ზვიად გაბროშვილი

სიმონ გელაშვილი

# სტატისტიკურ მონაცემთა დამუშავება და ანალიზი MS SQL Server-ის გამოყენებით

თბილისი
2025

# სტატისტიკურ მონაცემთა დამუშავება და ანალიზი MS SQL Server-ის გამოყენებით

Ivane Javakhishvili Tbilisi State University

Zviad Gabroshvili, Simon Gelashvili

# Statistical data processing and analysis using SQL Server

Textbook

Tbilisi
2025

ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტი

**ზვიად გაბროშვილი, სიმონ გელაშვილი**

# სტატისტიკურ მონაცემთა დამუშავება და ანალიზი MS SQL Server-ის გამოყენებით

სახელმძღვანელო

უნივერსიტეტის
გამომცემლობა

*წარმოდგენილი სახელმძღვანელო მომზადდა ანაღოგიური დასახელების სასწავლო საგნის ამჟამად მოქმედი სილაბუსის – „სტატისტიკუი მონაცემთა დამუშავება და ანალიზი SQL Server-ის გამოყენებით“ – შესაბამისად. ეს საგანი, რომელსაც სავალდებულო საგნის სტატუსი აქვს, ისწავლება ივ. ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის ეკონომიკის სამაგისტრო პროგრამის ორ მოდულში: „ეკონომიკური და ბიზნესის სტატისტიკა“ და „სოციალური და დემოგრაფიული სტატისტიკა“.*

*სახელმძღვანელო განკუთვნილია უმაღლესი სასწავლებლების ეკონომიკის, ბიზნესის ადმინისტრირებისა და სოციალურ მეცნიერებათა სამაგისტრო და სადოქტორო პროგრამების სტუდენტებისთვის. იგი დიდ დახმარებას გაუწევს, ასევე, საჯარო და კერძო სექტორებში დასაქმებულ იმ პრაქტიკოს სპეციალისტებს, რომლებიც ახორციელებენ რაოდენობრივ მონაცემთა, განსაკუთრებით დიდი მოცულობის, დამუშავებასა და ანალიზს. ამჟამად ასეთი საქმიანობის სფეროში უკვე მნიშვნელოვნად გაფართოვდა და, შესაბამისად, გაიზარდა მოთხოვნა ასეთი უნარ-ჩვევების მქონე სტატისტიკოს-ანალიტიკოსებზე.*

სამეცნიერო რედაქტორი: ეკონომიკურ მეცნიერებათა დოქტორი,
პროფესორი სიმონ გელაშვილი

რეცენზენტი: ასოცირებული პროფესორი ნინო თოფურია

იბეჭდება ივანე ჯავახიშვილის სახელობის თსუ ეკონომიკისა და ბიზნესის ფაკულტეტის საგამომცემლო საბჭოს დადგენილების საფუძველზე (ოქმი #3, 25.11.2024).

**ერთ-ერთი ავტორის წერილობითი ნებართვის გარეშე აკრძალულია ამ წიგნის, ან მისი ნაწილების ნებისმიერი ნაბეჭდი სახით ასლების გამრავლება და გავრცელება. ამ წესის დარღვევა ისჯება საქართველოში მოქმედი კანონმდებლობის შესაბამისად.**

© ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის გამომცემლობა, 2025

ISBN 978-9941-36-391-7

# წინასიტყვაობა

მოცემული სახელმძღვანელო სტუდენტებს საშუალებას მისცემს მარტივად შეძლონ რელაციურ მონაცემთა ბაზების გამოყენება და მათი დანერგვა პრაქტიკაში. მასში გადმოცემულია, თუ როგორ გამოვიყენოთ Microsoft SQL Server 2022, Microsoft SQL Server Management 2022 და Microsoft Office Access მონაცემებთან მუშაობაში, იქნება ეს საჭირო პროგრამის ინსტალაცია, ბაზებისა და ცხრილების შექმნა, თუ შემდგომ ამ ბაზებთან დაკავშირება, მართვა და საჭირო მონაცემების დამუშავება. ამასთან ერთად, ზემოაღნიშნული პროგრამები საკმაოდ ხშირად გამოიყენება საქართველოს საჯარო და კერძო სექტორებში და სტუდენტი შეძლებს შრომის ბაზარზე წარმოაჩინოს თავისი ტექნიკურ-ანალიზური უნარები, რაც მისცემს მას უპირატესობას. უნდა აღინიშნოს, რომ საკითხები განხილულია პრაქტიკული მაგალითებით თავისი საჭირო კოდებით, რაც საშუალებას აძლევს სტუდენტს პრაქტიკულად შეძლოს ნასწავლის გამოყენება რეალური ამოცანების შესრულებაში და, ასევე, დაეხმარება ლექტორსაც პრაქტიკული მეცადინეობა ააგოს განხილული მაგალითების შესრულებით. თითოეული თავის ბოლოს მოცემულია კითხვები თვითშემოწმებისთვის, რაც სტუდენტს შესაძლებლობას მისცემს განახორციელოს გავლილი საკითხების თვითშეფასება.

წიგნი შედგება 13 თავისგან. **პირველ თავში** ზოგადად განხილულია მონაცემთა ბაზები, მათი კლასიფიკაცია, რელაციური ალგებრა და საჭირო პროგრამების ინსტალაცია. აქ განიხილება, თუ რატომ უნდა შევისწავლოთ მონაცემთა ბაზების მართვის სისტემები, რა სხვაობაა რელაციურ და არარელაციურ ბაზებს შორის, მოკლედ მიმოიხილება რელაციური ალგებრა და ბაზების მართვისთვის საჭირო ალტერნატიული პროგრამები. **მეორე თავში** ახსნილია მომხმარებლები, სქემა და მონაცემთა ტიპები; განიხილება, თუ როგორ ხდება მომხმარებლებისა და სქემის შექმნა. **მესამე თავი** ეთმობა შექმნის, წაკითხვის, განახლებისა და წაშლის (CRUD) ოპერაციებს. აქ მკითხველი იწყებს მონაცემებთან მუშაობას. **მეოთხე თავში** უფრო დეტალურად განიხილება CRUD ოპერაციები, სადაც მოცემულია ოპერატორების, მონაცემთა ფილტრაციის, დახარისხების, აგრეგირების, ჩადგმული მოთხოვნების, საერთო ცხრილური გამოსახულებების და სხვადასხვა მეთოდით მონაცემთა გატანა შემდგომი გამოყენებისთვის. **მეხუთე თავში** განხილულია ცხრილებს შორის კავშირები, გაერთიანებები და ნორმალიზაციის თეორია. **მეექვსე თავში** ახსნილია მონაცემთა წარმოდგენები, მიმოიხილება სხვადასხვა ალგორითმი, ალგორითმული სირთულე და მონაცემთა ბაზებსა და ალგორითმულ სირთულეს შორის კავშირი. შემდეგ უკვე ხდება ინდექსირების განხილვა. ამ გზით სტუდენტი შეძლებს სხვადასხვა ცხრილთან მუშაობის ოპტიმიზაციას. ამ თავის შესწავლის შემდეგ სტუდენტს შეეძლება უკვე პრაქტიკაში გამოიყენოს მიღებული ცოდნა და იმუშაოს მონაცემთა ბაზებთან სხვადასხვა კერძო თუ სახელმწიფო სტრუქტურაში. მეშვიდე თავიდან იწყება შედარებით მაღალი დონის საკითხები. **მეშვიდე თავში** განიხილება პროგრამირებისა და ავტომატიზაციის საკითხები, სადაც ხდება პროგრამირების პარადიგმების მოკლე მიმოხილვა, პროგრამული ენის ელემენტების განხილვა T-SQL-ში, ტრანზაქციები, შენახული პროცედურები და მათი პრაქტიკული გამოყენება ავტომატიზაციაში SQL Server Agent-ის საშუალებით. ამ თავის

დასრულების შემდეგ სტუდენტი შეძლებს მონაცემთა ბაზებში გარკვეული სახის ამოცანების ავტომატიზებას, რაც საშუალებას მისცემს სხვადასხვა ორგანიზაციას უფრო ეფექტიანად განახორციელოს საქმიანობა, რომელიც ეხება მონაცემთა დამუშავებას და ანალიზს. **მერვე თავში** განიხილება სხვადასხვა დამხმარე ფუნქცია, იქნება ეს ტექსტური, მათემატიკური თუ თარიღებთან სამუშაოდ, რაც ისევ გამოადგება მკითხველს ავტომატიზაციაში. **მეცხრე თავი** ეთმობა ფუნქციებისა და ტრიგერების განხილვას. ფუნქციების გამოყენებით შესაძლებელი იქნება უფრო სუფთა კოდის წერა და დაწერილი ფუნქციების გამოყენება, რათა შევამციროთ კოდის გამეორება სხვადასხვა ოპერაციისთვის. ტრიგერები აქ განხილულია მონაცემთა ბაზებში ცხრილების აუდიტისთვის. **მეათე თავიდან** იწყება Microsoft Office Access-ის განხილვა, ცხრილებისა და ფორმების შექმნა. **მეთერთმეტე თავში** უკვე ხდება Access-ში მოთხოვნებისა და რეპორტების შექმნა. **მეთორმეტე თავში** განიხილება Access-ის დაკავშირება SQL Server 2022-თან და შესაბამისი CRUD ოპერაციების შექმნა და გამოყენება კოდის წერის გარეშე. **მეცამეტე თავი,** ზოგადად, უფრო მეტად აღწერითი ხასიათისაა და მოკლედ განიხილება მონაცემები თანამედროვე მსოფლიოში, მონაცემთა როლი გადაწყვეტილებების მიღებისას და მათი უსაფრთხოება, დიდი მონაცემები, რა პროგრამული ინსტრუმენტები და საშუალებები არსებობს მონაცემებთან მუშაობისას და მანქანური დასწავლა. ამ თავის შემდეგ სტუდენტი შეძლებს თავისი კარიერისთვის საჭირო მიმართულების განსაზღვრას, თუ სურს გააგრძელოს მუშაობა მონაცემთა ანალიზის მიმართულებით. მეცამეტე თავის შემდეგ მოცემულია მოკლე თავი დამატებითი რესურსების შესახებ, რაც მკითხველს საშუალებას მისცემს დამატებით გაეცნოს სხვადასხვა რესურსს და გაიღრმაოს ცოდნა მონაცემებთან მუშაობის მიმართულებით.

რაც შეეხება სწავლას და სწავლების პროცესს, მიზანშეწონილია სტუდენტი სახელმძღვანელოს ყველა თავს მიჰყვეს ინდივიდუალურად, გამოიყენოს დაწერილი კოდები და შეასრულოს ამოცანები და მაგალითები. თითოეულ თავს სასურველია სასწავლად დაეთმოს ერთი კალენდარული კვირა, გარდა მეშვიდე თავისა, სადაც, მასალის დიდი მოცულობიდან გამომდინარე, მიზანშეწონილია ორი აკადემიური კვირის დათმობა.

რა თქმა უნდა, წარმოდგენილ სახელმძღვანელოში ზოგიერთი საკითხი შეიძლება სადისკუსიო იყოს, ან მოიცავდეს გარკვეულ უზუსტობებს. მათი აღმოფხვრის მიზნით გამოთქმული ყველა მოსაზრება გათვალისწინებული იქნება ავტორთა მიერ წიგნის შემდგომი გამოცემისას.

*სამეცნიეო ხედაქტოხი,*
*პხოფესოხი სიმონ გეღაშვიღი*
*თბიღისი, სექტემბეხი, 2024*

# სარჩევი

# თავი 1. რელაციური და არარელაციური მონაცემთა ბაზების მიმოხილვა

## 1.1. რატომ უნდა ვისწავლოთ მონაცემთა ბაზების მართვის სისტემები

საქართველოში ბოლო პერიოდში ბევრი მნიშვნელოვანი ტექნოლოგიური ცვლილება მოხდა. ეკონომიკური ზრდის პარალელურად გაიზარდა ისეთი დარგების რაოდენობა, რომლებიც უშუალოდ ტექნოლოგიური მიმართულებით მუშაობენ. გარდა კერძო სექტორისა, სახელმწიფო სტრუქტურებშიც დაინერგა და, ასევე, მიმდინარეობს თანამედროვე ტექნოლოგიების დანერგვა სხვადასხვა მიმართულებით, იქნება ეს მონაცემთა შეგროვება, წინასწარი დამუშავება, ანალიზი, გავრცელება თუ სხვა. შესაბამისად, ყოველდღიურად იზრდება მოთხოვნა ისეთ კადრებზე, რომლებსაც შეუძლიათ თანამედროვე კომპიუტერულ ტექნოლოგიებთან მუშაობა. მეტიც, ზოგიერთი ტიპის დავალების ეფექტიანად და ეფექტურად შესასრულებლად მნიშვნელოვანი ხდება კონკრეტულ პროგრამებთან მუშაობის ცოდნა და გამოცდილება. ასეთ პროგრამებს შეუძლია მნიშვნელოვნად შეამციროს სამუშაო დრო ორგანიზაციაში და უფრო პროდუქტიული გახადოს შრომა, ასევე, ავტომატიზება გაუკეთოს განმეორებად დავალებებს და უკვე გამოთავისუფლებული დრო დაიხარჯოს უფრო მეტად შემოქმედებით საქმიანობაში, ახალი მეთოდოლოგიების დანერგვაში და ა. შ.

მეტად პოპულარული გახდა გადაწყვეტილებების მისაღებად მონაცემების გამოყენება. არსებული მონაცემების საფუძველზე ხდება სხვადასხვა ტიპის მოდელის შემუშავება, იქნება ეს ეკონომიკური, საბუნებისმეტყველო მეცნიერებების თუ სხვა. მონაცემთა ხარისხი და მოცულობა ხშირად გავლენას ახდენს გადაწყვეტილებებზეც. იმისათვის, რომ მონაცემებთან მუშაობა უფრო კომფორტული გახდეს, სასურველია გამოიყენებოდეს სპეციალური კომპიუტერული პროგრამები, რომლებიც ამარტივებენ მუშაობის პროცესს.

მონაცემების შესანახად სხვადასხვა საშუალება არსებობს. მათგან საკმაო პოპულარობით სარგებლობს მონაცემთა ბაზები, რომელთა დახმარებით შესაძლებელია მონაცემების შენახვა, მათზე ოპერაციების განხორციელება, გავრცელება და სხვა. ზოგიერთ შემთხვევაში უშუალოდ მონაცემთა ბაზაშივე ხდება შესაძლებელი გამოთვლითი ოპერაციების შესრულება. ამა თუ იმ ორგანიზაციაში არსებულ მონაცემთა ბაზაში ერთი მომხმარებლის მიერ შესრულებული სამუშაო, სურვილის შემთხვევაში, შესაძლებელია ხელმისაწვდომი გახდეს სხვა თანამშრომლის ან დეპარტამენტისთვისაც. შესაბამისად, მნიშვნელოვნად იზოგება დრო, რომელიც უნდა გამოყენებულიყო ორგანიზაციაში თანამშრომლებს შორის მონაცემთა მიმოცვლისთვის.

გარდა სტატისტიკოსებისა და მონაცემთა ანალიტიკოსებისა, მონაცემთა ბაზები გამოიყენება პროგრამული უზრუნველყოფის ინჟინრების მიერ სხვადასხვა აპლიკაციის შესაქმნელად, იქნება ეს ვებაპლიკაციები, მობილური აპლიკაციები თუ სხვ.

ზემოხსენებულიდან გამომდინარე, მონაცემთა ბაზებთან მუშაობის ცოდნა და უნარ-ჩვევები ნებისმიერ ადამიანს საშუალებას მისცემს შეძლოს ბევრ მონაცემთან მუშაობა, მათი ანალიზი, განახორციელოს განმეორებადი დავალებების ავტომატიზება, გამოიყენოს მონაცემთა ბაზები მონაცემთა გავრცელებისთვის და სხვ. ასევე, პიროვნებას საშუალება მიეცემა ჰქონდეს საჭირო უნარები ამ მიმართულებით, რომელიც მისი კარიერისთვის მნიშვნელოვანი დადებითი მხარე აღმოჩნდება.

## 1.2. რელაციური და არარელაციური მონაცემთა ბაზები

მონაცემთა ბაზების სხვადასხვა კლასიფიკაცია არსებობს. მონაცემთა ბაზების კლასიფიცირება შეიძლება მოვახდინოთ მონაცემების მოდელების მიხედვით. მონაცემების მოდელი არის მონაცემების სტრუქტურისა და მათი დამუშავების ოპერაციების ერთობლიობა. მონაცემების მოდელის მიხედვით შესაძლებელია ობიექტების სტრუქტურისა და მათ შორის არსებული კავშირების წარმოდგენა. არსებობს მონაცემების იერარქიული, ქსელური და რელაციური მოდელები. შესაბამისად, არსებობს იერარქიული, ქსელური და რელაციური მონაცემთა ბაზები [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 19].

ამ ქვეთავში განვიხილავთ მონაცემთა ბაზის ერთ-ერთ მნიშვნელოვან კლასიფიკაციას, რომლის მიხედვითაც, მონაცემთა ბაზა შეიძლება იყოს რელაციური, ან არარელაციური.

რელაციური მონაცემთა ბაზები იყენებენ რელაციურ მოდელს, რომელიც შეიმუშავა ამერიკელმა მეცნიერმა **ედგარ კოდმა** და ეს ცნობილი გახდა 1970 წელს. რელაციური მოდელის შემთხვევაში მონაცემების წარმოდგენა ხდება ბაზაში ორგანზომილებიანი ცხრილების სახით. ედგარ კოდმა მონაცემების დამუშავებისათვის შემოგვთავაზა სიმრავლეთა თეორიის აპარატი და დაამტკიცა, რომ მონაცემთა ნებისმიერი ნაკრები შეიძლება ორგანზომილებიანი ცხრილის სახით წარმოვადგინოთ. ამრიგად, რელაციური არის მონაცემთა ისეთი ბაზა, რომელშიც მონაცემები წარმოდგენილია სწორკუთხა ორგანზომილებიანი ცხრილებით [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 20].

რელაციური მოდელის გასაგებად მოვიყვანოთ მარტივი მაგალითი. პრაქტიკაში რელაციური მოდელის გამოყენების შემთხვევაში ბაზაში მოცემული შეიძლება იყოს სხვადასხვა ცხრილი. ეს ცხრილები შედგება სვეტებისგან (ინგლ. Columns) და სტრიქონებისგან (ინგლ. Rows). ცხრილში მოცემული სვეტები შეიძლება ერთმანეთთან დაკავშირებული იყოს კონკრეტული სვეტის ან სვეტების საფუძელზე. ასეთი კავშირების გამოყენების შემთხვევაში, ერთი ცხრილის მიბმა შესაძლებელია მეორეზე და არსებული კავშირების საშულებით კონკრეტული ოპერაციების განხორციელება. ცხრილში 1.1 მოცემულია NACE Rev. 2 კლასიფიკატორი სექციის დონეზე, ხოლო ცხრილში 1.2 მოცემულია გამოშვება მილიონ ფულად ერთეულში კლასიფიკატორის მიხედვით. დავუშვათ, გვსურს პირველი ცხრილის საფუძველზე გარდავქმნათ მეორე ცხრილი ისე, რომ მოიცავდეს სექციის დასახელებასა და გამოშვებას. ამ ცხრილების დაკავშირება შესაძლებელია სექციის სვეტით, რომელიც მოთავსებულია ორივე ცხრილში. შედეგად

მივიღებთ ცხრილს, რომელიც მოიცავს სექციის დასახელებასა და გამოშვებას მილიონ ფულად ერთეულში. შედეგი მოცემულია ცხრილში 1.3.

**ცხრილი 1.1. NACE Rev. 2 კლასიფიკატორი სექციის დონეზე**

| სექცია | სექციისდასახელება |
|---|---|
| A | სოფლის, სატყეო და თევზის მეურნეობა |
| B | სამთომოპოვებითი მრეწველობა |
| C | დამამუშავებელი მრეწველობა |
| D | ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰაერის მიწოდება |
| E | წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბინძურებისაგან გასუფთავების საქმიანობები |
| F | მშენებლობა |
| G | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოციკლების რემონტი |
| H | ტრანსპორტი და დასაწყობება |
| I | განთავსების საშუალებებით უზრუნველყოფის და საკვების მიწოდების საქმიანობები |
| J | ინფორმაცია და კომუნიკაცია |
| K | საფინანსო და სადაზღვევო საქმიანობები |
| L | უძრავ ქონებასთან დაკავშირებული საქმიანობები |
| M | პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები |
| N | ადმინისტრაციული და დამხმარე მომსახურების საქმიანობები |
| O | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო სოციალური უსაფრთხოება |
| P | განათლება |
| Q | ჯანდაცვა და სოციალური მომსახურების საქმიანობები |
| R | ხელოვნება, გართობა და დასვენება |
| S | სხვა სახის მომსახურება |
| T | შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები; არადიფერენცირებული საქონლის და მომსახურების წარმოება შინამეურნეობების მიერ საკუთარი მოხმარებისათვის |
| U | ექსტერიტორიული ორგანიზაციების და ორგანოების საქმიანობები |

**ცხრილი 1.2. სექცია და გამოშვება**

| სექცია | გამოშვება მილიონ ფულად ერთეულში |
|---|---|
| A | 100 |
| B | 200 |
| C | 150 |
| D | 350 |
| E | 140 |
| F | 180 |

**ცხრილი 1.3. საშედეგო ცხრილი**

| სექციის დასახელება | გამოშვება მილიონ ფულად ერთეულში |
|---|---|
| **სოფლის, სატყეო და თევზის მეურნეობა** | 100 |
| **სამთომოპოვებითი მრეწველობა** | 200 |
| **დამამუშავებელი მრეწველობა** | 150 |
| **ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰაერის მიწოდება** | 350 |
| **წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბინძურებისაგან გასუფთავების საქმიანობები** | 140 |
| **მშენებლობა** | 180 |

აღნიშნული მეთოდით ერთ მონაცემთა ბაზაში ხდება ერთი და იგივე მონაცემების გამეორების შემცირება, რასაც შეუძლია გამოათავისუფლოს სივრცე სერვერიდან. თუმცა, აქვე უნდა აღინიშნოს ისიც, რომ არის ოპერაციები, რომელთა განხორციელება დიდ დროს მოითხოვს და გამოთვლითი დანახარჯები უფრო მეტია, ვიდრე სივრცითი დანახარჯები. ასეთ შემთხვევაში მიზანშეწონილია განხორციელდეს ოპერაცია და შეინახოს ის ცალკე ცხრილად, ვიდრე სერვერიდან მისი ყოველი მოთხოვნისას დაიხარჯოს უფრო მეტი დრო. როდესაც ოპერაციის ხანგრძლივობა დიდია და ერთდროულად რამდენიმე მომხმარებელი იყენებს მონაცემთა ბაზას, ამ შემთხვევაში შესაძლებელია მსგავსმა ოპერაციებმა მონაცემთა ბაზა სხვა მომხმარებლებისთვის მიუწვდომელი გახადოს. ამიტომ კარგი ვარიანტია, თუ მოთხოვნის პასუხი, რომელიც ითხოვს დიდ დროს, შეინახოს მოთხოვნის შესრულებისას ერთხელ და შემდეგ მოხდეს უკვე შენახული მონაცემების გამოთხოვა მოგვიანებით სხვადასხვა მომხმარებლების მიერ, ვიდრე ყოველ ჯერზე დროითი დანახარჯების თვალსაზრისით შრომატევადი ოპერაციების შესრულება.

ზემოხსენებულის გარდა, მონაცემთა ბაზებში მუშაობისას ხშირად იყენებენ სპეციალურ კოდებს. მაგალითად, თუ გვაქვს კლასიფიკატორი, რომელიც მოიცავს 10 კლასს და ეს კლასები წარმოდგენილია ტექსტური სახით, 10 კლასისთვის შეიძლება გამოვიყენოთ ციფრები 0-დან 9-ის ჩათვლით, შევქმნათ ცხრილი, რომელიც თითოეულ ციფრს შეუსაბამებს კონკრეტულ კლასს, ხოლო მეორე ცხრილი მოიცავს კლასს და შესაბამის მონაცემს. სხვადასხვა ოპერაციებისთვის სივრცის შესამცირებლად, შეგვიძლია მეორე ცხრილში ეს კლასები ჩავანაცვლოთ კონკრეტული ციფრებით, რაც პირველ ცხრილში მივუთითეთ. მოთხოვნის შესრულებისა და საშედეგო ცხრილის მიღების შემდეგ შეგვიძლია უკვე ციფრებით დავაკავშიროთ ეს ცხრილი პირველ ცხრილს და მონაცემები წარმოვადგინოთ კლასის დასახელებისა და შესაბამისი მონაცემის შემცველ ცხრილად, რომელიც, მაგალითად, შეიძლება იყოს რაიმე ინდექსის გამოთვლის შედეგი და სხვა.

რელაციური მონაცემთა ბაზების მართვის სისტემები (RDBMS) იყენებენ სტრუქტურული მოთხოვნების ენას (Structured Query Language), რომელსაც მოკლედ მოიხსენიებენ როგორც SQL-ს. SQL არის პროგრამული ენა, რომელიც გამოიყენება მონაცემთა ბაზებთან მუშაობისას. ამასთან, არსებობს SQL-ის სხვადასხვა დიალექტი (მოდიფიკაცია).

რაც შეეხება არარელაციურ მონაცემთა ბაზებს, ამ შემთხვევაში არ ხდება ცხრილის ფორმით წარმოდგენა მონაცემთა ბაზაში. არის შემთხვევები, როდესაც უფრო მეტად მიზანშეწონილია არარელაციური მონაცემთა ბაზის მართვის სისტემის გამოყენება, როდესაც ცხრილური სახით მონაცემთა წარმოდგენა უფრო ნაკლებად ეფექტიანია კონკრეტული ოპერაციის შესრულებისას, ვიდრე სხვა ფორმით.

## 1.3. რელაციური ალგებრის მოკლე მიმოხილვა

რელაციური ალგებრა მონაცემთა მოდელირებისთვის იყენებს ალგებრულ სტრუქტურებს. რაც შეეხება ოპერატორებს, სიმრავლის თეორიიდან გამომდინარე, რელაციურ ალგებრაში მოცემულია სიმრავლეების გაერთიანება, მათი სხვაობა და დეკარტესეული ნამრავლი.

რელაციურ ალგებრაში მოცემულია, ასევე, პროექცია $\prod_{a1,a2,...an}(R)$, სადაც R არის კავშირი, ხოლო $a_1$, $a_2$,..., $a_n$ ატრიბუტებია. მაგალითისთვის წარმოვადგინოთ რაიმე ცხრილი S, რომლის სვეტები დეპარტამენტია, ასევე სახელი და გვარია და გვსურს ამ ცხრილიდან მხოლოდ დეპარტამენტი და გვარი ამოვიღოთ. მაშინ შესაბამისი პროექცია იქნება $\prod$დეპარტამენტი, გვარი(S).

რელაციურ ალგებრაში, ასევე, გამოიყენება ამორჩევა/შეზღუდვა (ინგლ. Selection/restriction) $\sigma_{a\theta b}(R)$, სადაც $a$ და $b$ არის ატრიბუტები, ხოლო $\theta$ არის ბინარული ოპერატორი სიმრავლიდან $\{<, \leq, =, >, \geq, \neq\}$, b-ს მაგივრად შეიძლება გამოიყენებოდეს კონკრეტული მნიშვნელობაც. მაგალითისთვის, თუ გვაქვს რაიმე ცხრილი T, რომელიც მოიცავს კონკრეტულ NACE Rev. 2 კლასს სექციის დონეზე და შესაბამის გამოშვებას მილიონ ფულად ერთეულში და გვსურს აღნიშნული ცხრილიდან ამოვარჩიოთ ისეთი საქმიანობის სახეები თავისი შესაბამისი გამოშვებით, რომელთა გამოშვება მეტია, დავუშვათ, 100 მილიონ ფულად ერთეულზე, რელაციურ ალგებრაში ეს ჩაიწერება როგორც $\sigma_{\text{გამოშვება}\geq 100000000}(T)$.

გადარქმევის ოპერატორი გამოიყენება კონკრეტული ატრიბუტისთვის სახელის შესაცვლელად. მაგალითად, $\rho_{a/b}(R)$, სადაც R არის კავშირი, a და b - ატრიბუტები, ხოლო b არის ის ატრიბუტი, რომელიც ეკუთვნის R-ს. თუ არსებობს ცხრილი, რომელიც მოიცავს ატრიბუტ b-ს და შესრულდა გადარქმევის ოპერაცია $\rho_{a/b}(R)$, მაშინ ახალ ცხრილში ატრიბუტ b-ს შეეცვლება სახელი და გახდება a.

არსებობს, ასევე, კავშირის და სხვა ოპერატორები. კავშირის შემთხვევაში არის სხვადასხვა ტიპის კავშირები. კავშირებში ბუნებრივი კავშირი აღინიშნება $\bowtie$ ოპერატორით. კავშირებს უფრო დეტალურად განვიხილავთ სხვა თავში.

## 1.4. არსებული კომპიუტერული პროდუქტების მიმოხილვა

ბაზარზე მონაცემთა ბაზის მართვის სხვადასხვა აპლიკაცია არსებობს. დიდი პოპულარობით სარგებლობს Microsoft SQL Server, MySQL, PostgreSQL, MongoDB და მო-

ნაცემთა ბაზის მართვის სხვა სისტემები. ქვემოთ მოკლედ განვიხილავთ აღნიშნულ პროდუქტებს.

Microsoft SQL Server არის Microsoft-ის პროდუქტი, რელაციურ მონაცემთა ბაზის მართვის სისტემა. მას ჩაშენებული აქვს საკმაოდ მძლავრი ფუნქციები, რომლებიც მონაცემებთან მუშაობას ამარტივებენ სხვადასხვა მიმართულებით. იგი იყენებს T-SQL-ს, რომელიც არის SQL-ის დიალექტი. აღნიშნული პროდუქტი, ძირითადად, ფასიანია, თუმცა, არსებობს მისი უფასოდ ხელმისაწვდომი ვერსიებიც. მას იყენებენ სხვადასხვა დანიშნულებით. საქართველოში ამჟამად სხვადასხვა საჯარო თუ კერძო ორგანიზაციაა მისი მომხმარებელი. მარტივად ინტეგრირდება Microsoft-ის პროდუქტებთან, როგორიცაა, მაგალითად, Microsoft Office. ორგანიზაციაში, რომელიც იყენებს Microsoft Windows-ის ოპერაციულ სისტემებს და აქვთ დომენის სისტემა აწყობილი, SQL Server-ის ჩართვისას სისტემაში შეუძლიათ აუთენთიკაცია განახორციელონ დომენის საშუალებით და ქსელში მარტივად გახადონ ხელმისაწვდომი SQL Server-ით აწყობილი მონაცემთა ბაზები. ამ წიგნშიც სწორედ SQL Server 2022-ია განხილული და ნაჩვენებია მისი პრაქტიკული გამოყენება.

MySQL ერთ-ერთი პოპულარული რელაციურ მონაცემთა ბაზის მართვის სისტემის პროდუქტია, რომელიც უფასოდ ხელმისაწვდომია და შეიცავს ღია კოდს. ის იყენებს SQL პროგრამულ ენას. იგი აქტიურად გამოიყენება სხვადასხვა დანიშნულებით ბევრი ორგანიზაციის მიერ საქართველოში და საზღვარგარეთ. პოპულარულია მისი გამოყენება ვებაპლიკაციების დეველოპმენტის დროს.

PostgreSQL იყენებს SQL-ს, არის უფასოდ ხელმისაწვდომი და ღია კოდის მატარებელი რელაციურ მონაცემთა ბაზის მართვის სისტემა. ისიც აქტიურად გამოიყენება სხვადასხვა მიმართულებით და, ხშირ შემთხვევაში, საკმაოდ სწრაფია. მისი დაკავშირება შესაძლებელია სხვადასხვა პროგრამებთანაც.

Oracle Database პოპულარული მრავალმოდელიანი მონაცემთა ბაზის მართვის სისტემაა, რომელიც გამოიყენება როგორც საზღვარგარეთ, ისე საქართველოში სხვადასხვა ორგანიზაციის მიერ მონაცემებთან მუშაობისთვის. იგი ფასიანია და იყენებს SQL-ს.

MongoDB დოკუმენტებზე ორიენტირებული არარელაციურ მონაცემთა ბაზის პროგრამაა, რომელიც აქტიურად გამოიყენება სხვადასხვა მიმართულებით, განსაკუთრებით დიდ მონაცემებთან მუშაობისას. ცხრილებისგან განსხვავებით, იყენებს JSON-ის მსგავს დოკუმენტებს.

## 1.5. Microsoft SQL Server 2022-ის ინსტალაცია

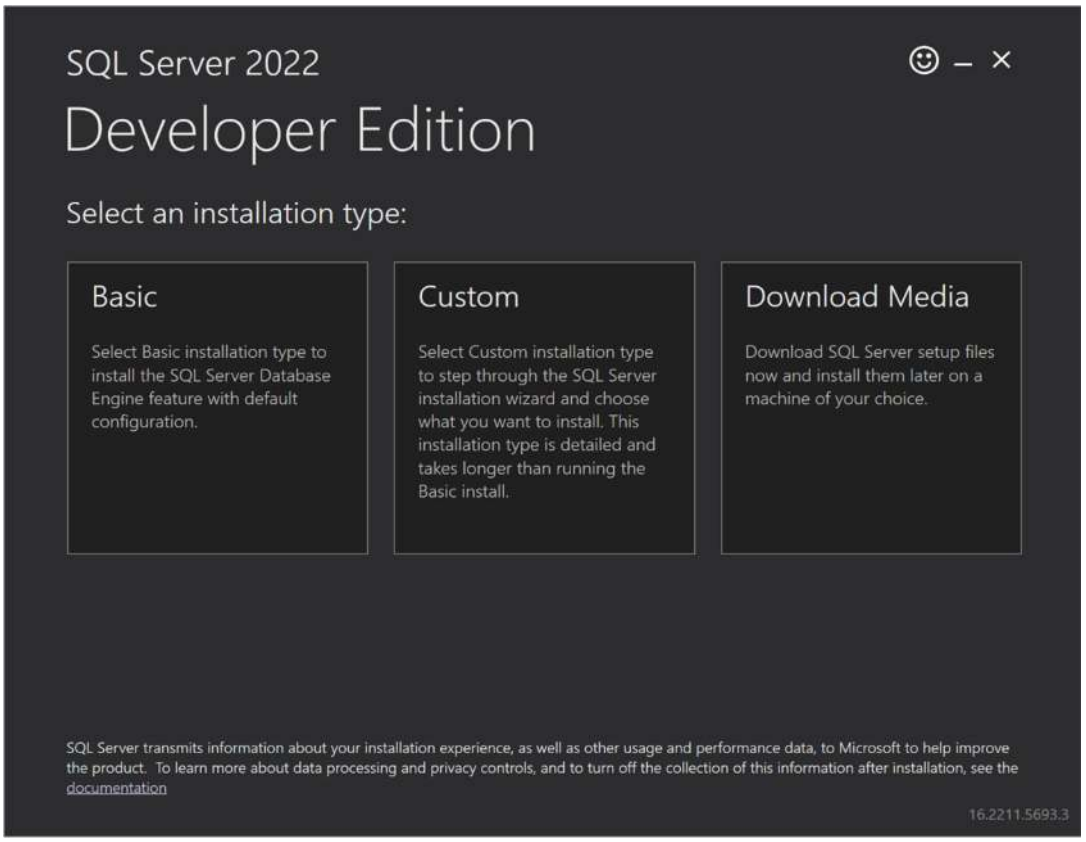
დავაყენოთ Microsoft SQL Server 2022-ის Developer ვერსია. ჯერ გადმოვწეროთ შემდეგი ლინკიდან ზემოხსენებული პროგრამის Developer ვერსია:

https://www.microsoft.com/en-us/sql-server/sql-server-downloads

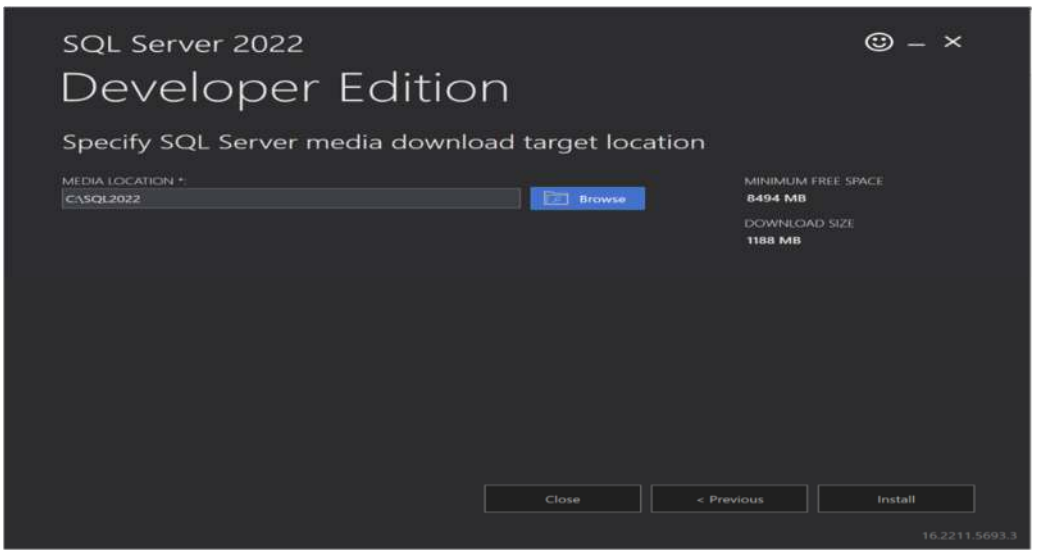
გავხსნათ ფაილი და გამოვა ფანჯარა, რომელიც მოითხოვს ადმინისტრატორის პრივილეგიას და დავეთანხმებით. შემდეგ გამოვა ფანჯარა, რომელიც სურათზე 1.1 გამოსახული ფანჯრის მსგავსი იქნება; იგი აგვარჩევინებს ინსტალაციის ტიპს. აქ უნდა ავირჩიოთ Custom. შემდეგ უნდა ავირჩიოთ ადგილმდებარეობა, თუ სად გვსურს გად-

მოიწეროს პროგრამა Browse ღილაკით. ამის შემდეგ დავაწკაპუნებთ Install ღილაკზე და დაიწყება გადმოწერა, რომელიც ასევე გვაჩვენებს პროგრესს.

**სურათი 1.1. ინსტალაციის ტიპის არჩევა**

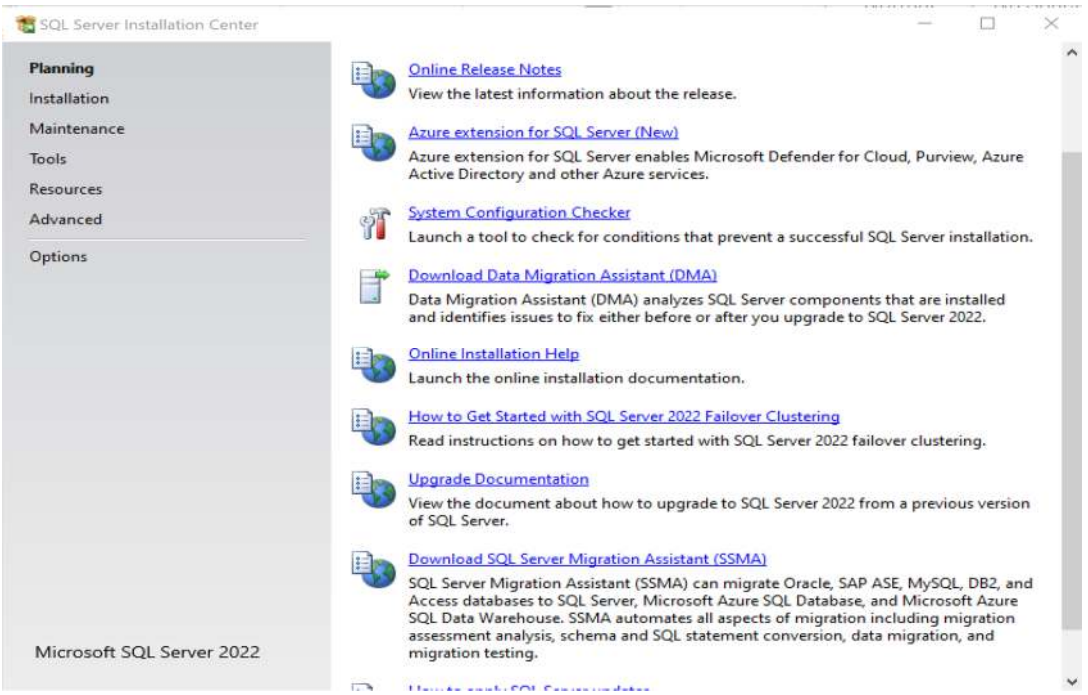



**სურათი 1.2. გადმოსაწერი ადგილმდებარეობის არჩევა**

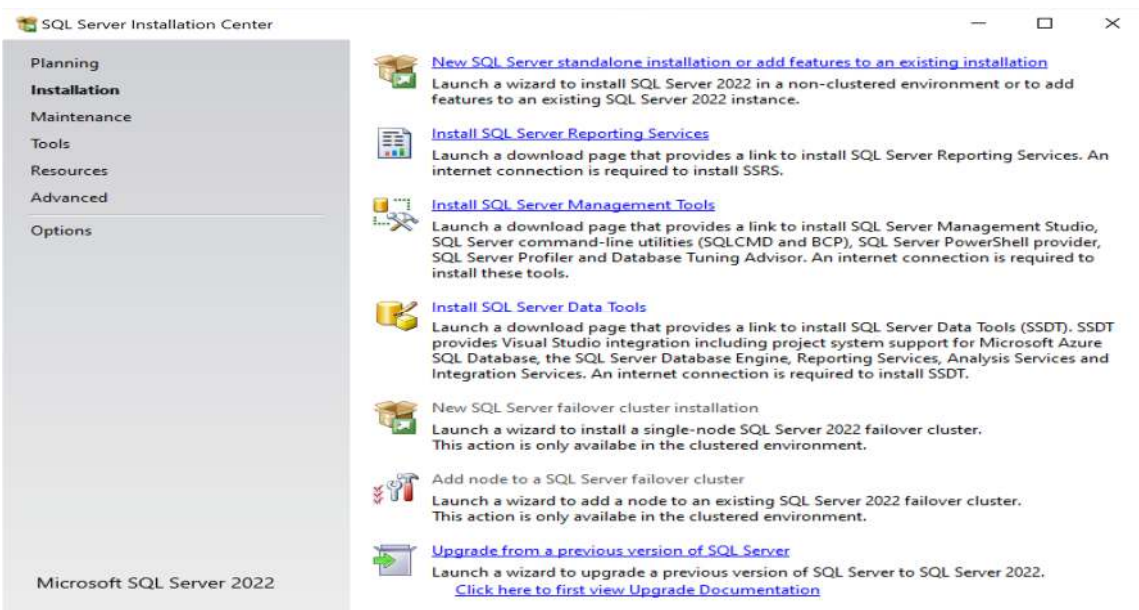


გადმოწერის შემდეგ გაეშვება საინსტალაციო, რომელიც გამოსახულია სურათზე 1.3. მოცემული ფანჯრის მარცხენა მენიუში ვირჩევთ Installation-ს, რის შემდეგაც პროგრამაში შეიცვლება მარჯვენა პანელში არსებული ღილაკები. შედეგი იქნება სურათზე 1.4 გამოსახული ფანჯრის მსგავსი, სადაც ავირჩევთ New SQL Server stanalone installation or add feature to existing installation-ს.

**სურათი 1.3. საინსტალაციო ფანჯარა**

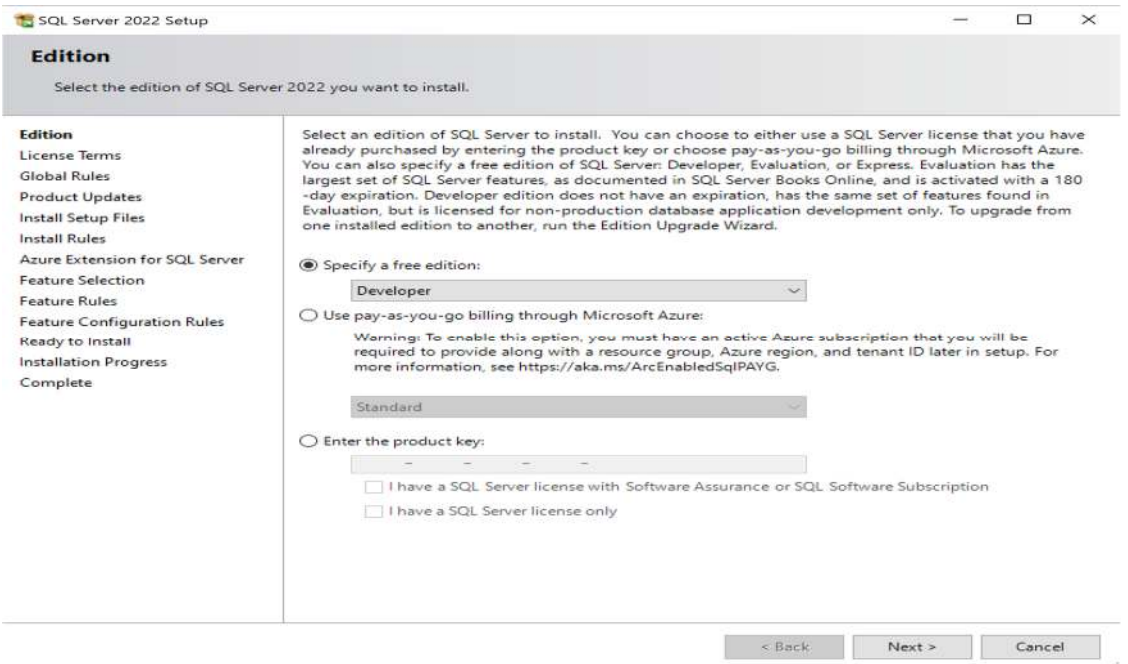



**სურათი 1.4. საინსტალაციო ფანჯარა**

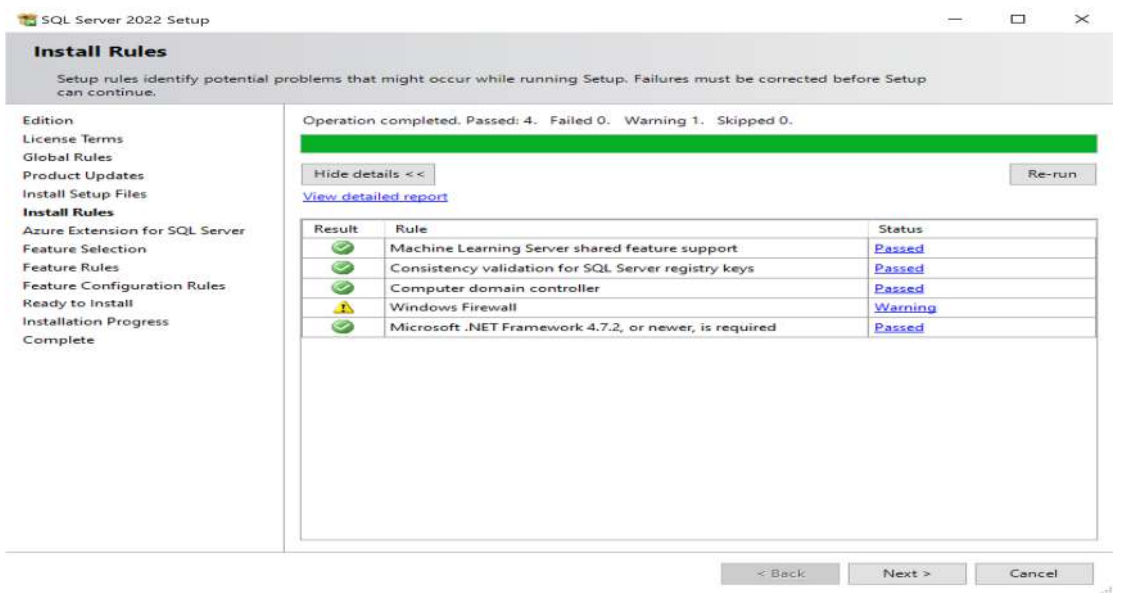


გამოვა ახალი ფანჯარა, რომელიც იქნება სურათზე 1.5 გამოსახული ფანჯრის მსგავსი. მნიშვნელოვანია, რომ პარამეტრები ავირჩიოთ ისე, როგორც არის მოცემული სურათზე 1.5. შემდეგ დავაწკაპუნოთ Next-ს. შემდეგ გვერდზე ვეთანხმებით ლიცენზიას, ვაგრძელებთ Next ღილაკზე დაწკაპუნებით. მივიღებთ შედეგს, როგორიც გამოსახულია სურათზე 1.6, სადაც ვაგრძელებთ Next ღილაკით.

**სურათი 1.5. საინსტალაციო ფანჯარა, ვერსიის არჩევა**




**სურათი 1.6. საინსტალაციო ფანჯარა**



შემდეგ გვერდზე გადასვლისას ვთიშავთ Azure Extension for SQL Server მონიშვნას და ისევ ვაგრძელებთ Next ღილაკით. შემდეგ გვერდზე ვირჩევთ ფუნქციებს, რომლის დაყენებაც გვსურს. შეგვიძლია ავირჩიოთ სხვადასხვა ფუნქცია და ინსტალაციის ადგილმდებარეობა, თუმცა, მნიშვნელოვანია ავირჩიოთ ის ფუნქციები, რომლებიც მოცემულია სურათზე 1.7. ამის შემდეგ ვაგრძელებთ ისევ Next ღილაკით და შემდეგ გვერდზე გადასვლისას ვიმეორებთ იმავეს.

**სურათ 1.7. SQL Server-ის ფუნქციები**

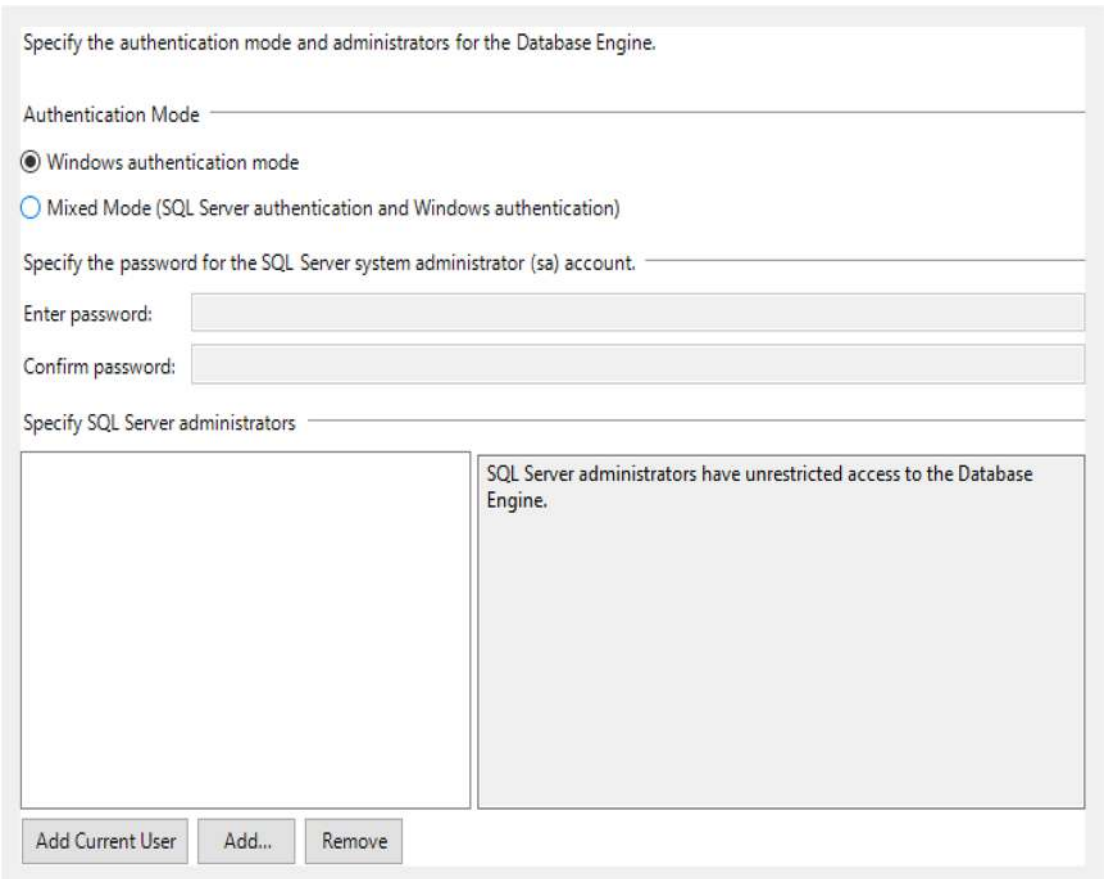

ზემოხსენებული ოპერაციების შესრულების შემდეგ, მომდევნო გვერდზე დავრწმუნდეთ, რომ პარამეტრები ისეა არჩეული, როგორც გამოსახულია სურათზე 1.8 და გავაგრძელოთ Next ღილაკით.

**სურათი 1.8. სერვერის კონფიგურაცია**

| Service | Account Name | Password | Startup Type |
|---|---|---|---|
| SQL Server Agent | NT Service\SQLSERVERAGENT | | Manual |
| SQL Server Database Engine | NT Service\MSSQLSERVER | | Automatic |
| SQL Server Browser | NT AUTHORITY\LOCAL SERVICE | | Disabled |

შემდეგ გვერდზე მონაცემთა ბაზის ძრავის კონფიგურაციაში, სერვერის კონფიგურაციის ჩანართზე, რომელიც გამოსახულია სურათზე 1.9, შეგვიძლია ავირჩიოთ Windows authentication mode ანMixed Mode (შერეული ვარიანტი), რომელიც მოგვთხოვს პაროლის დაყენებას SQL Server სისტემის ადმინისტრატორის (sa) ანგარიშზე. აქვე ვაწკაპუნებთ Add Current User ღილაკზე და ველოდებით, სანამ Specify SQL Server administrators-ს ველში არ გამოჩნდება შესაბამისი მომხმარებელი და შემდეგ ისევ ვაგრძელებთ Next ღილაკით.

სურათი 1.9. მონაცემთა ბაზის ძრავის კონფიგურაცია/სერვერის კონფიგურაცია




აღნიშნულის შესრულების შემდეგ ავტომატურად გადავალთ Ready to Install გვერდზე, სადაც ვაწკაპუნებთ Install ღილაკზე. ინსტალაცია დაიწყება, პროგრესი გამოსახული იქნება მწვანე პროგრესის ზოლზე. დაველოდებით ინსტალაციას. პროგრამა ავტომატურად გადავა მომდევნო გვერდზე, სადაც Close ღილაკით დავხურავთ საინსტალაციოს. თუ ინსტალაცია დასრულდა ხარვეზების გარეშე, მაშინ უკვე SQL Server 2022 Developer დაყენებულია და მუშაობს.

## 1.6. Microsoft SQL Server Management Studio 2019-ის ინსტალაცია

SQL Server 2022-ის ინსტალაციის შემდეგ შეგვიძლია დავუკავშირდეთ ბაზას, რომლის სხვადასხვა გზა არსებობს. ბაზასთან დაკავშირება შესაძლებელია Microsoft Office-ის აპლიკაციებით, როგორიცაა, მაგალითად, Excel და Access. მოგვიანებით განვიხილავთ Microsoft Office Access-ის გამოყენებას ბაზის სამართავად. გარდა ამისა, მოცემულ ბაზასთან დაკავშირება შესაძლებელია ისეთი პროგრამული ენებით, როგორე-

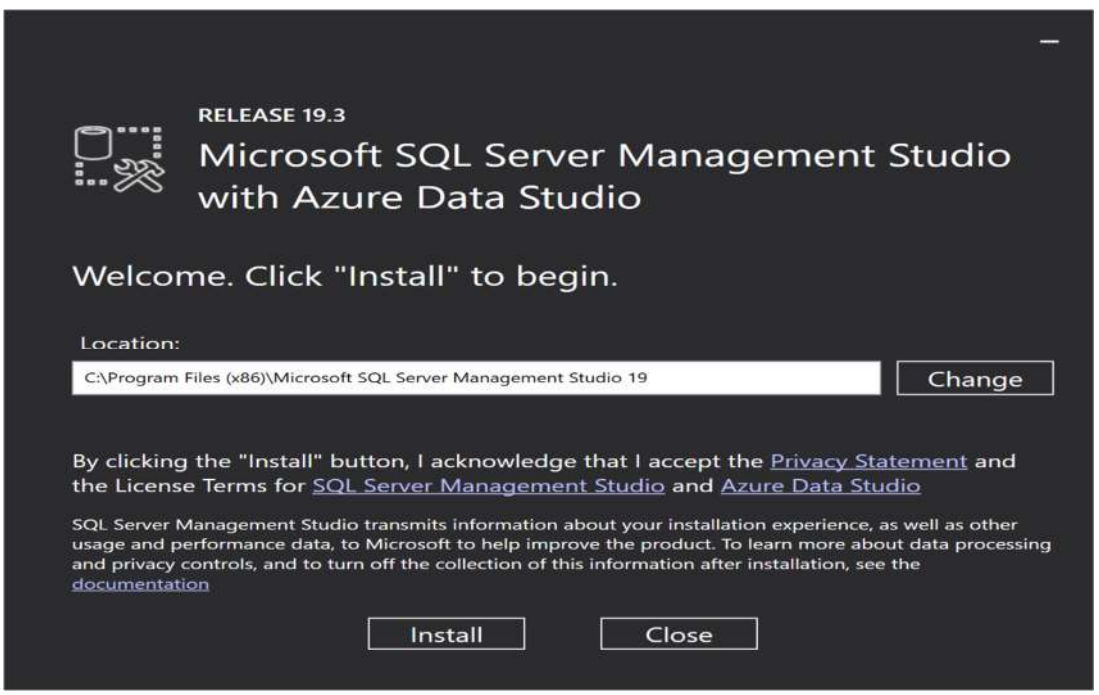
ბიცაა Python, R, C#, Java, C++ და სხვა. ამჯერად, ჩვენს შემთხვევაში, გამოვიყენოთ ინტეგრირებული დეველოპმენტის გარემო (IDE), რომელიც შექმნილია სპეციალურად SQL Server-ისთვის. ასეთი პროგრამა გახლავთ Microsoft SQL Server Management Studio 2019, ან მოკლედ SSMS 2019. აღნიშნულ პროგრამით შეგვეძლება დავუკავშირდეთ მონაცემთა ბაზას, დავწეროთ კოდები და მარტივად შევასრულოთ მონაცემთა ბაზის ისეთი ოპერაციები, როგორიცაა: მონაცემთა ბაზების შექმნა, მათში ცხრილების შექმნა, ავტომატიზება და სხვა. გადმოსაწერად შევიდეთ მოცემულ ბმულზე და გადმოვიწეროთ SSMS 19:

https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16#download-ssms

გადმოწერის შემდეგ გავხსნათ საინსტალაციო. გამოვა სურათზე 1.10 გამოსახული ფანჯარა, სადაც შეგვიძლია შევცვალოთ საინსტალაციო ადგილმდებარეობა, ხოლო შემდეგ გავაგრძლოთ Install ღილაკით. ამის შემდეგ პროგრამა დაიწყებს ინსტალაციას.

ინსტალაციის დასრულების შემდეგ მივიღებთ ფანჯარას, რომლის მსგავსი გამოსახულია სურათზე 1.11. ფანჯარა გვამცნობს, რომ პროგრამის დაყენება წარმატებით დასრულდა. შეგვიძლია აღნიშნული ფანჯარა დავხუროთ Close ღილაკზე დაწკაპუნებით.

**სურათი 1.10. SSMS 19-ის საინსტალაციო ფანჯარა**



სურათი 1.11. SSMS 19 ინსტალაციის დასრულება


### *კითხვები თვითშემოწმებისთვის*

1. რატომ უნდა ვისწავლოთ მონაცემთა ბაზების მართვის სისტემები?
2. რით განსხვავდება ერთმანეთისგან რელაციური და არარელაციური მონაცემთა ბაზები?
3. რა განსხვავებაა პროექციას და ამორჩევას/შეზღუდვას შორის რელაციურ ალგებრაში?
4. ჩამოთვალეთ მონაცემთა ბაზების მართვის თანამედროვე სისტემები.

# თავი 2. Microsoft SQL Server 2022-ის შესავალი

## 2.1. მომხმარებლები

SQL Server 2022-ში მომხმარებლები ერთ-ერთ საკვანძო კომპონენტს წარმოადგენს. აღნიშნულ პროგრამაში განსხვავებული ტიპის მომხმარებელი (USER) არსებობს. მონაცემთა ბაზასთან წვდომისათვის გამოიყენება აუთენტიფიცირების სხვადასხვა მეთოდი. ზოგჯერ იქმნება მომხმარებელი ბაზისთვის, რომელსაც აქვს გარკვეული წვდომის დონე მოცემულ ბაზაში და აუთენტიფიცირებისთვის იყენებს პაროლს. მაგრამ, რადგან Microsoft SQL Server 2022-ს აქვს Active Directory-ის მხარდაჭერა, აქედან გამომდინარე, იმ ორგანიზაციაში, რომელიც იყენებს Active Directory-ის, დამატებითი პაროლის გარეშეც შესაძლებელია აუთენტიფიცირება.

**მომხმარებლების აუთენტიფიცირების შემდეგი ძირითადი რეჟიმებია:**

- აუთენტიფიცირების რეჟიმი Windows NT-ის საშუალებებით (Windows NT Authentication);
- აუთენტიფიცირების შერეული რეჟიმი (Windows NT Authentication and SQL Server Authentication).

შერეული რეჟიმი საშუალებას გვაძლევს დავრეგისტრირდეთ როგორც სერვერის, ისე WindowsNT-ის საშუალებით [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 297].

აუთენტიფიცირების რეჟიმის არჩევის დროს უზრუნველყოფილი უნდა იყოს მაქსიმალური უსაფრთხოება და ადმინისტრირების სიმარტივე. ასე მაგალითად, თუ მცირე ზომის ფირმა გვაქვს და ქსელისა და მონაცემთა ბაზების ადმინისტრატორის ფუნქციებს ერთი ადამიანი ითავსებს, მაშინ უმჯობესია გამოვიყენოთ აუთენტიფიცირების რეჟიმი Windows NT-ის საშუალებებით. თუ ფირმა დიდია და ქსელისა და მონაცემთა ბაზების ადმინისტრატორის ფუნქციებს სხვადასხვა ადამიანი ასრულებს, მაშინ უმჯობესია გამოვიყენოთ აუთენტიფიცირების შერეული რეჟიმი. წინააღმდეგ შემთხვევაში მონაცემთა ბაზის ადმინისტრატორს მოუწევს ხშირი მიმართვები ქსელის ადმინისტრატორთან თავისი ფუნქციების შესასრულებლად, კერძოდ, ახალი მომხმარებლის შესაქმნელად, პაროლის შესაცვლელად, სხვა ჯგუფში მომხმარებლის გადასაყვანად და ა. შ., რაც, ბუნებრივია, არასწორია. მეორე მხრივ, თუ გამოიყენება აუთენტიფიცირების რეჟიმი Windows NT-ის საშუალებებით, მაშინ მონაცემთა ბაზის ადმინისტრატორს შეეძლება არსებული სააღრიცხვო ჩანაწერების გამოყენება [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 298].

### Windows NT-ის აუთენტიფიცირების რეჟიმი

როცა მომხმარებელი სერვერს უერთდება Windows NT-ის სააღრიცხვო ჩანაწერის გამოყენებით, ამ დროს მყარდება სანდო შეერთება. აუთენტიფიცირება Windows NT-ის საშუალებებით გარკვეულ უპირატესობებს იძლევა. მომხმარებლებზე ავტომატურად ვრცელდება უსაფრთხოების პოლიტიკის წესები. მაგალითად, ავტომატურად მოწმდე-

ბა პაროლის მინიმალური სიგრძე და მისი მოქმედების ვადა. თუ მომხმარებელი რამდენიმეჯერ არასწორად შეიტანს პაროლს, მაშინ გარკვეული დროით ხდება მისი სააღრიცხვო ჩანაწერის დაბლოკვა. გარდა ამისა, რეგისტრირების დროს ხდება პაროლების დაშიფვრა. ყოველივე ეს, სერვერის მონაცემების დაცულობას ზრდის [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 299].

Windows NT-ის საშუალებებით აუთენტიფიცირებისას დომენის მომხმარებლის სააღრიცხვო ჩანაწერი (loginID) ინახება სერვერის master მონაცემთა ბაზაში, ხოლო ინფორმაცია მომხმარებლის სახელის, პაროლის და ა. შ. შესახებ კი - დომენის მონაცემთა ბაზაში. ინფორმაცია მომხმარებლის სააღრიცხვო ჩანაწერისა და Windows NT-ის ჯგუფებში წევრობის შესახებ სერვერის მიერ წაიკითხება დომენის უსაფრთხოების სისტემის მონაცემთა ბაზიდან მომხმარებლის მიერთების დროს.

ამრიგად, Windows NT-ის საშუალებებით აუთენტიფიცირების რეჟიმში სერვერის სტანდარტული sysadmin ან securityadmin როლის წევრმა სერვერისთვის უნდა მიუთითოს, თუ Windows NT-ის რომელ მომხმარებლებს ან ჯგუფებს აქვთ სერვერთან მიმართვის უფლება. Windows NT-ის საშუალებებით აუთენტიფიცირების დროს მომხმარებელს არ სჭირდება სახელისა და პაროლის მითითება სერვერთან მიმართვის მიზნით, რადგან მომხმარებლის სახელისა და პაროლის შემოწმებას Windows NT-ის დომენის კონტროლერი ასრულებს [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 299].

## სერვერის აუთენტიფიცირების რეჟიმი

სერვერთან შეერთების დასამყარებლად, რომელიც იმყოფება დომენში, რომელთანაც არ არის დამყარებული სანდო ურთიერთობები, შეგვიძლია გამოვიყენოთ სერვერის აუთენტიფიცირების რეჟიმი. ეს რეჟიმი გამოიყენება მაშინაც, როცა არ არის დომენში დარეგისტრირების საშუალება, მაგალითად, როცა სერვერს ინტერნეტით ვუკავშირდებით.

სერვერის აუთენტიფიცირების გამოყენების შემთხვევაში, სერვერთან მიმართვა ხორციელდება სერვერის სააღრიცხვო ჩანაწერების საფუძველზე.

სერვერის საშუალებებით აუთენტიფიცირებისათვის სერვერის sysadmin ან securityadmin სტანდარტული როლის წევრმა მომხმარებლისთვის უნდა შექმნას სააღრიცხვო ჩანაწერი და შეასრულოს მისი კონფიგურირება. სააღრიცხვო ჩანაწერი შეიცავს სააღრიცხვო ჩანაწერის სახელს, სერვერის უნიკალურ იდენტიფიკატორსა და პაროლს. ეს ინფორმაცია შენახული იქნება master მონაცემთა ბაზაში. შექმნილ სააღრიცხვო ჩანაწერს კავშირი არ აქვს Windows NT-ის სააღრიცხვო ჩანაწერებთან. ამ რეჟიმში სერვერი თვითონ ამოწმებს სახელისა და პაროლის სისწორეს.

სერვერის აუთენტიფიცირების რეჟიმი შეგვიძლია გამოვიყენოთ Novell NetWare, Unix და ა. შ. მომხმარებლებისთვის, აგრეთვე, სერვერთან ინტერნეტით დაკავშირებისას, როცა დომენში რეგისტრირება არ ხდება.

ჩვეულებრივ, სერვერზე სააღრიცხვო ჩანაწერი იქმნება მიმართვის უფლების მისანიჭებლად, მაგრამ შესაძლებელია სააღრიცხვო ჩანაწერი შეიქმნას მიმართვის აკრძალვის მიზნით. ამისათვის უნდა შევქმნათ Windows NT-ის ჯგუფი, ავუკრძალოთ მას

სერვერთან მიმართვა და მასში ჩავრთოთ ის მომხმარებლები, რომლებსაც აკრძალული აქვთ სერვერთან მიმართვა [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 299].

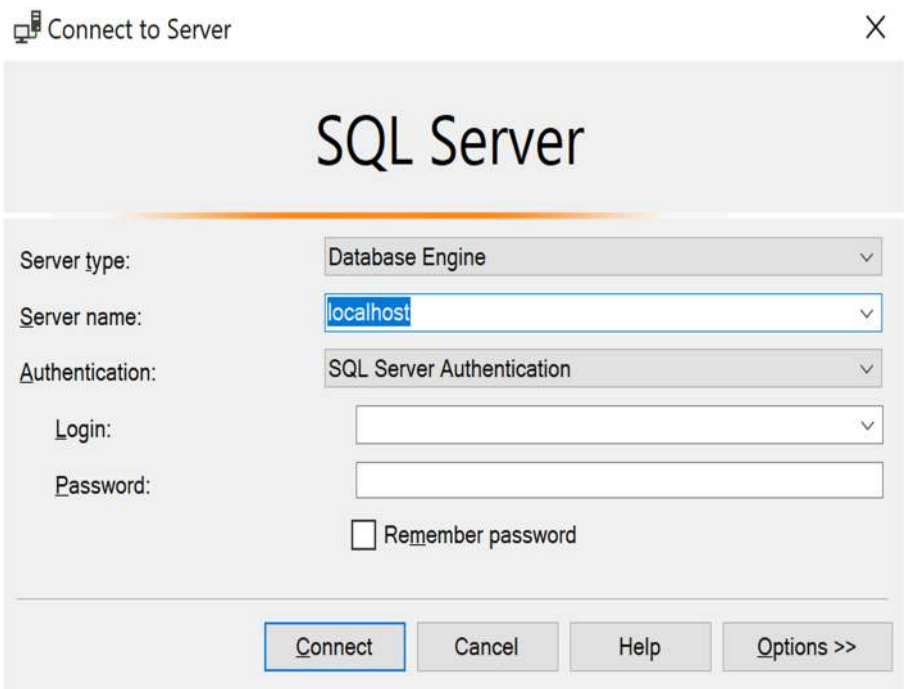
ზოგადად, არსებობს აუთენტიფიცირების სხვადასხვა ვარიანტი, მაგრამ აქედან გამოვყოფთ მხოლოდ ორს:

- Windows აუთენტიკაცია;
- SQL Server აუთენტიკაცია.

მომხმარებლები რომ დაუკავშირდნენ SQL Server-ს, ამისათვის საჭიროა შესვლა (LOGIN). შესვლა შეიძლება იყოს დაფუძნებული დომენის მომხმარებელზე, Windows-ის დომენის ჯგუფზე, ან უშუალოდ SQL Server-ის შესვლაზე.

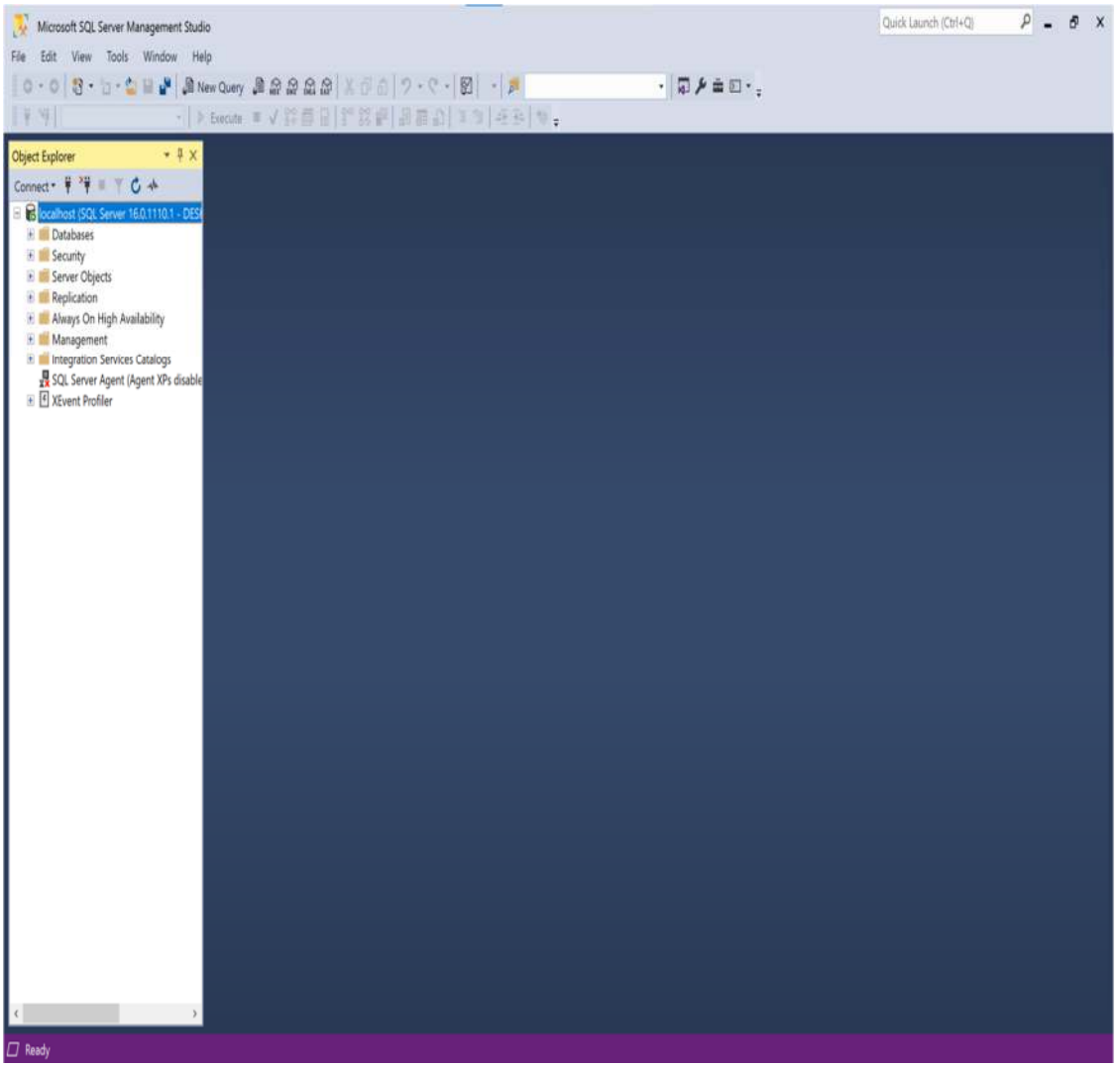
დავუკავშირდეთ მონაცემთა ბაზას. ამისათვის გავხსნათ SSMS 19, გამოვა ფანჯარა, რომლის მსგავსიც მოცემულია სურათზე 2.1. სერვერის სახელის გასწვრივ უნდა მიეთითოს სერვერის მისამართი, რომელიც შეიძლება იყოს კონკრეტული IP მისამართი, რომელიც ეკუთვნის მონაცემთა ბაზის სერვერს, ან ჰოსტის სახელი, რომელიც ორგანიზაციაში მითითებულია მონაცემთა ბაზად. ჩვენ შემთხვევაში, რადგან ბაზა დაყენებულია საკუთარ კომპიუტერზე, მაშინ აღნიშნულ ველში მივუთითოთ localhost. რაც შეეხება აუთენტიფიცირებას, გამოვიყენოთ Windows აუთენტიკაცია, რისთვისაც Authentication ველში SQL Server Authentication-დან გადავრთოთ Windows Authentication-ზე და შემდეგ დავაწკაპუნოთ Connect ღილაკზე.

**სურათი 2.1. სერვერთან დაკავშირება**



თუ ყველაფერი შესრულდა, მაშინ ბაზასთან უკვე კავშირი იქნება შემდგარი. დაკავშირების შემდეგ SSMS 19-ის ინტერფეისი გახდება დაახლოებით ისეთი, როგორიც ნაჩვენებია სურათზე 2.2.

**სურათი 2.2. SSMS 19-ის ინტერფეისი ბაზასთან დაკავშირების შემდეგ**




სურათზე 2.2 SSMS 19-ის მარცხენა ნაწილში ვხედავთ Object Explorer-ის პანელს, სადაც მოცემულია მონაცემთა ბაზის სერვერის ისეთი კომპონენტები, როგორიცაა: მონაცემთა ბაზები, უსაფრთხოება, სერვერის ობიექტები და სხვა. ფანჯრის ზედა ნაწილში მოთავსებულია სხვადასხვა ღილაკი; აქედან ვაწკაპუნებთ New Query ღილაკზე. შეიქმნება ახალი ჩანართი, რომელშიც შეგვეძლება ჩავწეროთ SQL კოდი, ხოლო შემდეგ Execute ღილაკზე მოქმედებით გავუშვათ ის.

თუ გვაქვს ჩართული Mixed Mode აუთენტიკაცია, მაშინ შეგვიძლია შევქმნათ ახალი შესვლა პაროლით. დავუშვათ, გვსურს შესვლის სახელად ავირჩიოთ statuser, ხოლო პაროლად კი stat123. ახლა გავუშვათ შემდეგი ბრძანება შესვლის შესაქმნელად:

```
CREATE LOGIN statuser WITH PASSWORD = 'stat123';
```

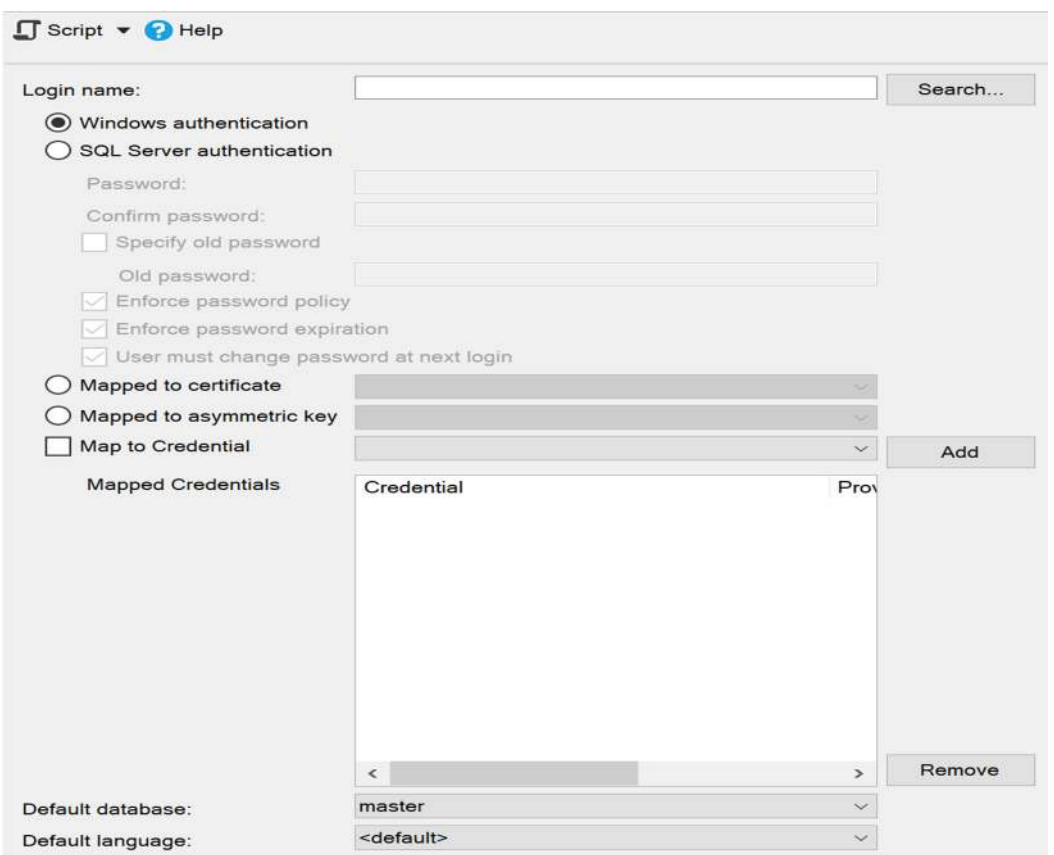
ამის შემდეგ შეგვეძლება დაკავშირება ბაზასთან ახალი შესვლით, რომელიც შევქმენით.

შესვლის შექმნის სხვა ვარიანტებიც არსებობს. თუ დომენის სახელია statistics, ხოლო, შესვლის სახელად ვირჩევთ stat1, მაშინ ასეთი შესვლის შესაქმნელად ვუშვებთ ბრძანებას:

```
CREATE LOGIN [statistics\stat1] FROM WINDOWS;
```

შესვლის შექმნა შესაძლებელია SSMS 19-დან სხვა გზითაც. ამისათვის Object Explorer-ში ვხსნით Security საქაღალდეს, შემდეგ Logins საქაღალდეზე ვაწკაპუნებთ მაუსის მარჯვენა ღილაკს და კონტექსტური მენიუდან ვირჩევთ New Login...-ს. ამის შემდეგ გამოვა ფანჯარა. General გვერდზე, რომელიც მოცემულია სურათზე 2.3, შეგვიძლია ავირჩიოთ Login name ველში შესვლის სახელი, ასევე ავირჩიოთ Windows-ის აუთენტიკაცია, თუ გვსურს SQL Server აუთენთიკაცია. Server Role გვერდზე შეგვიძლია ავირჩიოთ უსაფრთხოების პრივილეგიები, ხოლო OK ღილაკზე მოქმედებით შეიქმნება აღნიშნული შესვლა.

**სურათი 2.3. General გვერდი**



ზემოთ განვიხილეთ შესვლის შექმნის რამდენიმე ვარიანტი, თუმცა, სხვა გზებიც არსებობს. რადგან უკვე შეგვიძლია შევქმნათ შესვლა, ვაჩვენოთ, თუ როგორ შეიძლება მომხმარებლების შექმნა კონკრეტული მონაცემთა ბაზისთვის. ზოგადად, მონაცემთა ბაზის სერვერზე განთავსებულია სხვადასხვა ბაზა. აქედან თითოეული ბაზისთვის შესაძლებელია კონკრეტული მომხმარებლის შექმნა. მომხმარებლები შეიძლება დავყოთ რამდენიმე ჯგუფად, კერძოდ:

1. მომხმარებლები, რომლებიც დაფუძნებულია შესვლაზე master ბაზაში;
2. მომხმარებლები, რომლებიც აუთენტიკაციას ახდენენ ბაზაში;
3. მომხმარებლები, რომლებსაც არ შეუძლიათ აუთენტიკაცია;
4. მომხმარებლები, რომლებსაც შეუძლიათ აუთენტიკაცია Windows-ის ჯგუფებით.

თუ გვსურს შევქმნათ მომხმარებელი, რომლის სახელია statuser, უნდა გავუშვათ ბრძანება:

```
CREATE USER statuser;
```

თუ გვსურს შევქმნათ მომხმარებელი დომენიდან statistics და მისი სახელი იყოს stat1, მაშინ ვუშვებთ ამ ბრძანებას:

```
CREATE USER [statistics\stat1];
```

მომხმარებლის შესაქმნელად სახელით stat2 და პაროლით stat123 გამოიყენება შემდეგი ბრძანება:

```
CREATE USER stat2 WITH PASSWORD = 'stat123';
```

მომხმარებელს კონკრეტულ მონაცემთა ბაზაში შეიძლება ჰქონდეს სხვადასხვა პრივილეგია, რომელსაც განვიხილავთ შემდეგ ქვეთავში.

## 2.2. სქემები მონაცემთა ბაზებში

სქემა (SCHEMA) მოიცავს მონაცემთა ბაზის ობიექტებს. მისი საშუალებით შესაძელებელია კონკრეტულ მომხმარებელს მიენიჭოს პრივილეგია კონკრეტულ სქემაზე.

სქემის შესაქმნელად, რომლის სახელი იქნება stat, უნდა გაეშვას შემდეგი ბრძანება:

```
CREATE SCHEMA stat;
```

SQL Server 2022-ში ნაგულისხმევი სქემა არის dbo. კონკრეტული სქემის მესაკუთრეს აქვს პრივილეგიები სქემის ობიექტებზე. სქემის ობიექტები შეიძლება იყოს ცხრილები, წარმოდგენები, ფუნქციები და ა. შ. ხშირად dbo სქემით ხდება მომხმარებლის წვდომის დონის განსაზღვრა ობიექტებზე.

## 2.3. მონაცემთა ტიპები

როგორც აღვნიშნეთ, SQL Server-ის კონკრეტულ სერვერზე შეიძლება არსებობდეს სხვადასხვა მონაცემთა ბაზა, ხოლო ბაზებში იყოს მოთავსებული სხვადასხვა

ცხრილი. ცხრილი შედგება ერთი ან რამდენიმე სვეტისგან (COLUMN). თითოეული სტრიქონი მოიცავს სხვადასხვა სვეტს. მაგალითად, თუ გვაქვს ცხრილი, რომელიც მოიცავს ეკონომიკურ საქმიანობას შესაბამისი გამოშვებით (მილიონ ლარებში) კონკრეტული წლისათვის და ვიყენებთ სამ სვეტს, მაშინ ეკონომიკური საქმიანობის სვეტში ჩაწერილი მონაცემი იქნება ტექსტური ტიპის. კონკრეტული წლის სვეტი და გამოშვების მონაცემის სვეტი კი იქნება რიცხვითი ტიპის. SQL Server 2022 კიდევ უფრო შორს მიდის და კიდევ უფრო დეტალურად განარჩევს სხვადასხვა მონაცემთა ტიპს. ტექსტური მონაცემთა ტიპები მოცემულია ცხრილში 2.1, ხოლო რიცხვითი და დროითი კი - ცხრილში 2.2.

**ცხრილი 2.1. SQL Server-ის ტექსტური მონაცემთა ტიპები**

| მონაცემთა ტიპი | აღწერა | მაქსიმალური ზომა | მოცულობა |
|---|---|---|---|
| char(n) | ფიქსირებული ზომის სიმბოლოიანი ტექსტი | 8000 სიმბოლო | დაფიქსირებული ზომა |
| varchar(n) | ცვლადი ზომის სიმბოლოიანი ტექსტი | 8000 სიმბოლო | 2 ბაიტი + სიმბოლოების რაოდენობა |
| varchar(max) | ცვლადი ზომის სიმბოლოიანი ტექსტი | 1073741824 სიმბოლო | 2 ბაიტი + სიმბოლოების რაოდენობა |
| text | ცვლადი ზომის სიმბოლოიანი ტექსტი | 2GB ტექსტური მონაცემი | 4 ბაიტი + სიმბოლოების რაოდენობა |
| nchar | ფიქსირებული ზომის უნიკოდიანი ტექსტი | 4000 სიმბოლო | დაფიქსირებული ზომა * 2 |
| nvarchar | ცვლადი ზომის უნიკოდიანი ტექსტი | 4000 სიმბოლო | |
| nvarchar(max) | ცვლადი ზომის უნიკოდიანი ტექსტი | 536870912 სიმბოლო | |
| ntext | ცვლადი ზომის უნიკოდიანი ტექსტი | 2GB ტექსტური მონაცემი | |
| binary(n) | ფიქსირებული ზომის ორობითი ტექსტი | 8000 ბაიტი | |
| varbinary | ცვლადი ზომის ორობითი ტექსტი | 8000 ბაიტი | |
| varbinary(max) | ცვლადი ზომის ორობითი ტექსტი | 2 GB | |
| image | ცვლადი ზომის ორობითი ტექსტი | 2 GB | |

**ცხრილი 2.2. რიცხვითი და დროითი მონაცემთა ტიპები**

| მონაცემთა ტიპი | აღწერა | ზომა |
|---|---|---|
| **bit** | მთელი რიცხვი, 0, 1 ან NULL | |
| **tinyint** | მთელი რიცხვი 0-დან 255-ის ჩათვლით | 1 ბაიტი |
| **smallint** | მთელი რიცხვები -32768-დან 32767-ის ჩათვლით | 2 ბაიტი |
| **int** | მთელი რიცხვი -2147483648-დან 2147483647-ის ჩათვლით | 4 ბაიტი |
| **bigint** | -9223372036854775808-დან 9223372036854775807-ის ჩათვლით | 8 ბაიტი |
| **decimal(p,s)** | ფიქსირებული სიზუსტის რიცხვები $-10^{38}$ +1-დან $10^{38}$ –1-ის ჩათვლით, სადაც p არის მძიმის შემდეგ მარცხნივ და მარჯვნივ არსებული ციფრების რაოდენობა, რომლის ნაგულისხმევი მნიშვნელობაა 18 და შეიძლება შეიცვალოს 1-დან 38-ის ჩათვლით, ხოლო s არის მძიმის მარჯვნივ არსებული ციფრების რაოდენობა, ნაგულისხმევი მნიშვნელობაა 0. | 5-17 ბაიტი |
| **numeric(p,s)** | ფიქსირებული სიზუსტის რიცხვები $-10^{38}$ +1-დან $10^{38}$ –1-ის ჩათვლით, სადაც p არის მძიმის შემდეგ მარცხნივ და მარჯვნივ არსებული ციფრების რაოდენობა, რომლის ნაგულისხმევი მნიშვნელობაა 18 და შეიძლება შეიცვალოს 1-დან 38-ის ჩათვლით, ხოლო s არის მძიმის მარჯვნივ არსებული ციფრების რაოდენობა, ნაგულისხმევი მნიშვნელობაა 0. | 5-17 ბაიტი |
| **smallmoney** | მონეტარული მონაცემები -214,748.3648-დან 214,748.3647-ის ჩათვლით | 4 ბაიტი |
| **money** | მონეტარული მონაცემები -922337203685477.5808-დან 922,337,203,685,477.5807-ის ჩათვლით | 8 ბაიტი |
| **float** | ათწილადები -1.79E+308-დან -2.23E-308-მდე, 0 და 2.23E-308-დან 1.79E+308-მდე | დამოკიდებულია n პარამეტრზე |
| **real** | ათწილადები -3.40E + 38-დან -1.18E – 38-მდე, 0 და 1.18E – 38-დან 3.40E + 38-მდე | 4 ბაიტი |
| **datetime** | 01/01/1753-დან 31/12/9999-მდე | 8 ბაიტი |
| **datetime2** | 01/01/1-დან 31/12/9999-მდე | 6-8 ბაიტი |
| **smalldatetime** | 01/01/1900-დან 06/06/2079-მდე | 4 ბაიტი |
| **date** | ინახავს მხოლოდ თარიღს | 3 ბაიტი |
| **time** | ინახავს მხოლოდ დროს | 3-5 ბაიტი |
| **datetimeoffset** | datetime2 დროითი ზონის ეფექტით | 8-10 ბაიტი |
| **timestamp** | ინახავს უნიკალურ რიცხვს, რომელიც განახლდება ყოველ ჯერზე, როცა მწკრივი იქმნება ან იცვლება. დროის ანაბეჭდის მნიშვნელობა ეფუძნება შიდა საათს და არ შეესაბამება რეალურ დროს. | |

აქვე უნდა აღინიშნოს ისიც, რომ გარდა სვეტებისა, მონაცემთა ტიპები გააჩნიათ ცვლადებსაც და გამოსახულებებსაც. მათმა გაუთვალისწინებლობამ და არასწორმა შერჩევამ შეიძლება გამოიწვიოს სხვადასხვა ხარვეზი და ისეთი გართულება, როგორიცაა დროითი დანაკარგები გამოთვლისას და ცხრილების მოძიებისას.

### *კითხვები თვითშემოწმებისთვის*

1. რა არის შესვლა და როგორ შევქმნათ ის?
2. როგორ შევქმნათ მომხმარებელი მონაცემთა ბაზისთვის?
3. როგორ შევქმნათ სქემა და რაში გამოიყენება ის?
4. ჩამოთვალეთ მონაცემთა ტიპები და დაახასიათეთ ისინი.

# თავი 3. CRUD ოპერაციები

## 3.1. რას ნიშნავს CRUD ოპერაციები

სერვერზე შეიძლება იყოს სხვადასხვა მონაცემთა ბაზა, თითოეულ ბაზაში კი სხვადასხვა ცხრილი, ფუნქცია, პროცედურა და ა. შ. საჭიროა განხორციელდეს ოპერაციები, რომლებიც ქმნის, კითხულობს, განაახლებს, ან შლის კონკრეტულ ობიექტებს. აბრევიატურა CRUD ასე იშიფრება: Create, Read, Update, Delete, სადაც Create ნიშნავს შექმნას, Read - წაკითხვას, Update - განახლებას, ხოლო Delete - წაშლას.

მოცემულ თავში გავეცნობით CRUD ოპერაციებს და ვნახავთ, თუ როგორ გამოვიყენოთ ასეთი ოპერაციები მონაცემებთან მუშაობისას.

## 3.2. მონაცემთა ბაზის შექმნა

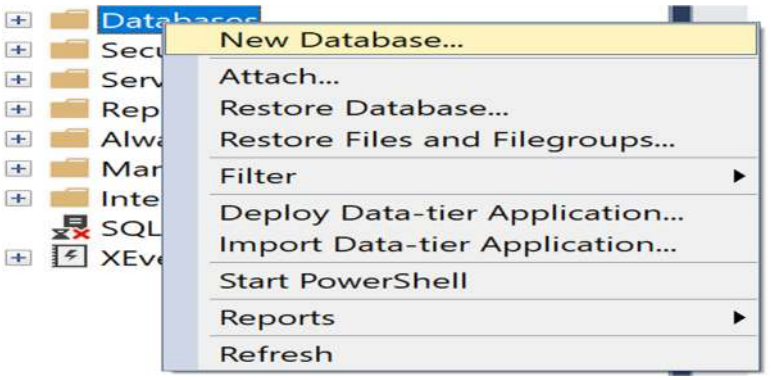
დავიწყოთ ბაზის შექმნის ოპერაციებით. წინა თავში მიმოვიხილეთ, თუ როგორ ხდება შესვლის, სქემისა და მომხმარებლის შექმნა, ამიტომ აქ აღარ გავჩერდებით ამ საკითხზე. პირველად სერვერზე შევქმნათ მონაცემთა ბაზა, რომელსაც შემდეგში გამოვიყენებთ მონაცემების განთავსებისთვის. დავუშვათ, გვსურს მონაცემთა ბაზას დავარქვათ statistika. ამისათვის გავუშვებთ შემდეგ ბრძანებას:

```
CREATE DATABASE statistika;
```

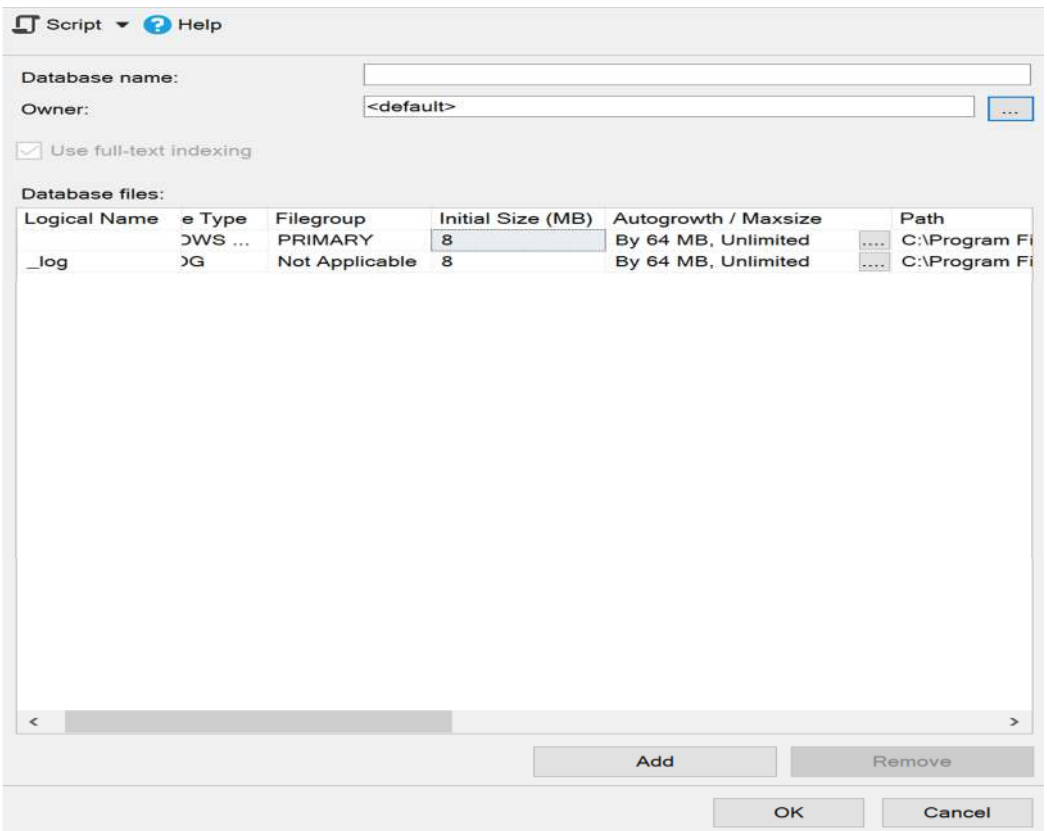
აღნიშნული მოთხოვნის გაშვების შემდეგ შეიქმნება ახალი მონაცემთა ბაზა, სახელად statistika.

რადგან ვიყენებთ SSMS 19-ს, შეგვიძლია კოდის დაწერის გარეშეც შევქმნათ მონაცემთა ბაზა. ამისათვის მივყვეთ ინსტრუქციებს. Object Explorer-ში დავინახავთ Databases საქაღალდეს, რომელზეც დავაწკაპუნებთ მაუსის მარჯვენა ღილაკით. გამოვა კონტექსტური მენიუ, რომელიც გამოსახულია სურათზე 3.1. მოცემული მენიუდან შესაძლებელია სხვადასხვა ოპერაციის განხორციელება. აქედან ვირჩევთ New Database... ღილაკს, რომელზე დაწკაპუნების შემდეგ გამოჩნდება ფანჯარა. ამ ფანჯარას გააჩნია სამი გვერდი. აქედან ვიყენებთ General-ს, რომელიც გამოსახულია სურათზე 3.2. Database name ველში ვწერთ მონაცემთა ბაზის სახელს, მაგალითად, statistika2 და დავაწკაპუნებთ OK ღილაკზე. ამის შემდეგ მონაცემთა ბაზა სახელით statistika2 შეიქმნება სერვერზე. თუ გავხსნით Databases საქაღალდეს მის მარცხნივ არსებული + ღილაკით და კონტექსტური მენიუდან ავირჩევთ Refresh-ს, მაშინ დავინახავთ ჩვენს მონაცემთა ორ ბაზას.

სურათი 3.1. კონტექსტური მენიუ




სურათი 3.2. მონაცემთა ბაზის შექმნა




ჩვენ უკვე შევქმენით მონაცემთა 2 ბაზა, აქედან გამოვიყენოთ statistika. მონაცემთა ბაზაში statistika შევქმნათ ცხრილები.

ცხრილი არის მონაცემთა ბაზის ობიექტი, რომელიც მონაცემებს ინახავს. ცხრილების და პროექტების დროს უნდა გადავწყვიტოთ შემდეგი საკითხები: რა მონაცემები უნდა მოვათავსოთ ცხრილებში? რომელი სვეტებისაგან უნდა შედგებოდეს ცხრილი? რომელი სვეტები შეიძლება შეიცავდეს NULL მნიშვნელობას? იქნება თუ არა სვეტებისათვის გამოყენებული მთლიანობაზე შეზღუდვები ან ავტომატური მნიშვნელობები?

რომელი სვეტების ინდექსირება უნდა მოხდეს? რომელი სვეტები უნდა შევიდეს პირველად და გარე გასაღებში? და ა. შ. [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 54].

ცხრილის შექმნის ორი ვარიანტი განვიხილოთ, ანუ კოდის დაწერით და მის გარეშე. ამის უკეთ გასააზრებლად განვიხილოთ ეს საკითხი ამოცანის მაგალითზე.

დავუშვათ, გვსურს ავაგოთ ცხრილები, რომლებიც მოგვცემენ მიმდინარე ფასებში გამოსახულ საქართველოს მთლიანი შიდა პროდუქტის (მშპ) შესახებ ინფორმაციას ეკონომიკური საქმიანობის სახეების და წლების მიხედვით. აღნიშნული ინფორმაცია შესაძლებელია მოვიძიოთ საქსტატის ვებგვერდზე:
https://www.geostat.ge/ka/modules/categories/23/mtliani-shida-produkti-mshp.

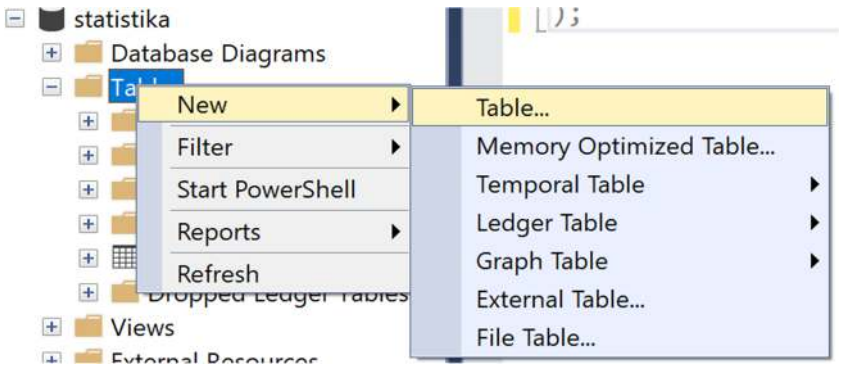
დაგვჭირდება ორი ცხრილი: პირველში მოთავსებული იქნება კლასიფიკატორი NACE Rev.2, სექციის და სექციის დასახელების სვეტით, ხოლო მეორე ცხრილში მოთავსებული იქნება წელი, სექცია და მშპ. ჯერ შევმნათ ეს ორი ცხრილი.

კლასიფიკატორის ცხრილის შესაქმნელად ავირჩიოთ მონაცემთა ტიპები. კლასიფიკატორში სექცია იყენებს ლათინური ანბანის ერთ ასოს; ავირჩოთ nchar(1) მონაცემთა ტიპი, ხოლო სვეტის სახელს დავარქვათ seqcia. ამასთან ვიცით, რომ მოცემულ სვეტში არ უნდა გამეორდეს სექციის კოდი. ამიტომ ამ სვეტს უნდა მიენიჭოს მთავარი გასაღები (PRIMARY KEY). რაც შეეხება მეორე სვეტს, რომელიც მოიცავს სექციის დასახელებას, რადგან იყენებს ქართულ ასოებს, გამოვიყენებთ nvarchar(200) მონაცემთა ტიპს, სადაც 200 აღნიშნავს მაქსიმალური სიმბოლოების რაოდენობას. ამ სვეტს დავარქვათ seqcia_dasaxeleba. ცხრილს დავარქვათ nace2. რადგან უკვე ვიცით ცხრილის სტრუქტურა, მის შესაქმნელად გავუშვათ მოთხოვნა, ამასთან დავრწმუნდეთ, რომ statistika მონაცემთა ბაზაში ვქმნით ცხრილს:

```
USE statistika; GO
CREATE TABLE nace2 (seqcia NCHAR (1) PRIMARY KEY, seqcia_dasaxeleba NVARCHAR (200));
```

კლასიფიკატორის ცხრილი უკვე შეიქმნა; ახლა კი შევქმნათ მეორე ცხრილი, რომელშიც იქნება მოთავსებული მონაცემები. ცხრილის შესაქმნელად გამოვიყენოთ გრაფიკული ვარიანტი SSMS 19-დან კოდის შეყვანის გარეშე. დაგვჭირდება სამი სვეტი: ის სვეტი, რომელიც მოიცავს წელს და მის მონაცემთა ტიპად უნდა აირჩეს smallint ან int მონაცემთა ტიპი და პირობითად ამ სვეტს ვუწოდოთ weli. მეორე სვეტში უნდა მოთავსდეს სექცია, რომელიც მოიცავს ერთ ლათინურ ასოს. მონაცემთა ტიპად ავირჩიოთ nchar(1) და ვუწოდოთ მას seqcia. სწორედ ამ სვეტით დავაკავშირებთ ამ ცხრილს კლასიფიკატორის ცხრილთან. მესამე სვეტში, სადაც მოთავსებული იქნება მშპ მიმდინარე ფასებში (მილიონ ლარებში), ვუწოდოთ მას mshp და მის მონაცემთა ტიპად ავირჩიოთ float. ცხრილს ვუწოდოთ mshp_seqcia. მის შესაქმნელად გავხსნათ მონაცემთა ბაზა + ღილაკით, Table საქაღალდეზე დავაწკაპუნოთ მაუსის მარჯვენა ღილაკით და ავირჩიოთ New>Table..., როგორც ნაჩვენებია სურათ 3.3-ზე.

სურათი 3.3. ცხრილის შექმნა




ამის შემდეგ გამოვა ახალი ჩანართი, რომლის შესაქმნელი ცხრილის შესაბამისად შევსებული ვარიანტი ნაჩვენებია სურათზე 3.4. Column Name სვეტში მოთავსებულია სვეტების დასახელებები, Data Type-ში მონაცემთა ტიპები, ხოლო Allow Nulls-ს სვეტში მოვნიშნავთ, შეიძლება თუ არა სვეტები შეიცავდნენ ცარიელ ჩანაწერს. SSMS 19-ის ფანჯრის მარჯვენა ნაწილში მოთავსებულია Properties პანელი, სადაც შეგვიძლია ავირჩიოთ ცხრილის სახელი, სქემა და ა. შ. სახელის ასარჩევად (Name) ველში ვწერთ mshp_seqcia-ს, ამის შემდეგ ვინახავთ Ctrl+S ღილაკების მოქმედებით.

სურათი 3.4. ცხრილის შექმნა

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| weli | smallint | ☑ |
| seqcia | nchar(1) | ☑ |
| mshp | float | ☑ |

ცხრილები უკვე გვაქვს. ახლა შევიტანოთ მათში შესაბამისი მონაცემები. ჯერ მოვამზადოთ კლასიფიკატორის ცხრილი. კლასიფიკატორი მოცემულია 3.1 ცხრილში.

ცხრილი 3.1. NACE Rev.2 კლასიფიკატორი

| NACE Rev. 2 | ეკონომიკური საქმიანობის სახეები |
|---|---|
| A | სოფლის, სატყეო და თევზის მეურნეობა |
| B | სამთომოპოვებითი მრეწველობა |
| C | დამამუშავებელი მრეწველობა |
| D | ელექტროენერგიის, აირის, ორთქლის დაკონდიცირებული ჰაერის მიწოდება |
| E | წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბინძურებისაგან გასუფთავების საქმიანობები |
| F | მშენებლობა |

| | |
|---|---|
| G | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოციკლების რემონტი |
| H | ტრანსპორტი და დასაწყობება |
| I | განთავსების საშუალებებით უზრუნველყოფის და საკვების მიწოდების საქმიანობები |
| J | ინფორმაცია და კომუნიკაცია |
| K | საფინანსო და სადაზღვევო საქმიანობები |
| L | უძრავ ქონებასთან დაკავშირებული საქმიანობები |
| M | პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები |
| N | ადმინისტრაციული და დამხმარე მომსახურების საქმიანობები |
| O | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო სოციალური უსაფრთხოება |
| P | განათლება |
| Q | ჯანდაცვა და სოციალური მომსახურების საქმიანობები |
| R | ხელოვნება, გართობა და დასვენება |
| S | სხვა სახის მომსახურება |
| T | შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები; არადიფერენცირებული საქონლის და მომსახურების წარმოება შინამეურნეობების მიერ საკუთარი მოხმარებისათვის |

***წყარო:*** *საქსტატის ეროვნული სამსახური*

რაც შეეხება მშპ-ის შესახებ მონაცემებს საბაზისო ფასებში, 2022 წლის მონაცემები მოთავსებულია ცხრილში 3.2.

**ცხრილი 3.2. მშპ საბაზისო ფასებში, 2022 წელი**

| წელი | სექცია | მშპ |
|---|---|---|
| 2022 | A | 4,313.2 |
| 2022 | B | 890.7 |
| 2022 | C | 7,078.2 |
| 2022 | D | 2,010.6 |
| 2022 | E | 428.0 |
| 2022 | F | 5,026.4 |
| 2022 | G | 9,990.8 |
| 2022 | H | 4,044.1 |
| 2022 | I | 2,329.2 |
| 2022 | J | 3,131.9 |
| 2022 | K | 2,961.4 |
| 2022 | L | 6,309.0 |
| 2022 | M | 1,286.9 |
| 2022 | N | 667.0 |

| | | |
|---|---|---|
| **2022** | O | 4,031.9 |
| **2022** | P | 2,807.9 |
| **2022** | Q | 2,321.1 |
| **2022** | R | 2,527.7 |
| **2022** | S | 574.7 |
| **2022** | T | 71.4 |

***წყარო:*** *საქსთვეღოს სტატისტიკის ეროვნული სამსახური*

ჯერ შევავსოთ კლასიფიკატორის ცხრილი. ამისათვის გამოვიყენოთ INSERT ბრძანება. მოთხოვნა გამოიყურება შემდეგნაირად:

```
INSERT INTO statistika.dbo.nace2
VALUES
(N'A'    ,          N' სოფლის, სატყეო და თევზის მეურნეობა'),
(N'B'    ,          N' სამთომოპოვებითი მრეწველობა'),
(N'C'    ,          N' დამამუშავებელი მრეწველობა'),
(N'D'    ,          N' ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰაერის მიწოდება'),
(N'E'    ,          N' წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბინძურებისაგან გასუფთავების საქმიანობები'),
(N'F'    ,          N' მშენებლობა'),
(N'G'    ,          N' საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოციკლების რემონტი'),
(N'H'    ,          N' ტრანსპორტი და დასაწყობება'),
(N'I'    ,          N' განთავსების საშუალებებით უზრუნველყოფის და საკვების მიწოდების საქმიანობები '),
(N'J'    ,          N' ინფორმაცია და კომუნიკაცია'),
(N'K'    ,          N' საფინანსო და სადაზღვევო საქმიანობები'),
(N'L'    ,          N' უძრავ ქონებასთან დაკავშირებული საქმიანობები'),
(N'M'    ,          N' პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები'),
(N'N'    ,          N' ადმინისტრაციული და დამხმარე მომსახურების საქმიანობები '),
(N'O'    ,          N' სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო სოციალური უსაფრთხოება'),
(N'P'    ,          N' განათლება'),
(N'Q'    ,          N' ჯანდაცვა და სოციალური მომსახურების საქმიანობები'),
(N'R'    ,          N' ხელოვნება, გართობა და დასვენება'),
(N'S'    ,          N' სხვა სახის მომსახურება'),
(N'T'    ,          N' შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები; არა-დიფერენცირებული საქონლის და მომსახურების წარმოება შინამეურნეობების მიერ საკუთარი მოხმარებისათვის');
```

მოცემულ ბრძანებაში მივუთითეთ მონაცემთა ბაზაც და სქემაც, თუმცა, შეგვიძლია მონაცემთა ბაზის დასახელება და სქემა გამოვტოვოთ, თუ ვიყენებთ შესაბამის მონაცემთა ბაზას და ნაგულისხმევ სქემას. შეგვიძლია, ასევე, მივუთითოთ კონკრეტული სვეტები. ამისათვის ზემოთ მოცემულ მოთხოვნაში შევცვლით ამ კოდს

```
INSERT INTO statistika.dbo.nace2
VALUES
```

ამ კოდით

```
INSERT INTO nace2(seqcia, seqcia_dasaxeleba)
VALUES
```

ახლა შევავსოთ mshp_seqcia ცხრილი 2022 წლის მონაცემებით, რისთვისაც გავუშვათ შემდეგი მოთხოვნა:

```
INSERT INTO [statistika].[dbo].[mshp_seqcia]
VALUES
(    2022    ,    N'A'    ,    4313.153279    ),
(    2022    ,    N'B'    ,    890.680132    ),
(    2022    ,    N'C'    ,    7078.223959    ),
(    2022    ,    N'D'    ,    2010.626158    ),
(    2022    ,    N'E'    ,    427.997953    ),
(    2022    ,    N'F'    ,    5026.424601    ),
(    2022    ,    N'G'    ,    9990.823914    ),
(    2022    ,    N'H'    ,    4044.070652    ),
(    2022    ,    N'I'    ,    2329.248793    ),
(    2022    ,    N'J'    ,    3131.881384    ),
(    2022    ,    N'K'    ,    2961.412246    ),
(    2022    ,    N'L'    ,    6309.015329    ),
(    2022    ,    N'M'    ,    1286.905117    ),
(    2022    ,    N'N'    ,    667.035041    ),
(    2022    ,    N'O'    ,    4031.916073    ),
(    2022    ,    N'P'    ,    2807.913642    ),
(    2022    ,    N'Q'    ,    2321.127977    ),
(    2022    ,    N'R'    ,    2527.685929    ),
(    2022    ,    N'S'    ,    574.733044    ),
(    2022    ,    N'T'    ,    71.360067    );
```

ამჯერად ის ცხრილები, რომლებიც შევქმენით, უკვე შევსებულია მონაცემებით.

## 3.3. წაკითხვის ოპერაციები

რადგან უკვე გვაქვს მონაცემთა ბაზა და მასში ცხრილები, ახლა განვიხილოთ, თუ როგორ წავიკითხოთ მონაცემები. წასაკითხად ვიყენებთ SELECT მოთხოვნას. მისი სინტაქსი შემდეგნაირია:

```
SELECT * FROM database.schema.table;
```

ამ ბრძანებაში* აღნიშნავს ყველა სვეტს database.schema.table ცხრილიდან, სადაც database.schema არის მონაცემთა ბაზა და შესაბამისი სქემა, ხოლო table არის ცხრილის დასახელება. ჩვენ შეგვიძლია ყველა სვეტის მაგივრად მხოლოდ ის სვეტები წავიკითხოთ, რომლებიც გვჭირდება. ამისთვის *-ის ნაცვლად დავწერთ სვეტების სახელებს და გამოვყოფთ მძიმეებით, მაგალითად: sveti1, sveti2. შეგვიძლია ბრძანებაში არ მივუთითოთ მონაცემთა ბაზა და სქემა, თუ კმაყოფილდება ანალოგიური პირობა, რომელიც წინა შექმნის ქვეთავში მონაცემთა ჩაწერისას გამოვიყენეთ.

დავუშვათ, გვსურს წავიკითხოთ ყველა მონაცემი mshp_seqcia ცხრილიდან, რომელიც შევქმენით. ამისათვის გამოვიყენოთ შემდეგი ბრძანება:

```
SELECT *
FROM
mshp_seqcia;
```

შედეგს მივიღებთ SSMS 19-ის ფანჯრის ქვედა ნაწილში არსებულ Results ჩანართში, რომელიც გამოსახულია ქვემოთ სურათზე 3.5:

**სურათი 3.5. შედეგები**

Results Messages

| | weli | seqcia | mshp |
|---|---|---|---|
| 1 | 2022 | A | 4313.153279 |
| 2 | 2022 | B | 890.680132 |
| 3 | 2022 | C | 7078.223959 |
| 4 | 2022 | D | 2010.626158 |
| 5 | 2022 | E | 427.997953 |
| 6 | 2022 | F | 5026.424601 |
| 7 | 2022 | G | 9990.823914 |
| 8 | 2022 | H | 4044.070652 |
| 9 | 2022 | I | 2329.248793 |
| 10 | 2022 | J | 3131.881384 |
| 11 | 2022 | K | 2961.412246 |
| 12 | 2022 | L | 6309.015329 |
| 13 | 2022 | M | 1286.905117 |
| 14 | 2022 | N | 667.035041 |
| 15 | 2022 | O | 4031.916073 |
| 16 | 2022 | P | 2807.913642 |
| 17 | 2022 | Q | 2321.127977 |
| 18 | 2022 | R | 2527.685929 |
| 19 | 2022 | S | 574.733044 |
| 20 | 2022 | T | 71.360067 |

როგორც ვხედავთ, მხოლოდ 2022 წლის მონაცემებია შეტანილი. დავუშვათ, გვსურს ამოვიღოთ მხოლოდ სექციისა და მშპ-ის სვეტები. ამისათვის მივუთითებთ მხოლოდ ორ სვეტს - seqcia და mshp და გამოვიყენებთ ასეთ მოთხოვნას:

```
SELECT seqcia, mshp
FROM mshp_seqcia;
```

აღნიშნული მოთხოვნა მოგვცემს შედეგს, რომელიც გამოსახულია სურათზე 3.6:

**სურათი 3.6. სექცია და მშპ**

Results Messages

| | seqcia | mshp |
|---|---|---|
| 1 | A | 4313.153279 |
| 2 | B | 890.680132 |
| 3 | C | 7078.223959 |
| 4 | D | 2010.626158 |
| 5 | E | 427.997953 |
| 6 | F | 5026.424601 |
| 7 | G | 9990.823914 |
| 8 | H | 4044.070652 |
| 9 | I | 2329.248793 |
| 10 | J | 3131.881384 |
| 11 | K | 2961.412246 |
| 12 | L | 6309.015329 |
| 13 | M | 1286.905117 |
| 14 | N | 667.035041 |
| 15 | O | 4031.916073 |
| 16 | P | 2807.913642 |
| 17 | Q | 2321.127977 |
| 18 | R | 2527.685929 |
| 19 | S | 574.733044 |
| 20 | T | 71.360067 |

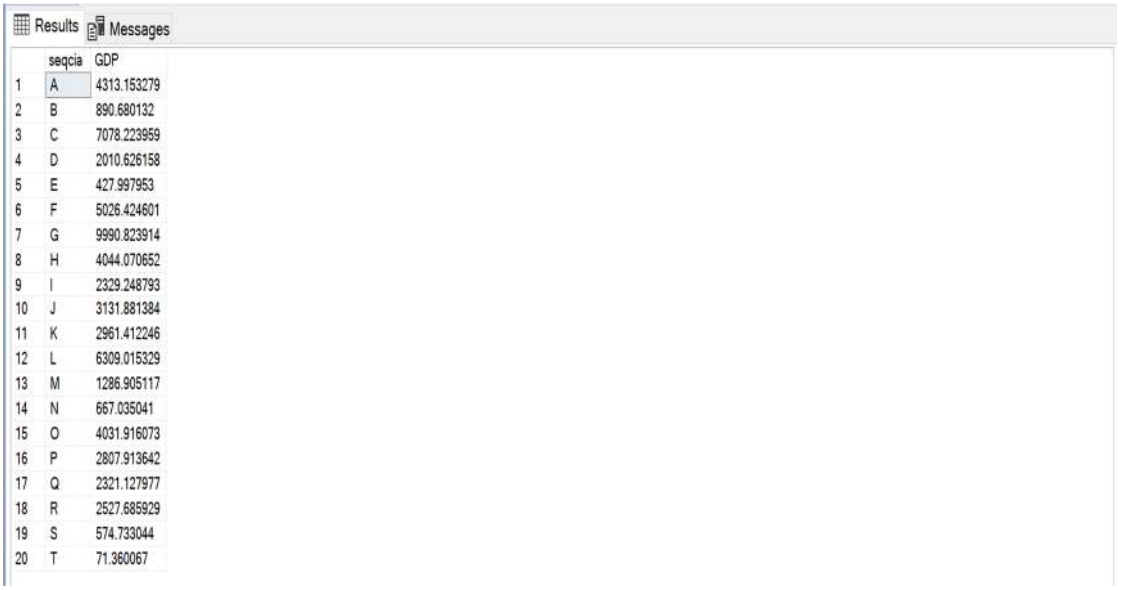
შეგვიძლია, ასევე, შედეგში შევცვალოთ სვეტის სახელები AS ბრძანებით.

მაგალითად, ამოვიღოთ ცხრილიდან სექციის სვეტი და მშპ-ის სვეტი, ხოლო mshp-ის მაგივრად გვსურს შედეგებში სვეტის დასახელებად დაეწეროს GDP. ამისათვის გამოვიყენებთ შემდეგ ბრძანებას:

```
SELECT seqcia, mshp AS GDP
FROM
mshp_seqcia;
```

შესაბამისი შედეგი მოცემულია სურათზე 3.7. აქვე უნდა აღინიშნოს ისიც, რომ არსებულ ბრძანებაში შეგვძლია „mshp AS GDP“-დან ამოვიღოთ AS და დავტოვოთ ასე: „mshp GDP“. შედეგი იგივე იქნება.

სურათი 3.7. AS ბრძანების გამოყენება



Results Messages

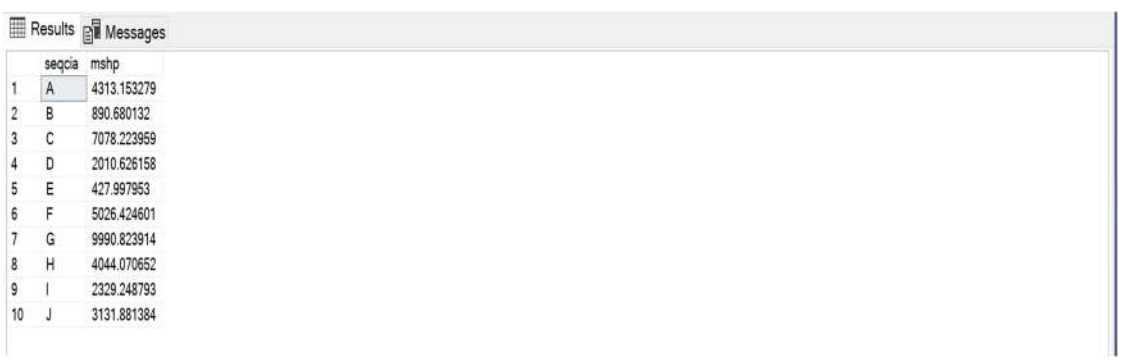
| | seqcia | GDP |
|---|---|---|
| 1 | A | 4313.153279 |
| 2 | B | 890.680132 |
| 3 | C | 7078.223959 |
| 4 | D | 2010.626158 |
| 5 | E | 427.997953 |
| 6 | F | 5026.424601 |
| 7 | G | 9990.823914 |
| 8 | H | 4044.070652 |
| 9 | I | 2329.248793 |
| 10 | J | 3131.881384 |
| 11 | K | 2961.412246 |
| 12 | L | 6309.015329 |
| 13 | M | 1286.905117 |
| 14 | N | 667.035041 |
| 15 | O | 4031.916073 |
| 16 | P | 2807.913642 |
| 17 | Q | 2321.127977 |
| 18 | R | 2527.685929 |
| 19 | S | 574.733044 |
| 20 | T | 71.360067 |

ასევე, შეგვიძლია მხოლოდ პირველი რამდენიმე მონაცემი ამოვიღოთ, მაგალითად, პირველი 10. ამისათვის გამოვიყენებთ TOP ბრძანებას. ამოვიღოთ mshp_seqcia ცხრილიდან პირველი 10 მონაცემი, მხოლოდ სექციისა და მშპ-ის სვეტებით. ამისათვის გამოვიყენოთ შემდეგი მოთხოვნა:

```
SELECT TOP 10
seqcia, mshp
FROM
mshp_seqcia;
```

შედეგი მოცემულია სურათზე 3.8.

სურათი 3.8. mshp_seqcia ცხრილიდან პირველი 10 მონაცემი



Results Messages

| | seqcia | mshp |
|---|---|---|
| 1 | A | 4313.153279 |
| 2 | B | 890.680132 |
| 3 | C | 7078.223959 |
| 4 | D | 2010.626158 |
| 5 | E | 427.997953 |
| 6 | F | 5026.424601 |
| 7 | G | 9990.823914 |
| 8 | H | 4044.070652 |
| 9 | I | 2329.248793 |
| 10 | J | 3131.881384 |

თუ მოცემული ცხრილიდან მხოლოდ წელს ამოვიღებთ ბრძანებით „SELECT weli FROM mshp_seqcia“, ვნახავთ, რომ მივიღეთ ერთი სვეტი და ოცი ჩანაწერი, სადაც ყვე-

ლა მათგანი აჩვენებს „2022“-ს. შეიძლება დაგვჭირდეს ბრძანება, რომელიც მხოლოდ უნიკალურ სტრიქონებს გვაჩვენებს. ამისათვის გამოიყენება ბრძანება DISTINCT. მის გამოსაყენებლად SELECT მოთხოვნაში ვუთითებთ DISTINCT ბრძანებას, რისთვისაც გავუშვათ ეს ბრძანება:

```
SELECT DISTINCT weli
FROM
mshp_seqcia;
```

ზემოხსენებული მოთხოვნის გაშვების შემდეგ დაბრუნდა მხოლოდ ერთი სტრიქონი და ის მითითებული იქნება წელის სვეტში მნიშვნელობით 2022.

წაკითხვისას, ასევე, შესაძლებელია კონკრეტული მონაცემი შეიცვალოს სხვა მონაცემით, მაგალითად, გვსურს წავიკითხოთ seqcia და mshp სვეტები, ხოლო სექციის სვეტში, თუ იქნება სექცია „A“, მაშინ შეიცვალოს ის სტრიქონებში „A1“-ად. წინააღმდეგ შემთხვევაში, გამოვიყენოთ ისევ სექციის სვეტში არსებული მონაცემი. ამისათვის გამოვიყენებთ CASE ოპერაციას:

```
SELECT
CASE
        WHEN seqcia=N'A' THEN N'A1'
        ELSE seqcia
END seqcia_axali,
mshp
FROM
mshp_seqcia;
```

შედეგი მოცემულია ქვემოთ სურათზე 3.9:

**სურათი 3.9. CASE-ის გამოყენება**

Results Messages

| | seqcia_axali | mshp |
|---|---|---|
| 1 | A1 | 4313.153279 |
| 2 | B | 890.680132 |
| 3 | C | 7078.223959 |
| 4 | D | 2010.626158 |
| 5 | E | 427.997953 |
| 6 | F | 5026.424601 |
| 7 | G | 9990.823914 |
| 8 | H | 4044.070652 |
| 9 | I | 2329.248793 |
| 10 | J | 3131.881384 |
| 11 | K | 2961.412246 |
| 12 | L | 6309.015329 |
| 13 | M | 1286.905117 |
| 14 | N | 667.035041 |
| 15 | O | 4031.916073 |
| 16 | P | 2807.913642 |
| 17 | Q | 2321.127977 |
| 18 | R | 2527.685929 |
| 19 | S | 574.733044 |
| 20 | T | 71.360067 |

აღნიშნულ ბრძანებაში CASE...END განაცხადით შეიქმნა seqcia_axali სვეტი შედეგებში, სადაც მივუთითეთ WHEN...THEN ბრძანებით, რომ თუ სექციის სვეტში, რომლის სახელია seqcia, არის ჩანაწერი, რომლის მნიშვნელობაა A, შეიცვალოს A1-ით, ხოლო წინააღმდეგ შემთხვევაში ELSE პირობით გამოვიყენოთ ისევ seqcia სვეტში არსებული ჩანაწერები.

ვკითხულობდით ყველა ჩანაწერს, თუმცა, SQL-ით შესაძლებელია კონკრეტული პირობების საფუძველზე მონაცემთა ფილტრაცია. ამისათვის გამოიყენება WHERE პირობა. მაგალითად, გვსურს ისეთი სექციების და შესაბამისი მშპ-ის მონაცემის ამოღება, რომელთა მშპ საბაზისო ფასებში 4 მლრდ. ლარზე მეტია (გაითვალისწინეთ, რომ მშპ ცხრილში მოცემულია მილიონებში). ამისათვის გამოვიყენოთ WHERE პირობა: WHERE mshp>4000. ეს პირობა მივუთითოთ SELECT მოთხოვნაში

შემდეგნაირად და გავუშვათ ბრძანება:

```
SELECT
seqcia, mshp
FROM
mshp_seqcia
WHERE mshp>4000;
```

შედეგი გამოსახულია სურათზე 3.10. 2022 წლისათვის ასეთი შვიდი სექცია მივიღეთ.

**სურათი 3.10. ფილტრაციის შედეგები**


| | seqcia | mshp |
|---|---|---|
| 1 | A | 4313.153279 |
| 2 | C | 7078.223959 |
| 3 | F | 5026.424601 |
| 4 | G | 9990.823914 |
| 5 | H | 4044.070652 |
| 6 | L | 6309.015329 |
| 7 | O | 4031.916073 |

გარდა მეტობისა და ნაკლებობისა, შესაძლებელია სხვა ოპერატორების გამოყენება, რომლებიც მოთავსებულია ცხრილში 3.3.

**ცხრილი 3.3. შედარების ოპერატორები**

| ოპერატორი | მნიშვნელობა |
|---|---|
| = | უდრის |
| != | არ უდრის |
| <> | არ უდრის |
| > | მეტია |
| >= | მეტია ან ტოლია |
| < | ნაკლებია |
| <= | ნაკლებია ან ტოლია |

აქვე უნდა აღინიშნოს ისიც, რომ მონაცემთა ჩასმა შესაძლებელია INSERT-ის მოქმედებით SELECT ბრძანებასთან ერთად. მანამდე ვიყენებდით VALUES პირობას მო-

ნაცემების ჩასასმელად, თუმცა, შეგვიძლია ერთი ცხრილის შედეგები გამოვიყენოთ სხვა, ან იმავე ცხრილში მონაცემების შესატანად. ამისათვის ჯერ ვწერთ INSERT ბრძანებას, ხოლო შემდეგ VALUES და შესაბამისი მნიშვნელობების მაგივრად ვწერთ კონკრეტულ SELECT მოთხოვნას და ვუშვებთ კოდს. მაგალითად, 2022 წლის მაგივრად 2025 წლის მონაცემები შევქმნათ, დავუშვათ რომ 2022 წლისა და 2025 წლის მონაცემები ტოლია. ამისათვის ვიყენებთ შემდეგ SELECT ბრძანებას:

```
SELECT
2025 weli, seqcia, mshp
FROM
mshp_seqcia;
```

როგორც ვხედავთ, აღნიშნულ მოთხოვნაში წლის მაგივრად შევქმენით ახალი სვეტი, რომელსაც ვუწოდეთ weli, ხოლო მნიშვნელობად ავირჩიეთ 2025. თუ გვსურს ეს შედეგი ჩავწეროთ იმავე, mshp_seqcia ცხრილში, ვწერთ შემდეგ ბრძანებას:

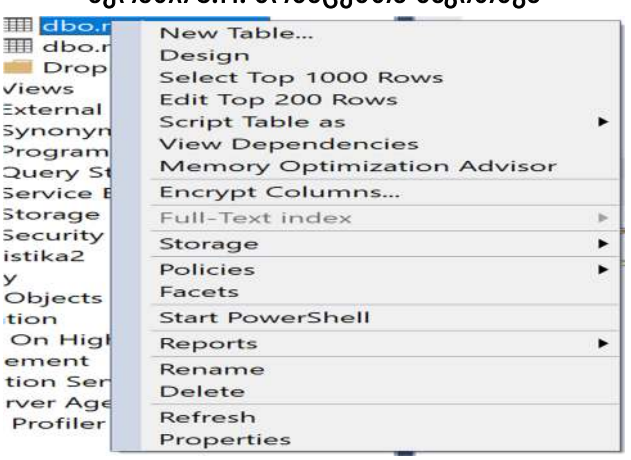
```
INSERT INTO mshp_seqcia(weli, seqcia, mshp)
SELECT
2025 weli, seqcia, mshp
FROM
mshp_seqcia;
```

აღნიშნული კოდის გაშვების შემდეგ, 2025 წლის სინთეზური მონაცემები ჩაიწერება mshp_seqcia ცხრილში.

WHERE ბრძანების მიერ დადებული ფილტრები შესაძლებელია გამოვიყენოთ როგორც შექმნისა და წაკითხვისას, ისე განახლებისა და წაშლის ოპერაციების განხორციელებისას.

მონაცემთა წაკითხვის დროს შეგვიძლია გამოვიყენოთ SSMS 19-ის გრაფიკული ინტერფეისი. ამისათვის კონკრეტულ ცხრილზე ვაწკაპუნებთ მაუსის მარჯვენა ღილაკს Object Explorer-ში, გამოვა კონტექსტური მენიუ, სადაც ავირჩევთ Select Top 1000 Rows ღილაკს, რომლის დაწკაპუნების შემდეგ გამოჩნდება ახალი ჩანართი შესაბამისი კოდითა და შედეგებით.

**სურათი 3.11. მონაცემთა წაკითხვა**



## 3.4. განახლების ოპერაციები

SQL სერვერზე შესაძლებელია მონაცემთა ბაზის ობიექტების განახლებაც. დავუშვათ გვსურს mshp_seqcia ცხრილში 2025 წელი შევცვალოთ 2024 წლით. ამისათვის გამოვიყენებთ UPDATE მოთხოვნას WHERE პირობასთან ერთად. წინააღმდეგ შემთხვევაში შეიცვლება მთელი ცხრილი. მისი სინტაქსი შემდეგნაირია:

```
UPDATE dabatabase.schema.table
SET sveti=1
WHERE sveti=2;
```

ამ ბრძანებაშიც ანალოგიურად შეგვიძლია გამოვტოვოთ მონაცემთა ბაზისა და სქემის დასახელება, თუ სრულდება შესაბამისი პირობები. ჩვენი მაგალითის შემთხვევაში გავუშვათ შემდეგი ბრძანება:

```
UPDATE [mshp_seqcia]
SET weli=2024
WHERE weli=2025;
```

აღნიშნულ ბრძანებაში „SET weli=2024“ ნიშნავს, რომ წლის სვეტში არსებული მნიშვნელობები უნდა შეიცვალოს 2024-ით, ხოლო „WHERE weli=2025“ პირობა კი ისეთ ჩანაწერებს იღებს, რომელთა წლის სვეტის მონაცემია 2025.

გარკვეულ შემთხვევებში შეგვიძლია, ასევე, შევცვალოთ ცხრილის სტრუქტურაც. SSMS 19-ით შეგვიძლია კონკრეტული ცხრილის სტრუქტურა გრაფიკული ინტერფეისის საშუალებით შევცვალოთ. ამისათვის სურათზე 3.11 გამოსახულ კონტექსტურ მენიუს თუ გამოვიძახებთ და ავირჩევთ Design ღილაკს, მაშინ გაიხსნება ჩანართი, რომელიც ცხრილის შექმნის დროს გახსნილი ჩანართის მსგავსია. თუ ინტერფეისზე განვახორციელებთ ცვლილებებს და გვსურს ახალი სტრუქტურის შენახვა, მაშინ ვიმოქმედებთ Ctrl+S ღილაკებით.

რაც შეეხება ბრძანებებს, სვეტების მონაცემთა ტიპის შესაცვლელად ვიყენებთ ასეთ ბრძანებას

```
ALTER TABLE cxrilis_saxeli
ALTER COLUMN svetis_saxelimonacemta_tipi;
```

სადაც cxrilis_saxeli არის ცხრილის სახელი ბაზაში, svetis_saxeli არის იმ სვეტის სახელი, რომლის მონაცემთა ტიპს ვცვლით, ხოლო monacemta_tipi არის ის მონაცემთა ტიპი, რომელიც გვსურს იყოს სვეტის ახალი მონაცემთა ტიპი (მაგალითად, INT, FLOAT და ა. შ.).

სვეტის სახელის შესაცვლელად გამოიყენება შემდეგი ბრძანება:

```
EXEC sp_RENAME 'cxrilis_saxeli.svetis_saxeli' , 'svetis_axali_saxeli', 'COLUMN';
```

სადაც cxrilis_saxeli არის იმ ცხრილის სახელი, რომლის რედაქტირებაც გვინდა, svetis_saxeli - იმ სვეტის სახელი, რომლის დასახელების შეცვლაც გვინდა, ხოლო svetis_axali_saxeli - ის დასახელება, რომელიც გვსურს დავარქვათ სვეტს.

ცხრილში სვეტის წასაშლელად ვიყენებთ ბრძანებას, სადაც ანალოგიურად cxrilis_saxeli და svetis_saxeli არის, შესაბამისად, ცხრილისა და სვეტის დასახელება:

```
ALTER TABLE cxrilis_saxeli DROP COLUMN svetis_saxeli;
```

## 3.5. მონაცემთა წაშლა

ახლა განვიხილოთ წაშლის ოპერაციები. თუ გვსურს ცხრილიდან წავშალოთ მონაცემები, მაშინ გამოიყენება DELETE მოთხოვნა. აღნიშნული მოთხოვნა შეგვიძლია გამოვიყენოთ WHERE პირობასთან ერთად, თუ გვსურს წაიშალოს ის მონაცემები, რომლებიც აკმაყოფილებენ რაიმე პირობას. დავუშვათ, გვსურს წავშალოთ mshp_seqcia ცხრილიდან ის მონაცემები, რომლებიც არის 2024 წლისათვის (წინა ქვეთავში აღნიშნული მონაცემები იყო 2025 წლის სინთეზური მონაცემების განახლება 2024 წლის მონაცემებად). ამისათვის გამოვიყენებთ შემდეგ ბრძანებას:

```
DELETE FROM [mshp_seqcia]
WHERE weli=2024;
```

გარდა მონაცემებისა, შეგვიძლია წავშალოთ ცხრილებიც მონაცემთა ბაზიდან. მაგალითისთვის შევქმნათ ახალი სატესტო ცხრილი რაიმე სტრუქტურით და ვუწოდოთ მას test. მის წასაშლელად გამოვიყენებთ ასეთ ბრძანებას:

```
DROP TABLE test;
```

წაშლა ასევე შესაძლებელია SSMS 19-ის გრაფიკული ინტერფეისის გამოყენებით. ამისათვის მონაცემთა ბაზაში არსებულ ცხრილზე ვიმოქმედებთ მაუსის მარჯვენა ღილაკით Object Explorer-ში. გამოვა კონტექსტური მენიუ, შემდეგ ავირჩევთ Delete ღილაკს, გამოვა ფანჯარა, დავაწკაპუნებთ გამოსულ ფანჯარაში OK ღილაკზე და ცხრილი წაიშლება.

ანალოგიურად შეიძლება წაიშალოს მონაცემთა ბაზა. მაგალითად, ჩვენ მიერ შექმნილი მონაცემთა ბაზის, სახელად statistika2-ის წასაშლელად შეგვიძლია გავუშვათ ბრძანება: DROP DATABASE statistika2, ან გამოვიყენოთ SSMS 19-ის ინტერფეისი. ამისათვის Object Explorer-ში შესაბამის მონაცემთა ბაზაზე მაუსის მარჯვენა ღილაკზე მოქმედებით გამოსული კონტექსტური მენიუდან ავირჩიოთ Delete და გამოსულ ფანჯარაში დავაწკაპუნოთ OK ღილაკზე. უნდა გავითვალისწინოთ ისიც, რომ წაშლისას არ უნდა გამოიყენებოდეს აღნიშნული მონაცემთა ბაზა.

ასევე, შესაძლებელია სქემის, მომხმარებლისა და შესვლის წაშლა შემდეგი ბრძანებებით: სქემის წაშლა: DROP SCHEMA სქემის სახელი. შესვლის წაშლა: DROP LOGIN shesvla; მომხმარებლის წაშლა: DROP USER momxmarebeli.

### *კითხვები თვითშემოწმებისთვის*

1. როგორ იქმნება მონაცემთა ბაზები?
2. როგორ იქმნება ცხრილები?
3. როგორ წავიკითხოთ ცხრილებიდან ინფორმაცია შესაბამისი ფილტრებით?
4. როგორ განვაახლოთ მონაცემები შესაბამისი კრიტერიუმებიდან გამომდინარე?
5. როგორ შევცვალოთ ცხრილის სტრუქტურა მონაცემთა ბაზაში?
6. როგორ წავშალოთ ცხრილიდან მონაცემები?
7. როგორ იშლება ცხრილი მონაცემთა ბაზიდან?

# თავი 4. გაღრმავებული CRUD ოპერაციები

## 4.1. ოპერატორები

SQL-ში შესაძლებელია სხვადასხვა არითმეტიკული ოპერაციის შესრულება, რომლებიც შეიძლება განხორციელდეს როგორც კონკრეტული სვეტების წაკითხვისას, ისე კონკრეტული WHERE პირობების გამოყენებისას. არითმეტიკული ოპერაციები მოცემულია ცხრილში 4.1.

**ცხრილი 4.1. არითმეტიკული ოპერაციები**

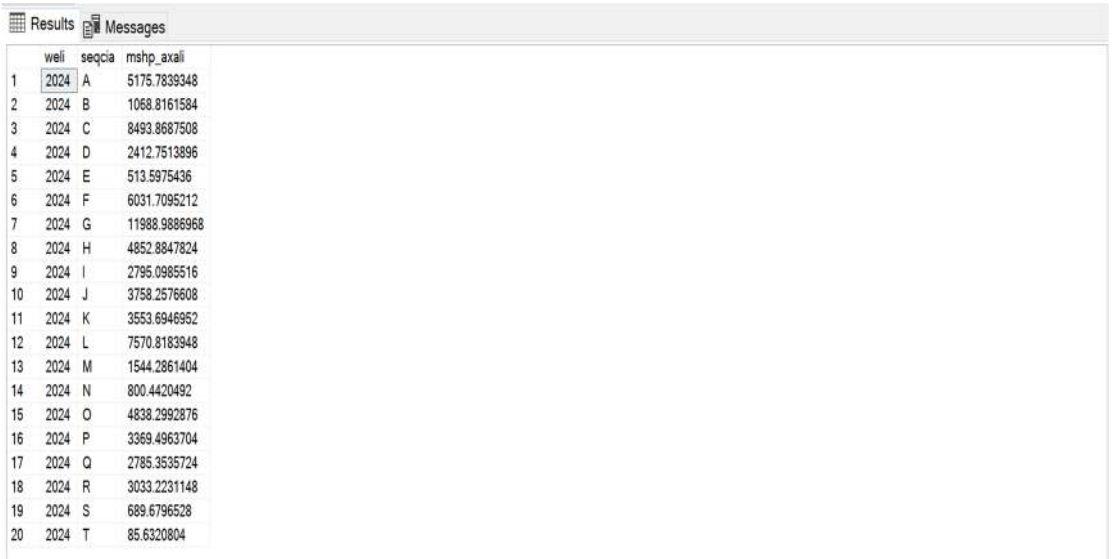
| ოპერატორი | განმარტება | მაგალითი |
|---|---|---|
| + | მიმატება | SELECT 2+2; |
| - | გამოკლება | SELECT 2-2; |
| * | გამრავლება | SELECT 2*2 |
| / | გაყოფა | SELECT 2/2 |
| % | ნაშთი | SELECT 2%2 |

mshp_seqcia ცხრილში, რომელიც ავაგეთ წინა თავში, მოთავსებულია 2020-2022 წლის მშპ საბაზისო ფასებში, კონკრეტული ეკონომიკური საქმიანობებისთვის სექციის დონეზე. მაგალითად, გვსურს 2024 წლის პროგნოზი გავაკეთოთ და დავუშვათ ვიცით, რომ 2024 წლის მშპ საბაზისო ფასებში ყველა ეკონომიკური საქმიანობისთვის გაიზრდება 20%-ით და გვჭირდება შესაბამისი ცხრილის გამოტანა. ამისათვის გამოვიყენებთ SQL ბრძანებას, რომელიც 2022 წლის ეკონომიკური საქმიანობების შესაბამის მშპ-ის მონაცემებს გაამრავლებს 1.2-ზე. ამ ამოცანისათვის მოთხოვნა შემდეგ სახეს მიიღებს:

```
SELECT 2024 [weli]
,[seqcia]
 ,[mshp] * 1.2 [mshp_axali]
FROM [mshp_seqcia]
WHERE weli = 2022;
```

შედეგი წარმოდგენილია სურათზე 4.1. ამ ბრძანებაში „WHERE weli = 2022“ პირობით წამოვიღეთ 2022 წლის მონაცემები და mshp სვეტში მოთავსებული თითოეული ჩანაწერი გავამრავლეთ 1.2-ზე და მიღებულ სვეტს ვუწოდეთ mshp_axali, ხოლო წლის მაგივრად 2024 შემოვიტანეთ და სვეტს ვუწოდეთ weli.

სურათი 4.1. საშედეგო მონაცემები



Results Messages

| | weli | seqcia | mshp_axali |
|---|---|---|---|
| 1 | 2024 | A | 5175.7839348 |
| 2 | 2024 | B | 1068.8161584 |
| 3 | 2024 | C | 8493.8687508 |
| 4 | 2024 | D | 2412.7513896 |
| 5 | 2024 | E | 513.5975436 |
| 6 | 2024 | F | 6031.7095212 |
| 7 | 2024 | G | 11988.9886968 |
| 8 | 2024 | H | 4852.8847824 |
| 9 | 2024 | I | 2795.0985516 |
| 10 | 2024 | J | 3758.2576608 |
| 11 | 2024 | K | 3553.6946952 |
| 12 | 2024 | L | 7570.8183948 |
| 13 | 2024 | M | 1544.2861404 |
| 14 | 2024 | N | 800.4420492 |
| 15 | 2024 | O | 4838.2992876 |
| 16 | 2024 | P | 3369.4963704 |
| 17 | 2024 | Q | 2785.3535724 |
| 18 | 2024 | R | 3033.2231148 |
| 19 | 2024 | S | 689.6796528 |
| 20 | 2024 | T | 85.6320804 |

## 4.2. WHERE პირობები

აქამდე WHERE პირობის გამოყენებისას მხოლოდ ერთ პირობას ვიყენებდით, თუმცა, უნდა აღინიშნოს, რომ შეგვიძლია ერთდროულად რამდენიმე პირობის გამოყენებაც. მაგალითისთვის დავუშვათ, რომ mshp_seqcia ცხრილიდან გვჭირდება 2022 წლის A სექციის მონაცემი. ამისათვის გვაქვს ორი პირობა:

1. mshp_seqcia ცხრილიდან უნდა ამოვიდეს 2022 წლის მონაცემები;
2. სექციის აღნიშვნა უნდა იყოს A.

თუ გვსურს, რომ ეს ორი პირობა ერთდროულად შესრულდეს, უნდა გამოვიყენოთ AND ოპერატორი. პირველი პირობა იქნება: „weli = 2022“, ხოლო მეორე პირობა ასე ჩაიწერება: „seqcia = N'A'“. თუ ორივე პირობას დავაკავშირებთ WHERE სინტაქსის გამოყენებით, მაშინ მოთხოვნა მიიღებს შემდეგ სახეს:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE weli = 2022 AND [seqcia] = N'A';
```

ამ ბრძანების გაშვების შემდეგ მივიღებთ 2022 წლის A სექციის მონაცემს.

დავუშვათ, რომ ცხრილიდან გვჭირდება მასში არსებული ყველა წლის A და B სექციის მონაცემები. ამისათვის უნდა გამოვიყენოთ OR ოპერატორი. შესაბამისად, ბრძანება მიიღებს შემდეგ სახეს:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [seqcia] = N'A' OR [seqcia] = N'B';
```

მაგრამ როგორ მოვიქცეთ მაშინ, თუ გვსურს ცხრილიდან A და B სექციების მონაცემები, ოღონდ მხოლოდ 2022 წლის? ამისათვის ჩვენი მოთხოვნა ორ პირობამდე დაიყვანება:

1. უნდა გამოვიყენოთ 2022 წლის მონაცემები;
2. სექციის აღნიშვნა უნდა იყოს A ან B.

შედეგად, პირველი პირობა „weli = 2022“ დაუკავშირდება მეორე პირობას „[seqcia] = N'A' OR [seqcia] = N'B'“ ოპერატორით AND, მაგრამ მეორე პირობა უნდა მოთავსდეს ფრჩხილებში. შედეგად, გასაშვები ბრძანება ასეთი იქნება:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE weli = 2022 AND ([seqcia] = N'A' OR [seqcia] = N'B');
```

წინა მაგალითში განხილული იყო მხოლოდ ორი სექცია. თუმცა, შეიძლება დაგვჭირდეს ისეთი მოთხოვნა, რომელიც საჭიროებს უფრო მეტ სექციას. ეს შესაძლებელია განვახორციელოთ OR ოპერატორების ჩასმით, მაგალითად: „[seqcia] = N'A' OR [seqcia] = N'B' OR [seqcia] = N'C'“ პირობებით, ან გამოვიყენოთ უფრო მარტივი ვარიანტი - IN ოპერატორი. მაგალითისთვის, გვჭირდება 2022 წლის A, B და C სექციის მონაცემები. ამისათვის მოთხოვნა მიიღებს შემდეგ სახეს:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE weli = 2022 AND seqcia IN (N'A', N'B', N'C');
```

აქ „seqcia IN (N'A', N'B', N'C');“ კოდი მიუთითებს, რომ seqcia სვეტის მონაცემი მოთავსებული უნდა იყოს შემდეგ მონაცემებში: (N'A', N'B', N'C').

რაც შეეხება რიცხვით მონაცემებს, ანალოგიურად შეიძლება პირობების დაკავშირება. თუ სვეტში მოცემული მონაცემებიდან გვჭირდება ისეთი მონაცემის ამოღება, რომელიც მოთავსებულია რაღაც დიაპაზონში, შესაძლებელია როგორც შედარების ოპერატორების გამოყენება შესაბამისი AND ოპერატორით, ან შეგვიძლია გამოვიყენოთ BETWEEN ოპერატორი. მაგალითად, mshp_seqcia ცხრილიდან თუ ვიღებთ ისეთ მონაცემებს, რომელიც არის 2020 და 2021 წლებს შორის, მაშინ შესაბამისი WHERE პირობა იქნება შემდეგ სახის: „WHERE weli BETWEEN 2020 AND 2021“.

შესაძლებელია სვეტში კონკრეტული მონაცემის ადგილი იყოს ცარიელი (NULL). ასეთი მონაცემების მოძიებისთვის გამოიყენება WHERE პირობა: „WHERE sveti IS NULL“, სადაც „sveti“ შესაბამისი სვეტის სახელია. ხოლო თუ გვსურს, რომ ისეთი პირობა გამოვიყენოთ, რომელიც გამორიცხავს ცარიელ მონაცემებს, მაშინ გამოვიყენებთ „IS NOT NULL“ კოდს.

ტექსტურ მონაცემებთან მუშაობისას, გარდა ტოლობისა, გამოიყენება მსგავსების ოპერატორიც LIKE. ამ ოპერატორით შესაძლებელია ისეთი ტექსტური მონაცემების მოძებნა, რომლებიც აკმაყოფილებენ რაღაც კანონზომიერებას.

განვიხილოთ მაგალითები:

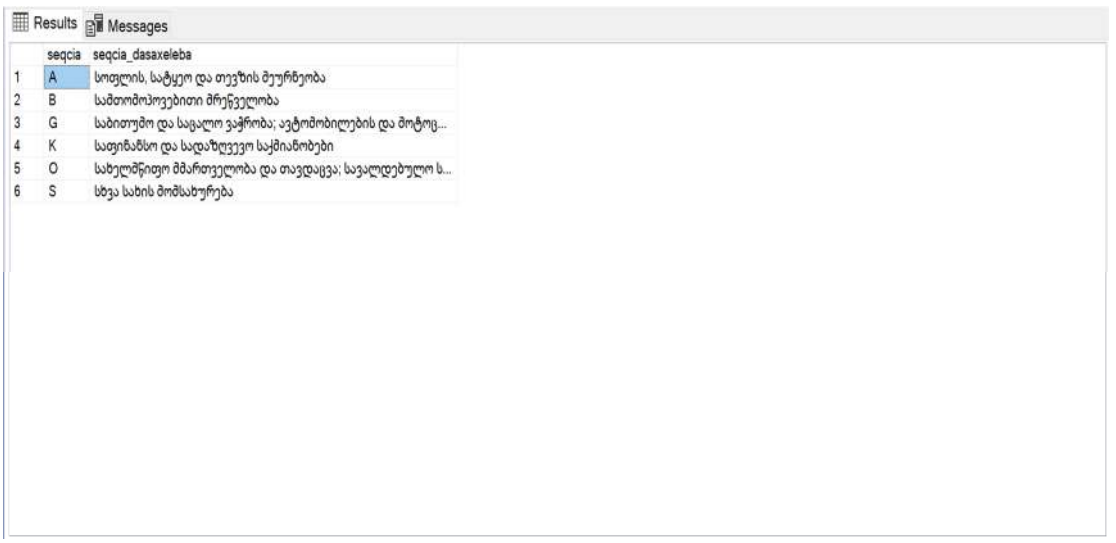
დავუშვათ, ჩვენ მიერ შექმნილი nace2 ცხრილიდან გვსურს ამოვიღოთ ისეთი მონაცემები, რომლის სექციის დასახელება იწყება „ს“ ასოზე. მაშინ WHERE ფილტრაციის გამოყენებისას გამოვიყენებთ LIKE ოპერატორს შემდეგი სინტაქსით:

„WHERE seqcia_dasaxeleba LIKE N'ს%';“, საბოლოოდ კოდი მიიღებს შემდეგ სახეს:

```
SELECT [seqcia], [seqcia_dasaxeleba]
FROM [statistika].[dbo].[nace2]
WHERE seqcia_dasaxeleba LIKE N'ს%';
```

მოთხოვნის შედეგი გამოსახულია სურათზე 4.2.

**სურათი 4.2. ტექსტის ფილტრაცია**


Results Messages

| | seqcia | seqcia_dasaxeleba |
|---|---|---|
| 1 | A | სოფლის, სატყეო და თევზის მეურნეობა |
| 2 | B | სამთომოპოვებითი მრეწველობა |
| 3 | G | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოც... |
| 4 | K | საფინანსო და სადაზღვევო საქმიანობები |
| 5 | O | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო ს... |
| 6 | S | სხვა სახის მომსახურება |

თუ გვსურს, რომ ტექსტი ბოლოვდებოდეს სიტყვით „მრეწველობა“, მაშინ შესაბამისი WHERE პირობა მიიღებს შემდეგ სახეს: „WHERE seqcia_dasaxeleba LIKE N'% მრეწველობა';“ ანალოგიურად შესაძლებელია კონკრეტული ტექსტის ნაწილი მოვძებნოთ მთლიან ტექსტში, მაგალითად, თუ დაგვჭირდა ისეთი ტექსტის ამოღება, რომელიც მოიცავს სიტყვას „და“, მაშინ შესაბამისი WHERE პირობა იქნება შემდეგი სახის: „WHERE seqcia_dasaxeleba LIKE N'%და%';“. LIKE ოპერატორით, ასევე, შესაძლებელია ისეთი ტექსტის ამოღება, რომლებიც იწყება (ან მოიცავს) სხვადასხვა ასოზე, მაგალითად, ისეთი ტექსტი, რომელიც იწყება ასო „ს“-ზე ან ასო „დ“-ზე. მაშინ შესაბამისი WHERE პირობის სახე იქნება: „WHERE seqcia_dasaxeleba LIKE N'[სდ]%';“. აღსანიშნავია, რომ არსებობს LIKE ოპერატორის სხვა პირობებიც. მაგალითად, „LIKE N'[^ს]%'“ ცხრილიდან ამოიღებს ისეთ ტექსტს, რომელიც არ იწყება „ს“ ასოზე, ხოლო პირობა „LIKE N'[A-I]%';“ ცხრილიდან დააბრუნებს ისეთ ტექსტს, რომლის შესაბამის სვეტში ჩანაწერი დაიწყება ანბანურად ასო A-დან ასო I-მდე. მაგალითისთვის, სექცია

A-დან სექცია I-ის ჩათვლით ჩანაწერების ამოსაღებად nace2 ცხრილიდან შეიძლება გამოვიყენოთ კოდი:

```
SELECT [seqcia], [seqcia_dasaxeleba]
FROM [statistika].[dbo].[nace2]
WHERE seqcia LIKE N'[A-I]';
```

LIKE ოპერატორთან გამოიყენება „_“ ოპერატორიც. მაგალითად, ბაზიდან გვსურს ამოვიღოთ ორი სიტყვა, რომლის პირველი ასო შეიძლება განსხვავდებოდეს, მაგრამ ვიცით, რომ არის ფიქსირებული ზომის სიტყვები და ბოლო სამი ასო არის მსგავსი. პირობითად, ასეთი სიტყვები შეიძლება იყოს „ფული“ და „ნული“. ასეთი სიტყვების ამოსაღებად შესაძლებელია გამოვიყენოთ LIKE ოპერატორი შემდეგი პირობით: „LIKE N'_ული'“.

## 4.3. დახარისხება

სტატისტიკური ანალიზის ჩატარებისას ერთ-ერთი ამოცანაა მონაცემთა რანჟირება, ანუ მათი დალაგება-დახარისხება ზრდადობით ან კლებადობით. ამისათვის SQL-ში არსებობს ORDER BY ბრძანება, რომელიც ჩვეულებრივ ახარისხებს მონაცემებს ზრდადობით, თუმცა, შეგვიძლია კლებადობითაც დავახარისხოთ. ზოგადად ის მიეთითება WHERE პირობის შემდეგ, თუ WHERE ფილტრაციას ვიყენებთ მონაცემთა ანალიზისას. მაგალითად, გვსურს დავახარისხოთ 2022 წლის მონაცემები mshp სვეტში არსებული ჩანაწერის ზრდადობით, მაშინ მოთხოვნის სინტაქსი შემდეგია:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE weli = 2022
ORDER BY mshp ASC;
```

რადგან ზრდადობით ვახარისხებთ, შეგვიძლია გამოვტოვოთ მოთხოვნაში ASC კოდი, თუმცა, თუ გვსურს კლებადობით დავახარისხოთ, მაშინ ბრძანებაში „ASC“-ის მაგივრად ჩავწერთ „DESC“-ს. კლებადობით დახარისხებული მონაცემების მისაღებად საჭირო ბრძანება 2022 წლისათვის იქნება:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE weli = 2022
ORDER BY mshp DESC;
```

შედეგი გამოსახულია სურათზე 4.3.

სურათი 4.3. კლებადობით დახარისხებული მონაცემების შედეგები

Results Messages

| | weli | seqcia | mshp |
|---|---|---|---|
| 1 | 2022 | G | 9990.823914 |
| 2 | 2022 | C | 7078.223959 |
| 3 | 2022 | L | 6309.015329 |
| 4 | 2022 | F | 5026.424601 |
| 5 | 2022 | A | 4313.153279 |
| 6 | 2022 | H | 4044.070652 |
| 7 | 2022 | O | 4031.916073 |
| 8 | 2022 | J | 3131.881384 |
| 9 | 2022 | K | 2961.412246 |
| 10 | 2022 | P | 2807.913642 |
| 11 | 2022 | R | 2527.685929 |
| 12 | 2022 | I | 2329.248793 |
| 13 | 2022 | Q | 2321.127977 |
| 14 | 2022 | D | 2010.626158 |
| 15 | 2022 | M | 1286.905117 |
| 16 | 2022 | B | 890.680132 |
| 17 | 2022 | N | 667.035041 |
| 18 | 2022 | S | 574.733044 |
| 19 | 2022 | E | 427.997953 |
| 20 | 2022 | T | 71.360067 |

## 4.4. მონაცემთა აგრეგირება

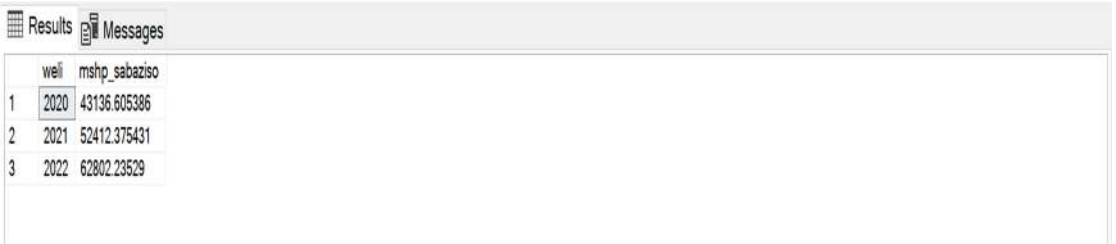
ფანჯრის ფუნქციებიდან (Window functions) საკმაოდ გამოყენებადია აგრეგირებული ფუნქციები. ზოგჯერ საჭიროა მონაცემების მთლიანობაში განხილვა, მაგალითად, ეს შეიძლება იყოს კონკრეტულ სვეტში არსებული მონაცემების შეჯამება, საშუალო არითმეტიკულის გამოთვლა, მინიმალური და მაქსიმალური მნიშვნელობების პოვნა და ა. შ. ამისათვის გამოიყენება აგრეგირებული ფუნქციები. მოცემულ ქვეთავში მაგალითების საფუძველზე განვიხილავთ რამდენიმე მათგანს.

mshp_seqcia ცხრილში გვაქვს 2020-2022 წლის საბაზისო ფასებში მოცემული მონაცემები სხვადასხვა ეკონომიკური საქმიანობის მიხედვით. წამოვიდგინოთ, რომ დაგვჭირდა წლების მიხედვით ამ მონაცემების შეჯამება, მაშინ უნდა გამოვიყენოთ შეჯამების აგრეგირებული ფუნქცია SUM, რომელიც შესაბამისი პირობის მიხედვით შეაჯამებს თითოეულ წელში არსებულ ყველა სექციას. ამ შემთხვევაში, შეჯამებისათვის უნდა გამოვიყენოთ GROUP BY ბრძანება, რომლითაც მივუთითებთ, რომ ვაჯამებთ წლის მიხედვით და SUM ფუნქცია mshp სვეტში არსებული მონაცემების დასაჯამებლად. შესაბამისი კოდი მიიღებს შემდეგ სახეს:

```
SELECT [weli], SUM([mshp]) AS mshp_sabaziso
FROM [statistika].[dbo].[mshp_seqcia]
GROUP BY [weli]
```

არსებული ბრძანების გაშვებისას მიღებული შედეგი გამოსახულია სურათზე 4.4.

სურათი 4.4. საბაზისო მშპ-ის შეჯამების შედეგები



Results Messages

| | weli | mshp_sabaziso |
|---|---|---|
| 1 | 2020 | 43136.605386 |
| 2 | 2021 | 52412.375431 |
| 3 | 2022 | 62802.23529 |

ანალოგიურად შეგვიძლია ვიპოვოთ საშუალო საბაზისო მშპ დარგების მონაცემებიდან კონკრეტული წლისათვის. ამისათვის გამოვიყენებთ AVG ფუნქციას და ზედა კოდში SUM ფუნქციას ჩავანაცვლებთ AVG ფუნქციით. შესაბამისად, ბრძანება იქნება შემდეგი:

```
SELECT [weli], AVG([mshp]) AS mshp_sabaziso_sashualo
FROM [statistika].[dbo].[mshp_seqcia]
GROUP BY [weli]
```

შედეგი გამოსახულია სურათზე 4.5.

სურათი 4.5. საშუალო საბაზისო მშპ წლის მიხედვით

Results Messages

| | weli | mshp_sabaziso_sashualo |
|---|---|---|
| 1 | 2020 | 2156.8302693 |
| 2 | 2021 | 2620.61877155 |
| 3 | 2022 | 3140.1117645 |

ანალოგიურად შეგვიძლია დავთვალოთ სტანდარტული გადახრაც, ამისათვის გამოვიყენებთ STDEV ფუნქცია AVG ფუნქციის მაგივრად და შესაბამისი მოთხოვნა იქნება:

```
SELECT [weli], STDEV([mshp]) AS mshp_sabaziso_standartuli
FROM [statistika].[dbo].[mshp_seqcia]
GROUP BY [weli];
```

აგრეგირებული ფუნქციებიდან შეგვიძლია გამოვყოთ მინიმალური და ის ფუნქციები, რომლებსაც შეუძლიათ მაქსიმალური მნიშვნელობის ამოღება. ამისათვის გამოიყენება MAX და MIN ფუნქციები. მაგალითად, დარგების მონაცემებიდან მაქსიმალური მნიშვნელობის ამოღება კონკრეტული წლისათვის შესაძლებელია განვახორციელოთ შემდეგი კოდით:

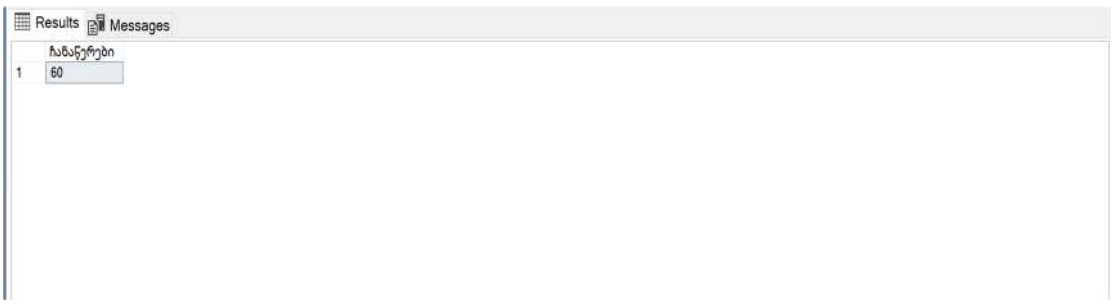
```
SELECT [weli], MAX([mshp]) max_
FROM [statistika].[dbo].[mshp_seqcia]
GROUP BY [weli]
```

გარდა ამისა, შეგვიძლია დავთვალოთ მონაცემები, რისთვისაც გამოიყენება COUNT ფუნქცია, რომლითაც შეიძლება დაითვალოს ჩანაწერების რაოდენობა. დავუშვათ, გვსურს ჩანაწერთა რაოდენობის დათვლა mshp_seqcia ცხრილში. ამისათვის ავირჩევთ ერთ-ერთ სვეტს (მაგალითად, weli) და გამოვიყენებთ ასეთ ბრძანებას:

```
SELECT COUNT([weli]) ჩანაწერები
FROM [statistika].[dbo].[mshp_seqcia];
```

შედეგები მოცემულია სურათზე 4.6.

**სურათი 4.6. ჩანაწერების რაოდენობა ცხრილში**

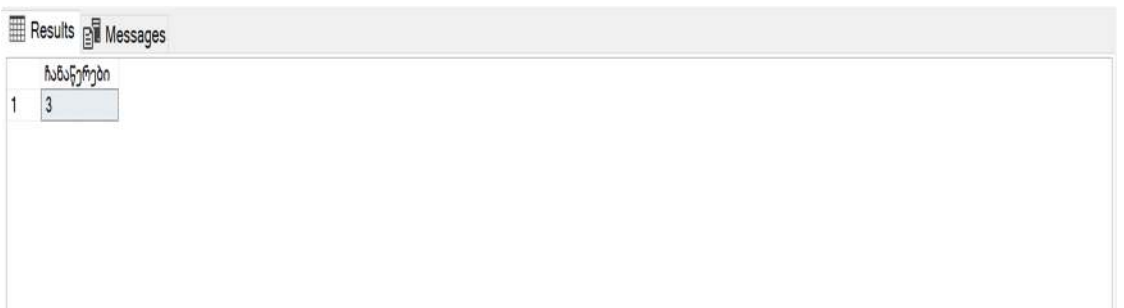



ზემოთ მოცემულმა ბრძანებამ მოგვცა ყველა ჩანაწერის რაოდენობა ცხრილში. მაგრამ თუ გვაინტერესებს უნიკალური ჩანაწერების რაოდენობა კონკრეტული სვეტის მიხედვით, მაგალითად, რამდენი წლის მონაცემია სულ mshp_seqcia ცხრილში, ამისთვის COUNT ფუნქციაში გამოვიყენებთ DISTINCT-ს. შესაბამისად, მოთხოვნა იქნება:

```
SELECT COUNT(DISTINCT [weli]) ჩანაწერები
FROM [statistika].[dbo].[mshp_seqcia]
```

შედეგი მოცემულია სურათზე 4.7.

**სურათი 4.7. უნიკალური წლები mshp_seqcia ცხრილში**

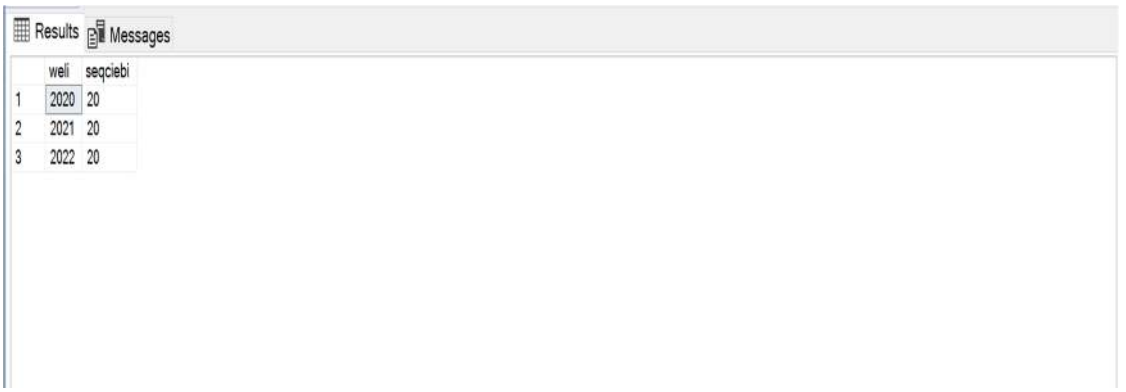



შესაძლოა, დაგვჭირდეს თითოეული წლისათვის არსებული ჩანაწერების რაოდენობა ცხრილში. ამისათვის COUNT ფუნქციასთან ერთად გამოვიყენებთ GROUP BY-ს და საშედეგო მოთხოვნა ასეთი იქნება:

```
SELECT [weli], COUNT(seqcia) seqciebi
FROM [statistika].[dbo].[mshp_seqcia]
GROUP BY [weli];
```

მოთხოვნის შედეგი მოცემულია სურათზე 4.8, სადაც სვეტი სახელწოდებით „seqciebi“ გვიჩვენებს სექციების რაოდენობას შესაბამისი წლისათვის.

**სურათი 4.8. სექციების რაოდენობა თითოეული წლისათვის**




აგრეგირების ფუნქციები შეგვიძლია გამოვიყენოთ მათემატიკურ ოპერატორებთანაც.

შესაძლებელია აგრეგირებულ ფუნქციებთან ერთად გამოვიყენოთ სხვა

მათემატიკური ოპერაციებიც, მაგალითად: დავუშვათ, გვაინტერესებს 2022 წლისთვის თითოეული სექციის რამდენი პროცენტი იყო მშპ საბაზისო ფასებში მთლიან საბაზისო ფასებში მშპ-თან. ამისთვის შეგვიძლია შევაჯამოთ კონკრეტული წლისთვის ყველა სექციაში არსებული საბაზისო მშპ, შემდეგ თითოეულ სექციაში არსებული საბაზისო მშპ გავყოთ დაჯამებულ მშპ-ზე და გავამრავლოთ 100-ზე. რაც შეეხება ამ ამოცანის SQL-ში ამოხსნას, გამოვიყენებთ „OVER (PARTITION BY sveti1, sveti2...)“ სინტაქსს, მაგრამ ჩვენ შემთხვევაში მხოლოდ ერთ სვეტს, ანუ წელს ვიყენებთ „GROUP BY“-ის გარეშე. ამოცანის ამოსახსნელად შესაბამისი ბრძანება იქნება:

```
SELECT [weli], [seqcia],
[mshp] / SUM(mshp) OVER (PARTITION BY weli) * 100 [wili]
FROM
mshp_seqcia
WHERE [weli] = 2022;
```

აქვე უნდა აღინიშნოს, რომ ასეთი აგრეგირება შესაძლებელია ერთდოულად რამდენიმე სვეტის გამოყენებისას. ასეთ შემთხვევაში „PARTITION BY“-ის შემდეგ მივუთითებთ სხვადასხვა სვეტს. აღნიშნული პრაქტიკულად შეგვიძლია გამოვიყენოთ მონაცემთა სტრატიფიცირებისთვის.

აგრეგირებული ფუნქციებიდან მიღებული შედეგები შეგვიძლია გავფილტროთ HAVING ფილტრაციის პირობებით, რომელიც ჰგავს WHERE-ის სინტაქსს, თუმცა, აგრე-

გირების შემთხვევაში HAVING თავსდება GROUP BY-ის შემდეგ და მუშაობს აგრეგირებულ ფუნქციებთან. მაგალითად, თუ მონაცემებიდან ვაჯამებთ საბაზისო ფასებში მშპ-ს mshp_seqcia ცხრილიდან წლების მიხედვით და გვსურს ამოვიღოთ იმ წლების მონაცემები, რომლის დროსაც საბაზისო მშპ იყო 50 მილიარდ ლარზე მეტი, მაშინ გამოვიყენებთ შემდეგ ბრძანებას:

```
SELECT [weli], SUM(mshp) mshp
FROM [statistika].[dbo].[mshp_seqcia]
GROUP BY [weli]
HAVING SUM(mshp) > 50000
```

შესაბამისი შედეგი გამოსახულია 4.9 სურათზე.

**სურათი 4.9. HAVING ფილტრაციის შედეგები**


## 4.5. ჩადგმული მოთხოვნები და საერთო ცხრილური გამოსახულებები

ერთ-ერთი საინტერესო შესაძლებლობა, რომელიც გააჩნია SQL-ს, არის ჩადგმული მოთხოვნები. ამ ტიპის მოთხოვნებში შესაძლებელია ერთი მოთხოვნიდან მეორე მოთხოვნამ მოიძიოს შესაბამისი მონაცემები. უკეთ გასააზრებლად განვიხილოთ ასეთი მაგალითი: ზოგჯერ ეროვნულ ანგარიშთა სისტემა იყენებს ზოგიერთი სექციის აგრეგაციას ერთი ტიპის ეკონომიკურ საქმიანობად. ეს დეტალურად განხილულია NACE Rev.2 კლასიფიკატორში (ასეთი სექციები შეიძლება იყოს B, C, D, E ერთად, G, H, I ერთად და ა. შ). დავუშვათ, ვაჯამებთ მშპ საბაზისო ფასებში სექციების მიხედვით 2022 წლის მონაცემებიდან, მაგრამ ამჯერად გვსურს უფრო მაღალი დონის აგრეგაცია, რომელიც განვიხილეთ. ამისათვის ჯერ მოვიძიოთ სახეშეცვლილი მონაცემები ცხრილიდან, სადაც სექციის სვეტის ნაცვლად იქნება აგრეგირებული ფუნქციები. ამისათვის მოთხოვნა მიიღებს შემდეგ სახეს:

```
SELECT [weli],
CASE
 WHEN [seqcia] IN (N'B', N'C', N'D', N'E') THEN N'BCDE'
 WHEN [seqcia] IN (N'G', N'H', N'I') THEN N'GHI'
```

```
 WHEN [seqcia] IN (N'M', N'N') THEN N'MN'
 WHEN [seqcia] IN (N'O', N'P', N'Q') THEN N'OPQ'
 WHEN [seqcia] IN (N'R', N'S', N'T', N'U') THEN N'RSTU'
 ELSE [seqcia]
END [seqcia_aggregated],
mshp
FROM [statistika].[dbo].[mshp_seqcia]
WHERE weli = 2022;
```

შედეგი გამოსახულია სურათზე 4.10.

**სურათი 4.10. აგრეგირებული სექციები**

Results Messages

| | weli | seqcia_aggregated | mshp |
|---|---|---|---|
| 1 | 2022 | A | 4313.153279 |
| 2 | 2022 | BCDE | 890.680132 |
| 3 | 2022 | BCDE | 7078.223959 |
| 4 | 2022 | BCDE | 2010.626158 |
| 5 | 2022 | BCDE | 427.997953 |
| 6 | 2022 | F | 5026.424601 |
| 7 | 2022 | GHI | 9990.823914 |
| 8 | 2022 | GHI | 4044.070652 |
| 9 | 2022 | GHI | 2329.248793 |
| 10 | 2022 | J | 3131.881384 |
| 11 | 2022 | K | 2961.412246 |
| 12 | 2022 | L | 6309.015329 |
| 13 | 2022 | MN | 1286.905117 |
| 14 | 2022 | MN | 667.035041 |
| 15 | 2022 | OPQ | 4031.916073 |
| 16 | 2022 | OPQ | 2807.913642 |
| 17 | 2022 | OPQ | 2321.127977 |
| 18 | 2022 | RSTU | 2527.685929 |
| 19 | 2022 | RSTU | 574.733044 |
| 20 | 2022 | RSTU | 71.360067 |

ამის შემდეგ ამ მოთხოვნიდან მიღებული შედეგებში უნდა დავაჯამოთ მონაცემები seqcia_aggregated სვეტის მიხედვით. ამისათვის ამ მოთხოვნას ჩავსვამთ სხვა მოთხოვნაში, რომელიც აჯამებს რიცხვებს seqcia_aggregated სვეტისა და წლის მიხედვით. ამისათვის გამოვიყენებთ შემდეგ კოდს:

```
SELECT weli, seqcia_aggregated, SUM(mshp) mshp
FROM(
SELECT [weli],
CASE
 WHEN [seqcia] IN (N'B', N'C', N'D', N'E') THEN N'BCDE'
 WHEN [seqcia] IN (N'G', N'H', N'I') THEN N'GHI'
 WHEN [seqcia] IN (N'M', N'N') THEN N'MN'
 WHEN [seqcia] IN (N'O', N'P', N'Q') THEN N'OPQ'
 WHEN [seqcia] IN (N'R', N'S', N'T', N'U') THEN N'RSTU'
```

```
 ELSE [seqcia]
END [seqcia_aggregated],
mshp
FROM [statistika].[dbo].[mshp_seqcia]
WHERE weli = 2022
) a1
GROUP BY weli, seqcia_aggregated
```

როგორც ვხედვათ, „FROM“-ის შემდეგ ცხრილის მაგივრად გამოიყენება მოთხოვნა, რომელსაც ვუწოდეთ პირობითად a1 და როგორც ცხრილიდან, ისე ჩადგმული მოთხოვნიდან მოვიძიეთ სვეტები. მოთხოვნების ასეთი ჩადგმა გამოიყენება სხვა ოპერატორებთანაც, მაგალითად, IN ოპერატორთან. ამ ამოცანის ამოსახსნელად გამოვიყენეთ ზემოხსენებული მეთოდი, თუმცა, შესაძლებელია სხვა მეთოდების გამოყენებაც. შედეგი მოცემულია სურათზე 4.11.

**სურათი 4.11. დაჯამებული მონაცემები ჩადგული მოთხოვნების საფუძველზე**

Results Messages

| | weli | seqcia_aggregated | mshp |
|---|---|---|---|
| 1 | 2022 | A | 4313.153279 |
| 2 | 2022 | BCDE | 10407.528202 |
| 3 | 2022 | F | 5026.424601 |
| 4 | 2022 | GHI | 16364.143359 |
| 5 | 2022 | J | 3131.881384 |
| 6 | 2022 | K | 2961.412246 |
| 7 | 2022 | L | 6309.015329 |
| 8 | 2022 | MN | 1953.940158 |
| 9 | 2022 | OPQ | 9160.957692 |
| 10 | 2022 | RSTU | 3173.77904 |

შეგვიძლია ჩადგმული მოთხოვნების საშუალებით დავატრიალოთ ცხრილები, რისთვისაც გამოიყენება PIVOT-ს. T-SQL საშუალებით შესაძლებელია მონაცემთა გაშლა სხვადასხვა ოპერაციის გამოყენებით. მათგან ერთ-ერთია PIVOT ოპერაცია, რომელიც გამოიყენება SELECT მოთხოვნის FROM განყოფილებაში. ის ამუშავებს საწყის ცხრილს ან ცხრილურ გამოსახულებას, გამოთვლის ჯამურ მნიშვნელობას და გასცემს შედეგობრივ ცხრილს. PIVOT ოპერაცია მოიცავს ლოგიკური დამუშავების ადრე აღწერილ სტადიებს (დაჯგუფება, გაშლა და შეჯამება), გაშლის იმავე ელემენტებით.

PIVOT ოპერაციის შემცველი მოთხოვნის სინტაქსია:

```
SELECT
. . .
FROM<საწყისი_ცხრილი_ან_ცხრილური_გამოსახულება>PIVOT(<აგრეგირების_ფუნქცია>(<შედეგობრივი_ელემენტი>)
FOR <გასაშლელი_ელემენტი>
IN (<შედეგობრივი_სვეტების_სია>)) AS <შედეგობრივი_ცხრილი ფსევდონიმი>
```

PIVOT ცხრილური ოპერაციის ფრჩხილებში მოიცემა აგრეგირების ფუნქცია (SUM()) შედეგობრივი ელემენტი, გასაშლელი ელემენტი და შედეგობრივი სვეტების სია (A,B,C,D). PIVOT ოპერაციის მრგვალი ფრჩხილების შემდეგ მიეთითება შედეგობრივი ცხრილის ფსევდონიმი.

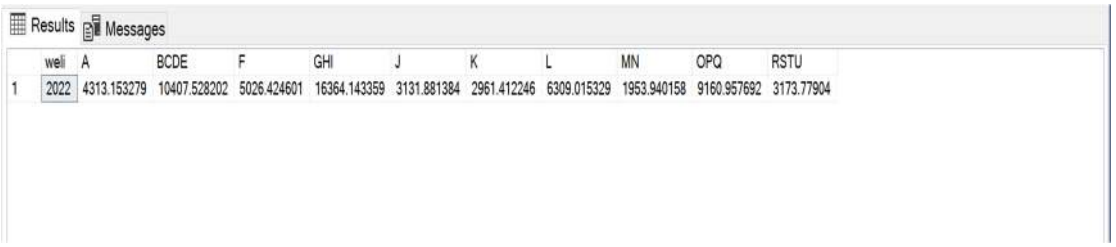
მნიშვნელოვანია აღვნიშნოთ, რომ PIVOT ცხრილურ ოპერაციაში აშკარად არ მიეთითება მაჯგუფებელი ელემენტი. ამით გამოირიცხება მოთხოვნაში GROUP BY განყოფილების ჩასმის აუცილებლობა. PIVOT ოპერაცია ირიბად განსაზღვრავს მაჯგუფებელ ელემენტებს, როგორც ყველა სვეტს საწყისი ცხრილიდან (ან ცხრილური გამოსახულებიდან). საწყისი ცხრილი PIVOT ოპერაციისათვის არ უნდა შეიცავდეს არანაირ სვეტს, გარდა მაჯგუფებელი, გასაშლელი და შედეგობრივი ელემენტებისა. ამას შეგვიძლია მივაღწიოთ, თუ გამოვიყენებთ PIVOT ცხრილურ ოპერაციას უშუალოდ ცხრილური გამოსახულების მიმართ, რომელიც მხოლოდ საჭირო სვეტებს მოიცავს [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 235].

ზემოთ არსებული მოთხოვნით მიღებული მონაცემებიდან ჩვენს შემთხვევაში თითოეულ სექციას შეგვიძლია მივანიჭოთ ცალკე სვეტი. ასეთ შემთხვევაში კოდი მიიღებს შესაბამის სახეს:

```
SELECT weli, [A], [BCDE], [F], [GHI], [J], [K], [L], [MN], [OPQ], [RSTU]
FROM
(
SELECT weli, seqcia_aggregated, SUM(mshp) mshp
FROM(
SELECT [weli],
CASE
 WHEN [seqcia] IN (N'B', N'C', N'D', N'E') THEN N'BCDE'
 WHEN [seqcia] IN (N'G', N'H', N'I') THEN N'GHI'
 WHEN [seqcia] IN (N'M', N'N') THEN N'MN'
 WHEN [seqcia] IN (N'O', N'P', N'Q') THEN N'OPQ'
 WHEN [seqcia] IN (N'R', N'S', N'T', N'U') THEN N'RSTU'
 ELSE [seqcia]
END [seqcia_aggregated],
mshp
FROM [statistika].[dbo].[mshp_seqcia]
WHERE weli = 2022
) a1
GROUP BY weli, seqcia_aggregated
) a2
PIVOT(
MAX(mshp)
FOR seqcia_aggregated IN ([A], [BCDE], [F], [GHI], [J], [K], [L], [MN], [OPQ], [RSTU])
) a3
```

ამ მოთხოვის შედეგი მოცემულია სურათზე 4.12.

სურათი 4.12. ცხრილის დატრიალება PIVOT-ით



Results Messages

| | weli | A | BCDE | F | GHI | J | K | L | MN | OPQ | RSTU |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2022 | 4313.153279 | 10407.528202 | 5026.424601 | 16364.143359 | 3131.881384 | 2961.412246 | 6309.015329 | 1953.940158 | 9160.957692 | 3173.77904 |

ჩვენ უკვე დავატრიალეთ შეჯამებული მონაცემები, თუმცა, შესაძლებელია ცხრილის დატრიალება დაუჯამებელი მონაცემების არსებობისას და დატრიალებისას მისი შეჯამება. ასეთ შემთხვევაში კოდი უფრო გამარტივდება და მიიღებს შემდეგ სახეს, შედეგი კი იგივე იქნება:

```
SELECT [weli], [A], [BCDE], [F], [GHI], [J], [K], [L], [MN], [OPQ], [RSTU]
FROM
(
SELECT [weli],
CASE
 WHEN [seqcia] IN (N'B', N'C', N'D', N'E') THEN N'BCDE'
 WHEN [seqcia] IN (N'G', N'H', N'I') THEN N'GHI'
 WHEN [seqcia] IN (N'M', N'N') THEN N'MN'
 WHEN [seqcia] IN (N'O', N'P', N'Q') THEN N'OPQ'
 WHEN [seqcia] IN (N'R', N'S', N'T', N'U') THEN N'RSTU'
 ELSE [seqcia]
END [seqcia_aggregated],
mshp
FROM [statistika].[dbo].[mshp_seqcia]
WHERE weli = 2022
) a1
PIVOT(
SUM(mshp)
FOR seqcia_aggregated IN ([A], [BCDE], [F], [GHI], [J], [K], [L], [MN], [OPQ], [RSTU])
) a2
```

არსებობს ასევე UNPIVOT, რომელიც PIVOT-ის საპირისპირო ოპერაციას ახორციელებს.

გარდა ჩადგმული მოთხოვნებისა, შეგვიძლია გამოვიყენოთ საერთო ცხრილური გამოსახულებები (ინგლ. Common table expressions, CTE). განვიხილოთ ამ ქვეთავში არსებული პირველი მაგალითი ეროვნულ ანგარიშთა სისტემაში მაღალი დონის აგრეგირების შესახებ. CTE-ის გამოყენებისას საჭიროა გამოვაცხადოთ CTE შესაბამისი სევეტებით, მასში მოვათავსოთ მოთხოვნა და შემდეგ მოთხოვნიდან მიღებულ მონაცემებზე განვახორციელოთ სხვადასხვა ოპერაცია. ამისათვის გამოიყენება შემდეგი კოდის მსგავსი სინტაქსი: „WITH CTE (sveti1, sveti2) AS (<motxovna1>) SELECT * FROM CTE“. პრაქტიკაში, ჩვენი ამოცანიდან გამომდინარე, კოდი მიიღებს შემდეგ სახეს:

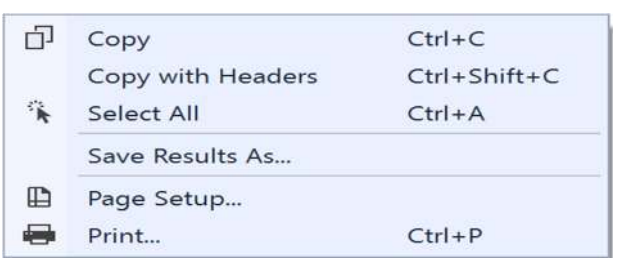
```
WITH CTE ([weli], [seqcia_aggregated], [mshp]) AS
(
SELECT [weli],
CASE
 WHEN [seqcia] IN (N'B', N'C', N'D', N'E') THEN N'BCDE'
 WHEN [seqcia] IN (N'G', N'H', N'I') THEN N'GHI'
 WHEN [seqcia] IN (N'M', N'N') THEN N'MN'
 WHEN [seqcia] IN (N'O', N'P', N'Q') THEN N'OPQ'
 WHEN [seqcia] IN (N'R', N'S', N'T', N'U') THEN N'RSTU'
 ELSE [seqcia]
END [seqcia_aggregated],
mshp
FROM [statistika].[dbo].[mshp_seqcia]
WHERE weli = 2022
)
SELECT [weli], [seqcia_aggregated], SUM(mshp) mshp
FROM CTE
GROUP BY [weli], [seqcia_aggregated]
```

რაც შეეხება შედეგს, ის იგივე იქნება, რაც მივიღეთ ჩადგმული მოთხოვნების გამოყენების დროს.

## 4.6. მონაცემთა გატანა სხვადასხვა მეთოდით

მონაცემები შეგვიძლია გავიტანოთ სხვადასხვა ფორმატში. SSMS 19-ში გაშვებული მოთხოვნიდან მიღებული მონაცემები შეგვიძლია დავაკოპიროთ და ჩავსვათ სხვადასხვა პროგრამაში, იქნება ეს Microsoft Excel, Microsoft Word თუ სხვა. ამისათვის მიღებულ შედეგებზე დავაწკაპუნებთ მაუსის მარჯვენა ღილაკით, გამოვა კონტექსტური მენიუ, რომელიც გამოსახულია სურათზე 4.13 და ავირჩევთ „Copy“-ის. თუ გვსურს მონაცემთა დაკოპირება სვეტების სათაურების გარეშე, ხოლო მონაცემების სვეტების სათაურებთან ერთად გასატანად, შეგვიძლია გამოვიყენოთ „Copy with Headers ღილაკი“. შესაძლებელია ასევე მონაცემების ამობეჭდვა „Print“ ღილაკით და შენახვაც „Save Results As...“ ღილაკით.

**სურათი 4.13. მიღებულ მონაცემთა გატანა**



შესაძლებელია, ასევე, SSMS 19-ში დაწერილი მოთხოვნების შენახვაც, რომელსაც გამოვიყენებთ მოგვიანებით. ამისათვის დაწერილ მოთხოვნას ვინახავთ კლავიატურაზე არსებული Ctrl+S ღილკაკებით ან File>Save <motxovnis_saxeli>.sql ღილაკით, სადაც <motxovnis_saxeli> არის შესანახი მოთხოვნის დასახელება.

***კითხვები თვითშემოწმებისთვის:***

1. ახსენით არითმეტიკული ოპერატორების გამოყენება SQL-ში.
2. როგორ ხდება მონაცემთა დახარისხება ზრდადობით ან კლებადობით?
3. ახსენით აგრეგირებული ფუნქციების გამოყენება, მოიფიქრეთ ამოცანა და ამოხსენით აგრეგირებული ფუნქციებით.
4. როგორ ხდება SQL-ში საშუალო არითმეტიკულისა და სტანდარტული გადახრის გამოთვლა?
5. რაში გამოიყენება ჩადგმული მოთხოვნები?
6. როგორ ხდება ცხრილების დატრიალება PIVOT-ით?
7. როგორ გაიტანთ მოთხოვნიდან მიღებულ მონაცემებს?

# თავი 5. ცხრილების გაერთიანებები და კავშირები

## 5.1. ცხრილების გაერთიანებები

SQL-ში შესაძლებელია ცხრილების გაერთიანება, რისთვისაც გამოიყენება UNION ოპერატორი. ამ ოპერატორით შეიძლება არა მარტო ცხრილების, არამედ ცალკეული მოთხოვნების შედეგად მიღებული ინფორმაციის გაერთიანება. ყველაფრის უკეთ გასააზრებლად საკითხი ავხსნათ კონკრეტული მაგალითის გამოყენებით.

დავუშვათ, რომ გვაქვს ამოცანა, რომ ჩვენ მიერ შექმნილი mshp_seqcia ცხრილიდან მოვახდინოთ პროგნოზირება ისე, რომ 2024 წლის თითოეული სექციის საბაზისო მშპ იყოს 2022 წლის შესაბამისი სექციის საბაზისო მშპ-თან 20%-ით გაზრდილი. ამისათვის კოდის შემუშავება მარტივია:

```
SELECT 2024 weli, [seqcia], [mshp] * 1.2 mshp
FROM [mshp_seqcia]
WHERE weli = 2022;
```

მაგრამ, ამასთან ერთად, გვთხოვენ, რომ 2025 წლის საბაზისო მშპ თითოეული სექციისთვის იყოს 2022 წლის მშპ-თან შედარებით 35%-ით გაზრდილი და შესაბამისი მონაცემი უნდა გამოვიტანოთ 2024 წლის საპროგნოზო მონაცემთან ერთად. ამისათვის ჯერ დავწერთ ბრძანებას, რომელიც არსებული მეთოდის საფუძველზე 2025 წლისათვის ახდენს პროგნოზირებას, ხოლო შემდეგ ამ ორ მოთხოვნას დავაკავშირებთ UNION ოპერატორის საშუალებით. შედეგად, შესაბამისი მოთხოვნა იქნება:

```
SELECT 2024 weli, [seqcia], [mshp] * 1.2 mshp
FROM [mshp_seqcia]
WHERE weli = 2022
UNION
SELECT 2025, [seqcia], [mshp] * 1.35 mshp
FROM [mshp_seqcia]
WHERE weli = 2022;
```

ამ კოდის შესაბამისი შედეგი მოცემულია სურათზე 5.1.

სურათი 5.1. საპროგნოზო მონაცემების გაერთიანება

| | weli | seqcia | mshp |
|---|---|---|---|
| 1 | 2024 | A | 5175.7839348 |
| 2 | 2024 | B | 1068.8161584 |
| 3 | 2024 | C | 8493.8687508 |
| 4 | 2024 | D | 2412.7513896 |
| 5 | 2024 | E | 513.5975436 |
| 6 | 2024 | F | 6031.7095212 |
| 7 | 2024 | G | 11988.9886968 |
| 8 | 2024 | H | 4852.8847824 |
| 9 | 2024 | I | 2795.0985516 |
| 10 | 2024 | J | 3758.2576608 |
| 11 | 2024 | K | 3553.6946952 |
| 12 | 2024 | L | 7570.8183948 |
| 13 | 2024 | M | 1544.2861404 |
| 14 | 2024 | N | 800.4420492 |
| 15 | 2024 | O | 4838.2992876 |
| 16 | 2024 | P | 3369.4963704 |
| 17 | 2024 | Q | 2785.3535724 |
| 18 | 2024 | R | 3033.2231148 |
| 19 | 2024 | S | 689.6796528 |
| 20 | 2024 | T | 85.6320804 |
| 21 | 2025 | A | 5822.75692665 |
| 22 | 2025 | B | 1202.4181782 |
| 23 | 2025 | C | 9555.60234465 |
| 24 | 2025 | D | 2714.3453133 |
| 25 | 2025 | E | 577.79723655 |
| 26 | 2025 | F | 6785.67321135 |
| 27 | 2025 | G | 13487.6122839 |
| 28 | 2025 | H | 5459.4953802 |
| 29 | 2025 | I | 3144.48587055 |
| 30 | 2025 | J | 4228.0398684 |
| 31 | 2025 | K | 3997.9065321 |
| 32 | 2025 | L | 8517.17069415 |
| 33 | 2025 | M | 1737.32190795 |
| 34 | 2025 | N | 900.49730535 |
| 35 | 2025 | O | 5443.08669855 |
| 36 | 2025 | P | 3790.6834167 |
| 37 | 2025 | Q | 3133.52276895 |
| 38 | 2025 | R | 3412.37600415 |
| 39 | 2025 | S | 775.8896094 |
| 40 | 2025 | T | 96.33609045 |

შეგვიძლია სხვა წლებისთვისაც გავიანგარიშოთ პროგნოზი და გავაერთიანოთ მონაცემები, რისთვისაც მოთხოვნებს შორის ჩავსვამთ UNION ოპერატორს. აღსანიშნავია ისიც, რომ, ზოგადად, UNION ოპერატორი გაერთიანებისას, თუ ორი სტრიქონი აბსოლუტურად ემთხვევა ერთმანეთს, მაშინ მხოლოდ ერთ სტრიქონს იღებს, იმისათვის რომ ყველა სტრიქონი გამოიყენოს და რამდენჯერაც სტრიქონი მეორდება, იმდენჯერ ჩასვას შედეგში, მაშინ UNION-თან ვიყენებთ ALL-ს. მოთხოვნების გაერთიანებიდან მიღებული ინფორმაციის დახარისხება შესაძლებელია ORDER BY ბრძანებით.

გარდა გაერთიანებისა, SQL-ში მხარდაჭერილია თანაკვეთისა და სხვაობის ოპერატორებიც. აქედან, თანაკვეთის ოპერატორია INTERSECT, ხოლო სხვაობის - EXCEPT.

## 5.2. ცხრილებს შორის კავშირები

რელაციურ მონაცემთა ბაზებში მეტად მნიშვნელოვანი საკითხია ცხრილებს შორის კავშირები, რომლებსაც მსუბუქად შევეხეთ შესავალში რელაციური ალგებრის საფუძველზე. ზოგადად, ცხრილების დაკავშირების სხვადასხვა მეთოდი და სახე არსებობს. კავშირების საფუძველზე შეგვიძლია განვახორციელოთ სხვადასხვა ოპერაცია ცხრილებში. **ზოგჯერ კონკრეტული სტატისტიკური თუ სხვა ტიპის მონაცემის გამოსათვლე-**

**ლად არსებული მონაცემები მოთავსებულია სხვადასხვა ცხრილში. შედეგად, გამოსათვლელად საჭიროა ცხრილების დაკავშირება და უკვე დაკავშირებული ცხრილებიდან გამომდინარე, შესაბამისი ოპერაციის განხორციელება.**

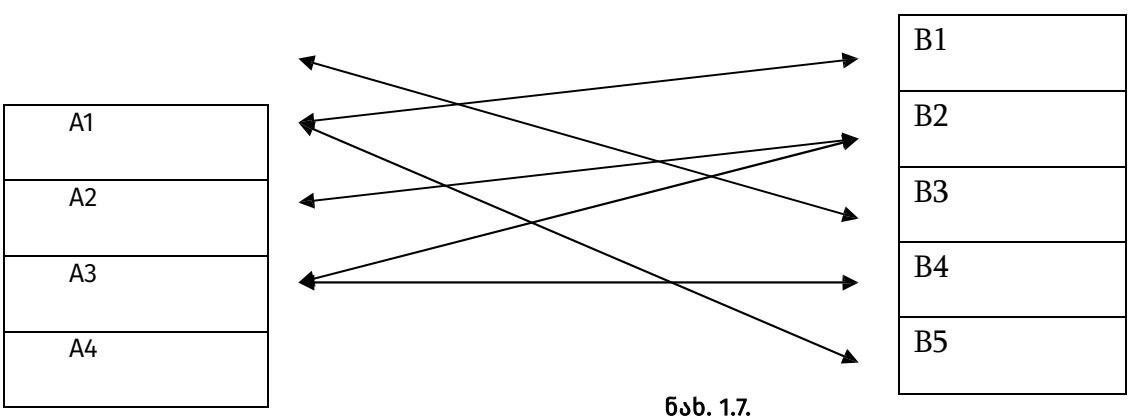
ცხრილებსა და მათ სტრიქონებს შორის ურთიერთკავშირი შეიძლება იყოს „ერთი-ბევრთან“ ან „ბევრი-ბევრთან“.

თუ პირველი (მთავარი) ცხრილის კონკრეტული სტრიქონი დროის ნებისმიერ მომენტში დაკავშირებულია მეორე (დამოკიდებული) ცხრილის ნულ, ერთ ან მეტ სტრიქონთან და მეორე ცხრილის ნებისმიერი სტრიქონი დაკავშირებულია პირველი ცხრილის მხოლოდ ერთ სტრიქონთან, მაშინ მათ შორის დამყარებულია „ერთი-ბევრთან“ კავშირი.

ასეთი კავშირის მაგალითია კავშირი რომელიმე ორგანიზაციის ან ფირმის თანამშრომლებსა და განყოფილებებს შორის. დროის ნებისმიერ მომენტში თითოეულ განყოფილებაში მუშაობს რამდენიმე თანამშრომელი და თითოეული თანამშრომელი მხოლოდ ერთ განყოფილებაში მუშაობს. ამიტომ, განყოფილებების შემცველი ცხრილი იქნება მთავარი, რომელსაც დაუკავშირდება თანამშრომლების ცხრილი. შედეგად, განყოფილებების ცხრილის თითოეულ სტრიქონს თანამშრომლების ცხრილის რამდენიმე სტრიქონი შეესაბამება და თანამშრომლების ცხრილის თითოეულ სტრიქონს კი - განყოფილებების ცხრილის მხოლოდ ერთი სტრიქონი. თუ პირველი (მთავარი) ცხრილის კონკრეტული სტრიქონი დროის ნებისმიერ მომენტში დაკავშირებულია მეორე (დამოკიდებული) ცხრილის ნულ, ერთ ან მეტ სტრიქონთან და, პირიქით, მაშინ მათ შორის დამყარებულია „ბევრი-ბევრთან“ კავშირი (ნახ.1.7). ასეთი კავშირის მაგალითებია:

ა) კავშირი ორგანიზაციის ან ფირმის თანამშრომლებსა და პროექტს შორის: დროის ნებისმიერ მომენტში ერთ პროექტზე შეიძლება ფირმის რამდენიმე თანამშრომელი მუშაობდეს და, პირიქით, ერთი თანამშრომელი შეიძლება რამდენიმე პროექტზე მუშაობდეს;

ბ) კავშირი წიგნის ავტორებსა და წიგნებს შორის: ერთ წიგნს შეიძლება რამდენიმე ავტორი ჰყავდეს და, პირიქით, ერთ ავტორს შეიძლება რამდენიმე წიგნი ჰქონდეს [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 33].




**ნახ. 1.7.**

თავდაპირველად განვიხილოთ მარტივი მაგალითი, რომელშიც გამოვიყენებთ ცხრილების დაკავშირებას. ჩვენ მონაცემთა ბაზაში statistika გვაქვს ორი ცხრილი: ერთი, რომელიც შეიცავს მონაცემებს და მოიცავს სექციის მიხედვით საბაზისო ფასებში მშპ-ს (ცხრილი mshp_seqcia), ხოლო მეორე მოიცავს კლასიფიკატორს (ცხრილი nace2). თითოეულ სექციას გააჩნია თავისი დასახელება და აღნიშვნა შესაბამისი ასოთი. დავუშვათ, დასახულია ამოცანა, რომ ამოვიღოთ 2022 წლის მონაცემები mshp_seqcia ცხრილიდან ისე, რომ გარდა სექციის აღმნიშვნელი ასოსი, გააჩნდეს შესაბამისი დასახელებაც ქართულ ენაზე. დასახელება ქართულ ენაზე მოცემულია კლასიფიკატორის ცხრილში. აქედან გამომდინარე, ეს ორი ცხრილი უნდა დაუკავშირდეს რომელიმე სვეტის საფუძველზე. ასეთი სვეტი ორივე ცხრილში ჩვენ შემთხვევაში არის seqcia, რომელიც მოიცავს კლასიფიკატორში არსებული სექციის შესაბამის აღნიშვნას. შესაბამისად, mshp_seqcia და nace2 ცხრილები ერთმანეთს უნდა დაუკავშირდეს seqcia სვეტების საფუძველზე და ორივე ცხრილიდან წამოიღონ შესაბამისი მონაცემები. ამისათვის გამოიყენება JOIN ბრძანება. მისი სინტაქსი შემდეგია:

```
SELECT a.sveti1, a.sveti2, b.sveti1, b.sveti2
FROM cxrili1 a
INNER JOIN cxrili2 b ON a.sveti1 = b.sveti1;
```

ამ შემთხვევაში მოვიძიებთ მონაცემებს cxrili1 და cxrili2-დან. სიმარტივისთვის cxrili1-ს ვარქმევთ a-ს, ხოლო cxrili2-ს - b-ს. შედეგში უნდა იყოს ოთხი სვეტი: აქედან cxrili1-დან sveti1 და sveti2, ხოლო cxrili2-დან sveti1 და sveti2. სვეტების სახელები და მონაცემები შეიძლება იყოს ერთმანეთის მსგავსი ან განსხვავებული. JOIN-ის წინ ჩავწერეთ INNER, რაც იმას ნიშნავს, რომ ამ ორი ცხრილიდან ამოღება მათ შორის თანაკვეთაა, ხოლო ON-ის შემდეგ „a.sveti1 = b.sveti1“ ნიშნავს, რომ cxrili1-ში მოთავსებული sveti1-ში არსებული მონაცემი უნდა იყოს ტოლი cxrili2-ში არსებული sveti1-ის მონაცემისა. ამ შემთხვევაში გამოვიყენეთ კავშირი მხოლოდ ერთი სვეტის საფუძველზე. თუ იქნებოდა ისეთი შემთხვევა, რომელიც რამდენიმე სვეტის ტოლობას მოითხოვდა, ამისთვის გამოვიყენებდით AND ოპერატორს კავშირებისთვის, მაგალითად „INNER JOIN cxrili2 b ON a.sveti1 = b.sveti1 AND a.sveti2 = b.sveti2“. ტოლობის გარდა კავშირი შეიძლება შედგეს სხვა ოპერატორებითაც, ასევე სვეტის მაგივრად შეიძლება გამოვიყენოთ გამოსახულებაც. შესაძლებელია, აგრეთვე, ორზე მეტი ცხრილის დაკავშირებაც. ამისათვის შემოვიტანთ დაკავშირების შესაბამისი რაოდენობის პირობებს მოთხოვნაში. დავუბრუნდეთ ჩვენ მაგალითს mshp_seqcia-დან მონაცემების ამოღების შესახებ. კავშირის სინტაქსის გამოყენებით მიღებული ბრძანება იქნება:

```
SELECT [weli], a.[seqcia], b.seqcia_dasaxeleba, [mshp]
FROM [mshp_seqcia] a
INNER JOIN nace2 b ON a.seqcia = b.seqcia
WHERE weli = 2022;
```

მოთხოვნის შედეგი მოცემულია სურათზე 5.2.

სურათი 5.2. mshp_seqcia და nace2 ცხრილების დაკავშირება

| | weli | seqcia | seqcia_dasaxeleba | mshp |
|---|---|---|---|---|
| 1 | 2022 | A | სოფლის, სატყეო და თევზის მეურნეობა | 4313.153279 |
| 2 | 2022 | B | სამთომოპოვებითი მრეწველობა | 890.680132 |
| 3 | 2022 | C | დამამუშავებელი მრეწველობა | 7078.223959 |
| 4 | 2022 | D | ელექტროენერგიის, აირის, ორთქლის და კონდიციირებული ჰა... | 2010.626158 |
| 5 | 2022 | E | წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბ... | 427.997953 |
| 6 | 2022 | F | მშენებლობა | 5026.424601 |
| 7 | 2022 | G | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოც... | 9990.823914 |
| 8 | 2022 | H | ტრანსპორტი და დასაწყობება | 4044.070652 |
| 9 | 2022 | I | განთავსების საშუალებებით უზრუნველყოფის და საკვების ... | 2329.248793 |
| 10 | 2022 | J | ინფორმაცია და კომუნიკაცია | 3131.881384 |
| 11 | 2022 | K | საფინანსო და სადაზღვევო საქმიანობები | 2961.412246 |
| 12 | 2022 | L | უძრავ ქონებასთან დაკავშირებული საქმიანობები | 6309.015329 |
| 13 | 2022 | M | პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები | 1286.905117 |
| 14 | 2022 | N | ადმინისტრაციული და დამხმარე მომსახურების საქმიანობე... | 667.035041 |
| 15 | 2022 | O | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო ს... | 4031.916073 |
| 16 | 2022 | P | განათლება | 2807.913642 |
| 17 | 2022 | Q | ჯანდაცვა და სოციალური მომსახურების საქმიანობები | 2321.127977 |
| 18 | 2022 | R | ხელოვნება, გართობა და დასვენება | 2527.685929 |
| 19 | 2022 | S | სხვა სახის მომსახურება | 574.733044 |
| 20 | 2022 | T | შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები; ... | 71.360067 |

ჩვენ განვიხილეთ კავშირის სახე, რომელიც იყო თანაკვეთა და გამოვიყენეთ INNER JOIN ბრძანება. არსებობს კავშირის სხვა სახეებიც. მიზანშეწონილია პრაქტიკაში მათი გამოყენება სხვადასხვა შემთხვევაში. ჩვენ მაგალითში მარცხენა ცხრილი იყო mshp_seqcia, ხოლო მარჯვენა - nace2. თუ მარჯვენა ცხრილში არ იქნებოდა მონაცემი, ხოლო მარცხენაში იქნებოდა, მაშინ თანაკვეთის შემთხვევაში არსებული მონაცემი შედეგში არ აღმოჩნდებოდა. ამ პრობლემის მოსაგვარებლად გამოვიყენებდით კავშირის სხვა სახეს, კერძოდ, მარცხენა კავშირს LEFT JOIN ბრძანებას. შესაბამისად, შედეგს მივიღებდით მოთხოვნის გაშვების შემდეგ, მაგრამ seqcia_dasaxeleba სვეტში იქნებოდა ცარიელი მნიშვნელობა NULL. ანალოგიურად, თუ მარცხენა ცხრილში არ იქნებოდა მონაცემი და მაინც დაგვჭირდებოდა შედეგში, გამოვიყენებდით მარჯვენა კავშირს RIGHT JOIN ბრძანებას და შესაბამის მარცხენა სვეტში იქნებოდა NULL მნიშვნელობა. ხოლო თუ დაგვჭირდებოდა მოთხოვნა, რომელიც ამოიღებდა ისეთ მონაცემებს, რომლის სვეტში ჩანაწერები შეიძლება ერთდროულად ყოფილიყო ორივე ცხრილში ან ერთ-ერთში, გამოვიყენებდით FULL JOIN ბრძანებას.

SQL-ში კავშირების ერთ-ერთი ძლიერი მხარეა მოთხოვნების დაკავშირებაც. ჩვენ შეგვიძლია არა მარტო ცხრილები, არამედ მოთხოვნებიც დავაკავშიროთ. ამის პრაქტიკული მაგალითი შეიძლება იყოს თვითდაკავშირება. ამ შემთხვევაში თვითდაკავშირებას განვიხილავთ თითოეულ სექციაში საბაზისო ფასებში მშპ-ის ზრდის მაგალითზე. დავუშვათ, გვაინტერესებს რამდენი პროცენტით გაიზარდა მშპ საბაზისო ფასებში თითოეული სექციისთვის, 2022 წელს 2021 წელთან შედარებით. ამისათვის ჯერ გვჭირდება შესაბამისი ფორმულა. თითოეული სექციისთვის პროცენტული ზრდა ტოლი იქნება:

$$k = \left(\frac{Y_{2022}}{Y_{2021}} - 1\right) * 100$$

სადაც $k$ არის პროცენტული ზრდა, $Y_{2022}$ - შესაბამის სექციაში არსებული მშპ საბაზისო ფასებში 2022 წლისათვის, ხოლო $Y_{2021}$ - შესაბამის სექციაში არსებული ანალოგიური მაჩვენებელი 2021 წლისათვის. იმისათვის, რომ გამოვთვალოთ სასურველი მონაცემი, ჯერ საჭიროა დავწეროთ მოთხოვნა, რომელიც ამოიღებს ცხრილიდან შესაბამის მონაცემებს. 2021 წლისათვის ეს მოთხოვნა იქნება:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2021;
```

ხოლო 2022 წლისათვის კი მოთხოვნა ასე გამოიყურება:

```
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2022;
```

ახლა საჭიროა ეს ორი მოთხოვნა ერთმანეთს დაუკავშირდეს რაიმე სვეტით, რომელშიც იქნება საერთო მონაცემი. ჩვენ შემთხვევაში ასეთია seqcia. დასაკავშირებლად გამოვიყენებთ ჩადგმულ მოთხოვნებს, დავაკავშირებთ INNER JOIN კავშირით და ამოვიღებთ მონაცემებს. ჯერ ამოვიღოთ კავშირის შედეგად მიღებული ყველა სვეტი. ამისათვის საჭირო მოთხოვნა იქნება:

```
SELECT *
FROM
(
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2021
) a
INNER JOIN(
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2022) b ON a.seqcia = b.seqcia
```

ამ მოთხოვნის შედეგი გამოსახულია სურათზე 5.3.

**სურათი 5.3. მოთხოვნების კავშირის შედეგები**

| | weli | seqcia | mshp | weli | seqcia | mshp |
|---|---|---|---|---|---|---|
| 1 | 2021 | A | 3880.01202 | 2022 | A | 4313.153279 |
| 2 | 2021 | B | 819.616668 | 2022 | B | 890.680132 |
| 3 | 2021 | C | 5921.596189 | 2022 | C | 7078.223959 |
| 4 | 2021 | D | 1684.638979 | 2022 | D | 2010.626158 |
| 5 | 2021 | E | 465.761124 | 2022 | E | 427.997953 |
| 6 | 2021 | F | 3929.947494 | 2022 | F | 5026.424601 |
| 7 | 2021 | G | 8085.252991 | 2022 | G | 9990.823914 |
| 8 | 2021 | H | 3304.976646 | 2022 | H | 4044.070652 |
| 9 | 2021 | I | 1783.647591 | 2022 | I | 2329.248793 |
| 10 | 2021 | J | 1828.450353 | 2022 | J | 3131.881384 |
| 11 | 2021 | K | 2700.719619 | 2022 | K | 2961.412246 |
| 12 | 2021 | L | 5324.400032 | 2022 | L | 6309.015329 |
| 13 | 2021 | M | 1160.100363 | 2022 | M | 1286.905117 |
| 14 | 2021 | N | 490.289962 | 2022 | N | 667.035041 |
| 15 | 2021 | O | 3416.085583 | 2022 | O | 4031.916073 |
| 16 | 2021 | P | 2454.155815 | 2022 | P | 2807.913642 |
| 17 | 2021 | Q | 2527.473511 | 2022 | Q | 2321.127977 |
| 18 | 2021 | R | 2162.10567 | 2022 | R | 2527.685929 |
| 19 | 2021 | S | 425.625926 | 2022 | S | 574.733044 |
| 20 | 2021 | T | 47.518895 | 2022 | T | 71.360067 |

როგორც სურათზე 5.3 გამოსახული შედეგებიდან დავრწმუნდით, თითოეული სტრიქონი გვიჩვენებს ერთსა და იმავე სექციას, მაგრამ განსხვავებულია წელი და ამ შემთხვევაში, შესაბამისად, ჩაწერილი მონაცემიც. ახლა საჭიროა სათანადო გამოთვლების განხორციელება. ამისათვის გამოვიტანთ სექციისა და 2022 წლის პროცენტული ზრდის სვეტს. შედეგად მოთხოვნა იქნება ასეთი სახის:

```
SELECT a.seqcia [სექცია],
(b.mshp / a.mshp - 1) * 100 [პროცენტული ზრდა 2022 წელს]
FROM
(
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2021
) a
INNER JOIN(
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2022) b ON a.seqcia = b.seqcia
```

შედეგი გამოსახულია სურათზე 5.4.

**სურათი 5.4. თითოეული სექციის საბაზისო ფასებში გამოსახული მშპ-ის პროცენტული ზრდა 2022 წლისათვის**

| | სექცია | პროცენტული ზრდა 2022 წელს |
|---|---|---|
| 1 | A | 11.1633999267868 |
| 2 | B | 8.67032928617795 |
| 3 | C | 19.5323648064446 |
| 4 | D | 19.3505660894482 |
| 5 | E | -8.10784091975869 |
| 6 | F | 27.90055359961 |
| 7 | G | 23.5684761518429 |
| 8 | H | 22.3630628946962 |
| 9 | I | 30.5890695422693 |
| 10 | J | 71.2861045891356 |
| 11 | K | 9.6527097876427 |
| 12 | L | 18.4925116648335 |
| 13 | M | 10.93049860549 |
| 14 | N | 36.0490919045167 |
| 15 | O | 18.0273729986348 |
| 16 | P | 14.4146441247863 |
| 17 | Q | -8.16410273349845 |
| 18 | R | 16.9085287584487 |
| 19 | S | 35.0324331511704 |
| 20 | T | 50.1719831658543 |

დავუშვათ, რომ გვჭირდება სექციის დასახელება სექციის აღმნიშვნელი ასოს მაგივრად. ამისათვის გამოვიყენებთ კიდევ ერთ დაკავშირებას და დავაკავშირებთ nace2 ცხრილთან. ამ შემთხვევაში შესაბამისი კოდი იქნება:

```
SELECT c.seqcia_dasaxeleba [სექცია],
(b.mshp / a.mshp - 1) * 100 [პროცენტული ზრდა 2022 წელს]
FROM
(
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2021
) a
INNER JOIN(
SELECT [weli], [seqcia], [mshp]
FROM [mshp_seqcia]
WHERE [weli] = 2022) b ON a.seqcia = b.seqcia
INNER JOIN nace2 c ON b.seqcia = c.seqcia
```

შედეგი გამოსახულია სურათზე 5.5.

**სურათი 5.5. შედეგების დაკავშირება სექციის დასახელებასთან**

| | სექცია | პროცენტული ზრდა 2022 წელს |
|---|---|---|
| 1 | სოფლის, სატყეო და თევზის მეურნეობა | 11.1633999267868 |
| 2 | სამთომოპოვებითი მრეწველობა | 8.67032928617795 |
| 3 | დამამუშავებელი მრეწველობა | 19.5323648064446 |
| 4 | ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰა... | 19.3505660894482 |
| 5 | წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბ... | -8.10784091975869 |
| 6 | მშენებლობა | 27.90055359961 |
| 7 | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოც... | 23.5684761518429 |
| 8 | ტრანსპორტი და დასაწყობება | 22.3630628946962 |
| 9 | განთავსების სამუალებებით უზრუნველყოფის და საკვების ... | 30.5890695422693 |
| 10 | ინფორმაცია და კომუნიკაცია | 71.2861045891356 |
| 11 | საფინანსო და სადაზღვევო საქმიანობები | 9.6527097876427 |
| 12 | უძრავ ქონებასთან დაკავშირებული საქმიანობები | 18.4925116648335 |
| 13 | პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები | 10.93049860549 |
| 14 | ადმინისტრაციული და დამხმარე მომსახურების საქმიანობე... | 36.0490919045167 |
| 15 | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო ს... | 18.0273729986348 |
| 16 | განათლება | 14.4146441247863 |
| 17 | ჯანდაცვა და სოციალური მომსახურების საქმიანობები | -8.16410273349845 |
| 18 | ხელოვნება, გართობა და დასვენება | 16.9085287584487 |
| 19 | სხვა სახის მომსახურება | 35.0324331511704 |
| 20 | შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები; ... | 50.1719831658543 |

მონაცემთა ბაზაში კავშირების საშუალებით შესაძლებელია სხვადასხვა სტატისტიკური ინდექსის გაანგარიშება.

არის შემთხვევები, როდესაც კონკრეტულ პერიოდში მონაცემების უფრო მეტად აგრეგირებული ვარიანტები გროვდებოდა და, შესაბამისად, არ არსებობდა წინა პერიოდის მონაცემი, რომლითაც უნდა გამოთვლილიყო შესაბამისი ინდექსი. ან შეიძლება იყოს შემთხვევა, როდესაც შეიცვალა კლასიფიკატორი და, შესაბამისად, კონკრეტული კოდი ახალ კლასიფიკატორში განსხვავდება ძველი კლასიფიკატორისგან. ასეთ შემთხვევებში შეგვიძლია გამოვიყენოთ გადამყვანი ცხრილები, რომლებსაც შესაბამისად დავუკავშირებთ მონაცემებს, ან მარტივად გამოვიყენოთ CASE სინტაქსი მოთხოვნაში, მაგალითად, შემოვიღოთ ჰიპოთეტური კლასიფიკატორი, რომლის პირველ ვერსიაში კონკრეტული სექცია აღინიშნება „A1“-ით, ხოლო მეორე ვერსიაში ეს სექცია აღინიშნება „A2“-ით. ამ შემთხვევაში ინდექსის გასაანგარიშებლად გამოვიყენებთ ჩადგმულ მოთხოვნას, რომელშიც კონკრეტული სვეტის მაგივრად იქნება CASE სინტაქსი:

```
CASE
        WHEN seqcia = 'A1' THEN 'A2'
        ELSE seqcia
END seqcia
```

სტატისტიკური ანალიზისას ხშირად საჭიროა აგრეგირებული მონაცემების გამოყენება. წინა თავში განვიხილეთ მაღალი დონის აგრეგირება ეროვნულ ანგარიშთა სისტემაში NACE Rev.2 კლასიფიკატორით და გამოვიყენეთ CASE პირობები, თუმცა, ამის სხვა ვარიანტიც არსებობს. ამისათვის შეგვიძლია შევქმნათ ცხრილი, რომელიც დაგვეხმარება აგრეგირებაში. NACE Rev.2 კლასიფიკატორისთვის მონაცემთა ბაზაში შევიტანოთ 5.1 ცხრილი სახელით nace2_agregireba.

**ცხრილი 5.1. NACE Rev.2 მაღალი დონის აგრეგირების ცხრილი**

| seqcia_aggregated | Seqcia |
|---|---|
| A | A |
| BCDE | B |
| BCDE | C |
| BCDE | D |
| BCDE | E |
| F | F |
| GHI | G |
| GHI | H |
| GHI | I |
| J | J |
| K | K |
| L | L |
| MN | M |
| MN | N |
| OPQ | O |
| OPQ | P |
| OPQ | Q |
| RSTU | R |
| RSTU | S |
| RSTU | T |
| RSTU | U |

ახლა ვცადოთ ამ ცხრილის დაკავშირება mshp_seqcia ცხრილთან ორივე ცხრილში არსებული seqcia სვეტით და ამოვიღოთ 2022 წლის მონაცემები. ამისათვის შესაბამისი მოთხოვნა ასეთი იქნება:

```
SELECT weli, a.seqcia, b.seqcia_agregirebuli, a.mshp
FROM mshp_seqcia a
INNER JOIN nace2_agregireba b ON a.seqcia = b.seqcia
WHERE weli = 2022;
```

მოთხოვნის შედეგი გამოსახულია სურათზე 5.6.

**სურათი 5.6. აგრეგირების ცხრილთან დაკავშირება**

| | weli | seqcia | seqcia_agregirebuli | mshp |
|---|---|---|---|---|
| 1 | 2022 | A | A | 4313.153279 |
| 2 | 2022 | B | BCDE | 890.680132 |
| 3 | 2022 | C | BCDE | 7078.223959 |
| 4 | 2022 | D | BCDE | 2010.626158 |
| 5 | 2022 | E | BCDE | 427.997953 |
| 6 | 2022 | F | F | 5026.424601 |
| 7 | 2022 | G | GHI | 9990.823914 |
| 8 | 2022 | H | GHI | 4044.070652 |
| 9 | 2022 | I | GHI | 2329.248793 |
| 10 | 2022 | J | J | 3131.881384 |
| 11 | 2022 | K | K | 2961.412246 |
| 12 | 2022 | L | L | 6309.015329 |
| 13 | 2022 | M | MN | 1286.905117 |
| 14 | 2022 | N | MN | 667.035041 |
| 15 | 2022 | O | OPQ | 4031.916073 |
| 16 | 2022 | P | OPQ | 2807.913642 |
| 17 | 2022 | Q | OPQ | 2321.127977 |
| 18 | 2022 | R | RSTU | 2527.685929 |
| 19 | 2022 | S | RSTU | 574.733044 |
| 20 | 2022 | T | RSTU | 71.360067 |

ახლა შევაჯამოთ mshp სვეტში არსებული მონაცემები seqcia_agregirebuli სვეტის საფუძველზე 2022 წლისათვის. ამისათვის შესაბამისი კოდი იქნება:

```
SELECT weli, b.seqcia_agregirebuli, SUM(mshp) mshp
FROM mshp_seqcia a
INNER JOIN nace2_agregireba b ON a.seqcia = b.seqcia
WHERE weli = 2022
GROUP BY weli, seqcia_agregirebuli;
```

შედეგი ნაჩვენებია 5.7 სურათზე.

**სურათი 5.7. მაღალი დონის აგრეგირების შედეგი აგრეგირების ცხრილის გამოყენებით**

| | weli | seqcia_agregirebuli | mshp |
|---|---|---|---|
| 1 | 2022 | A | 4313.153279 |
| 2 | 2022 | BCDE | 10407.528202 |
| 3 | 2022 | F | 5026.424601 |
| 4 | 2022 | GHI | 16364.143359 |
| 5 | 2022 | J | 3131.881384 |
| 6 | 2022 | K | 2961.412246 |
| 7 | 2022 | L | 6309.015329 |
| 8 | 2022 | MN | 1953.940158 |
| 9 | 2022 | OPQ | 9160.957692 |
| 10 | 2022 | RSTU | 3173.77904 |

კავშირების დახმარებით შეგვიძლია სხვა რთული მათემატიკური გამოთვლების განხორციელებაც, მაგალითად, ეს შეიძლება იყოს პირსონის კორელაციის კოეფიციენტი. დავუშვათ, გვაქვს ორი ცხრილი: tansacmeli და fexsacmeli და ამოვიღეთ ყოველთვიური სამომხმარებლო ფასების ინდექსები ფეხსაცმლისა და ტანსაცმლის მიხედვით 2023 წლისათვის საქართველოს სტატისტიკის ეროვნული სამსახურის ვებგვერდიდან: https://www.geostat.ge/ka/modules/categories/26/samomkhmareblo-fasebis-indeksi-inflatsia

ცხრილების სვეტებია weli, tveდა indeqsi და მოიცავს, შესაბამისად, წლის, თვისა და ინდექსის მონაცემს. დავაკავშიროთ ეს ცხრილები და ამოვიღოთ წელი, თვე და შესაბამისი ინდექსები. ამისათვის გამოვიყენებთ ასეთ ბრძანებას:

```
SELECT a.weli, a.tve, a.indeqsitansacmeli, b.indeqsifexsacmeli
FROM tansacmeli a
INNER JOIN fexsacmeli b ON a.weli = b.weli AND a.tve = b.tve;
```

ბრძანების შედეგად მივიღებთ ისეთ ცხრილს, როგორიც გამოსახულია სურათზე 5.8.

**სურათი 5.8. ტანსაცმლისა და ფეხსაცმლის ფასების ინდექსები**

| | weli | tve | tansacmeli | fexsacmeli |
|---|---|---|---|---|
| 1 | 2023 | 1 | 97.5047 | 98.3927 |
| 2 | 2023 | 2 | 99.4122 | 97.2867 |
| 3 | 2023 | 3 | 101.1005 | 103.6106 |
| 4 | 2023 | 4 | 102.7119 | 100.3737 |
| 5 | 2023 | 5 | 99.5126 | 99.994 |
| 6 | 2023 | 6 | 100.0586 | 100.5596 |
| 7 | 2023 | 7 | 98.7055 | 96.1126 |
| 8 | 2023 | 8 | 98.1799 | 97.6398 |
| 9 | 2023 | 9 | 99.9829 | 102.6435 |
| 10 | 2023 | 10 | 101.9483 | 107.0344 |
| 11 | 2023 | 11 | 100.8413 | 102.5213 |
| 12 | 2023 | 12 | 100.4816 | 94.6756 |

ამის შემდეგ გავიანგარიშოთ კორელაციის კოეფიციენტი მიღებული tansacmeli და fexsacmeli სვეტში არსებულ მონაცემებს შორის. შედეგად ბრძანება მიიღებს შემდეგ სახეს:

```
WITH CTE AS (
SELECT a.weli, a.tve, a.indeqsitansacmeli, b.indeqsifexsacmeli
FROM tansacmeli a
INNER JOIN fexsacmeli b ON a.weli = b.weli AND a.tve = b.tve
)
SELECT
(AVG(tansacmeli * fexsacmeli) - AVG(tansacmeli) * AVG(fexsacmeli))
/ (STDEVP(tansacmeli) * STDEVP (fexsacmeli)) Correlation
FROM CTE;
```

გავუშვათ ბრძანება. შედეგი გამოსახულია სურათზე 5.9.

**სურათი 5.9. გამოთვლილი კორელაცია ტანსაცმლისა და ფეხსაცმლის სამომხმარებლო ფასების ინდექსებს შორის (2023 წლისათვის)**


როგორც ვნახეთ, SQL Server-ის საშუალებით შეგვიძლია გავიანგარიშოთ სტატისტიკური მაჩვენებლები იმ შემთხვევაშიც, თუ ეს მონაცემები მოთავსებულია სხვადასხვა ცხრილში.

## 5.3. ნორმალიზაციის თეორია

მონაცემებთა ბაზებთან მუშაობაში ერთ-ერთი მნიშვნელოვანი საკითხია ნორმალიზაციის თეორია. ნორმალიზაცია წარმოადგენს მონაცემთა ბაზაში მონაცემების ორგანიზაციის პროცესს. ამ დროს ხდება მონაცემთა შორის ინტეგრაციის გაუმჯობესება და გამეორებების შემცირება. როდესაც ხდება მონაცემების არასაჭიროებისამებრ გამეორება, ამით ზედმეტად ივსება სივრცე და, შესაბამისად, ჩნდება სივრცის არაეფექტიანი მოხმარება. ამასთან ერთად, ასეთ არაეფექტიანობებთან მუშაობის დროს ხშირია დამატებითი დროითი დანახარჯებიც, რადგან, თუ მონაცემები მეორდება, მაშინ მონაცემთა ცვლილებისას საჭიროა დამატებით ყველგან შეიცვალოს მონაცემები. ამასთან ერთად, პრობლემას წარმოადგენს არათანმიმდევრული დამოკიდებულებებიც ცხრილებს შორის, რომელსაც შეუძლია შექმნას ხარვეზები მონაცემებთან დაკავშირების დროს.

ცხრილების ნორმალიზება არის მონაცემების წარმოდგენის პროცესი მარტივი ორგანზომილებიანი ცხრილებით, რაც საშუალებას გვაძლევს თავიდან ავიცილოთ მონაცემების დუბლირება და უზრუნველვყოთ ბაზაში შენახული მონაცემების არაწინააღმდეგობრიობა. ნორმალიზების მიზანია მონაცემთა ბაზის ისეთი პროექტის მიღება, რომელშიც ინფორმაციის ნებისმიერი ნაწილი შენახული იქნება მხოლოდ ერთ ადგილზე, ე. ი. გამორიცხული იქნება ინფორმაციის სიჭარბე. ეს კეთდება არა მარტო ადგილის ეკონომიის, არამედ შენახული მონაცემების წინააღმდეგობრიობის გამორიცხვის მიზნით.

ცხრილი ითვლება ნორმალიზებულად გარკვეულ დონეზე, როცა ის აკმაყოფილებს ნორმალიზების შესაბამისი ფორმის მოთხოვნებს. ნორმალიზების პროცესი წარმოადგენს ცხრილის სტრუქტურის მიმდევრობით ცვლილებას მანამ, სანამ ის არ დააკმაყოფილებს ნორმალიზების რომელიმე ფორმის მოთხოვნებს [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 23].

არსებობს მონაცემთა ბაზის ნორმალიზაციის წესები. თუ ეს წესები შესრულდა მონაცემთა ბაზების არქიტექტურის დროს, მაშინ გამოჩნდება შესაბამისი ნორმალური ფორმა. არსებობს სხვადასხვა ნორმალური ფორმა. თუმცა, ზოგადად, განიხილავენ ექვს ნორმალურ ფორმას. ესენია:

1. პირველი ნორმალური ფორმა (1NF);
2. მეორე ნორმალური ფორმა (2NF);
3. მესამე ნორმალური ფორმა (3NF);
4. ბოის-კოდის ნორმალური ფორმა (BCNF);
5. მეოთხე ნორმალური ფორმა (4NF);
6. მეხუთე ნორმალური ფორმა (5NF).

**პირველი ნორმალური ფორმა** უზრუნველყოფს, რომ ცხრილში თითოეული სვეტი შეიცავდეს ატომურ მნიშვნელობებს. ამ დროს აუცილებელია, ცხრილის კონკრეტულ უჯრაში მონაცემები ისე იყოს განთავსებული, რომ შეიცავდეს მხოლოდ ერთ მნიშვნელობას და არ საჭიროებდეს დამატებით დაყოფას. მაგალითად, თუ ჩვენს მშპ-ის ცხრილში, მშპ სვეტში გამოყოფილი იქნება მძიმეებით მშპ-ის შესაბამისი მნიშვნელობები და არა ცალ-ცალკე, თითოეული წლისათვის შესაბამის სტრიქონში, მაშინ ასეთ მონაცემებთან მუშაობა იქნებოდა უფრო მეტად რთული. ეს პრობლემა რომ შეგვქმნოდა, ოპტიმიზაციისთვის დავყოფდით ამ მონაცემებს და მოვათავსებდით ცალკე სტრიქონებად. ამ შემთხვევაში ოპტიმიზაციის განხორციელებისას დავაკმაყოფილებდით პირველ ნორმალურ ფორმას.

**მეორე ნორმალური ფორმა** აკმაყოფილებს პირველი ნორმალური ფორმის მოთხოვნებს და დამატებით მოითხოვს, რომ არ არსებობდეს ნაწილობრივი დამოკიდებულებები პირველად გასაღებზე. აქედან გამომდინარე, თითოეული არაგასაღებური ატრიბუტი ფუნქციონალურად უნდა იყოს დამოკიდებული პირველად გასაღებზე. უფრო მარტივად, თითოეული სვეტი, რომელიც არ არის პირველადი გასაღები, დამოკიდებული უნდა იყოს პირველად გასაღებზე.

**მესამე ნორმალური ფორმა** აკმაყოფილებს მეორე ნორმალური ფორმის მოთხოვნებს და დამატებით მოითხოვს ტრანზიტული დამოკიდებულებების მოსპობას. ტრანზიტული დამოკიდებულების მოსპობისთვის ბაზაში უნდა მოხდეს ცვლილებები ისე, რომ არაგასაღებური ატრიბუტი არ იყოს დამოკიდებული სხვა არაგასაღებურ ატრიბუტზე. თუ მოხდა ისე, რომ არაგასაღებური ატრიბუტი დამოკიდებულია სხვა არაგასაღებურ ატრიბუტზე, რომელიც დამოკიდებულია გასაღებურ ატრიბუტზე, მაშინ სახეზე გვაქვს ტრანზიტული დამოკიდებულება.

**ბოის-კოდის ნორმალური ფორმა** უნდა აკმაყოფილებდეს მესამე ნორმალური ფორმის მოთხოვნებს, მაგრამ, ამავე დროს, ცხრილში ყველა დეტერმინანტი (ატრიბუტთა სიმრავლე, რომლითაც ხდება სტრიქონის უნიკალურად იდენტიფიცირება), უნდა იყოს კანდიდატური გასაღები (ატრიბუტთა მინიმალური სიმრავლე, რომელიც უნიკალურად აიდენტიფიცირებს სტრიქონს). აქედან გამომდინარე, შესაძლებელი ხდება, რომ ყველა დეტერმინანტი ასრულდებდეს პირველადი გასაღების ფუნქციას.

**მეოთხე ნორმალური ფორმა** აკმაყოფილებს ბოის-კოდის ნორმალური ფორმის მოთხოვნებს და, ასევე, დამატებით მოითხოვს მრავალი მრავალმნიშვნელობიანი დამოკიდებულებების მოსპობას. ნორმალიზაციის ეს ფორმა გამოიყენება იმ შემთხვევაში, როდესაც ერთ ატრიბუტს შეიძლება გააჩნდეს მრავალი დამოკიდებული ატრიბუტი, რომელიც პირდაპირ არ უკავშირდება პირველად გასაღებს.

**მეხუთე ნორმალური ფორმა** აკმაყოფილებს მეოთხე ნორმალური ფორმის მოთხოვნებს და დამატებით მოითხოვს, რომ თუ დაცულია მეოთხე ნორმალური ფორმის მოთხოვნები, მაშინ ცხრილების გაერთიანებისას კანდიდატი გასაღებებით არ უნდა დაიკარგოს ორგიგინალური ჩანაწერები და არც ახალი უნდა შეიქმნას.

როგორც უკვე ავღნიშნეთ, გარკვეულწილად, ნორმალიზაცია გვეხმარება ეფექტიანობის მიღწევაში, მაგრამ აქვე უნდა გავითვალისწინოთ ისიც, რომ ზოგიერთ შემთხვევაში ნორმალიზაციის უფრო მაღალი ფორმების შესრულებამ შესაძლოა გამოიწვიოს დროითი დანაკარგები მონაცემებთან მუშაობაში იმ გაგებით, რომ შეიძლება საჭირო გახდეს დამატებითი გაერთიანებები ცხრილებთან მუშაობისას და ამან შეიძლება გაზარდოს მოთხოვნის შესრულების დრო. ამ მხრივ შესაძლოა გაჩნდეს ალტერნატივა სივრცით და მონაცემთა დამუშავებაში დროით დანაკარგებს შორის.

***კითხვები თვითშემოწმებისთვის:***

1. როგორ ხდება ცხრილების გაერთიანება SQL-ში?
2. აღწერეთ კავშირების სახეები ცხრილებს შორის.
3. როგორ ხდება ცხრილების დაკავშირება SQL-ში?
4. როგორ განვახორციელოთ მონაცემთა აგრეგირება აგრეგირების ცხრილების დახმარებით?
5. აღწერეთ რას ნიშნავს ნორმალიზაციის თეორია?
6. რომელ ნორმალურ ფორმებს იცნობთ?
7. ახსენით ნორმალიზაციის გამოყენებისას ალტერნატივა მოთხოვნის შესრულების დროით და სივრცით დანახარჯებს შორის.

# თავი 6. მონაცემთა წარმოდგენები და ინდექსირება ბაზებში

## 6.1. მონაცემთა წარმოდგენები ბაზებში

წარმოდგენები (ინგლ. VIEWS) მხარდაჭერილია SQL Server-ში. მონაცემთა წარმოდგენები შეიძლება შეიცავდეს ერთი ან რამდენიმე ცხრილიდან მონაცემებს, რომელიც მიიღება მოთხოვნის საფუძველზე. ხშირად წარმოდგენებს ვირტუალურ ცხრილებსაც უწოდებენ, მაგრამ წარმოდგენები ვირტუალური ცხრილების ერთ-ერთი სახეა. მონაცემთა ბაზაში წარმოდგენის სახით ინახება მოთხოვნა, რომლის გამოძახების შედეგად ამოიღება სხვადასხვა მონაცემი.

მაშასადამე, წარმოდგენა არის ვირტუალური ცხრილის ერთ-ერთი სახე და შედგება სვეტებისა და სტრიქონებისაგან, რომლებიც დინამიკურად ამოირჩევა ერთი ან მეტი ცხრილიდან და/ან წარმოდგენიდან. ფიზიკურად წარმოდგენას SELECT მოთხოვნის სახე აქვს, რომლის საფუძველზეც ხდება მონაცემების ამორჩევა.

წარმოდგენა ხშირად გამოიყენება ისეთი სვეტების დასამალად, რომელიც კონფიდენციალურ ინფორმაციას შეიცავს, როგორიცაა, მაგალითად, ხელფასი, ასაკი და ა. შ. წარმოდგენაში შეგვიძლია ისეთი სვეტების გამოჩენა, რომელთა ნახვაც ნებადართულია მომხმარებლებისთვის. თუ წარმოდგენაში ჩართული არ არის ცხრილის რომელიმე სვეტი, მაშინ ცხრილზე დადებულია ვერტიკალური ფილტრი. თუ მოთხოვნა შეიცავს სტრიქონების ამორჩევის პირობებს, მაშინ ცხრილზე დადებულია ჰორიზონტალური ფილტრი.

წარმოდგენა შეიძლება შეიცავდეს ბმული ცხრილების სვეტებს. მაგალითად, წარმოდგენა შეიძლება შეიცავდეს Personali ცხრილიდან თანამშრომლების გვარებს და Xelshekruleba ცხრილიდან მათთან გაფორმებული ხელშეკრულებების თანხებს, ან Personali ცხრილიდან თანამშრომლების გვარებს და Shemkveti ცხრილიდან მათთან ხელშეკრულების დამდები კლიენტების გვარებს და ა. შ.

წარმოდგენასთან მიმართვის დროს სერვერი ამოწმებს იმ ობიექტების არსებობას, რომლებიც საჭიროა წარმოდგენის განმსაზღვრელი SELECT მოთხოვნის შესასრულებლად. თუ მოთხოვნაში მითითებული რომელიმე ცხრილი წაშლილია, მაშინ წარმოდგენა ვერ იმუშავებს და გაიცემა შეტყობინება შეცდომის შესახებ. თუ წაშლილი ცხრილის ნაცვლად შევქმნით იმავე სახელისა და სტრუქტურის მქონე ცხრილს, მაშინ წარმოდგენა იმუშავებს. თუ ახალ ცხრილს განსხვავებული სტრუქტურა აქვს, მაშინ წარმოდგენა უნდა წავშალოთ და ხელახლა შევქმნათ. წარმოდგენას შეუძლია მონაცემების მიღება სხვა წარმოდგენებიდანაც. წარმოდგენა ყოველთვის იქმნება მიმდინარე მონაცემთა ბაზაში. განაწილებული მოთხოვნების გამოყენებით შეგვიძლია მივმართოთ მიმდინარე სერვერის სხვა მონაცემთა ბაზებში შექმნილ ცხრილებსა და წარმოდგენებს [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 199].

წინა თავში აგრეგირების ცხრილის დახმარებით ბაზიდან ამოვიღეთ მონაცემები. იგივეს თუ გავიმეორებთ ყველა წლისათვის, შესაბამისი მოთხოვნა იქნება:

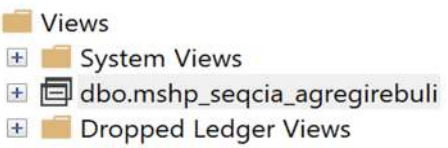
```
SELECT weli, b.seqcia_agregirebuli, SUM(mshp) mshp
FROM mshp_seqcia a
INNER JOIN nace2_agregireba b ON a.seqcia = b.seqcia
GROUP BY weli, seqcia_agregirebuli;
```

ჩვენ შეგვიძლია მონაცემთა ბაზაში შევინახოთ არსებული მოთხოვნა წარმოდგენის სახით, რომლიდანაც შემდეგში შეგვეძლება მონაცემების ამოღება. ამისათვის უნდა შევქმნათ წარმოდგენა და პირობითად დავარქვათ მას „mshp_seqcia_agregirebuli“. შესაბამისი მოთხოვნა იქნება:

```
CREATE VIEW [mshp_seqcia_agregirebuli] AS
SELECT weli, b.seqcia_agregirebuli, SUM(mshp) mshp
FROM mshp_seqcia a
INNER JOIN nace2_agregireba b ON a.seqcia = b.seqcia
GROUP BY weli, seqcia_agregirebuli;
```

თუ Object Explorer-ში გავხსნით Views საქაღალდეს, ვიმოქმედებთ მასზე მაუსის მარჯვენა ღილაკით და ავირჩევთ Refresh-ს კონტექსტური მენიუდან, მაშინ გამოჩნდება ჩვენი შექმნილი წარმოდგენა.

**სურათი 6.1. მონაცემთა წარმოდგენები**




წარმოდგენიდან მონაცემის ამოღებისათვის ვიქცევით ისე, როგორც ამას ცხრილების შემთხვევაში ვაკეთებთ. მაგალითისათვის ამოვიღოთ ყველა მონაცემი ჩვენ მიერ შექმნილი წარმოდგენიდან. ამისათვის შესაბამისი კოდი იქნება:

```
SELECT *
FROM [statistika].[dbo].[mshp_seqcia_agregirebuli]
```

აქვე შეგვიძლია გამოვიყენოთ WHERE სინტაქსი მონაცემთა ფილტრაციისთვის და, ასევე, T-SQL-ის სხვადასხვა ბრძანებაც.

წარმოდგენის შესაცვლელად გამოიყენება „CREATE OR ALTER VIEW“ სინტაქსი. თუ გვსურს წარმოდგენა ისე შევცვალოთ, რომ აჩვენებდეს მხოლოდ 2022 წლის მონაცემებს, მაშინ მის გასაახლებლად გამოვიყენებთ მოთხოვნას:

```
CREATE OR ALTER VIEW [mshp_seqcia_agregirebuli] AS
SELECT weli, b.seqcia_agregirebuli, SUM(mshp) mshp
FROM mshp_seqcia a
INNER JOIN nace2_agregireba b ON a.seqcia = b.seqcia
WHERE weli=2022
GROUP BY weli, seqcia_agregirebuli;
```

დავაბრუნოთ წარმოდგენა ისევ საწყის ვარიანტზე.

წარმოდგენის წასაშლელად გამოიყენება „DROP VIEW warmodgena1“, სადაც „warmodgena1“ არის იმ წარმოდგენის სახელი, რომლის წაშლაც გვინდა.

## 6.2. ალგორითმებისა და ალგორითმული სირთულის მოკლე მიმოხილვა

ალგორითმული სირთულე კომპიუტერული მეცნიერების ერთ-ერთ საკვანძო საკითხს წარმოადგენს. ძირითადად განიხილება ალგორითმული დროითი და სივრცითი სირთულეები. ალგორითმების შემუშავებისას საჭირო ხდება აღნიშნული სირთულეების გათვალისწინება, რათა შედგეს ოპტიმიზებული ალგორითმი. თუ ალგორითმი არ არის ოპტიმიზებული, მაშინ შეიძლება დაიკარგოს რესურსები, იქნება ეს დროითი, ფინანსური და სხვა. ამ ქვეთავში განვიხილავთ ძებნის ალგორითმებს და მაგალითის საფუძველზე გავარჩევთ აღნიშნულ საკითხს.

განვიხილოთ ძებნის ალგორითმები. პირველად განვახორციელოთ წრფივი ძებნა. დავუშვათ, გვაქვს მონაცემთა მასივი, რომელიც შედგება 10 მონაცემისგან და პირობითად ვუწოდოთ მას $a$.

$$a = \{1, \quad 3, \quad 2, \quad 7, \quad 9, \quad 10, \quad 6, \quad 4, \quad 5, \quad 8\}$$

დავუშვათ, გვსურს მოცემულ მასივში მოვძებნოთ რაიმე მონაცემი, მაგალითად, 4. წრფივი ძებნის შემთხვევაში ალგორითმი მიმდევრობით შეამოწმებს ყველა მონაცემს და იპოვის ინდექსს. შესაბამისი იტერაციები იქნება:

1. თუ 1 = 4, მაშინ ინდექსია 1, თუ არა, გაგრძელდეს ძებნა;
2. თუ 3 = 4, მაშინ ინდექსია 2, თუ არა, გაგრძელდეს ძებნა;
3. თუ 2 = 4, მაშინ ინდექსია 3, თუ არა, გაგრძელდეს ძებნა;
4. თუ 7 = 4, მაშინ ინდექსია 4, თუ არა, გაგრძელდეს ძებნა;
5. თუ 9 = 4, მაშინ ინდექსია 5, თუ არა, გაგრძელდეს ძებნა;
6. თუ 10 = 4, მაშინ ინდექსია 6, თუ არა, გაგრძელდეს ძებნა;
7. თუ 6 = 4, მაშინ ინდექსია 7, თუ არა, გაგრძელდეს ძებნა;
8. თუ 4 = 4, მაშინ ინდექსია 8, თუ არა, გაგრძელდეს ძებნა;
9. თუ 5 = 4, მაშინ ინდექსია 9, თუ არა, გაგრძელდეს ძებნა;
10. თუ 8 = 4, მაშინ ინდექსია 10, თუ არა, გაგრძელდეს ძებნა;

შესაბამისად, ალგორითმი მოიცემა შემდეგი ფსევდოკოდით (Knuth, 1988):

1. i<-1; j<-10; index = 0;
2. თუ $a_i$= 4, მაშინ index <- i, ციკლი გაჩერდეს, წინააღმდეგ შემთხვევაში გაგრძელდეს;

3. i<- i + 1;
4. თუ i< j, დავბრუნდეთ მეორე ნაბიჯზე, თუ არა ციკლი დასრულდება უშედეგოდ.

როგორც ზემოთ აღნიშნულ ფსევდოკოდში ჩანს, გამოჩნდა სამი ცვლადი. აქედან, i აღნიშნავს იტერაციის ინდექსს, რომლის მნიშვნელობა უნდა შეიცვალოს,

თუმცა, თავდაპირველი მნიშვნელობა არის 1. ცვლადი j აღნიშნავს a მასივში მოთავსებული ელემენტების რაოდენობას, ხოლო index ცვლადი - იმ ინდექსს, რომელიც უნდა დაემთხვეს ელემენტის ადგილს მასივში, ან მისი მნიშვნელობა იქნება 0, თუ ვერ მოხერხდება მოძებნა. მეორე ეტაპზე a მასივის i-ური ელემენტი ედრება 4-ს. თუ ტოლობა შედგა, მაშინ index ცვლადი მიიღებს i-ს მნიშვნელობას და ციკლი გაჩერდება, თუ არა და გაგრძელდება. მესამე ეტაპზე, i-ს მნიშვნელობა გაიზრდება 1-ით. მეოთხე ეტაპზე შემოწმდება i არის თუ არა ნაკლები j-ზე, თუ შესრულდება ეს უტოლობა, დაბრუნდება მეორე ეტაპზე, ამავდროულად, მესამე ეტაპიდან გამომდინარე, i-ს მნიშვნელობა იქნება 1-ით გაზრდილი. თუ არ შედგება უტოლობა, მაშინ ციკლი გაჩერდება და შედეგი არ გვექნება, index ცვლადი 0 დარჩება.

როგორც ვხედავთ, ალგორითმი გაჩერდება მერვე იტერაციაზე, რადგან მერვე მონაცემი უდრის 4-ს. შესაბამისად, ინდექსი იქნება 8. როგორც ვხედავთ, ამ შემთხვევაში დაგვჭირდა 8 იტერაცია 10-დან. აქ დავსვათ ასეთი შეკითხვა: რა მოხდებოდა, თუ საძიებო რიცხვი იქნებოდა სხვა? ამ შემთხვევაში დაგვჭირდებოდა იტერაციათა განსხვავებული რაოდენობა. რა მოხდებოდა, თუ მასივში განსხვებული რიცხვები იქნებოდა და, ამავდროულად, მასივის ზომა იქნებოდა განსხვავებული? რამდენი იტერაცია დაგვჭირდებოდა? ამის გასაზომად შემუშავდა სხვადასხვა მეთოდი. ერთ-ერთი ყველაზე პოპულარულია უარესი შემთხვევის სირთულე (ინგლ. Worst case complexity). ჩვენი მასივის შემთხვევაში, რომელიც გამოვაცხადეთ, ყველაზე უარეს შემთხვევაში საჭირო გახდებოდა 10 იტერაციის განხორციელება, რადგან მასივში ელემენტების რაოდენობა 10-ის ტოლია და, ამავდროულად, რამდენი ელემენტიცაა, იმდენჯერ ხდება იტერაცია. თუ მასივი მოიცავს n ელემენტების სიმრავლეს, მაშინ საჭირო იქნებოდა n რაოდენობის იტერაციის შესრულება, შესაბამისად, მისი ალგორითმული დროითი სირთულე იქნებოდა O(n). აქ „O(n)“ აღნიშვნას უწოდებენ დიდი O-ს ნოტაციას (ინგლ. Big O notation) და გვიჩვენებს ყველაზე უარეს შემთხვევაში განსახორციელებელი კონკრეტული ოპერაციების რაოდენობას.

ჩვენ განვიხილეთ წრფივი ძებნის ალგორითმი. არსებობს ძებნის სხვა ალგორითმებიც. ამასთან ერთად, მასივში ელემენტები დალაგებული არ იყო. აქ დაისმის ასეთი შეკითხვა: თუ მწკრივში ელემენტები დალაგდებოდა და ძებნის ალგორითმს შევცვლიდით, რიცხვი 4-ის მოსაძებნად რამდენი იტერაცია დაგვჭირდებოდა? ეს ყველაფერი დამოკიდებულია ალგორითმზე. ძებნის ერთ-ერთი ალგორითმია ორობითი ძებნა. განვიხილოთ ისევ a მასივი და მოვძებნოთ ისევ რიცხვი 4. იმისათვის, რომ ამ ალგორითმმა იმუშაოს, ჯერ საჭიროა მონაცემები იყოს დალაგებული. დავუშვათ, გვაქვს იგივე მასივი, ოღონდ ამჯერად უკვე დალაგებული. შესაბამისად, უკვე დალაგებულ მასივს ვუწოდოთ b და წარმოვადგინოთ შემდეგი სახით:

$$b = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

ორობითი ძებნის შემთხვევაში ვირჩევთ შუა ინდექსს. ამისათვის უმცირეს ინდექსს, რომელიც მასივშია და უდიდეს ინდექსს, შევკრებთ და ჯამს ვყოფთ 2-ზე. მიღებულ შედეგს ვამრგვალებთ კლებადობთ; ჩვენ შემთხვევაში მივიღებთ 5-ს. შემდეგ შუა ინდექსის შესაბამის ელემენტს b მასივში ვადარებთ საძიებელ ელემენტს, ჩვენ შემთხვევაში 4-ს. თუ ემთხვევა, მაშინც ციკლი გაჩერდება და ინდექსად დაბრუნდება

5. თუ საძიებელი ელემენტი ნაკლებია ან მეტია b მასივის მე-5 ელემენტზე, მაშინ ახალ იტერაციაში ძებნა გაგრძელდება შესაბამის ინდექსამდე, ხოლო თუ მეტია, მაშინ ძებნა გაგრძელდება მე-5 ელემენტის შემდეგ. ჩვენ შემთხვევაში ძებნა გაგრძელდება ქვემასივში, რომელიც იქნება:

$$b_1 = \{1, 2, 3, 4\}$$

როგორც ვნახეთ, პირველი იტერაციის შემდეგ შესადარებელი ელემენტების რაოდენობა შემცირდა. მეორე იტერაციაზეც ანალოგიურად მოვიქცევით. შესაბამისად, შესადარებელი ელემენტები ისევ შემცირდება და ამჯერად შესადარებელი ელემენტების მასივი იქნება

$$b_3 = \{3, 4\}$$

მესამე იტერაციის შემდეგ შესადარებელი ელემენტების რაოდენობა ისევ შემცირდება და დაგვრჩება ერთი ელემენტი.

როგორც ვნახეთ, თუ მონაცემები დალაგებულია მასივში და ვიყენებთ ძებნის ორობით ალგორითმს, მაშინ ელემენტის ინდექსის მოსაძებნად იტერაციათა უფრო ნაკლები რაოდენობა გვჭირდება. ამ შემთხვევაში, ალგორითმული დროითი სირთულეა O(logn), სადაც n=10. შესაბამისად, მასივში ელემენტების ზრდის შემთხვევაში, თუ ელემენტები დალაგებულია, დრო, რომელიც დაგვჭირდება ინდექსის მოსაძებნად, გაიზრდება ლოგარითმულად. თუ გამოვიყენებთ ამ ალგორითმს, წრფივი ძებნის შემთხვევაში დრო წრფივად გაიზრდებოდა.

ზემოხსენებულიდან გამომდინარე, ამოცანაზე მორგებული ალგორითმის შემუშავებისას, მისმა ოპტიმალურმა ვარიანტმა შეიძლება მოგვცეს მნიშვნელოვანი დროითი რესურსების დანაზოგი. მარტივ მაგალითზე უკვე გავარჩიეთ აღნიშნული საკითხი. რა მოხდება, თუ მონაცემს ვეძებთ უფრო დიდ მასივში? სხვა თანაბარ პირობებში, თუ მასივში, რომელიც შედგება, დავუშვათ, 1 მილიონი ელემენტისგან და მოძებნას წრფივი ძებნის ალგორითმით მოვახდენთ, ყველაზე უარეს შემთხვევაში სჭირდება t წამი, მაშინ ბევრად უფრო ნაკლები დრო დასჭირდება მოძებნას იმავე ელემენტების შემცველ მასივში, თუ მონაცემები დალაგებულია და ვიყენებთ ორობითი ძებნის ალგორითმს. ეს ყველაზე მეტად გამოსადეგია მაშინ, როდესაც ხშირად ხდება მონაცემის მოძიება უკვე ერთხელ დალაგებული მასივიდან, როგორც ეს შეიძლება მოხდეს მონაცემთა ბაზების შემთხვევაში. როგორც აღვნიშნეთ, ორობითი ძებნის ალგორითმის გამოყენებისთვის საჭიროა მონაცემები იყოს დალაგებული. მონაცემთა დალაგების სხვადასხვა ალგორითმი არსებობს, თუმცა, მონაცემთა დალაგებასაც სჭირდება გარკვეული დრო.

მონაცემების დასალაგებლად მასივში გამოიყენება სხვადასხვა ალგორითმი, რომელთაც განსხვავებული ალგორითმული დროითი სირთულე შეიძლება გააჩნდეთ. აქედან საკმაოდ პოპულარულია Merge Sort და Quick Sort ალგორითმები. აქედან Merge Sort ალგორითმის ალგორითმული დროითი სირთულე უარეს შემთხვევაში არის O(nlogn), ხოლო Quick Sort ალგორითმის არის $O(n^2)$.

ზემოთ განვიხილეთ, თუ რა მნიშნელობა აქვს ალგორითმის ეფექტიანობას, გავიაზრეთ ძებნის ორი ალგორითმის მუშაობის პრინციპი, ვაჩვენეთ, თუ რატომ შეიძლება გახდეს მონაცემთა დალაგება საჭირო. მონაცემთა ბაზებშიც დაახლოებით მსგავსი პრინციპები მოქმედებს.

## 6.3. ინდექსირება

მონაცემთა ბაზების მართვის სისტემებში ერთ-ერთი საკვანძო საკითხია ინდექსირება. რა თქმა უნდა, ინდექსირება შესაძლებელია MS SQL Server 2022-შიც. აღნიშნულ პროგრამაში შესაძლებელია როგორც ცხრილების, ისე წარმოდგენების ინდექსირებაც. პროგრამაში ინდექსი (index) არის ცხრილთან ან წარმოდგენასთან დაკავშირებული სტრუქტურა და განკუთვნილია შესაბამის ცხრილში ან წარმოდგენაში ინფორმაციის ძებნის დასაჩქარებლად. ინდექსი განისაზღვრება ერთი ან მეტი სვეტისთვის, რომლებსაც ინდექსირებული სვეტები ეწოდება. ინდექსი შეიცავს ინდექსირებული სვეტის ან სვეტების დახარისხებულ მნიშვნელობებს საწყისი ცხრილის ან წარმოდგენის შესაბამის სტრიქონზე მიმართვებთან ერთად.

მწარმოებლურობის ზრდა მიიღწევა სვეტის მონაცემების დახარისხების ხარჯზე. თუ სვეტი დაუხარისხებელია, მაშინ საჭირო მნიშვნელობის საპოვნელად მიმდევრობით უნდა გაისინჯოს ყველა მნიშვნელობა. როცა საჭიროა ინდექსირებულ სვეტში მნიშვნელობის პოვნა, მაშინ მისი ძებნა სრულდება ინდექსში. ინდექსების გამოყენება მნიშვნელოვნად ზრდის მონაცემების ძებნის მწარმოებლურობას, მაგრამ ითხოვს დამატებით სივრცეს მონაცემთა ბაზაში [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 38].

განიხილება კლასტერირებული და არაკლასტერირებული ინდექსირება. კლასტერირებული ინდექსირების დროს ხდება უშუალოდ ცხრილშივე მონაცემების დალაგება და, აქედან გამომდინარე, ცხრილს შეიძლება ჰქონდეს მხოლოდ ერთი კლასტერირებული ინდექსი. არაკლასტერირებული ინდექსირების შემთხვევაში ხდება სხვა მონაცემთა სტრუქტურის შექმნა, რომელიც უკავშირდება ინდექსირებულ მონაცემებთა სტრიქონებს.

ცხრილების არაკლასტერირებული ინდექსირებისთვის გამოიყენება „CREATE INDEX“ სინტაქსი, რომელიც მიიღებს შემდეგ სახეს:

```
CREATE INDEX indexis_saxeli
ON cxrili (svetis_saxeli)
```

ამ შემთხვევაში იქმნება ინდექსი, რომელსაც ჰქვია „indexis_saxeli“ ცხრილისთვის, რომლის სახელია „cxrili“ და იმ სვეტისთვის, რომლის სახელია „svetis_saxeli“. უნიკალური არაკლასტერირებული ინდექსირებისთვის გამოვიყენებთ „CREATE UNIQUE INDEX“ სინტაქსს. ასევე, შესაძლებელია რამდენიმე სვეტის გამოყენებაც და დახარისხება როგორც ზრდადობით, ისე კლებადობით.

კლასტერირებული ინდექსირებისთვის გამოიყენება სინტაქსი:

```
CREATE CLUSTERED INDEX indexi_2
```

```
ON cxrili (svetis_saxeli)
```

ამ შემთხვევაში იქმნება კლასტერირებული ინდექსი სახელად „indexi_2“ იმ ცხრილისთვის, რომლის სახელია „cxrili“, ხოლო სვეტი კი არის „svetis_saxeli“.

გასათვალისწინებელია ის ფაქტიც, რომ ყველა შემთხვევაში ცხრილების ინდექსაცია არ არის ოპტიმალური. როდესაც ხდება ცხრილების განახლება, შესაბამისად, საჭირო ხდება მონაცემების თავიდან დალაგებაც. ასეთ დროს ჩაწერისა და განახლების ოპერაციები უფრო მეტად შენელდება.

განვიხილოთ მაგალითი და განვახორციელოთ mshp_seqcia ცხრილის ინდექსირება. გავითვალისწინოთ, რომ ეს მაგალითია და, ზოგადად, ასეთი მცირე ცხრილების ინდექსირება დიდ შედეგს არ მოგვიტანს. დავუშვათ, გვჭირდება არაკლასტერირებული ინდექსირება ორი სვეტისთვის, ანუ weli და seqcia სვეტებისთვის. მაშინ შესაბამისი მოთხოვნა ამ ოპერაციის განსახორციელებლად იქნება:

```
CREATE INDEX index1
ON [dbo].[mshp_seqcia] (weli, seqcia);
```

თუ SSMS 19-ის Object Explorer-ში ვნახავთ შესაბამის ცხრილს და მის მარცხნივ მოთავსებულ „+“ ღილაკს დავაწკაპუნებთ, დავინახავთ საქაღალდეებს, როგორებიც გამოსახულია სურათზე 6.2. გამოჩნდება Indexes საქაღალდე, რომლის გახსნის შემთხვევაში გამოჩნდება ჩვენი შექმნილი ინდექსი.

ინდექსის წასაშლელად გამოიყენება „DROP INDEX index1 ON cxrili“ სინტაქსი, სადაც „index1“ არის ინდექსის, ხოლო „cxrili“ კი ცხრილის სახელი. ამ შემთხვევაში, ჩვენ მიერ შექმნილი ინდექსის წასაშლელად გამოვიყენებთ კოდს:

```
DROP INDEX index1
ON dbo.mshp_seqcia;
```

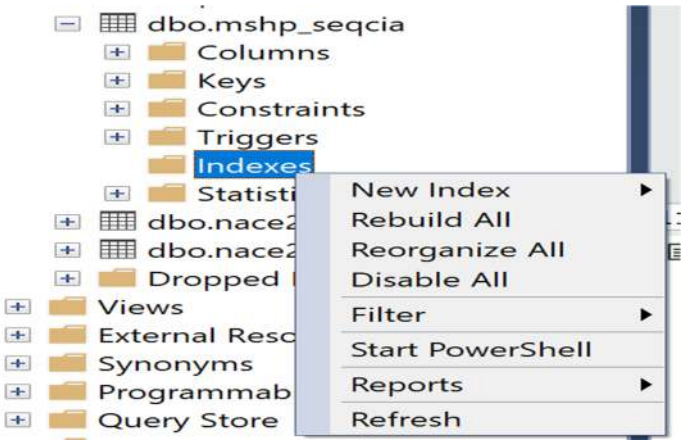
ამ ბრძანების გაშვების შემთხვევაში წაიშლება index1 ინდექსი.

**სურათი 6.2. mshp_seqcia ცხრილის ინდექსები**


SSMS 19-ით შესაძლებელია ინდექსების შექმნა გრაფიკული ინტერფეისის საშუალებით. ამისთვის Object Explorer-ში შესაბამისი ცხრილის Indexes საქაღალდეზე ვი-

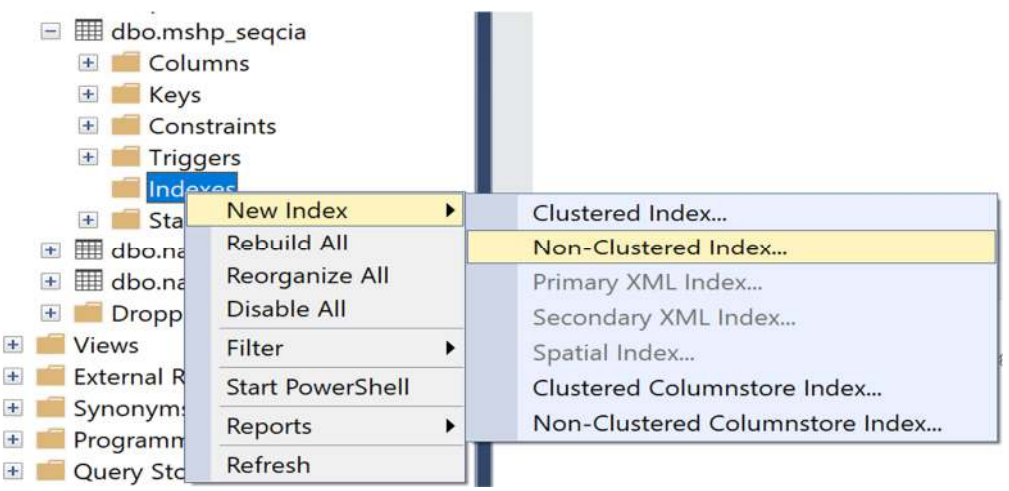
მოქმედოთ მაუსის მარჯვენა ღილაკით. გამოვა კონტექსტური მენიუ, როგორიც გამოსახულია სურათზე 6.3.

**სურათი 6.3. ინდექსების კონტექსტური მენიუ**




ახალი ინდექსის შესაქმნელად გამოტანილ კონტექსტურ მენიუში მივიტანოთ მაუსი „New Index“ ღილაკთან. შედეგად გამოიტანება ახალი მენიუ, რომელიც გამოსახულია სურათზე 6.4.

**სურათი 6.4. ინდექსების შექმნის კონტექსტური მენიუ**

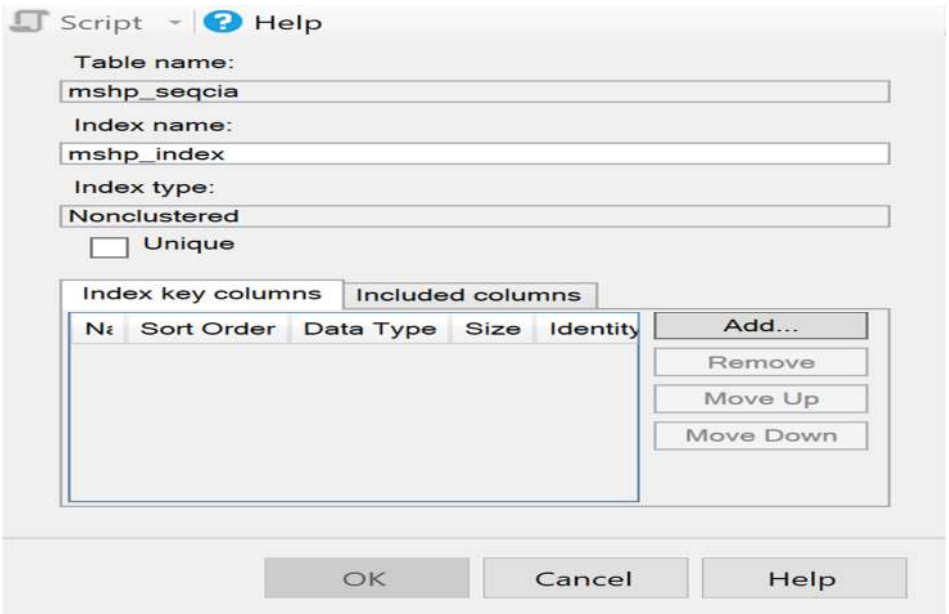



როგორც ვხედავთ, არსებობს სხვადასხვა ინდექსის შექმნის ვარიანტი გრაფიკული ინტერფეისით. კლასტერირებული ინდექსის შესაქმნელად გამოიყენება „Clustered Index...“, ხოლო არაკლასტერირებულისთვის კი „Non-Clustered Index...“ ღილაკი.

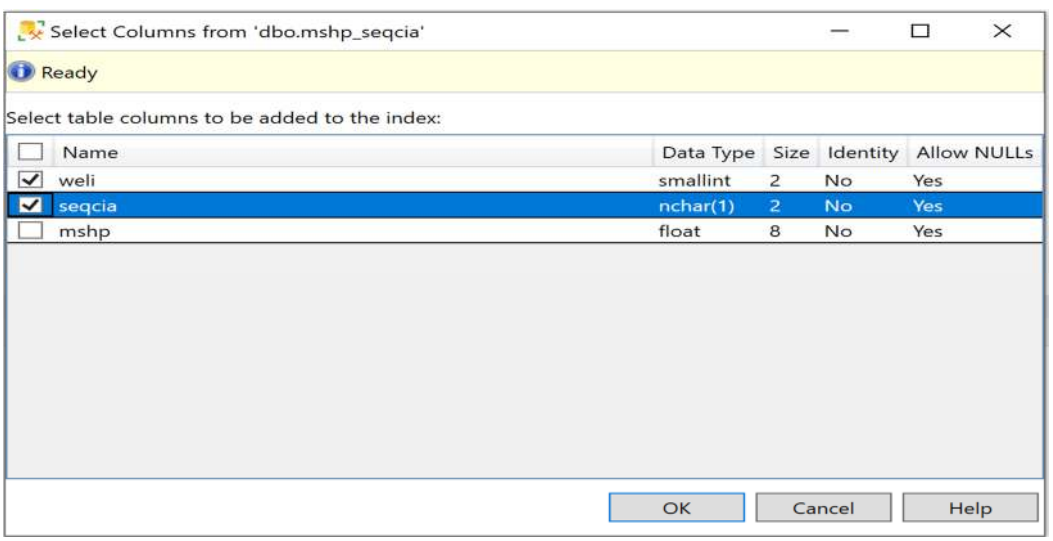
პირობითად შევქმნათ არაკლასტერირებული ინდექსი mshp_seqcia ცხრილისთვის SSMS 19-ის გრაფიკული ინტერფეისის გამოყენებით. ამისათვის დავაწკაპუნოთ „Non-Clustered Index...“ ღილაკზე. შედეგად გამოვა ახალი ფანჯარა, რომელიც გამოსახულია სურათზე 6.5. ახალი ინდექსის შესაქმნელ ფანჯარაში გამოიტანება General

გვერდი, სადაც Table name ველში მოთავსებულია ცხრილის სახელი, Index name ველში კი შესაქმნელი ინდექსის სახელი. ჩვენს შემთხვევაში მას დავარქვათ „mshp_index“ და ჩავწეროთ შესაბამის ველში. Index type ველი გვიჩვენებს ინდექსის ტიპს. ამ შემთხვევაში ჩანს, რომ ინდექსი არაკლასტერირებულია. ამ ველის ქვემოთ არსებული Unique მოსანიშნი ველით შესაძლებელია დავაზუსტოთ, გვინდა თუ არა უნიკალური არაკლასტერირებული ინდექსის შექმნა. ქვემოთ არსებულ ჩანართებში შეგვიძლია მივუთითოთ შესაბამისი სვეტები. ამისათვის Index key columns ჩანართი გავააქტიუროთ და დავაწკაპუნოთ „Add...“ ღილაკზე. გამოვა ახალი ფანჯარა, რომელიც გამოსახულია სურათზე 6.6.

სურათი 6.5. ახალი ინდექსის შექმნა, General გვერდი

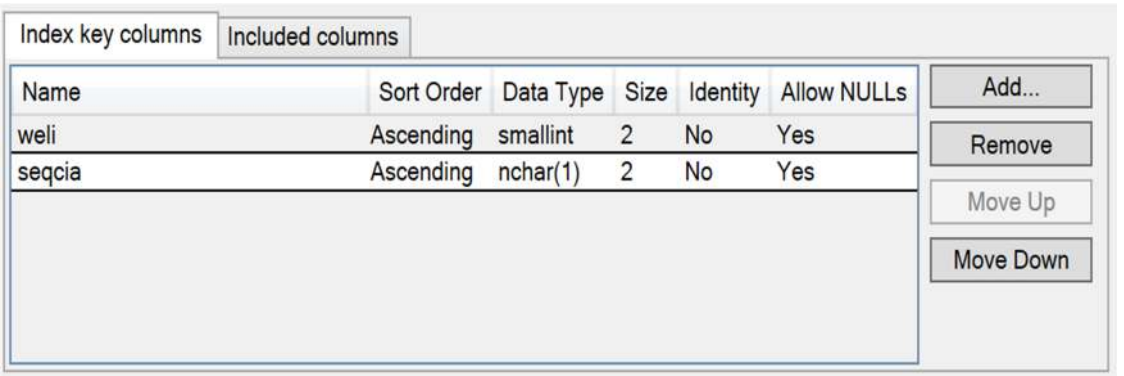



სურათი 6.6. ინდექსაციისთვის სვეტების არჩევა



ამ შემთხვევაში ავირჩიოთ ორი სვეტი - „weli“ და „seqcia“, მოცემული სვეტების მონიშვნით. ამის შემდეგ დავაწკაპუნოთ OK ღილაკზე. შედეგად, General გვერდზე გამოსახულ Index key columns ჩანართში ჩაიწერება ჩვენ მიერ არჩეული სვეტები ისე, როგორც ეს გამოსახულია სურათზე 6.7. ამავე ჩანართში შეგვიძლია სვეტები გადაავადგილოთ მათი მონიშვნით და Move Up და Move Down ღილაკებით, ან წავშალოთ კონკრეტული სვეტის მონიშვნით და Remove ღილაკზე დაწკაპუნებით.

**სურათი 6.7. არჩეული სვეტები ინდექსირებისთვის**




რადგან უკვე შევარჩიეთ შესაბამისი პარამეტრები ინდექსის შესაქმნელად, ახლა დავაწკაპუნოთ OK ღილაკზე, რომელიც გამოსახულია სურათზე 6.5 და ინდექსიც შეიქმნება. ინდექსის წაშლა შესაძლებელია გრაფიკულადაც. ამისათვის მოვნიშნოთ კონკრეტული ინდექსი Object Explorer-ში და დავაწკაპუნოთ მაუსის მარჯვენა ღილაკით. გამოვა კონტექსტური მენიუ, რომელიც გამოსახულია სურათზე 6.8 და დავაწკაპუნოთ Delete ღილაკზე; გამოვა ფანჯარა, რომელშიც დავაწკაპუნებთ OK ღილაკზე და ინდექსიც წაიშლება.

**სურათი 6.8. ინდექსის კონტექსტური მენიუ**

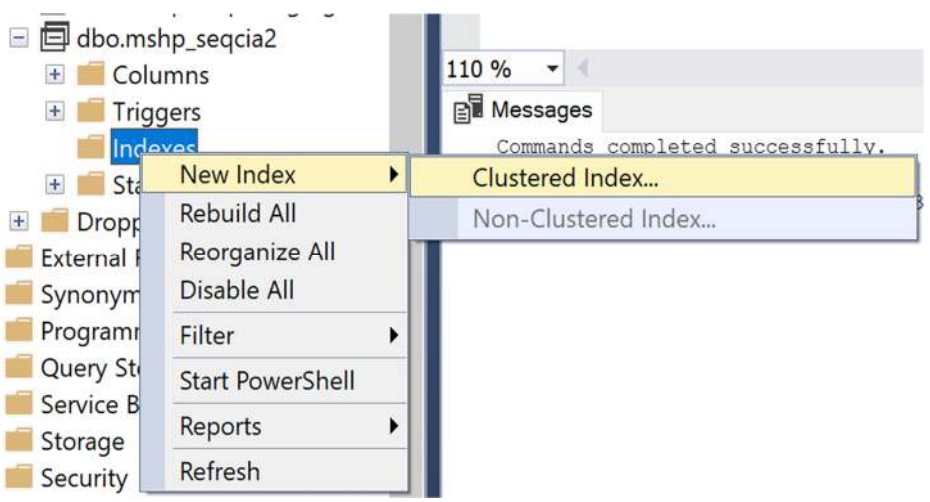

როგორც აღვნიშნეთ, შესაძლებელია ინდექსირებული წარმოდგენის შექმნა. ამ დროს წარმოდგენისთვის საჭირო მონაცემების დისკზე ჩაწერაც ხდება. თუ გვსურს

შევქმნათ ინდექსირებული წარმოდგენა, ამისათვის ჯერ საჭიროა შევქმნათ წარმოდგენა „WITH SCHEMABINDING“ პარამეტრით. მაგალითად, შევქმნათ წარმოდგენა, რომელიც ჩვენ მიერ შექმნილი mshp_seqcia ცხრილს დააკავშირებს nace2 ცხრილთან და ამოიღებს წლის, სექციის დასახელებისა და მშპ-ის მონაცემს. დავურთოთ ზემოხსენებული პარამეტრი და დავარქვათ mshp_seqcia2, შედეგად მოთხოვნა ასეთი იქნება:

```
CREATE VIEW [dbo].[mshp_seqcia2]
WITH SCHEMABINDING
AS
SELECT weli, seqcia_dasaxeleba, mshp
FROM [dbo].mshp_seqcia a
INNER JOIN dbo.nace2 b ON a.seqcia = b.seqcia;
```

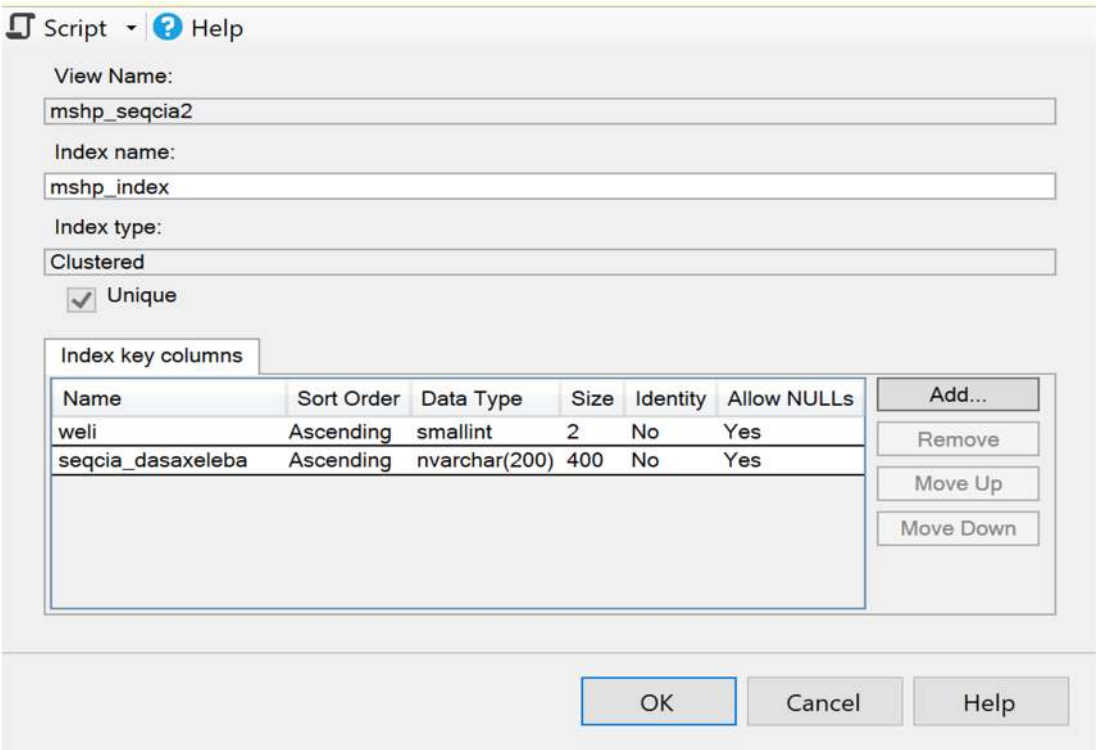
გავითვალისწინოთ, რომ ზემოხსნებული პარამეტრის გამოყენებისას აუცილებელია თითოეულ ცხრილს მივუთითოთ შესაბამისი სქემა და გავუშვათ მოთხოვნა. ამის შემდეგ ვცადოთ ჩვენ მიერ შექმნილი წარმოდგენის ინდექსირება. ამისათვის SSMS 19-ის Object Explorer-ში მოვნახოთ ჩვენ მიერ შექმნილი წარმოდგენა, დავაწკაპუნოთ მის მარცხნივ არსებულ „+“ ღილაკს და Index საქაღალდეზე მაუსის მარჯვენა ღილაკით გამოვიძახოთ კონტექსური მენიუ, ავირჩიოთ „New Index >Clustered Index...“, როგორც ეს გამოსახულია სურათზე 6.8.

**სურათი 6.9. წარმოდგენის ინდექსირება**




გამოვა ფანჯარა და გრაფიკული ინტერფეისიდან ავირჩიოთ სვეტები და პარამეტრები ისე, როგორც ეს ნაჩვენებია სურათზე 6.10. დავაწკაპუნოთ OK ღილაკზე, რის შემდეგაც მოხდება წარმოდგენის ინდექსირება.

სურათი 6.10. წარმოდგენის ინდექსირება




***კითხვები თვითშემოწმებისთვის:***

1. რა არის წარმოდგენა მონაცემთა ბაზებში?
2. როგორ იქმნება წარმოდგენა მონაცემთა ბაზებში?
3. აღწერეთ წრფივი ძებნის ალგორითმი.
4. აღწერეთ ორობითი ძებნის ალგორითმი.
5. რაში გამოიყენება ინდექსირება?
6. როგორ ხდება ცხრილების ინდექსირება?
7. როგორ ხდება წარმოდგენების ინდექსირება?

# თავი 7. პროგრამირება და ავტომატიზაცია

## 7.1. შესავალი პროგრამირებაში

Microsoft SQL Server 2022-ის ერთ-ერთ დიდ შესაძლებლობას წარმოადგენს მისი ავტომატიზების საშუალება. სანამ გადავიდოდეთ ავტომატიზებაზე, მანამდე განვიხილოთ ის ძირითადი თემები, რომლებიც პროგრამირებას შეეხება.

პროგრამა ასრულებს ბრძანებებს, ამისთვის კი მას სჭირდება გარკვეული ტიპის ინფორმაცია. პროგრამის წერისას ხდება სხვადასხვა ცვლადის გამოცხადება, რომლის საშუალებით უნდა განხორციელდეს რაიმე ოპერაცია. მაგალითად, გამოვაცხადოთ რაიმე ცვლადი და ვუწოდოთ მას a, მივანიჭოთ ამ ცვლადს მთელი რიცხვის ტიპი. ამ შემთხვევაში ვუშვებთ შეზღუდვას, რომ ცვლად a-ში შეიძლება ისე მოთავსდეს მონაცემი, რომელიც იქნება მთელი რიცხვი. შემდეგ მივანიჭოთ ცვლად a-ს რაიმე მნიშვნელობა, მაგალითად, დავუშვათ, რომ a-ს მნიშვნელობა არის 2. ამ ცვლადის საშუალებით ვიპოვოთ რიცხვი 2-ის ნამრავლი, მაგალითად, 4-ზე და გამოვიტანოთ მიღებული მნიშვნელობა. ამ ამოცანის შესრულების შემდეგ გამოიტანება რიცხვი 8. ფსევდოკოდი ამ შემთხვევაში იქნება:

```
გამოცხადდეს ცვლადი a;
მიენიჭოს ცვლად a-ს მნიშვნელობა 2;
დაიბეჭდოს a-ს ნამრავლი 2-ზე;
```

ამ შემთხვევაში ჩვენ შევზღუდეთ ცვლადი a-ს მონაცემთა ტიპი. ასეთ დროს ვამბობთ, რომ გამოყენებულია სტატიკური ტიპირება. რა მოხდებოდა, თუ ცვლად a-ს შეეძლებოდა მიეღო ერთ შემთხვევაში მთელი რიცხვის მნიშვნელობა, სხვა შემთხვევაში ათწილადის, ხოლო კიდევ სხვა შემთხვევაში საერთოდ ტექსტური მონაცემი გამხდარიყო. ამ შემთხვევაში ვამბობთ, რომ გამოყენებულია დინამიკური ტიპირება. ზოგადად, პროგრამირების ენები განსხვავდებიან ტიპირების სისტემითაც. არსებობს დინამიკური და სტატიკური ტიპირების მქონე პროგრამული ენები. მაგალითად, პროგრამული ენები C, C++, Java არიან სტატიკური ტიპირების მქონე ენები, ხოლო Python, JavaScript და pHp - დინამიკური ტიპირების მქონე პროგრამული ენები.

მოკლედ მიმოვიხილოთ პროგრამირების ზოგიერთი პარადიგმა. ზოგადად, ერთმანეთისაგან განსხვავდება იმპერატიული და დეკლარატიული პროგრამირების პარადიგმები.

იმპერატიული პარადიგმების შემთხვევაში, პროგრამული უზრუნველყოფა იყენებს სხვადასხვა განაცხადს პროგრამის მდგომარეობის შესაცვლელად. ამ დროს ხდება ნაბიჯების დაზუსტება, თუ როგორ უნდა შესრულდეს რაიმე ბრძანება. პროცედურული პროგრამირება არის იმპერატიული პროგრამირების ერთ-ერთი სახე, როდესაც ხდება პროცედურების გამოყენება. ამ დროს შესაძლებელია პროცედურებმა გამოიძახონ ერთმანეთი და შესრულდეს სხვადასხვა ოპერაცია. არსებობს ობიექტზე

ორიენტირებული პროგრამირების პარადიგმაც, რომელიც იმპერატიული პროგრამირების ერთ-ერთი სახეა.

ობიექტზე ორიენტირებული დაპროგრამების შემთხვევაში საკვანძო წერტილი არის ობიექტი. წინა მაგალითში ვნახეთ, როცა დავადეკლარირეთ ცვლადი, რომელმაც მიიღო რაღაც მნიშვნელობა და შემდეგ განვახორციელეთ მასზე 4-ზე გამრავლების ოპერაცია. ობიექტზე ორიენტირებული პროგრამირებისას ობიექტი მოიცავს როგორც მონაცემს, ისე მეთოდებს. მაგალითად, არსებობს ობიექტი b, რომელიც მოიცავს მონაცემს a-ს და, ასევე, მოიცავს მეთოდებს, როგორიცაა გამრავლება 2-ზე და გაყოფა 2-ზე. თუ გამოვიძახებთ ობიექტის 2-ზე გამრავლების მეთოდს, მაშინ გამოიტანება რიცხვი 8, ხოლო თუ გამოვიძახებთ 2-ზე გაყოფის მეთოდს, მაშინ დაიბეჭდება რიცხვი 2. გამოვიძახოთ 2-ზე გამრავლების მეთოდი. ფსევდოკოდი მიიღებს ასეთ სახეს:

```
ობიექტი b{
ცვლადი a = 4;
გამრავლებისმეთოდი{
        დაიბეჭდოს a-ს ნამრავლი 2-ზე;
}
გაყოფისმეთოდი{
        დაიბეჭდოს a-ს განაყოფი 2-ზე;
}
}
გამოვიძახოთ b ობიექტის გამრავლების მეთოდი.
```

განვიხილეთ ობიექტზე ორიენტირებული დაპროგრამების საკმაოდ მარტივი შემთხვევა. თუმცა, ამ პარადიგმის სიძლიერე ბევრად მასშტაბურია. ობიექტზე ორიენტირებული პროგრამირების ერთ-ერთი ძლიერი თვისება არის კლასები. კლასი შეიძლება წარმოვიდგინოთ, როგორც ერთგვარი ყალიბი, რომლიდანაც იქმნება სხვადასხვა ობიექტი. ამ შემთხვევაში იქმნება ერთი კლასი, რომელსაც აქვს მახასიათებლები და მეთოდები. ჩვენი მაგალითის შემთხვევაში იქმნება კლასი, რომელიც მოიცავს a ცვლადს და ზემოთ განხილულ ოპერაციებს, მაგრამ ამ შემთხვევაში a-ს მნიშვნელობა უცნობია. ერთ-ერთი მეთოდი, რომელიც გახდება საჭირო, არის კონსტრუქტორი. კონსტრუქტორი არის ერთგვარი მეთოდი, რომელიც ეშვება კლასიდან ობიექტის შექმნის დროს და ხდება გარკვეული ოპერაციების განხორციელება. მაგალითად, ჩვენს ვარიანტში კონსტრუქტორი მიანიჭებს a-ს გარკვეულ მნიშვნელობას ობიექტის შექმნისას. შემდეგ შეგვიძლია თითოეულ ობიექტზე ვიმოქმედოთ განსხვავებულად.

განიხილება ობიექტზე ორიენტირებული დაპროგრამების ოთხი სვეტი, ესენია: აბსტრაქცია, ენკაპსულაცია, მემკვიდრეობა და პოლიმორფიზმი. აბსტრაქციის შემთხვევაში ხდება პროგრამის იმპლემენტაციის დეტალების დამალვა გარე სამყაროსთვის. ამ შემთხვევაში შეიძლება კლასს გააჩნდეს ისეთი ცვლადები, რომლებიც დამალულია და მხოლოდ გარკვეული მეთოდებით შეიძლება მათზე მოქმედება. ენკაპსულაციის დროს ხდება მონაცემებისა და მახასიათებლების განთავსება კონკრეტულ

კლასში. შესაბამისად, თითოეული ობიექტი მოიცავს გარკვეულ მონაცემს მეთოდებთან ერთად. მემკვიდრეობა საკმაოდ საინტერესო რამაა. ამ დროს შესაძლებელია შეიქმნას კლასი, რომელიც გახდება ერთგვარი ყალიბი სხვა კლასებისთვის. მაგალითად, წარმოვიდგინოთ კომპანიის კლასი, რომელსაც გააჩნია სახელი, დასაქმებულების რაოდენობა, მოგება. ამავე დროს, სხვადასხვა კომპანიას შეიძლება გააჩნდეს სხვადასხვა მახასიათებელი თუნდაც იქიდან გამომდინარე, რომ ეს კომპანიები შეიძლება იყვნენ სხვადასხვა სახის. მაგალითად, ასეთი განსხვავება შეიძლება იყოს ის, რომ ზოგი შეიძლება იყოს შპს, ზოგი სააქციო საზოგადოება, ზოგი კომანდიტური საზოგადოება და ა.შ. შესაბამისად, დავადეკლარირებთ ერთ კლასს, რომელიც მოიცავს ისეთ მახასიათებლებსა და მეთოდებს, რომლებიც ახასიათებს მოდელირებისთვის განკუთვნილ კომპანიებს, მაგრამ, ამავე დროს, დავადეკლარირებთ სხვა კლასებსაც, რომლებიც დამემკვიდრდებიან მშობელი კლასისგან და მისგან მიიღებენ მეთოდებსა და მახასიათებლებს. შედეგად, შვილი კლასებისგან შექმნილ ობიექტებს ექნება მახასიათებლები და მეთოდები, რომლებიც მიიღეს შვილმა კლასებმა მშობელი კლასებისგან. რაც შეეხება პოლიმორფიზმს, წარმოვიდგინოთ, რომ კლასს გააჩნია მახასიათებლები, რომელთაც შეიძლება გააჩნდეთ სხვადასხვა მონაცემთა ტიპი, თუმცა, ერთი მეთოდის დასახელებით შესაძლებელია გახდეს სხვადასხვა მონაცემთა ტიპისთვის სხვადასხვა ოპერაციის შესრულება კონტექსტიდან გამომდინარე. ზოგადად, განიხილება პოლიმორფიზმის სხვადასხვა სახე. ესენი შეიძლება იყოს ფუნქციების გადატვირთვა, ოპერატორების გადატვირთვა და ვირტუალური ფუნქციები.

დეკლარატიულ პროგრამირების პარადიგმის შემთხვევაში კონტროლის ნაკადების აღწერა არ ხდება. დეკლარატიული პროგრამირების პარადიგმებში ერთ-ერთი ყველაზე ცნობილი ფუნქციონალური პროგრამირებაა. ფუნქციონალური პროგრამირების შემთხვევაში ხდება შესაბამისი შედეგების ფუნქციების საშუალებით მიღება. მაგალითად, შეიძლება შევქმნათ რაიმე სტატისტიკური ინდექსის გაანგარიშების ფუნქცია და იგი იღებდეს რაიმე არგუმენტებს. შედეგად, ცვლადმა შესაძლებელია მიიღოს მნიშვნელობები ამ ფუნქციის გამოყენებით, თუ ფუნქციას მივაწვდით შესაბამის არგუმენტებს და შემდეგ ფუნქციას დავუბრუნებთ შესაბამის მნიშვნელობას.

სხვადასხვა კლასიფიკაციიდან გამომდინარე, პროგრამულ ენებს ხშირად ანსხვავებენ ერთმანეთისგან. ერთ-ერთი კლასიფიკაცია ადარებს პროგრამულ ენებს იმის მიხედვით, ეს ენა კომპილირებადია თუ ინტერპრეტირებადი. კომპილირებადი ენების შემთხვევაში პროგრამული კოდი ჯერ საჭიროა დაკომპილირდეს შესაბამის მანქანურ კოდზე და შემდეგ ისე მოხდეს მისი გაშვება. ინტერპრეტირებადი პროგრამული ენის შემთხვევაში ხდება დაწერილი კოდის წაკითხვა ინტერპრეტერის მიერ და შესაბამისი ბრძანებების გაშვება.

## 7.2. პროგრამული ენის ელემენტები T-SQL-ში

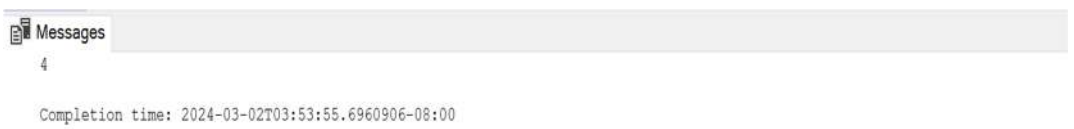
წინა ქვეთავში მოკლედ განვიხილეთ ის თემები, რომლებიც საჭიროა პროგრამირების გასააზრებლად. ამჯერად განვიხილოთ T-SQL-ში პროგრამული ენის ელემენტების გამოყენება.

ჯერ დავიწყოთ ცვლადების გამოცხადებით. მივუბრუნდეთ ამ თავში მოცემულ ჩვენს პირველ მაგალითს. გამოვაცხადოთ a ცვლადი, რომელიც შეიცავს მთელ რიცხვს 4-ს და დავბეჭდოთ მისი მნიშვნელობა. შესაბამისი კოდი იქნება:

```
DECLARE @a INT = 4;
PRINT(@a)
```

ამ კოდში DECLARE აცხადებს a ცვლადს, INT ნიშნავს, რომ გამოცხადებული ცვლადის ტიპი უნდა იყოს მთელი რიცხვი. ზემოთ დაწერილ ბრძანებაში ტოლობა ნიშნავს მინიჭების ოპერატორს და ცვლად a-ს ანიჭებს მნიშვნელობას 4. ხოლო PRINT არის დაბეჭდვის ფუნქცია, რომელსაც ვაწვდით არგუმენტ a-ს და შედეგად ბეჭდავს a ცვლადის მნიშვნელობას. შედეგი გამოსახულია სურათზე 7.1.

**სურათი 7.1. ცვლადის გამოცხადება და მისი მნიშვნელობის გამოტანა**

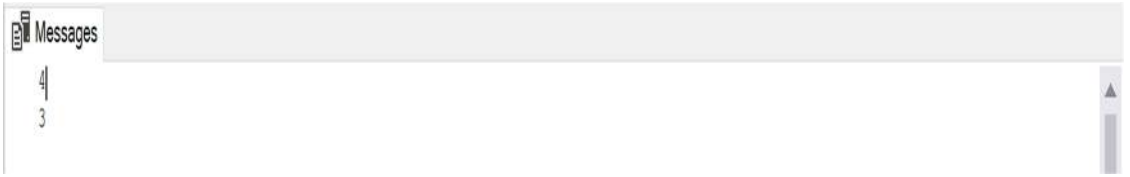



შეგვიძლია ცვლადი ჯერ გამოვაცხადოთ და შემდეგ მივანიჭოთ მას მნიშვნელობა. დავუშვათ, გვსურს ჯერ დავბეჭდოთ a ცვლადის მნიშვნელობა, შემდეგ შევცვალოთ მისი მნიშვნელობა 3-ით და დავბეჭდოთ მისი ახალი მნიშვნელობა. ამ შემთხვევაში კოდი მიიღებს შემდეგ სახეს:

```
DECLARE @a INT = 4;
PRINT(@a)
SET @a = 3;
PRINT(@a)
```

აქ SET ბრძანება ცვლის a ცვლადის მნიშვნელობას. შედეგად „SET @a = 3“ კოდი ნიშნავს, რომ a ცვლადს ენიჭება ახალი მნიშვნელობა 3. შედეგი გამოსახულია სურათზე 7.2.

**სურათი 7.2. ცვლადის მნიშვნელობის შეცვლის შედეგი**

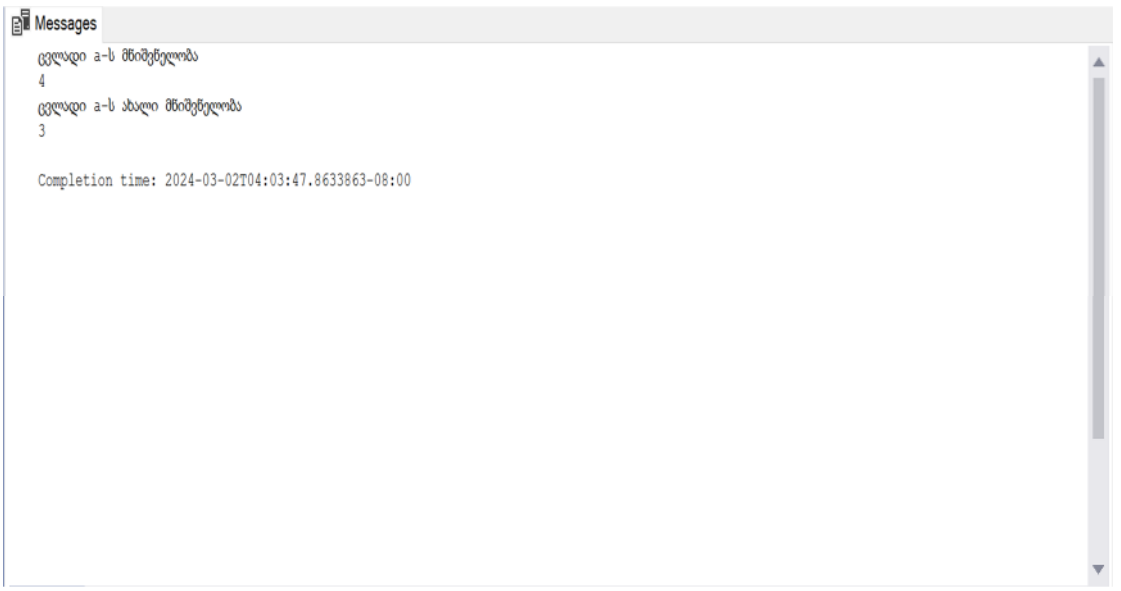



PRINT ფუნქციით შეგვიძლია გამოვიტანოთ ასევე ტექსტიც, მივუთითოთ ტექსტით ზემოხსენებული ცვლადის თავდაპირველი და ახალი მნიშვნელობის შესახებ. ამისათვის შესაბამისი კოდი იქნება:

```
DECLARE @a INT = 4;
PRINT(N'ცვლადი a-ს მნიშვნელობა')
PRINT(@a)
PRINT(N'ცვლადი a-ს ახალიმნიშვნელობა')
SET @a = 3;
PRINT(@a)
```

შედეგი გამოსახულია სურათზე 7.3.

**სურათი 7.3. ტექსტის გამოტანა PRINT ფუნქციით**



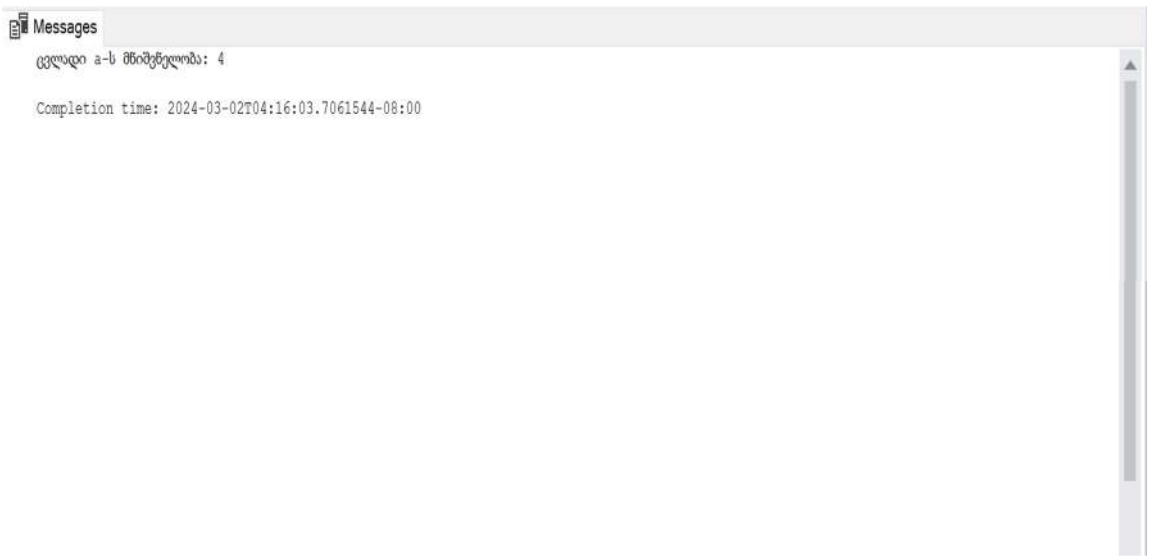

ჩვენ შეგვიძლია კონკრეტული ცვლადისთვის მონაცემთა ტიპის შეცვლა და ისე დაბეჭდვა. მაგალითად, a ცვლადი ვაქციოთ NVARCHAR მონაცემთა ტიპის მქონედ და ისე დავბეჭდოთ მისი ძველი მნიშვნელობა. ამ შემთხვევაში გამოვიყენებთ CAST სინტაქსს, რომელიც შემდეგნაირად გამოიყურება: „CAST (@cvladi AS NVARCHAR)“, სადაც cvladi არის ცვლადის სახელი, რომლის მონაცემთა ტიპიც გვსურს შევცვალოთ, ხოლო NVARCHAR - a ცვლადის ახალი მონაცემთა ტიპი. შედეგად, შეგვეძლება არსებულ

ტექსტს დავუმატოთ ახალი ტექსტი, რომელშიც მოთავსებულია ცვლადის მონაცემი „+“ ოპერატორით. ამ შემთხვევაში ჩვენი კოდი იქნება:

```
DECLARE @a INT = 4;
PRINT(N'ცვლადი a-ს მნიშვნელობა: '+CAST(@a AS NVARCHAR))
```

შედეგი გამოსახულია სურათზე 7.4.

**სურათი 7.4. მონაცემთა ტიპის შეცვლა**

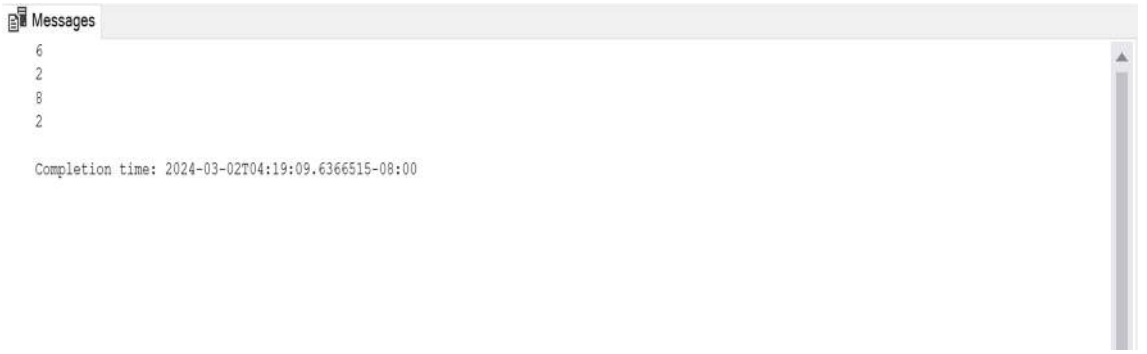



შესაძლებელია ცვლადებზე მათემატიკური ოპერაციების განხორციელებაც, როგორც ამას ადრე ვაკეთებდით. მაგალითად, გამოვაცხადოთ ცვლადი a, მივანიჭოთ მას მნიშვნელობა 4 და დავბეჭდოთ მისი ჯამი 2-თან, მისი სხვაობა 2-თან, მისი ნამრავლი 2-თან და მისი განაყოფი 2-თან. შედეგად, შესაბამისი კოდი იქნება:

```
DECLARE @a INT = 4;
PRINT(@a + 2);
PRINT(@a - 2);
PRINT(@a * 2);
PRINT(@a / 2);
```

ამ კოდის გაშვების შედეგი გამოსახულია სურათზე 7.5.

სურათი 7.5. მათემატიკური ოპერაციების შედეგი

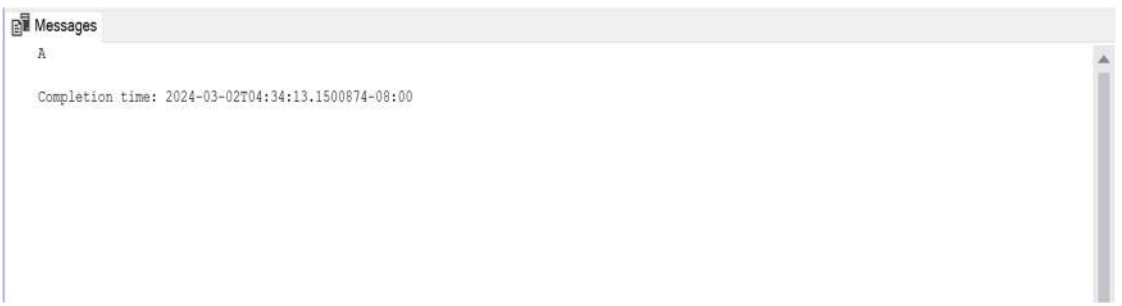



T-SQL-ში შეგვიძლია გამოვიყენოთ პირობებიც. მუშაობის პრინციპი დაახლოებით მსგავსია, როგორიც CASE სინტაქსის გამოყენების დროს იყო, თუმცა, ამჯერად გამოვიყენებთ IF-ს. მაგალითად, დავუშვათ, რომ არის შეფასების შემდეგი კრიტერიუმები: თუ სტუდენტი მიიღებს 91-დან 100 ქულამდე, მაშინ დავუწერთ A-ს, თუ 81-დან 90 ქულამდე, მაშინ B-ს, ხოლო სხვა შემთხვევაში დავუწერთ „სხვა შედეგინ. დავუშვათ, მიღებული ქულის ცვლადია a და მისი მნიშვნელობა არის 95. შესაბამისად, კოდი მიიღებს შემდეგ სახეს:

```
DECLARE @a INT = 95;
IF @a > 91 AND @a <= 100
BEGIN
        PRINT(N'A')
END
ELSE IF @a > 81 AND @a <= 90
BEGIN
        PRINT(N'B')
END
ELSE
BEGIN
        PRINT(N'სხვაშედეგი')
END
```

კოდის შედეგი გამოსახულია სურათზე 7.6.

სურათი 7.6. IF, ELSE IF, ELSE განაცხადების გამოყენება



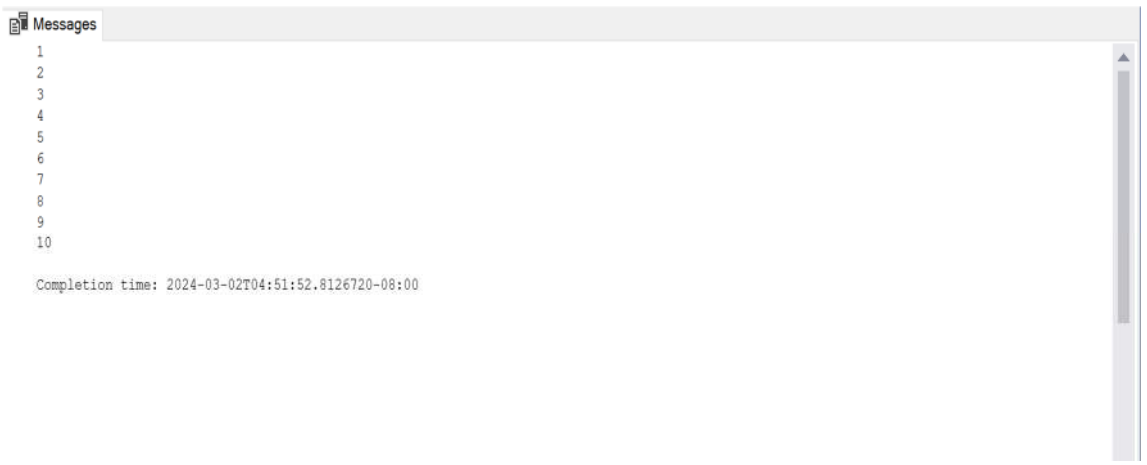
აქვე უნდა აღინიშნოს, რომ შესაძლებელია ჩადგმული IF-ების გამოყენება, მაგალითად, როდესაც შესრულდა რაღაც პირობა და გვსურს კიდევ სხვა პირობაც შევამოწმოთ არსებული შესრულებული პირობის შიგნით.

T-SQL-ს აქვს ციკლის მხარდაჭერაც. მაგალითად, შესაძლებელია რომელიმე ოპერაცია ხორციელდებოდეს მანამ, სანამ სრულდება რაიმე პირობა. განვიხილოთ ერთერთი მარტივი მაგალითი. დავუშვათ, შემოვიღეთ მთელი რიცხვის შემცველი ცვლადი a, რომლის მნიშვნელობაა 1 და გვსურს დავთვალოთ 10-მდე, ანუ პროგრამამ დაბეჭდოს რიცხვები 1-დან 10-ის ჩათვლით. შეგვიძლია ეს განვახორციელოთ ინდივიდუალურად რიცხვების შეყვანით და PRINT ფუნქციის გამოყენებით, მაგალითად PRINT(1); PRINT(2) და ა. შ., ან გამოვიყენოთ ციკლები და დავბეჭდოთ a ცვლადის მნიშვნელობა, შემდეგ a-ს მნიშვნელობა გავზარდოთ 1-ით და გავიმეოროთ სანამ a-ს მნიშვნელობა იქნება 10-ზე ნაკლები ან ტოლი. ამისათვის უნდა გამოვიყენოთ WHILE ციკლი. შედეგად, შესაბამისი კოდი მიიღებს შემდეგ სახეს:

```
DECLARE @a INT = 1;
WHILE @a <= 10
BEGIN
        PRINT(@a);
        SET @a = @a + 1;
END
```

შედეგი გამოსახულია სურათზე 7.7.

**სურათი 7.7. WHILE ციკლის გამოყენება**




შეგვიძლია კოდი შევცვალოთ ისე, რომ თუ a ცვლადის მნიშვნელობა გახდება 9, მაშინ გაგრძელდეს ციკლი და არ დაბეჭდოს a-ს მნიშვნელობა, მაგრამ ცვლადი გაიზარდოს 1-ით. სხვა შემთხვევაში დაიბეჭდოს ცვლადი. ამისათვის გამოვიყენებთ CONTINUE ბრძნებას და კოდი ასეთი იქნება:

```
DECLARE @a INT = 1;
WHILE @a <= 10
BEGIN
	IF @a = 9
	BEGIN
		SET @a = @a + 1;
		CONTINUE
	END
	ELSE
	BEGIN
		PRINT(@a);
		SET @a = @a + 1;
	END
END
```

შესაბამისად, დაიბეჭდება რიცხვები 1-დან 10-ის ჩათვლით, 9-ის გარდა. თუ CONTINUE ბრძანებას შევცვლით BREAK-ით, ციკლი გაჩერდება მაშინ, როდესაც a-ს მნიშვნელობა იქნება 9 და დაიბეჭდება რიცხვები 1-დან 8-ის ჩათვლით.

კომენტარებისთვის შეგვიძლია გამოვიყენოთ „--“ სინტაქსი. შედეგად, დაკომენტარებული კოდი, ან კომენტარში მოთავსებული ტექსტი არ გაეშვება. მრავალხაზიანი კომენტარებისთვის გამოიყენება „/*...*/“. იხილეთ შემდეგი მაგალითი:

```
--PRINT(1)
PRINT(3)
/*
	ეს არის მრავალსტრიქონიანი კომენტარი, რომელშიც ტექსტის იგნორირება მოხდება

PRINT(8)
*/
PRINT(10)
```

ამ შემთხვევაში დაიბეჭდება მხოლოდ 3 და 10, რადგან PRINT(1) და PRINT(8) დაკომენტარებულია. შესაბამისად, კომენტარში მითითებული ქართული ტექსტი არ გამოიწვევს ხარვეზს ბრძანების გაშვებისას.

## 7.3. შენახული პროცედურები

SQL Server 2022-ს გააჩნია შენახული პროცედურების მხარდაჭერა. შენახული პროცედურებით შესაძლებელია სხვადასხვა შესასრულებელი ოპერაციის ავტომატიზაცია.

ვთქვათ, გვაქვს mshp_seqcia2 წარმოდგენა და დავუშვათ, გვსურს ამ წარმოდგენის საფუძველზე დაითვალოს თითოეული სექციის პროცენტული წილი მშპ-ში კონ-

კრეტული წლისათვის და ეს მონაცემები შევიდეს სხვა ცხრილში. ჯერ დავწეროთ მოთხოვნა, რომელიც დათვლის პროცენტულ წილებს პირობითად 2022 წლისათვის. შედეგად, მოთხოვნა ასეთი იქნება:

```
SELECT [weli], [seqcia_dasaxeleba],
[mshp] / SUM(mshp) OVER (PARTITION BY weli) * 100 mshp_procenti
FROM [statistika].[dbo].[mshp_seqcia2]
WHERE weli = 2022;
```

შედეგი მოცემულია სურათზე 7.8.

**სურათი 7.8. პროცენტული წილები მშპ-სთვის**

Results Messages

| | weli | seqcia_dasaxeleba | mshp_procenti |
|---|---|---|---|
| 1 | 2022 | ადმინისტრაციული და დამხმარე მომსახურების საქმიანობე... | 1.06211990372612 |
| 2 | 2022 | განათლება | 4.47104092558805 |
| 3 | 2022 | განთავსების სამუალებებით უზრუნველყოფის და საკების ... | 3.70886288082629 |
| 4 | 2022 | დამამუშავებელი მრენველობა | 11.2706560941901 |
| 5 | 2022 | ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰა... | 3.20152005532222 |
| 6 | 2022 | ინფორმაცია და კომუნიკაცია | 4.98689476503186 |
| 7 | 2022 | მშენებლობა | 8.00357595201768 |
| 8 | 2022 | პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები | 2.04913903312119 |
| 9 | 2022 | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოც... | 15.9083890372145 |
| 10 | 2022 | სამთომოპოვებითი მრენველობა | 1.41822998478817 |
| 11 | 2022 | საფინანსო და სადაზღვევო საქმიანობები | 4.71545675456483 |
| 12 | 2022 | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო ს... | 6.42002001104251 |
| 13 | 2022 | სოფლის, სატყეო და თევზის მეურნეობა | 6.86783401750476 |
| 14 | 2022 | სხვა სახის მომსახურება | 0.915147432804059 |
| 15 | 2022 | ტრანსპორტი და დასაწყობება | 6.43937374732892 |
| 16 | 2022 | უძრავ ქონებასთან დაკავშირებული საქმიანობები | 10.0458451834828 |
| 17 | 2022 | მიწამეურნეობების, როგორც დაბინავებლის, საქმიანობები; ... | 0.113626635533724 |
| 18 | 2022 | ჩყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბ... | 0.681501145657709 |
| 19 | 2022 | ხელოვნება, გართობა და დასვენება | 4.02483433484171 |
| 20 | 2022 | ჯანდაცვა და სოციალური მომსახურების საქმიანობები | 3.6959321054128 |

ახლა შევქმნათ ახალი ცხრილი, რომელშიც მსგავსი მონაცემები უნდა მოთავსდეს. დავარქვათ მას mshp_procentebi და მივანიჭოთ სტრუქტურა, როგორიც ეს გამოსახულია სურათზე 7.9.

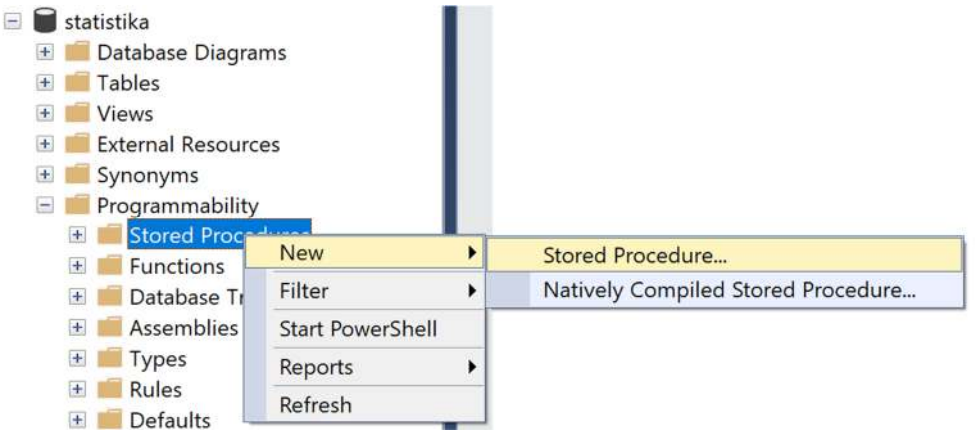
**სურათი 7.9. mshp_procentebi ცხრილის სტრუქტურა**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| weli | smallint | ☑ |
| seqcia_dasaxeleba | nvarchar(200) | ☑ |
| mshp_procenti | float | ☑ |
| | | ☐ |

ცხრილის შექმნის შემდეგ გადავიდეთ პროცედურის შექმნაზე. პროცედურის შესაქმნელად SSMS 19-ის Object Explorer-ში მონაცემთა ბაზის ქვეშ არსებული საქაღალდეებიდან ავირჩიოთ Programmability საქაღალდე, დავაწკაპუნოთ მაუსის მარჯვენა

ღილაკი და ავირჩიოთ New>Stored Procedure..., როგორც ეს გამოსახულია სურათზე 7.10. გამოვა ახალი ჩანართი, როგორიც ნაჩვენებია სურათზე 7.11.

**სურათი 7.10. ახალი პროცედურის შექმნა**




**სურათი 7.11. ახალი პროცედურის შექმნის ჩანართი**

```
-- ================================================
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- ================================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
```

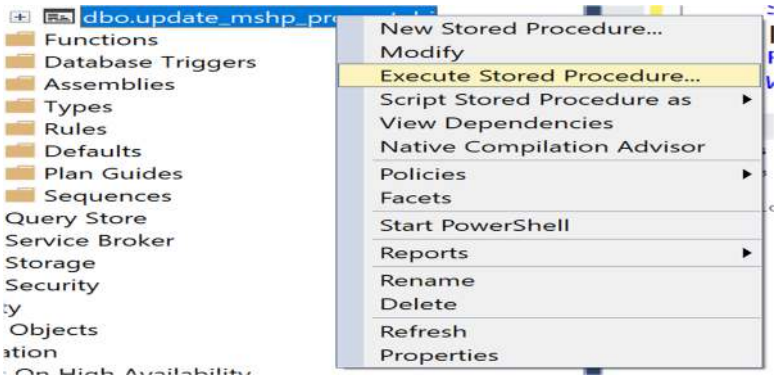
ახალ ჩანართში დავწეროთ პროცედურა და დავარქვათ მას update_mshp_procenti. პროცედურამ უნდა მიიღოს პარამეტრი, ჩვენ შემთხვევაში წელი, და უნდა წაშალოს mshp_procentebi ცხრილიდან ის მონაცემები, რომლის weli სვეტში არსებული ჩანაწერი ემთხვევა მიღებულ პარამეტრს, შემდეგ mshp_procenti ცხრილში შეიტანოს მონაცემები, რომელიც მივიღეთ წარმოდგენიდან, ოღონდ, WHERE ფილტრაციაში უნდა იყოს მითითებული პარამეტრიდან მიღებული მონაცემი weli სვეტით ფილტრაციისას. შენახული პროცედურის შესაქმნელად შესაბამისი მოთხოვნა იქნება:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		სახელი, გვარი
-- Create date: 02/03/2024
-- Description:	ანახლებსmshp_procentebiცხრილს
-- =============================================
CREATE PROCEDURE update_mshp_procentebi
	@weli INT
AS
BEGIN
	SET NOCOUNT ON;
--მონაცემებისწაშლა
 DELETE FROM mshp_procentebi WHERE weli = @weli;

--მონაცემებისჩასმა
	INSERT INTO mshp_procentebi(weli, seqcia_dasaxeleba, mshp_procenti)
	SELECT [weli], [seqcia_dasaxeleba],
	[mshp] / SUM(mshp) OVER (PARTITION BY weli) * 100 mshp_procenti
	FROM [statistika].[dbo].[mshp_seqcia2]
	WHERE weli = @weli;

END
GO
```

გავუშვათ მოთხოვნა. თუ ყველაფერი კარგად წავიდა, მოთხოვნა შეინახება, რომელსაც ვნახავთ მონაცემთა ბაზის შესაბამის საქაღალდეში, ანუ Object Explorer-ში Programmability>Stored Procedures -ში.

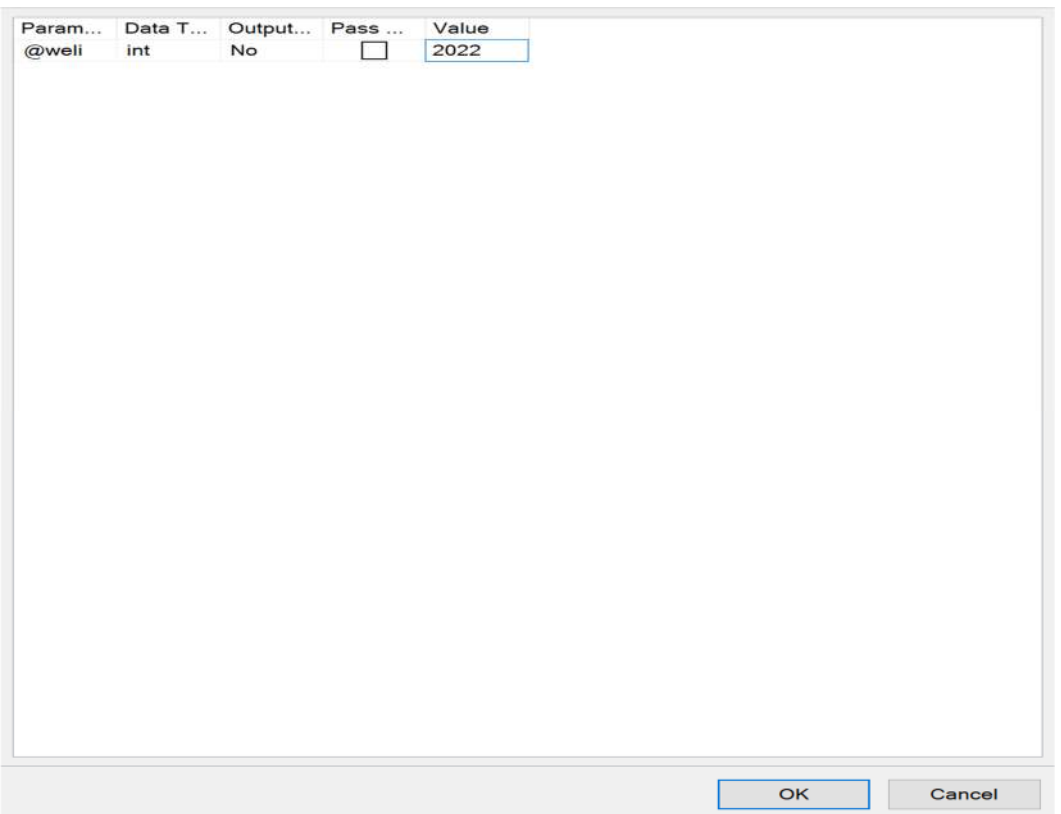
საჭიროების შემთხვევაში შეგვიძლია შენახული პროცედურა გავუშვათ. ამისათვის მასზე ვაწკაპუნებთ მაუსის მარჯვენა ღილაკით, გამოვა კონტექსტური მენიუ, როგორიც ეს გამოსახულია სურათზე 7.12 და ვაწკაპუნებთ Execute Stored Procedure... ღილაკზე.

სურათი 7.12. შენახული პროცედურის კონტექსტური მენიუ



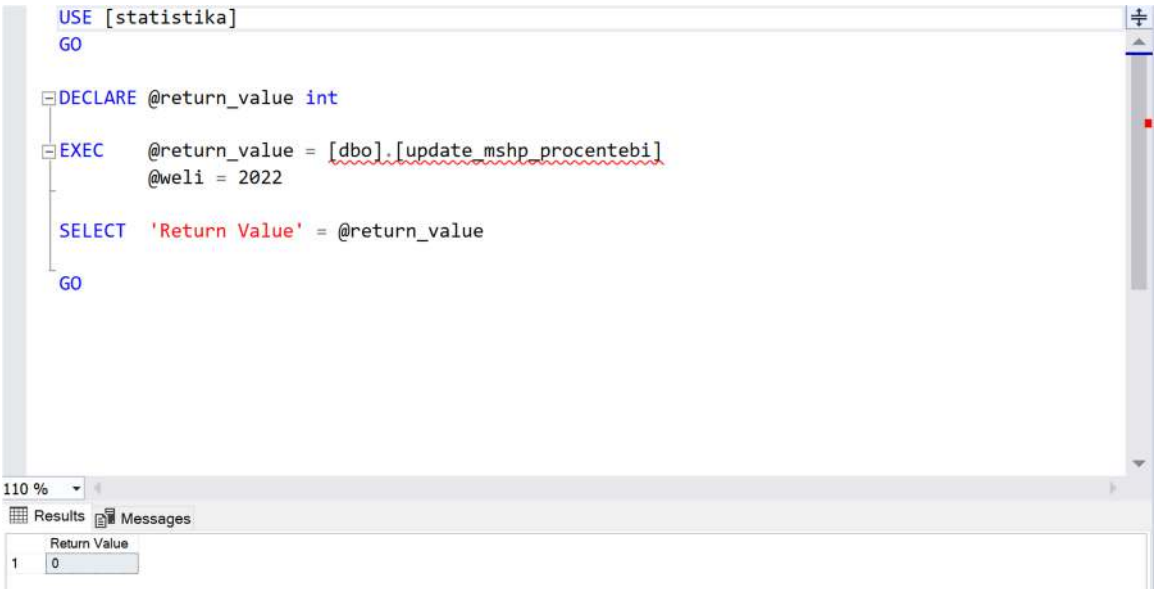

ამ პროცესის შესრულების შემდეგ გამოვა ახალი ფანჯარა, რომელიც General გვერდზე მოგვთხოვს შესაბამისი პარამეტრის შეყვანას. მოცემული გვერდი გამოსახულია სურათზე 7.13.

სურათი 7.13. შენახული პროცედურისთვის პარამეტრების შეყვანა




Value ველში ავირჩიოთ, მაგალითად, 2022, თუ გვსურს 2022 წლის მონაცემების შეტანა ახალ ცხრილში და დავაწკაპუნოთ OK ღილაკს. გამოვა ახალი ჩანართი, რომელიც ავტომატურად დაწერს პროცედურის გაშვების მოთხოვნას და გაუშვებს მას. შედეგებში მივიღებთ ცხრილს, რომელიც აბრუნებს 0-ს, როგორც ეს გამოსახულია სურათზე 7.14.

სურათი 7.14. პროცედურის გაშვების შედეგი




როგორც ვხედავთ, პროცედურის გაშვება შეგვიძლია მოთხოვნითაც, EXEC ბრძანებით, მაგალითად, ჩვენს შემთხვევაში ის იქნება:

```
EXEC [dbo].[update_mshp_procentebi] @weli = 2022;
```

თუ გადავამოწმებთ mshp_procentebi ცხრილს, ვნახავთ, რომ მასში მონაცემები ჩაწერილია.

პროცედურის შექმნისას, თუ გვსურს სხვა პარამეტრების შეყვანაც, მაშინ შეგვიძლია შესაბამისი პარამეტრის სახელი და მისი მონაცემთა ტიპი მძიმით გამოვყოთ პარამეტრიდან „CREATE PROCEDURE proceduris_saxeli“ კოდის შემდეგ. აქვე უნდა აღინიშნოს ისიც, რომ შეიძლება ისეთი პროცედურების შექმნა, რომელიც არ საჭიროებს პარამეტრის მიღებას მომხმარებლისაგან.

შეგვიძლია უკვე შექმნილი პროცედურა შევცვალოთ და წავშალოთ. შეცვლისათვის

სურათზე 7.12 გამოსახული კონტექსტური მენიუდან ავირჩევთ Modify ღილაკს და ახალ გამოსულ ჩანარში დავარედაქტირებთ კოდს, ხოლო წაშლისათვის კი გამოვიყენებთ ამავე კონტექსტურ მენიუში Delete ღილაკს და გამოსულ ფანჯარაში დავაწკაპუნებთ OK ღილაკზე.

## 7.4. ტრანზაქციულობა და მისი გამოყენება

SQL Server 2022-ს გააჩნია ტრანზაქციულობის მხარდაჭერა. სწორედ ტრანზაქციულობა არის საკვანძო საკითხი ACID პირობებთან მიმართებაში, რომელიც გულისხმობს ატომურობას (მთლიანი ოპერაცია სრულდება ერთიანად, ან საერთოდ არ სრულდება), თანმიმდევრულობას (მონაცემთა ბაზა უნდა იყოს თანმიმდევრული

ტრანზაქციამდე და მის შემდეგ), იზოლაციასა (ტრანზაქციები არ უშლიან ხელს ერთმანეთს) და გამძლეობას (ტრანზაქციის შესრულების შემთხვევაში შენახული მონაცემები არ დაიკარგება). ჩვენს შექმნილ პროცედურაში გამოვიყენეთ მონაცემთა წაშლისა და ჩაწერის ოპერაციები. რა შეიძლება მოხდეს იმ შემთხვევაში, თუ მონაცემების ჩაწერისას მოხდა ხარვეზი და ძველი მონაცემები წაიშალა?

ასეთი პრობლემების გადასაწყვეტად შეგვიძლია გამოვიყენოთ ტრანზაქციები. ამ შემთხვევაში ჯერ გამოვიყენებთ TRY...CATCH სინტაქსს. BEGIN TRY ბლოკში დავიწყებთ ტრანზაქციას, შემდეგ დავიწყებთ ოპერაციის განხორციელებას და შემდეგ დავასრულებთ ტრანზაქციას, ავსახავთ მონაცემებს, თუ შესრულდება კარგად. ხოლო BEGIN CATCH ბლოკში ტრანზაქციას უკან დავაბრუნებთ და შედეგად ცხრილში მონაცემები არც წაიშლება და არც შეიცვლება. ამ ბლოკში არსებული კოდის გაშვება მოხდება მაშინ, თუ დაფიქსირდება ხარვეზი. ტრანზაქციის დასაწყებად გამოიყენება BEGIN TRANSACTION TRANSACTION1 კოდი, სადაც TRANSACTION1 არის პირობითი სახელი, რაც შეიძლება დავარქვათ ტრანზაქციას. ტრანზაქციის დასასრულებლად გამოიყენება COMMIT TRANSACTION TRANSACTION1 კოდი, ხოლო მის უკან დასაბრუნებლად კი გამოვიყენებთ ROLLBACK TRANSACTION TRANSACTION1 კოდს. შევცვალოთ ჩვენ მიერ შექმნილი პროცედურა და ამჯერად დავუმატოთ მას ტრანზაქციები. შედეგად, გასაშვები ბრძანება მიიღებს შემდეგ სახეს:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  სახელი, გვარი
-- Create date: 02/03/2024
-- Description:   ანახლებსmshp_procentebiცხრილს
-- =============================================
ALTER PROCEDURE update_mshp_procentebi
        @weli INT
AS
BEGIN
        SET NOCOUNT ON;
--მონაცემებისწაშლა
        BEGIN TRY
                BEGIN TRANSACTION TRANSACTION1;

                DELETE FROM mshp_procentebi WHERE weli = @weli;

    --მონაცემებისჩასმა
                INSERT INTO mshp_procentebi(weli, seqcia_dasaxeleba, mshp_procenti)
                SELECT [weli], [seqcia_dasaxeleba],
                [mshp] / SUM(mshp) OVER (PARTITION BY weli) * 100 mshp_procenti
                FROM [statistika].[dbo].[mshp_seqcia2]
```

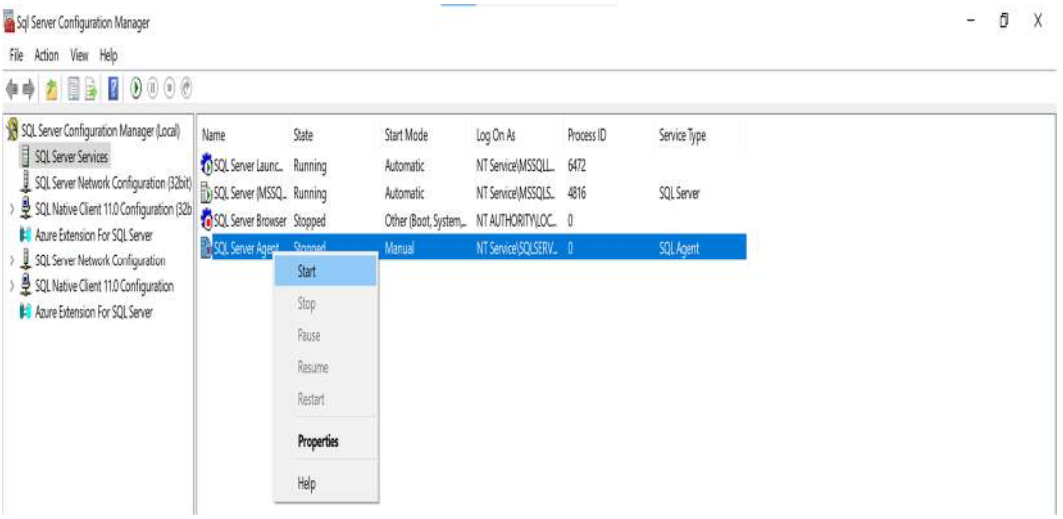
```
                WHERE weli = @weli;

                COMMIT TRANSACTION TRANSACTION1;
        END TRY
        BEGIN CATCH
ROLLBACK TRANSACTION TRANSACTION1;
    --ამკოდითგამოვიტანთშესაბამისხარვეზს
                SELECT ERROR_NUMBER() AS ErrorNumber,
                ERROR_MESSAGE() AS ErrorMessage;
        END CATCH
END
GO
```

ამ მოთხოვნის გაშვების შემდეგ გვექნება განახლებული პროცედურა ტრანზაქციულობის თვისებებით.

## 7.4. SQL Server Agent

წარმოიდგინეთ, რომ ხართ დასაქმებული ისეთ ორგანიზაციაში, რომელიც შედგება სხვადასხვა დეპარტამენტისგან, რომლებიც ამუშავებენ ინფორმაციას და სხვა დეპარტამენტების მიერ დამუშავებული ინფორმაცია შეიძლება თქვენ დაგჭირდეთ. თანამშრომლებს შეუძლიათ დამუშავებული ინფორმაცია გადმოგიგზავნონ, ან, თუ ის მონაცემთა ბაზაშია, ავტომატურად მიიღოთ გარკვეული მოთხოვნების გაშვების საშუალებით. შეიძლება ეს იყოს შენახული პროცედურა და სხვა. დავუშვათ, ერთი დეპარტამენტი აქვეყნებს მონაცემებს ყოველი თვის 23 რიცხვში 11:00 საათზე, ხოლო შენ გჭირდება ეს მონაცემები და ამ მონაცემების საფუძველზე ითვლი სხვადასხვა სტატისტიკურ მაჩვენებლებს. ამოცანის სირთულიდან გამომდინარე, დასათვლელი ცხრილების რაოდენობა შეიძლება ათეულსაც კი ასცდეს. თითოეული ცხრილისთვის შეიძლება არსებობდეს ცალკეული პროცედურა და ამ პროცედურის გაშვების საშუალებით ხდებოდეს ცხრილების განახლება, თუმცა, ამ შემთხვევაში შეიძლება დაიკარგოს დრო, რადგან საჭირო გახდება ინდივიდუალურად გაეშვას პროცედურები. SQL Server Agent-ს შეუძლია გაუშვას მოთხოვნები, როდესაც დადგება შესაბამისი დრო, მაგალითად, 12 საათზე კონკრეტული დეპარტამენტის ბაზიდან შემოიტანოს მონაცემები თქვენი დეპარტამენტის ბაზაში და განახორციელოს სხვადასხვა გაანგარიშებები ავტომატურად და ეს შეიძლება განმეორდეს პერიოდულადაც, მაგალითად, ყოველთვიურად, ყოველკვარტალურად, ყოველწლიურად და ა. შ. ასეთი ავტომატიზაციის განსახორციელებლად საჭიროა გააქტიურებული იყოს SQL Server Agent. ამისათვის გავხსნათ შესაბამის სერვერზე Sql Server Configuration Manager, დავაწკაპუნოთ მაუსის მარჯვენა ღილაკით SQL Server Agent-ს და ავირჩიოთ Start კონტექსტურ მენიუში, როგორც ეს გამოსახულია სურათზე 7.15.

სურათი 7.15. SQL Server Agent-ის ჩართვა



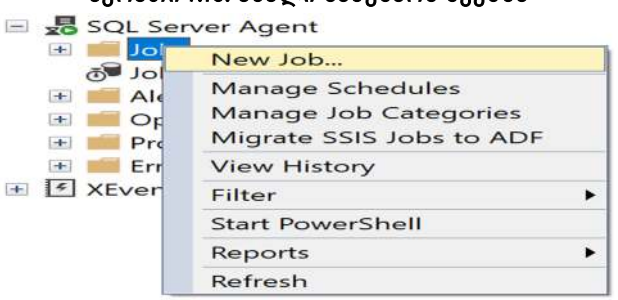

ახლა შეგვიძლია შევქმნათ სხვადასხვა სამუშაო SQL Server Agent-ში, რომელიც ავტომატურად შესრულდება.

მაგალითისთვის შევქმნათ ისეთი სამუშაო, რომელიც ყოველი თვის 23 რიცხვში 12 საათზე გაუშვებს შემდეგ ბრძანებას:

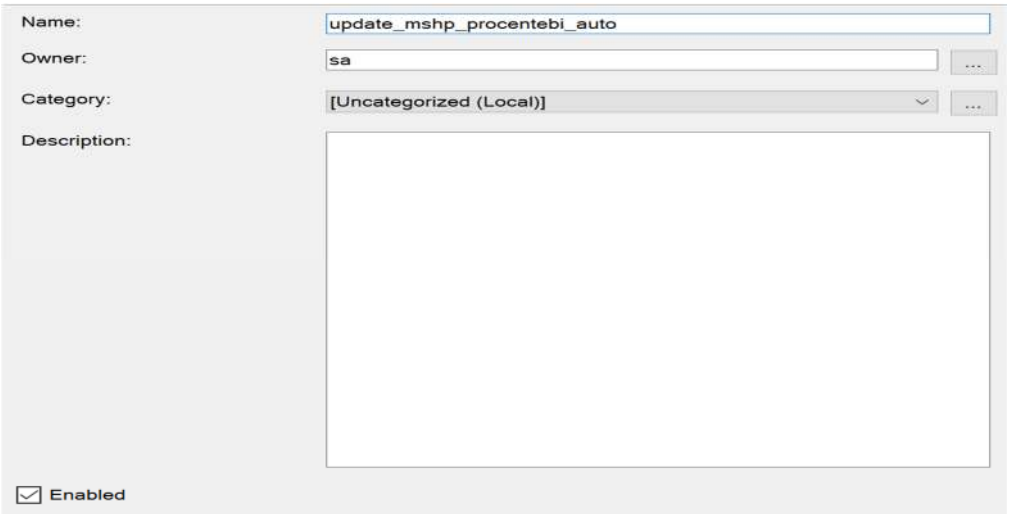
```
EXEC [dbo].[update_mshp_procentebi] @weli = 2022;
```

ამისთვის SSMS 19-ში SQL Server Agent>Jobs-ისთვის გავხსნით კონტექსტურ მენიუს და ავირჩევთ New Job...-ს, როგორც ეს გამოსახულია სურათზე 7.16. გამოვა ფანჯარა, რომლის General გვერდზე ავირჩევთ სამუშაოს სახელს, მესაკუთრეს, კატეგორიასა და აღწერას, როგორც ეს ნაჩვენებია სურათზე 7.17.

სურათი 7.16. ახალი სამუშაოს შექმნა



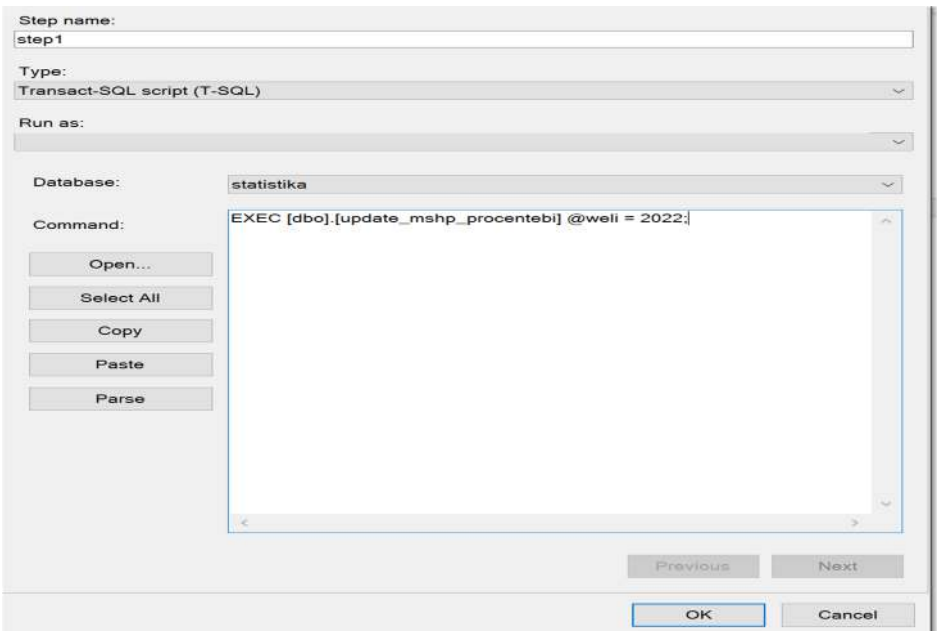
სურათი 7.17. ახალი სამუშაოს შექმნა, General გვერდი



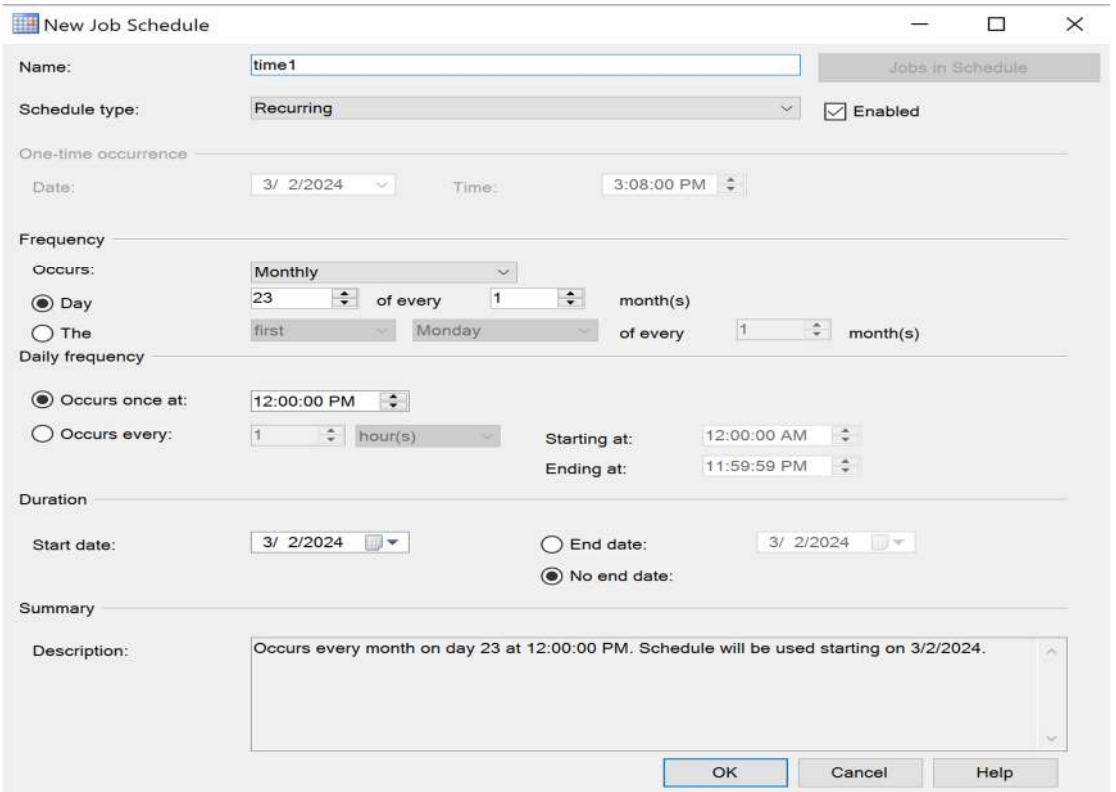

ამის შემდეგ გადავდივართ Steps გვერდზე, სადაც ვაწკაპუნებთ New... ღილაკზე, გამოვა ფანჯარა, რომელიც გამოსახულია სურათზე 7.18, სადაც ვირჩევთ ნაბიჯის სახელს, მონაცემთა ბაზას და გასაშვებ ბრძანებას, როგორც ეს მოცემულია სურათზე და ვაწკაპუნებთ OK ღილაკზე. გადავდივართ Schedules გვერდზე, სადაც ვაწკაპუნებთ New ღილაკზე და გამოვა ახალი ფანჯარა, როგორც ეს გამოსახულია სურათზე 7.19. ყველაფერს ვირჩევთ ისე, როგორც გამოსახულია სურათზე 7.19, ვაწკაპუნებთ OK ღილაკზე და შემდეგ New Job ფანჯრის OK ღილაკზე. შედეგად შეიქმნება ახალი სამუშაო, რომელიც შეგვიძლია მოვნახოთ SQL Server Agent>Jobs საქაღალდეში SSMS 19-ის Object Explorer-ში.

ამრიგად, ჩვენ შევქმენით ახალი სამუშაო და შეგვიძლია ამ პროცესის ავტომატიზაცია. მის წასაშლელად შესაბამისი სამუშაოს კონტექსტური მენიუდან ვიმოქმედოთ Delete ღილაკით.

სურათი 7.18. ნაბიჯების შექმნა



სურათი 7.19. დროის არჩევა




### *კითხვები თვითშემოწმებისთვის:*

1. მოკლედ მიმოიხილეთ პროგრამირების პარადიგმები.
2. აღწერეთ, თუ როგორ ხდება ცვლადების გამოცხადება T-SQL-ში.
3. აღწერეთ WHILI ციკლი და IF განაცხადები.
4. როგორ იქმნება შენახული პროცედურები?
5. რას ნიშნავს ტრანზაქციულობა და როგორ გამოვიყენოთ ის?
6. აღწერეთ, რაში გამოიყენება SQL Server Agent?
7. როგორ ხდება ბრძანებების ავტომატიზება SQL Server Agent-ით?

# თავი 8. სხვადასხვა დამხმარე ფუნქცია

## 8.1. მათემატიკური და სტატისტიკური ფუნქციები

T-SQL-ში მოცემულია სხვადასხვა მათემატიკური და სტატისტიკური ფუნქცია, რომელიც შეგვიძლია გამოვიყენოთ მონაცემებთან მუშაობისას. ამ ქვეთავში მიმოვიხილავთ ზოგიერთ მათგანს.

წინა თავებში მუშაობისას განვიხილეთ ისეთი აგრეგირებული ფუნქციები, როგორიცაა SUM (ჯამი), COUNT (დათვლა), STDEVP (სტანდარტული გადახრა პოპულაციისთვის), STDEV (სტანდარტული გადახრა შერჩევისთვის), MIN (მინიმალური მნიშვნელობა) და MAX (მაქსიმალური მნიშვნელობა). ვნახეთ, თუ როგორ ხდება აგრეგირებული ფუნქციების გამოყენება და როგორ შეიძლება დაგვეხმარონ ისინი სხვადასხვა ამოცანის გადასაჭრელად.

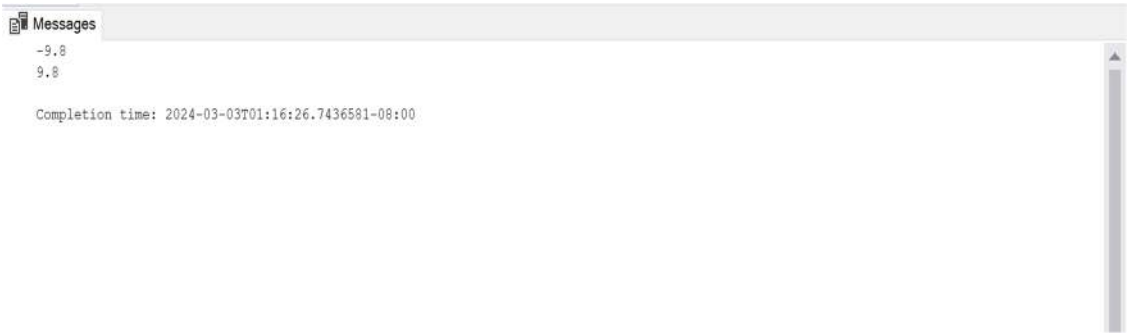
სხვა სტატისტიკური ფუნქციები, რომლებსაც ვიყენებთ T-SQL-ში აგრეგირებული ფუნქციების მსგავსად, არის: VARP (დისპერსია პოპულაციისთვის) და VAR (დისპერსია შერჩევისთვის).

განვიხილოთ მათემატიკური ფუნქციები: მაგალითად, შეიძლება დაგვჭირდეს აბსოლუტური მნიშვნელობა რომელიმე სვეტისთვის. ამ შემთხვევაში გამოვიყენებთ ABS ფუნქციას. დავუშვათ, შემოვიტანეთ რაიმე ათწილადი, პირობითად -9.8; მისი აბსოლუტური მნიშვნელობის გასაგებად გამოვიყენებთ შემდეგ კოდს:

```
DECLARE @num FLOAT = -9.8;
PRINT(@num);
PRINT(ABS(@num));
```

აქ შემოვიტანეთ ცვლადი num, მივანიჭეთ მნიშვნელობა, დავბეჭდეთ როგორც თავდაპირველი, ისე მათემატიკური გარდაქმნის შედეგად მიღებული მნიშვნელობა. ეს ფუნქცია, ასევე, შეგვიძლია გამოვიყენოთ ცხრილებთან მუშაობისას. შედეგი გამოსახულია სურათზე 8.1.

სურათი 8.1. რიცხვის აბსოლუტური მნიშვნელობის გაანგარიშება



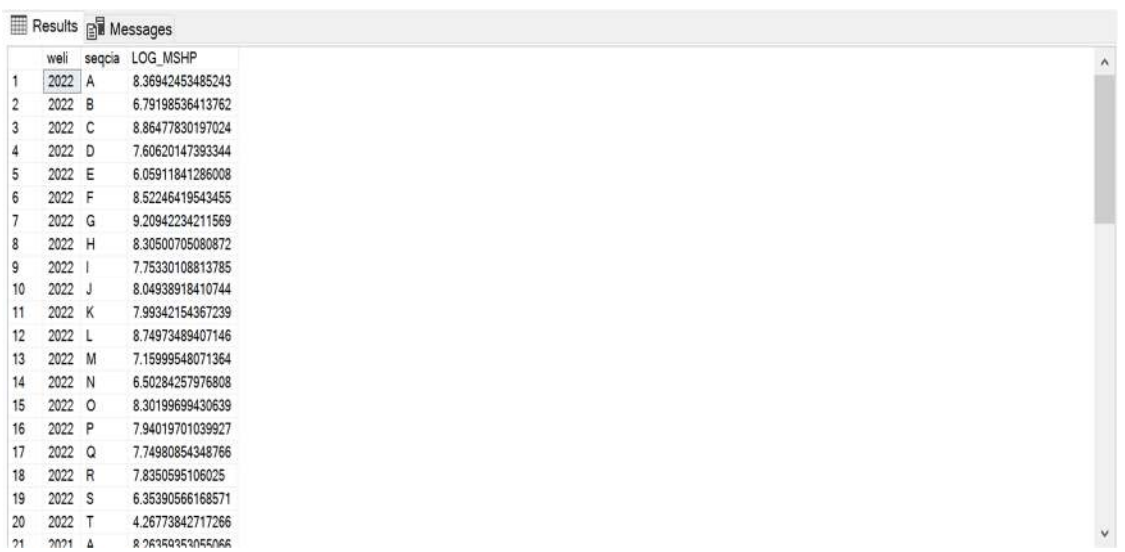

მხარდაჭერილია, ასევე, ტრიგონომეტრიული ფუნქციებიც. ესენია: ACOS (არქკოსინუსი), ASIN (არქსინუსი), ATAN (არქტანგესი), COS (კოსინუსი), COT (კოტანგესი), SIN (სინუსი), TAN (ტანგესი), RADIANS (გრადუსების რადიანებში გადაყვანა), DEGRESS (რადიანების გრადუსებში გადაყვანა), PI (აბრუნებს PI-ის მნიშვნელობას).

ეკონომიკაში ხშირია გარკვეული ცვლადების ლოგარითმებში გამოსახვა, მაგალითად, ეს შეიძლება დაგვჭირდეს აპროქსიმაციისთვის და გარკვეული განტოლებების გაწრფივებისთვის. დავუშვათ, გვსურს mshp სვეტში არსებული მონაცემების გალოგარითმება ნატურალური ლოგარითმით mshp_seqcia ცხრილში. ამისათვის შეგვიძლია გამოვიყენოთ LOG ფუნქცია. შედეგად, მოთხოვნა ასეთი იქნება:

```
SELECT [weli], [seqcia],
LOG([mshp]) LOG_MSHP
FROM [statistika].[dbo].[mshp_seqcia];
```

მოთხოვნის შედეგი გამოსახულია სურათზე 8.2.

სურათი 8.2. მონაცემთა გალოგარითმების შედეგი




| | weli | seqcia | LOG_MSHP |
|---|---|---|---|
| 1 | 2022 | A | 8.36942453485243 |
| 2 | 2022 | B | 6.79198536413762 |
| 3 | 2022 | C | 8.86477830197024 |
| 4 | 2022 | D | 7.60620147393344 |
| 5 | 2022 | E | 6.05911841286008 |
| 6 | 2022 | F | 8.52246419543455 |
| 7 | 2022 | G | 9.20942234211569 |
| 8 | 2022 | H | 8.30500705080872 |
| 9 | 2022 | I | 7.75330108813785 |
| 10 | 2022 | J | 8.04938918410744 |
| 11 | 2022 | K | 7.99342154367239 |
| 12 | 2022 | L | 8.74973489407146 |
| 13 | 2022 | M | 7.15999548071364 |
| 14 | 2022 | N | 6.50284257976808 |
| 15 | 2022 | O | 8.30199699430639 |
| 16 | 2022 | P | 7.94019701039927 |
| 17 | 2022 | Q | 7.74980854348766 |
| 18 | 2022 | R | 7.8350595106025 |
| 19 | 2022 | S | 6.35390566168571 |
| 20 | 2022 | T | 4.26773842717266 |
| 21 | 2021 | A | 8.26359353055066 |

ამავე ფუნქციით შეგვიძლია არა მარტო ნეპერის რიცხვის ფუძით გავალოგარითმოთ, არამედ სხვა ფუძეც ავირჩიოთ, მაგალითად, 2. ამისათვის LOG ფუნქციას მივაწვდით მეორე არგუმენტს, სადაც მივუთითებთ ფუძეს და ამ არგუმენტს გამოვყოფთ მძიმით. შედეგად შესაბამისი ბრძანება იქნება:

```
SELECT [weli], [seqcia],
LOG([mshp], 2) LOG_MSHP
FROM [statistika].[dbo].[mshp_seqcia];
```

შედეგი იხილეთ სურათზე 8.3.

**სურათი 8.3. მონაცემთა გალოგარითმება ორობითი ფუძით**

Results Messages

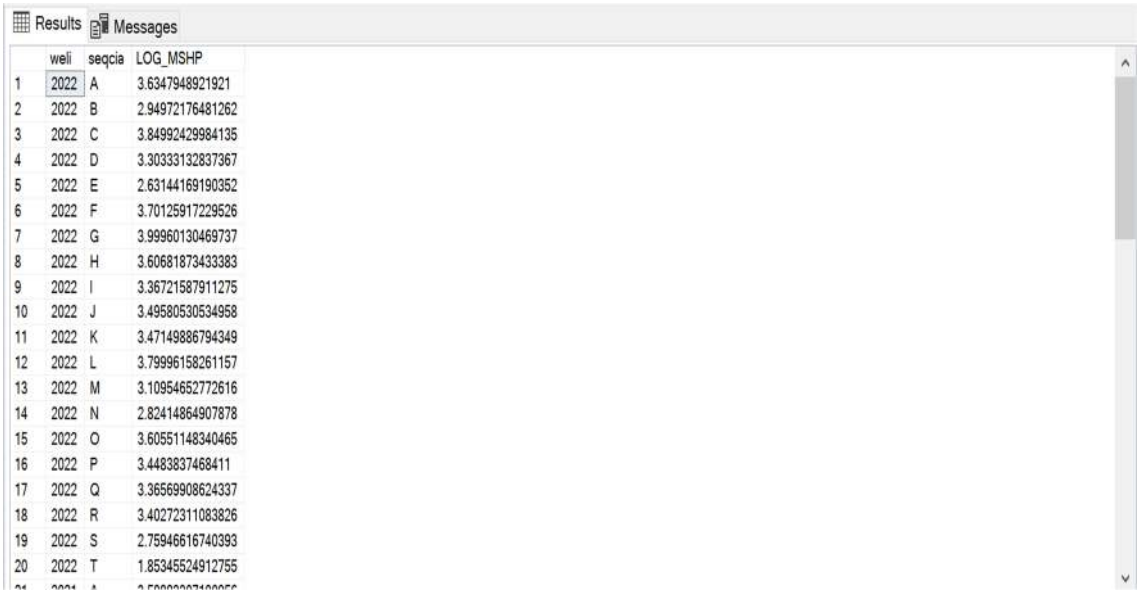
| | weli | seqcia | LOG_MSHP |
|---|---|---|---|
| 1 | 2022 | A | 12.074527271526 |
| 2 | 2022 | B | 9.79876360263176 |
| 3 | 2022 | C | 12.7891716948325 |
| 4 | 2022 | D | 10.9734291464461 |
| 5 | 2022 | E | 8.74146008639224 |
| 6 | 2022 | F | 12.2953168309072 |
| 7 | 2022 | G | 13.2863879424223 |
| 8 | 2022 | H | 11.9815924867496 |
| 9 | 2022 | I | 11.1856490303755 |
| 10 | 2022 | J | 11.6128138580971 |
| 11 | 2022 | K | 11.5320696207912 |
| 12 | 2022 | L | 12.62319914077 |
| 13 | 2022 | M | 10.329689972813 |
| 14 | 2022 | N | 9.381618741513 |
| 15 | 2022 | O | 11.9772498931609 |
| 16 | 2022 | P | 11.4552828505844 |
| 17 | 2022 | Q | 11.1806103535286 |
| 18 | 2022 | R | 11.3036015010161 |
| 19 | 2022 | S | 9.16674818839028 |
| 20 | 2022 | T | 6.15704506469326 |
| 21 | 2021 | A | 11.9218454064476 |

მონაცემთა გალოგარითმება შეგვიძლია ფუძით 10 უფრო მარტივი ფუნქციით, ვიდრე LOG ფუნქციაა. ამისათვის მხოლოდ LOG10 ფუნქციას გამოვიყენებთ და შედეგად, ჩვენ შემთხვევაში, კოდი იქნება ასეთი:

```
SELECT [weli], [seqcia],
LOG10([mshp]) LOG_MSHP
FROM [statistika].[dbo].[mshp_seqcia];
```

შედეგი იხილეთ სურათზე 8.4.

სურათი 8.4. მონაცემთა გალოგარითმება LOG10 ფუნქციით



Results Messages

| | weli | seqcia | LOG_MSHP |
|---|---|---|---|
| 1 | 2022 | A | 3.6347948921921 |
| 2 | 2022 | B | 2.94972176481262 |
| 3 | 2022 | C | 3.84992429984135 |
| 4 | 2022 | D | 3.30333132837367 |
| 5 | 2022 | E | 2.63144169190352 |
| 6 | 2022 | F | 3.70125917229526 |
| 7 | 2022 | G | 3.99960130469737 |
| 8 | 2022 | H | 3.60681873433383 |
| 9 | 2022 | I | 3.36721587911275 |
| 10 | 2022 | J | 3.49580530534958 |
| 11 | 2022 | K | 3.47149886794349 |
| 12 | 2022 | L | 3.79996158261157 |
| 13 | 2022 | M | 3.10954652772616 |
| 14 | 2022 | N | 2.82414864907878 |
| 15 | 2022 | O | 3.60551148340465 |
| 16 | 2022 | P | 3.4483837468411 |
| 17 | 2022 | Q | 3.36569908624337 |
| 18 | 2022 | R | 3.40272311083826 |
| 19 | 2022 | S | 2.75946616740393 |
| 20 | 2022 | T | 1.85345524912755 |

გარდა გალოგარითმებისა, შეგვიძლია მონაცემების ახარისხებაც. ამისათვის გამოიყენება POWER ფუნქცია, რომლის პირველი არგუმენტი იქნება ის რიცხვი, რომლის ახარისხებაც გვინდა, ხოლო მეორე, თუ რა ხარისხში გვინდა ავახარისხოთ. დავუშვათ, გვსურს რიცხვი 2-ის მესამე ხარისხში აყვანა. ამისათვის მოთხოვნა ასეთი იქნება:

```
SELECT POWER(2,3);
```

თუ გვსურს რიცხვის კვადრატი, მაშინ შეგვიძლია გამოვიყენოთ SQUARE ფუნქცია. მაგალითად, რიცხვი 8-ის კვადრატის საპოვნელად გამოვიყენებთ მოთხოვნას:

```
SELECT SQUARE(8);
```

T-SQL-ის ერთ-ერთი ფუნქცია ასევე არის მაჩვენებლიანი ფუნქცია EXP, რომლიდანაც შეგვიძლია მივიღოთ $e^x$, სადაც $x$ არის არგუმენტი. დავუშვათ $x$-ის მნიშვნელობა არის 8, მაშინ შესაბამისი მოთხოვნა ამ ფუნქციის მნიშვნელობის მისაღებად იქნება:

[

```
SELECT EXP(8);
```

ზოგადად, შეგვიძლია ფუნქციების კომპოზიციაც. მაგალითად, დავუშვათ გავალოგარითმეთ რაიმე რიცხვი, მაგალითად 10 ფუძით e და გვსურს ამ ფუნქციის მნიშვნელობა მიეწოდოს EXP ფუნქციას. მაშინ შესაბამისი კოდი იქნება:

```
SELECT EXP(LOG(10));
```

შედეგად, ამ მოთხოვნის გაშვების შემდეგ მივიღებთ რიცხვს 10.

ასევე, შეგვიძლია ამოვიღოთ კვადრატული ფესვი რაიმე რიცხვიდან SQRT ფუნქციით, დავუშვათ 64-დან. ამისათვის შესაბამისი მოთხოვნა იქნება:

```
SELECT SQRT(64);
```

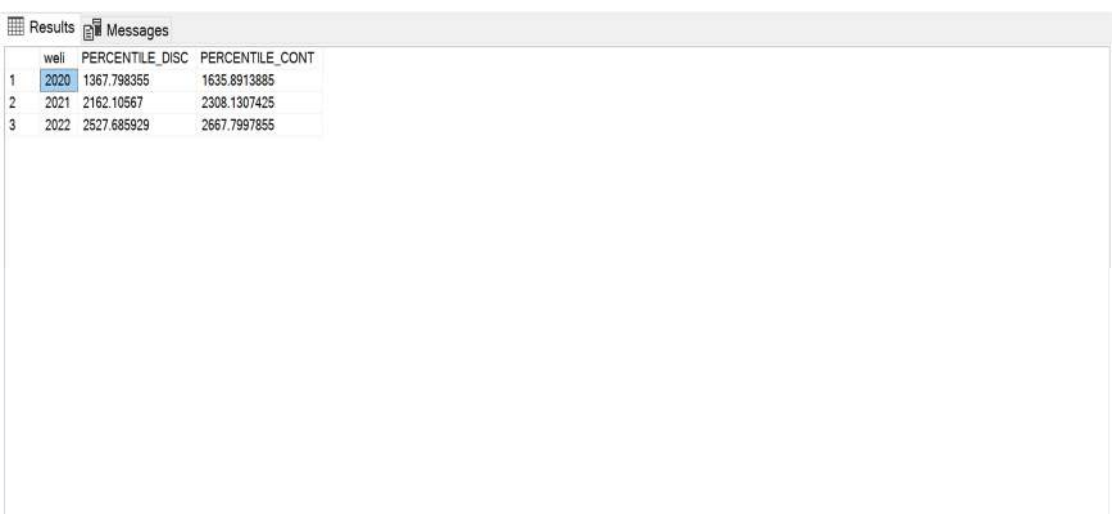
რაც შეეხება დამრგვალებებს, აქ გვაქვს სამი შემთხვევა: 1) კლებადობით დამრგვალებისთვის გამოიყენება FLOOR ფუნქცია; 2) მეტობით დამრგვალებისთვის CEILING ფუნქცია და 3) თუ გვსურს დამრგვალება სპეციფიკური სიზუსტით, მაშინ გამოვიყენებთ ROUND ფუნქციას, სადაც პირველი არგუმენტი იქნება რიცხვი, ხოლო მეორე - სიზუსტე.

განვიხილოთ შედარებით რთული ფუნქცია და ვიპოვოთ პერცენტილი მონაცემებში. მედიანის საპოვნელად უნდა ავიღოთ 50-ე პერცენტილი. პერცენტილი T-SQL-ში შესაძლებელია იყოს როგორც უწყვეტი, ისე დისკრეტული. დავთვალოთ ორივე მათგანი mshp_seqcia ცხრილის მიხედვით თითოეული წლისათვის. ამისთვის ჩვენი მოთხოვნა ასეთი იქნება:

```
SELECT DISTINCT [weli],
PERCENTILE_DISC (0.5) WITHIN GROUP (ORDER BY mshp) OVER (PARTITION BY weli) PERCENTILE_DISC,
PERCENTILE_CONT (0.5) WITHIN GROUP (ORDER BY mshp) OVER (PARTITION BY weli) PERCENTILE_CONT
FROM [statistika].[dbo].[mshp_seqcia];
```

მიღებული შედეგი გამოსახულია სურათზე 8.5.

**სურათი 8.5. 50-ე პერცენტილის პოვნა**


Results Messages

| | weli | PERCENTILE_DISC | PERCENTILE_CONT |
|---|---|---|---|
| 1 | 2020 | 1367.798355 | 1635.8913885 |
| 2 | 2021 | 2162.10567 | 2308.1307425 |
| 3 | 2022 | 2527.685929 | 2667.7997855 |

## 8.2. ფუნქციები თარიღებთან სამუშაოდ

T-SQL-ის გამოყენებით შეგვიძლია ვიმუშაოთ თარიღებთანაც. აქედან მიმოვიხილავთ რამდენიმეს.

მიმდინარე თარიღის მისაღებად გამოიყენება GETDATE() ფუნქცია. შესაბამისად, თუ გვსურს გავიგოთ მიმდინარე თარიღი თავისი დროით, მაშინ ამისათვის შესაბამისი მოთხოვნა იქნება:

```
SELECT GETDATE();
```

ჩვენ შეგვიძლია ამ მონაცემიდან ამოვიღოთ წამი (და უფრო მცირე დროის ნაწილებიც), წუთი, საათი, დღე, კვირა, თვე, კვარტალი და წელი. ამისათვის უნდა გამოვიყენოთ DATEPART ფუნქცია. თუ გვსურს ამოვიღოთ კვარტალი, მაშინ შესაბამისი მოთხოვნა ასეთი იქნება:

```
SELECT DATEPART(QUARTER, GETDATE());
```

სხვა დროითი ნაწილების ამოსაღებად QUARTER-ს შევცვლით მოთხოვნაში შესაბამისი დროის მონაკვეთით, რომელიც მოთავსებული ცხრილში 8.1. იხილეთ არგუმენტის სვეტი.

**ცხრილი 8.1. დროითი მონაკვეთები DATEPART ფუნქციისთვის**

| დასახელება | არგუმენტი |
|---|---|
| **წამი** | SECOND |
| **წუთი** | MINUTE |
| **საათი** | HOUR |
| **დღე** | DAY |
| **კვირა** | WEEK |
| **თვე** | MONTH |
| **კვარტალი** | QUARTER |
| **წელი** | YEAR |

წინა თავში შევქმენით პროცედურა და შემდეგ მოვახდინეთ მისი ავტომატიზაცია. დავუშვათ, გვჭირდება, რომ პროცედურა გამეორდეს ყოველ სამ თვეში, ავტომატურად მოიძიოს თარიღი, აქედან ამოიღოს კვარტალი და წელი და შემდეგ გამოიძახოს პროცედურა, რომელსაც მიაწვდის პარამეტრს, რომელიც იქნება შესაბამისად წინა კვარტალი და, შესაძლოა, წინა წელიც. შესაბამისად, ავტომატურად გადაითვლის მთელ რიგ მონაცემებს. ამ ყველაფერში გვეხმარება დროითი ფუნქციებიც. ასე რომ, ავტომატიზებაში დროითი ფუნქციების გამოყენება საკმაოდ მნიშვნელოვანია.

## 8.3. ფუნქციები ტექსტებთან სამუშაოდ

როგორც ცნობილია, SQL Server 2022-ში შესაძლებელია ტექსტურ მონაცემებთან მუშაობა. ამ ქვეთავში განვიხილავთ ზოგიერთ დამხმარე ფუნქციას ტექსტურ მონაცემებთან სამუშაოდ.

ტექსტების გასაერთიანებლად გამოიყენება **CONCAT** ფუნქცია. გავაერთიანოთ nace2

ცხრილში მოთავსებული seqcia და seqcia_dasaxeleba სვეტებში არსებული მონაცემები. ამისთვის შესაბამისი კოდი იქნება:

```
SELECT CONCAT([seqcia], [seqcia_dasaxeleba])
FROM [statistika].[dbo].[nace2];
```

შედეგი იხილეთ სურათზე 8.6. აქ გავაერთიანეთ ორი ტექსტი, თუმცა, შესაძლებელია უფრო მეტი ტექსტური მონაცემის გაერთიანება.

**სურათი 8.6 CONCAT ფუნქციის შედეგი**

Results Messages

| | (No column name) |
|---|---|
| 1 | A სოფლის, სატყეო და თევზის მეურნეობა |
| 2 | B სამთომოპოვებითი მრეწველობა |
| 3 | C დამამუშავებელი მრეწველობა |
| 4 | D ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰ... |
| 5 | E წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და და... |
| 6 | F მშენებლობა |
| 7 | G საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტო... |
| 8 | H ტრანსპორტი და დასაწყობება |
| 9 | I განთავსების სამუალებებით უზრუნველყოფის და საკვების ... |
| 10 | J ინფორმაცია და კომუნიკაცია |
| 11 | K საფინანსო და სადაზღვევო საქმიანობები |
| 12 | L უძრავ ქონებასთან დაკავშირებული საქმიანობები |
| 13 | M პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები |
| 14 | N ადმინისტრაციული და დამხმარე მომსახურების საქმიანო... |
| 15 | O სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო... |
| 16 | P განათლება |
| 17 | Q ჯანდაცვა და სოციალური მომსახურების საქმიანობები |
| 18 | R ხელოვნება, გართობა და დასვენება |
| 19 | S სხვა სახის მომსახურება |
| 20 | T შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები... |

ზოგჯერ შეიძლება საჭირო გახდეს ტექსტის გარდაქმნა ისე, რომ არსებული ტექსტი შეიცვალოს შესაბამისი პატარა ასოებით, ან დიდი ასოებით. პატარა ასოების ტექსტად გარდაქმნისათვის გამოიყენება LOWER ფუნქცია, ხოლო დიდ ასოებად გარდაქმნისათვის კი ვიყენებთ UPPER ფუნქციას.

ტექსტურ მონაცემში სიმბოლოების რაოდენობის დასათვლელად გამოიყენება LEN ფუნქცია. მაგალითად, nace2 ცხრილში გავიგოთ seqcia_dasaxeleba სვეტში არსე-

ბულ თითოეულ ჩანაწერში სიმბოლოების რაოდენობა. ამისთვის გამოვიყენებთ ასეთ კოდს:

```
SELECT [seqcia_dasaxeleba],
LEN(seqcia_dasaxeleba) simboloebi
FROM [statistika].[dbo].[nace2];
```

შედეგი იხილეთ სურათზე 8.7.

**სურათი 8.7. LEN ფუნქციის შედეგი**

Results Messages

| | seqcia_dasaxeleba | simboloebi |
|---|---|---|
| 1 | სოფლის, სატყეო და თევზის მეურნეობა | 34 |
| 2 | სამთომომპოვებითი მრეწველობა | 26 |
| 3 | დამამუშავებელი მრეწველობა | 25 |
| 4 | ელექტროენერგიის, აირის, ორთქლის და კონდიცირებული ჰა... | 64 |
| 5 | წყალმომარაგება; კანალიზაცია, ნარჩენების მართვა და დაბ... | 92 |
| 6 | მშენებლობა | 10 |
| 7 | საბითუმო და საცალო ვაჭრობა; ავტომობილების და მოტოც... | 65 |
| 8 | ტრანსპორტი და დასაწყობება | 25 |
| 9 | განთავსების საშუალებებით უზრუნველყოფის და საკვების ... | 73 |
| 10 | ინფორმაცია და კომუნიკაცია | 25 |
| 11 | საფინანსო და სადაზღვევო საქმიანობები | 36 |
| 12 | უძრავ ქონებასთან დაკავშირებული საქმიანობები | 43 |
| 13 | პროფესიული, სამეცნიერო და ტექნიკური საქმიანობები | 48 |
| 14 | ადმინისტრაციული და დამხმარე მომსახურების საქმიანობე... | 53 |
| 15 | სახელმწიფო მმართველობა და თავდაცვა; სავალდებულო ს... | 69 |
| 16 | განათლება | 9 |
| 17 | ჯანდაცვა და სოციალური მომსახურების საქმიანობები | 47 |
| 18 | ხელოვნება, გართობა და დასვენება | 31 |
| 19 | სხვა სახის მომსახურება | 22 |
| 20 | შინამეურნეობების, როგორც დამქირავებლის, საქმიანობები; ... | 153 |

ზოგჯერ შეიძლება დაგვჭირდეს კონკრეტული ტექსტიდან მისი ნაწილის ამოჭრა. ამისათვის შეგვიძლია გამოვიყენოთ LEFT, RIGHT და SUBSTRING ფუნქციები. RIGHT ფუნქცია არგუმენტებად იღებს კონკრეტულ ტექსტურ მონაცემს და სიმბოლოების რაოდენობას, რომელიც უნდა ამოვიდეს მარჯვნიდან, მაგალითად, სექციის დასახელებიდან თითოეული სექციის ბოლო 8 ასოს ამოსაღებად გამოვიყენებთ კოდს:

```
SELECT RIGHT([seqcia_dasaxeleba], 8)
FROM [statistika].[dbo].[nace2];
```

შედეგი გამოსახულია სურათზე 8.8.

სურათი 8.8. RIGHT ფუნქციის შედეგი

| | (No column name) |
|---|---|
| 1 | ეურზეობა |
| 2 | ენველობა |
| 3 | ენველობა |
| 4 | მიწოდება |
| 5 | იანობები |
| 6 | ენებლობა |
| 7 | რემონტი |
| 8 | ანყობება |
| 9 | ანობები |
| 10 | უნიჯაცია |
| 11 | იანობები |
| 12 | იანობები |
| 13 | იანობები |
| 14 | ანობები |
| 15 | ფრთხოება |
| 16 | ანათლება |
| 17 | იანობები |
| 18 | ასვეზება |
| 19 | საბურება |
| 20 | ბისათვის |

LEFT ფუნქციაც მსგავსად მუშაობს, ოღონდ ამ შემთხვევაში მარცხნიდან ტექსტის იმ სიმბოლოების რაოდენობას ვიღებთ, რომელსაც მივუთითებთ მეორე არგუმენტად. SUBSTRING ფუნქცია ტექსტის შუა ნაწილიდან იღებს ქვეტექსტს, იღებს სამ არგუმენტს: ტექსტურ მონაცემს, დასაწყისს და ზომას. მაგალითად, სექციის დასახელებიდან თუ გვსურს, რომ ამოვიღოთ ქვეტექსტი, რომელიც დაიწყება მეორე ასოდან და ამოიღებს 10 სიმბოლოს, მაშინ შესაბამის კოდი იქნება:

```
SELECT SUBSTRING([seqcia_dasaxeleba], 2, 10)
FROM [statistika].[dbo].[nace2];
```

ზოგჯერ შეიძლება ტექსტური მონაცემების შეყვანის დროს მოხდეს შეცდომა და მარცხნივ ან მარჯვნივ ტექსტს ჩაეწერა გამოტოვებები, რამაც შეიძლება გამოიწვიოს სხვადასხვა პროგრამის მუშაობა ხარვეზებით. ასეთი გამოტოვებების მოსაშორებლად გამოიყენება LTRIM და RTRIM ფუნქციები, რომელიც შესაბამისად, მარცხნიდან და მარჯვნიდან შემოჭრის გამოტოვებებს. ამასთან, შესაძლებელია LTRIM ფუნქციამ არგუმენტად მიიღოს RTRIM ფუნქციის შედეგი და პირიქით.

ტექსტის შესაბრუნებლად გამოიყენება REVERSE ფუნქცია, რომელიც იღებს არგუმენტად ტექსტურ მონაცემებს და აბრუნებს მის შებრუნებულ ვერსიას.

ტექსტში გარკვეული ქვეტექსტის ჩასანაცვლებლად გამოიყენება REPLACE ფუნქცია. ეს ფუნქცია იღებს სამ არგუმენტს: ტექსტურ მონაცემს, რომელშიც უნდა მოხდეს ჩანაცვლება; ქვეტექსტს, რომელიც მოთავსებულია ტექსტურ მონაცემში და გვსურს ამ ტექსტის ჩანაცვლება და მესამე არგუმენტი, თუ რით შეგვიძლია შევცვალოთ ქვეტექსტი ტექსტურ მონაცემში. მაგალითად, თუ გვსურს შევცვალოთ სექციის დასახელებაში A სექციის მონაცემში სიტყვა „სოფლის“ სიტყვა „სოფლების“ ტექსტით, მაშინ შესაბამისი კოდი იქნება:

```
SELECT REPLACE([seqcia_dasaxeleba], N'სოფლის', N'სოფლების')
FROM [statistika].[dbo].[nace2];
```

***კითხვები თვითშემოწმებისთვის:***

1. ჩამოთვალეთ, რომელ მათემატიკურ და სტატისტიკურ ფუნქციებს იცნობთ T-SQL-ში?
2. დაასახელეთ ფუნქციები, რომლებიც გვეხმარება თარიღებთან მუშაობაში.
3. განიხილეთ ტექსტებთან მუშაობის დამხმარე ფუნქციები.

# თავი 9. ფუნქციები და ტრიგერები

## 9.1. ფუნქციები

წინა თავებში T-SQL-ში განვიხილეთ ცხრილები, წარმოდგენები, შენახული პროცედურები, ფუნქციები და სხვა. კიდევ ერთი საინტერესო თემაა თვით ფუნქციების შექმნა და გამოყენება.

ზოგჯერ შეიძლება რთული გაანგარიშებები იყოს საჭირო, იქნება ეს კონკრეტული ინდექსის გამოთვლა თუ სხვა მაჩვენებელი და ეს ოპერაცია მრავალჯერ დაგვჭირდეს. ამისათვის კარგი იქნება, თუ შეიქმნება ისეთი ფუნქცია, რომელიც დაგვეხმარება გაანგარიშებებში.

არსებობს ფუნქციების სხვადასხვა სახე. ზოგიერთი შესაძლებელია აბრუნებდეს ცხრილებს, ხოლო ზოგი კი სკალარულ მონაცემს. სიმარტივისთვის შევქმნათ ფუნქცია, რომელიც დააბრუნებს სკალარულ მონაცემს. მაგალითად, შევქმნათ ფუნქცია, რომელიც დააბრუნებს რიცხვის 20%-ს. ამ ფუნქციას დავარქვათ, მაგალითად, get20. ფუნქცია უნდა იღებდეს პარამეტრს num-ს, რომლის მონაცემთა ტიპი იქნება ათწილადი. ეს ფუნქცია დააბრუნებს მიღებული პარამეტრის ნამრავლს 0.2-ზე. ფუნქციის შესაქმნელად გამოიყენება CREATE FUNCTION<funqciis_saxeli>სინტაქსი, ხოლო მისი რედაქტირებისთვის ALTER FUNCTION<funqciis_saxeli>სინტაქსი.

ფუნქციის შესაქმნელად გავუშვათ შემდეგი მოთხოვნა:

```
USE [statistika]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:                  სახელი, გვარი
-- Create date: 03/03/2024
-- Description:   ითვლისრიცხვის 20%-ს
-- =============================================
CREATE FUNCTION [dbo].[get20]
(
        @num FLOAT
)
RETURNS FLOAT
AS
BEGIN
        RETURN @num * 0.2
END
```

შექმნის შემდეგ აღნიშნული ფუნქცია შეგვიძლია გამოვიყენოთ, მაგრამ გამოყენებამდე სასურველია, რომ ჩვენი შექმნილი ფუნქცია შევამოწმოთ mshp_seqcia ცხრილის mshp სვეტზე. ამისათვის გამოვიყენოთ ასეთი ბრძანება:

```
SELECT weli, seqcia, mshp,
dbo.get20(mshp) [20 procenti]
FROM mshp_seqcia;
```

შედეგი გამოსახულია სურათზე 9.1.

**სურათი 9.1. get20 ფუნქციის გაშვების შედეგი**

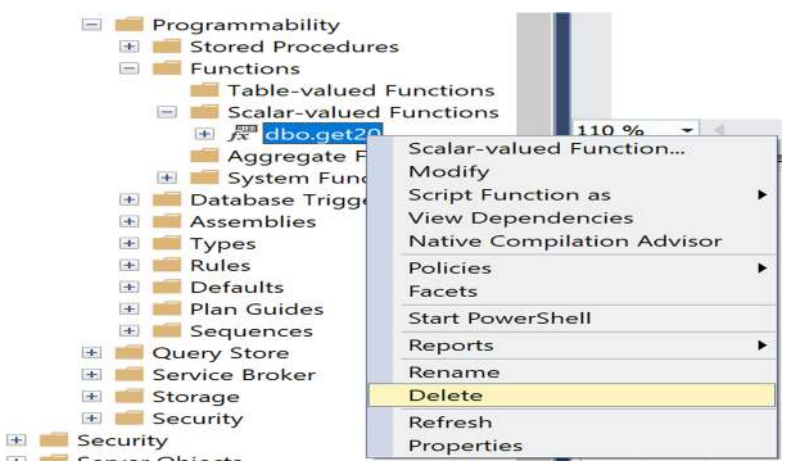
Results Messages

| | weli | seqcia | mshp | 20 procenti |
|---|---|---|---|---|
| 1 | 2022 | A | 4313.153279 | 862.6306558 |
| 2 | 2022 | B | 890.680132 | 178.1360264 |
| 3 | 2022 | C | 7078.223959 | 1415.6447918 |
| 4 | 2022 | D | 2010.626158 | 402.1252316 |
| 5 | 2022 | E | 427.997953 | 85.5995906 |
| 6 | 2022 | F | 5026.424601 | 1005.2849202 |
| 7 | 2022 | G | 9990.823914 | 1998.1647828 |
| 8 | 2022 | H | 4044.070652 | 808.8141304 |
| 9 | 2022 | I | 2329.248793 | 465.8497586 |
| 10 | 2022 | J | 3131.881384 | 626.3762768 |
| 11 | 2022 | K | 2961.412246 | 592.2824492 |
| 12 | 2022 | L | 6309.015329 | 1261.8030658 |
| 13 | 2022 | M | 1286.905117 | 257.3810234 |
| 14 | 2022 | N | 667.035041 | 133.4070082 |
| 15 | 2022 | O | 4031.916073 | 806.3832146 |
| 16 | 2022 | P | 2807.913642 | 561.5827284 |
| 17 | 2022 | Q | 2321.127977 | 464.2255954 |
| 18 | 2022 | R | 2527.685929 | 505.5371858 |
| 19 | 2022 | S | 574.733044 | 114.9466088 |
| 20 | 2022 | T | 71.360067 | 14.2720134 |

ბრძანების გაშვების შედეგად მივიღებთ mshp მონაცემის 20%-ს თითოეული სვეტისთვის.

ჩვენ საკმაოდ მარტივი ფუნქცია შევქმენით, თუმცა შეგვიძლია ანალოგიურად უფრო რთული ფუნქციები შევქმნათ იმით, რაც ვისწავლეთ ამ წიგნში. მის წასაშლელად შეგვიძლია გამოვიყენოთ კონტექსტური მენიუ, რომელიც გამოსახულია სურათზე 9.2 და დავაწკაპუნოთ DELETE ღილაკზე.

**სურათი 9.2. ფუნქციის კონტექსტური მენიუ**



შესაძლებელია შევქმნათ ისეთი ფუნქცია, რომელიც დააბრუნებს ცხრილს, მაგალითად, ცხრილი კონკრეტული წლისათვის, რომელიც დაზუსტდება პარამეტრით დამshp_seqcia ცხრილის mshp სვეტს გაამრავლებს 1.2-ზე. ამ ფუნქციას ვუწოდოთ mshp_axali. მის შესაქმნელად გამოვიყენოთ შემდეგი მოთხოვნა:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:                  სახელი, გვარი
-- Create date: 03/03/2024
-- Description:   აბრუნებსწელს, სექტორებსადამშპ-ისნამრავლს 1.2-ზე
-- ============================================
CREATE FUNCTION mshp_axali
(
        @weli INT
)
RETURNS TABLE
AS
RETURN
(
        SELECT weli, seqcia, mshp * 1.2 mshp_axali
        FROM mshp_seqcia
        WHERE weli = @weli
)
GO
```

თუ გვსურს ცხრილის მიღება, ამ ფუნქციის საშუალებით გამოვიყენოთ ასეთი ბრძანება:

```
SELECT *
FROM dbo.mshp_axali(2022);
```

აქ mshp_axali ფუნქციას მივაწოდეთ პარამეტრი 2022 და მიღებული ცხრილიდან წამოვიღეთ ყველა მონაცემი. შედეგი გამოსახულია სურათზე 9.3.

სურათი 9.3. mshp_axali ფუნქციის გამოძახების შედეგი

Results Messages

| | weli | seqcia | mshp_axali |
|---|---|---|---|
| 1 | 2022 | A | 5175.7839348 |
| 2 | 2022 | B | 1068.8161584 |
| 3 | 2022 | C | 8493.8687508 |
| 4 | 2022 | D | 2412.7513896 |
| 5 | 2022 | E | 513.5975436 |
| 6 | 2022 | F | 6031.7095212 |
| 7 | 2022 | G | 11988.9886968 |
| 8 | 2022 | H | 4852.8847824 |
| 9 | 2022 | I | 2795.0985516 |
| 10 | 2022 | J | 3758.2576608 |
| 11 | 2022 | K | 3553.6946952 |
| 12 | 2022 | L | 7570.8183948 |
| 13 | 2022 | M | 1544.2861404 |
| 14 | 2022 | N | 800.4420492 |
| 15 | 2022 | O | 4838.2992876 |
| 16 | 2022 | P | 3369.4963704 |
| 17 | 2022 | Q | 2785.3535724 |
| 18 | 2022 | R | 3033.2231148 |
| 19 | 2022 | S | 689.6796528 |
| 20 | 2022 | T | 85.6320804 |

## 9.2. ტრიგერები

ზოგჯერ ცხრილებში გარკვეული მონაცემების ცვლილება ხდება. ტრიგერებით შესაძლებელია ცვლილებების შედეგების ასახვა. ტრიგერი (trigger) არის შენახული პროცედურის სპეციალური ტიპი, რომელიც სერვერის მიერ ავტომატურად გაიშვება ცხრილში მონაცემების შეცვლის ოპერაციების შესრულების დროს. ტრიგერი სხვადასხვა დანიშნულებით გამოიყენება. მათი საშუალებით შესაძლებელია ბმულ ცხრილებში კასკადური ცვლილებების შესრულება, მონაცემების შემოწმების ალგორითმების რეალიზება და ა. შ.

თითოეული ტრიგერი დაკავშირებულია კონკრეტულ ცხრილთან. მაგალითად, ცხრილში მონაცემების შეცვლის წინ სერვერი ავტომატურად გაუშვებს ტრიგერს. თუ მისი შესრულება წარმატებით დამთავრდა, მაშინ შესრულდება ცხრილში მონაცემების ცვლილება. ტრიგერის მიერ შესრულებული მოქმედებები განიხილება, როგორც ერთი ტრანზაქცია. ტრიგერების გამოყენება შეიძლება, აგრეთვე, ჩვენი შეხედულებისამებრ.

ტრიგერი არ უნდა გამოვიყენოთ მარტივი შემოწმებების შესასრულებლად (რისთვისაც შეიძლება CHECK მთლიანობის შეზღუდვის გამოყენება). ტრიგერი არ უნდა გამოვიყენოთ იმ მოქმედებების შესასრულებლად, რომლებიც შეიძლება შენახული პროცედურებითაც Transact-SQL-ის პაკეტით შესრულდეს. ტრიგერების გამოყენება არ არის სასურველი იმიტომაც, რომ ისინი ბლოკავენ რესურსს მუშაობის დამთავრებამდე და სხვა მომხმარებლებს არ ეძლევა ამ რესურსთან მიმართვის შესაძლებლობა [სამხარაძე, გაჩეჩილაძე, 2016; გვ. 226].

არსებობს სხვადასხვა სახის ტრიგერი, თუმცა, ჩვენ განვიხილავთ DML (მონაცემთა მანიპულაციის ენა) ტიპის ტრიგერებს.

ტრიგერი შევქმნათ mshp_seqcia ცხრილისათვის. როდესაც ამ ცხრილში მოხდება ჩანაწერის გაკეთება, მაშინ სხვა ცხრილში, რომელსაც დავარქმევთ mshp_seqcia_audit-ს, აისახება შესაბამისი მონაცემები და დრო, როდესაც განხორციელდა ცვლილება. ამისათვის ჯერ შევქმნათ mshp_seqcia_audi ცხრილი, რომლის სტრუქტურაც მოცემულია სურათზე 9.4.

**სურათი 9.4. mshp_seqcia_audit ცხრილის სტრუქტურა**

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| weli | smallint | ☑ |
| seqcia | nchar(1) | ☑ |
| mshp | float | ☑ |
| cvlilebis_dro | datetime | ☑ |
| | | ☐ |

ცხრილის შექმნის შემდეგ შევიმუშაოთ შესაბამისი ტრიგერი, რომელიც INSERTდა UPDATE ოპერაციის განხორციელებისას mshp_seqcia-ში ჩაწერს mshp_seqcia_audit ცხრილში შესაბამის ცვლილებს და, ასევე, ცვლილების დროს. დავარქვათ ტრიგერს audit_mshp_seqcia. შესაბამისი მოთხოვნა ტრიგერის შესაქმნელად ასეთი იქნება:

```
USE statistika
GO
CREATE TRIGGER audit_mshp_seqcia ON dbo.mshp_seqcia
        FOR INSERT, UPDATE
AS
BEGIN
 INSERT INTO dbo.mshp_seqcia_audit (weli, seqcia, mshp, cvlilebis_dro)
 SELECT weli, seqcia, mshp, GETDATE()
 FROM inserted;
END
GO
```

გავუშვათ ბრძანება. ტრიგერის შექმნის შემდეგ ვცადოთ mshp_seqcia ცხრილში ჩანაწერის დამატება. მაგალითად, გამოვიყენოთ ასეთი ბრძანება:

```
INSERT INTO [statistika].[dbo].[mshp_seqcia] ([weli], [seqcia], [mshp])
VALUES
(2023, N'A', 4500);
```

თუ ამ ბრძანებას გავუშვებთ, მაშინ mshp_seqcia_audit ცხრილში დაემატება შესაბამისი ჩანაწერი, დაახლოებით მსგავსი, როგორიც გამოსახულია სურათზე 9.5.

**სურათი 9.5. ტრიგერის შედეგი mshp_seqcia_audit ცხრილში**


UPDATE ოპერაციის შესრულების შემთხვევაშიც მსგავსი რამ მოხდება. მაგალითად, თუ გამოვიყენებთ შემდეგ მოთხოვნას, მაშინ აუდიტის ცხრილში გაჩნდება ასეთი ახალი ჩანაწერი:

```
UPDATE [statistika].[dbo].[mshp_seqcia]
SET seqcia = N'B'
WHERE weli = 2023;
```

თუ ამოვიღებთ მონაცემებს mshp_seqcia_audit ცხრილიდან, ვნახავთ შესაბამის ჩანაწერებს, დაახლოებით მსგავსს, როგორიც ნაჩვენებია სურათზე 9.6.

**სურათი 9.6. აუდიტის ცხრილის შედეგები**


წაშლის ტრიგერის შემთხვევაში ტრიგერის შექმნისას გამოყენებული inserted ცხრილის მაგივრად გამოვიყენებთ deleted ცხრილს.

ტრიგერის წასაშლელად გამოიყენება DROP <trigeris_saxeli> სინტაქსი, სადაც „<trigeris_saxeli>“ არის შესაბამისი ტრიგერის დასახელება.

### *კითხვები თვითშემოწმებისთვის:*

1. როგორ იქმნება ფუნქციები T-SQL-ში?
2. ახსენით სხვაობა ცხრილური და სკალარული მნიშვნელობის ფუნქციებს შორის.
3. რაში გამოიყენება ტრიგერები T-SQL-ში?
4. აღწერეთ DML ტიპის ტრიგერის შექმნის პროცესი.

# თავი 10. Microsoft Office Access-ის შესავალი

## 10.1. MS Access-ის მოკლე მიმოხილვა

Microsoft Office Access არის პროგრამული უზრუნველყოფა, რომლის დახმარებით შეგვიძლია შევქმნათ და ვმართოთ მონაცემთა ბაზები. წინა თავებში განვიხილეთ SQL Server 2022-ის ძირითადი ელემენტები, ვისწავლეთ, თუ როგორ შევქმნათ მონაცემთა ბაზა, დავამატოთ მომხმარებლები, შევქმნათ ცხრილები, გავუშვათ მოთხოვნები, ასევე, CRUD ოპერაციების განხორციელებისთვის საჭირო ბრძანებები, წარმოდგენები, შენახული პროცედურების შექმნა და გამოყენება, ტრანზაქციულობა და მისი გამოყენება, ახალი ფუნქციებისა და ტრიგერების შექმნა და გამოყენება. ამჯერად გავეცნობით Microsoft Office Access-ს, რომლის დახმარებითაც შესაძლებელია მონაცემთა ბაზებისა და ცხრილების, მონაცემების შემყვანი ფორმებისა და მოთხოვნების შექმნა და მოგვიანებით მისი დაკავშირება SQL Server-თან. უნდა ითქვას, რომ MS Access-ით მსგავსი ოპერაციების განხორციელებისას, ძირითადად, გამოიყენება გრაფიკული ინტერფეისი, თუმცა, შესაძლებელია SQL კოდის გამოყენებაც.

გრაფიკული ინტერფეისის გამოყენება მონაცემების ამოღებისას ზოგჯერ მნიშვნელოვნად ამარტივებს შესასრულებელ საქმეს, თანაც MS Access-ის თავსებადობა SQL Server სისტემასთან საშუალებას აძლევს ამა თუ იმ ორგანიზაციის თანამშრომლებს უფრო მარტივად მიმოცვალონ ერთმანეთში ინფორმაცია. ამ პროგრამის საშუალებით შესაძლებელია SQL Server-დან მონაცემების როგორც ჩამოტვირთვა, ისე მასში მონაცემების ატვირთვა. ამავდროულად, მისი გამოყენება SQL Server-ის გარეშეც შეიძლება. შესაბამისად, შესაძლებელი იქნება მონაცემთა ბაზის აწყობა, რომელიც მოთავსებული იქნება ლოკალურ ფაილში და ამ ბაზაში შესაბამისი ოპერაციების განხორციელება.

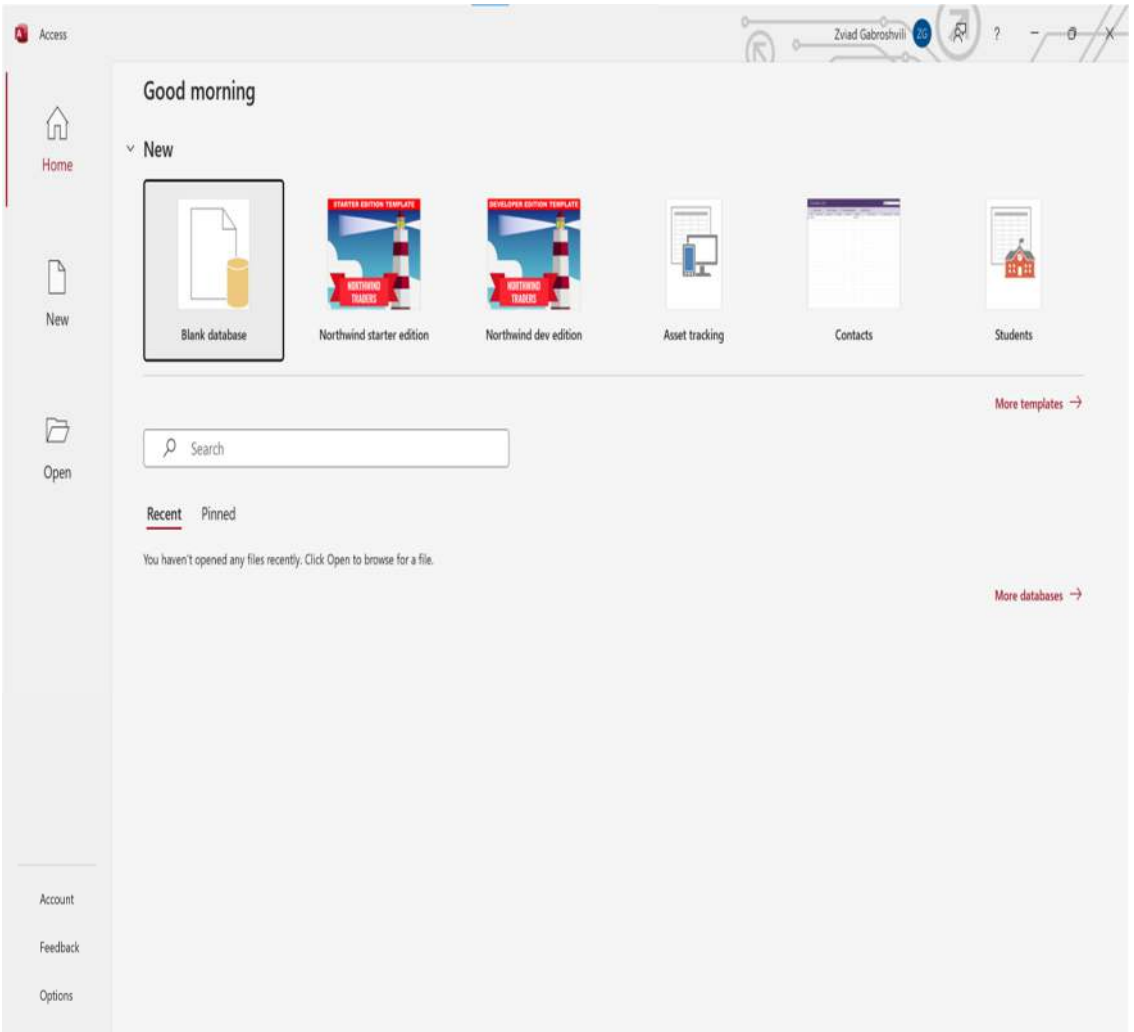
როგორც საზღვარგარეთ, ისე საქართველოში დღეს სხვადასხვა სახელმწიფო თუ კერძო ორგანიზაცია იყენებს MS Access-ს, შესაბამისად, მისი ცოდნა სასარგებლოა დასაქმებისა და კარიერული თვალსაზრისით და მონაცემთა ბაზების მართვაშიც მნიშვნელოვნად დაგვეხმარება.

## 10.2. ცხრილების შექმნა

გავხსნათ Microsoft Office Access; გამოვა ინტერფეისი, როგორიც გამოსახულია სურათზე 10.1. საწყის ფანჯარაზე ახალი მონაცემთა ბაზის შესაქმნელად ავირჩიოთ Blank database მასზე დაწკაპუნებით. შედეგად, გამოვა ისეთი ფანჯარა, როგორიც გამოსახულია სურათზე 10.2. აქ შეგვეძლება ავირჩიოთ სახელი, თუ რას დავარქმევთ მონაცემთა

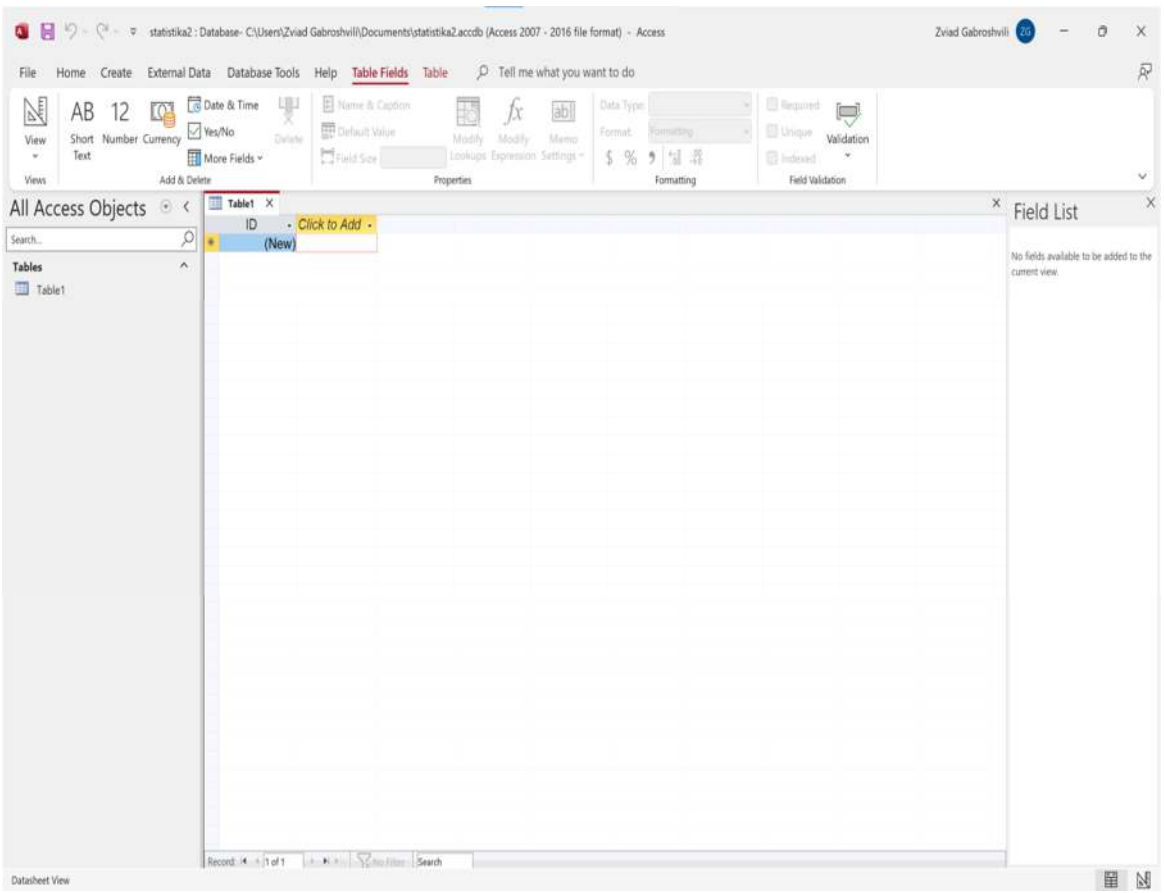
ბაზას და სად შევინახავთ მას. პირობითად დავარქვათ მას statistika2 (როგორც გამოსახულია სურათზე 10.2) და შევინახოთ სასურველ ადგილას. ამის შემდეგ დავაწკაპუნოთ აღნიშნულ ფანჯარაში Create ღილაკზე, შედეგად მონაცემთა ბაზა შეიქმნება ჩვენს სასურველ ადგილზე. ამის შემდეგ გამოვა ფანჯარა, როგორიც გამოსახულია სურათზე 10.3, რომელსაც მიმოვიხილავთ.

**სურათი 10.1. MS Access-ის საწყისი ფანჯარა**



სურათი 10.2. მონაცემთა ბაზისთვის ადგილმდებარეობისა და სახელის არჩევა

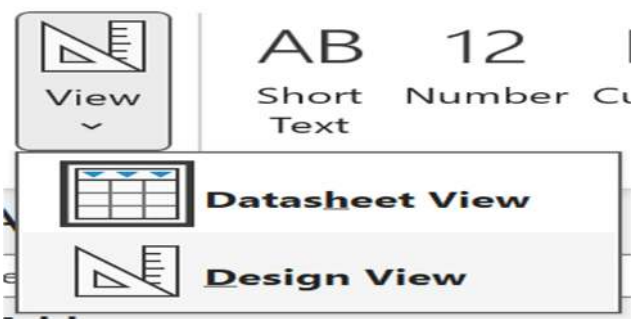

სურათი 10.3. MS Access ინტერფეისი



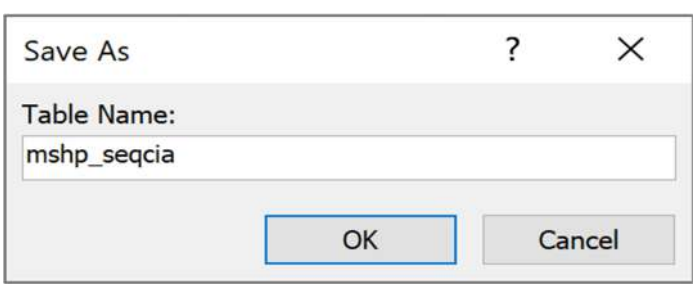
სურათზე 10.3 გამოსახულ ფანჯარაში ზედა ნაწილში ვხედავთ ჩანართებს, დაახლოებით მსგავსს, როგორიც გააჩნია Microsoft Office-ის სხვა აპლიკაციებს. მარცხნივ გამოსახულია All Access Objects პანელი, სადაც მოთავსებულია Access-ის ობიექტები. ასეთი ობიექტები შეიძლება იყოს ცხრილები, ფორმები, მოთხოვნები, რეპორტები, მაკროსები და მოდულები. ამ ეტაპზე Access ფანჯრის ცენტრში გვაჩვენებს Table1 ცხრილს ერთ-ერთ ჩანართად, რომელიც არ არის შენახული. აქ შეგვიძლია შევიტანოთ და ვნახოთ მონაცემები. ამჯერად ცხრილს Datasheet View მდგომარეობაში ვხედავთ, თუმცა, შეგვიძლია მისი Design View ინტერფეისზე გადასვლა, საიდანაც შესაძლებელია ცხრილის სტრუქტურის განსაზღვრა. ამ პროგრამაში სვეტებს ეწოდებათ ველები (Fields), ხოლო ცხრილის სტრიქონებს - ჩანაწერები (Records). პირველი ველი, რომელსაც გვთავაზობს და ავტომატურადაა შექმნილი, არის ID. ცხრილის ქვემოთ გვაჩვენებს ჩანაწერების რაოდენობას.

გადავიყვანოთ ცხრილი Design View მდგომარეობაში. ამისთვის Table Fields ჩანართიდან ავირჩიოთ View>Design View, როგორც მოცემულია სურათზე 10.4. გამოვა ფანჯარა, როგორიც ნაჩვენებია 10.5 სურათზე, სადაც ავირჩევთ სახელს. ჩვენს შემთხვევაში მას დავარქვათ mshp_seqcia და დავაწკაპუნოთ OK ღილაკზე.

**სურათი 10.4. ცხრილის სტრუქტურის დიზაინის ინტერფეისზე გადაყვანა**

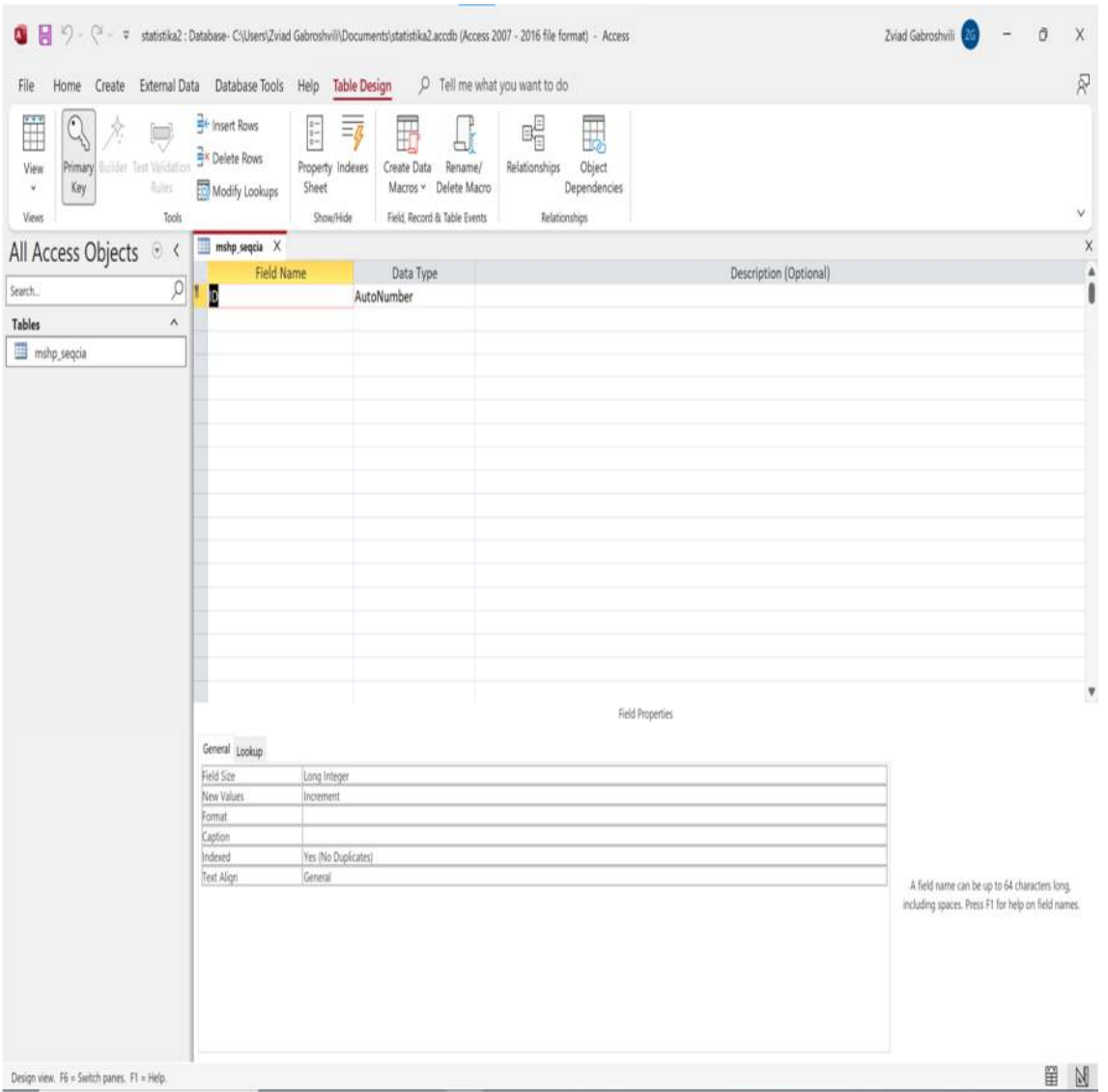



**სურათი 10.5. ცხრილის სახელის არჩევა**




ცხრილი შეინახება შესაბამისი სახელით და გამოჩნდება Design View ინტერფეისი, როგორიც ნაჩვენებია სურათზე 10.6.

სურათი 10.6. Design View ინტერფეისი

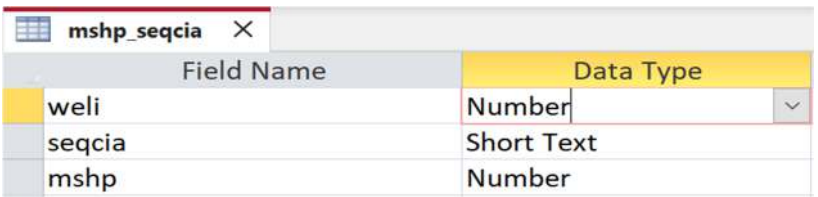



ამ ინტერფეისზე გააქტიურებულია Table Design ჩანართი, Field Name სვეტში მოთავსებულია ცხრილის სვეტების სახელები, ხოლოდ Data Type სვეტში - მონაცემთა ტიპი შესაბამისი სვეტისთვის. მონიშნულია ID ველი და მას აქვს მინიჭებული მთავარი გასაღებური ველის სტატუსი. როგორც ვხედავთ, გააქტიურებულია Primary Key ღილაკი. ამასთან ერთად, ავტომატურად გაიზრდება ეს რიცხვი თითოეული ჩანაწერისთვის, შეტანილი ჩანაწერის შედეგად, რადგან მონაცემთა ტიპში მას აქვს AutoNumber ტიპი. Field Properties პანელში, General ჩანართში მოცემულია მონიშნული ველის თვისებები.

წავშალოთ ID ველი, Table Design ჩანართში არსებული Delete Rows-ზე დაწკაპუნებთ. შევქმნათ SQL Server-ში არსებული mshp_seqcia ცხრილის მსგავსი სვეტები Access-ის ცხრილისთვის. შესაბამისი ველები თავისი მონაცემთა ტიპებით ნაჩვენებია

სურათზე 10.7, ხოლო თითოეული ველის პარამეტრები მოცემულია შემდეგ სურათებზე: weli - სურათი 10.8, seqcia - სურათი 10.9 და mshp - სურათი 10.10.

**სურათი 10.7. mshp_seqcia ცხრილის ველები და მონაცემთა ტიპები**




**სურათი 10.8. weli ველის პარამეტრები**

General | Lookup

| | |
|---|---|
| Field Size | Integer |
| Format | |
| Decimal Places | 0 |
| Input Mask | |
| Caption | |
| Default Value | 0 |
| Validation Rule | |
| Validation Text | |
| Required | No |
| Indexed | No |
| Text Align | General |

**სურათი 10.9. seqcia ველის პარამეტრები**

General | Lookup

| | |
|---|---|
| Field Size | 1 |
| Format | |
| Input Mask | |
| Caption | |
| Default Value | |
| Validation Rule | |
| Validation Text | |
| Required | No |
| Allow Zero Length | Yes |
| Indexed | No |
| Unicode Compression | Yes |
| IME Mode | No Control |
| IME Sentence Mode | None |
| Text Align | General |

სურათი 10.10. mshp ველის პარამეტრები

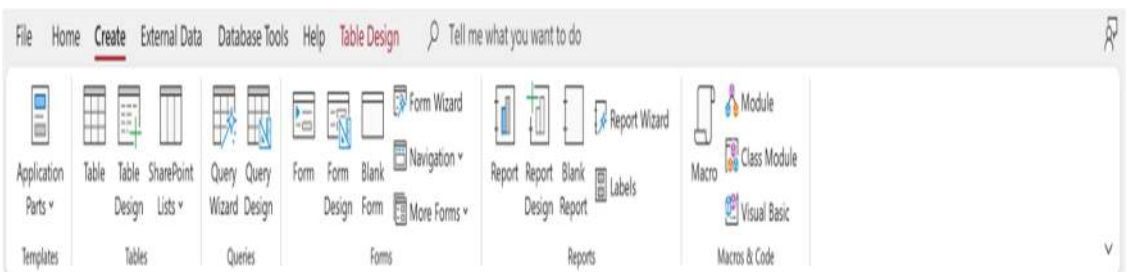
General Lookup

| | |
|---|---|
| Field Size | Single |
| Format | |
| Decimal Places | Auto |
| Input Mask | |
| Caption | |
| Default Value | 0 |
| Validation Rule | |
| Validation Text | |
| Required | No |
| Indexed | No |
| Text Align | General |

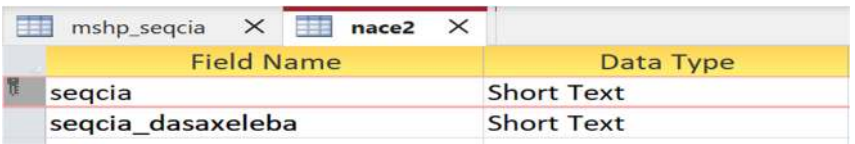
თუ დავაკვირდებით სურათებს, ვნახავთ, რომ weli ველს მივუთითეთ Field Size-ში Integer, რაც ნიშნავს, რომ გამოყენებული იქნება მთელი რიცხვები; seqcia ველში მითითებულია FieldSize ველში 1, რაც ნიშნავს, რომ გამოყენებული იქნება 1 სიმბოლო, ხოლო mshp ველში მივუთითეთ Single, ანუ გამოიყენება 32=-ბიტიანი ათწილადის მონაცემთა ტიპი. შევინახოთ ცვლილებები კლავიატურაზე Ctrl+S ღილაკების გამოყენებით.

mshp_seqcia ცხრილი უკვე შექმილია. ახალ შევქმნათ SQL Server-ში მოცემული nace2 ცხრილის ანალოგიური ცხრილი. ამისთვის გადავიდეთ Create ჩანართში და დავაწკაპუნოთ Tables პანელში არსებულ Table Design ღილაკზე. ჩანართი ნაჩვენებია სურათზე 10.11. გამოვა ცხრილის სტრუქტურის დიზაინის ინტერფეისი და მივუთითებთ შესაბამის ველებსა და მონაცემთა ტიპებს, როგორც ეს გამოსახულია სურათზე 10.12. მივანიჭოთ seqcia ველს გასაღებული ველის სტატუსი და Field Size-ში მივუთითოთ 1, ხოლო seqcia_dasaxeleba Field Size-ში 200. შევინახოთ ცხრილი და დავარქვათ მას nace2.

სურათი 10.11. Create ჩანართი



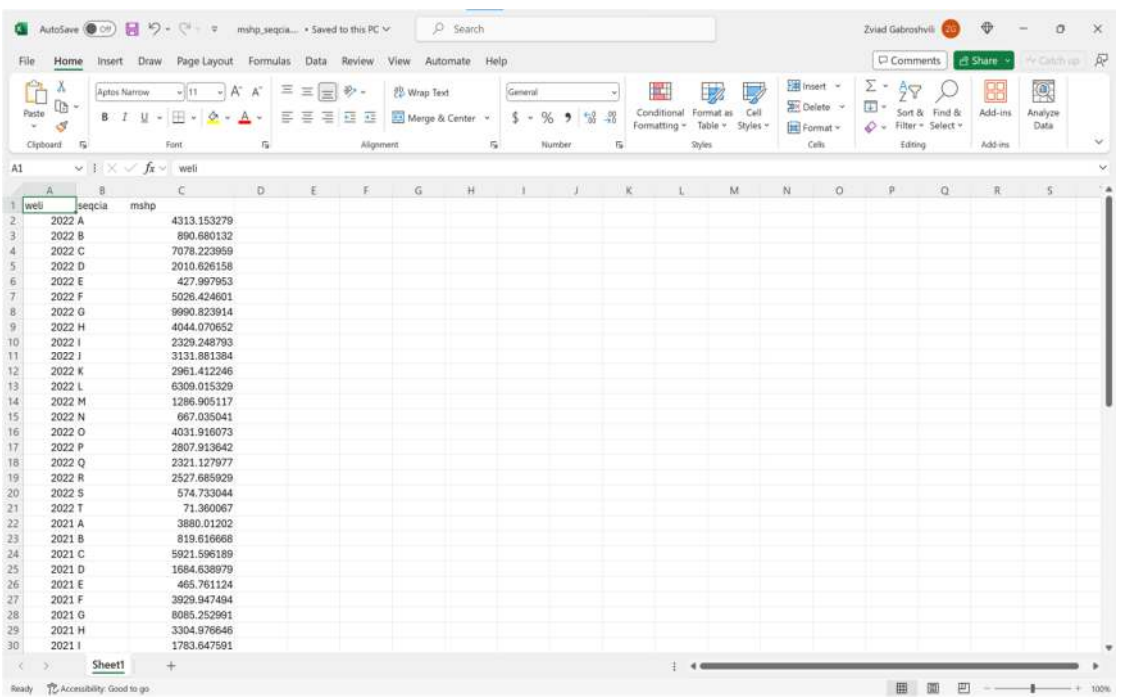
**სურათი 10.12. nace2 ცხრილის სტრუქტურა**




ჩვენს მონაცემთა ბაზაში ცხრილები უკვე შექმნილია, მაგრამ არ არის შესაბამისი მონაცემები. მონაცემთა შესატანად სხვადასხვა გზა არსებობს. მონაცემების შესატანად და მათი რედაქტირებისათვის შეგვიძლია გამოვიყენოთ ცხრილის Datasheet View ინტერფეისი, სადაც შეგვიძლია შევიტანოთ მონაცემები, შევცვალოთ, ან თითოეულ ჩანაწერზე ვიმოქმედოთ კონტექსტური მენიუთი და წავშალოთ მონაცემი ამავე მენიუში მოცემული Delete ღილაკით.

არსებობს მონაცემთა შეტანის სხვა ვარიანტებიც. ამჯერად განვიხილოთ MS Excel-ის ფაილიდან მონაცემების შემოტანა. ამისათვის შევქმნათ MS Excel-ის ფაილი შესაბამისი მონაცემებით, რომელიც შეიძლება ამოვიღოთ ბაზიდან, ან სხვა წყაროდან და დავარქვათ mshp_seqcia.xlsx. ფაილის მაგალითი ნაჩვენებია სურათზე 10.13. ფაილში მოთავსებულია SQL Server-ზე არსებული mshp_seqcia ცხრილიდან ამოღებული მონაცემები 2020-2022 წლებისათვის.

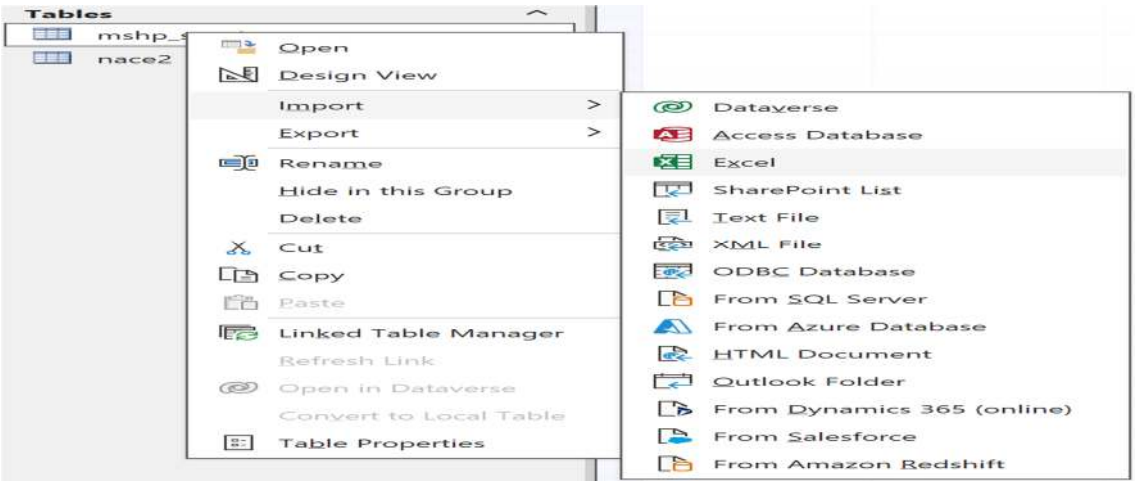
**სურათი 10.13. mshp_seqcia.xlsx ფაილის მაგალითი**



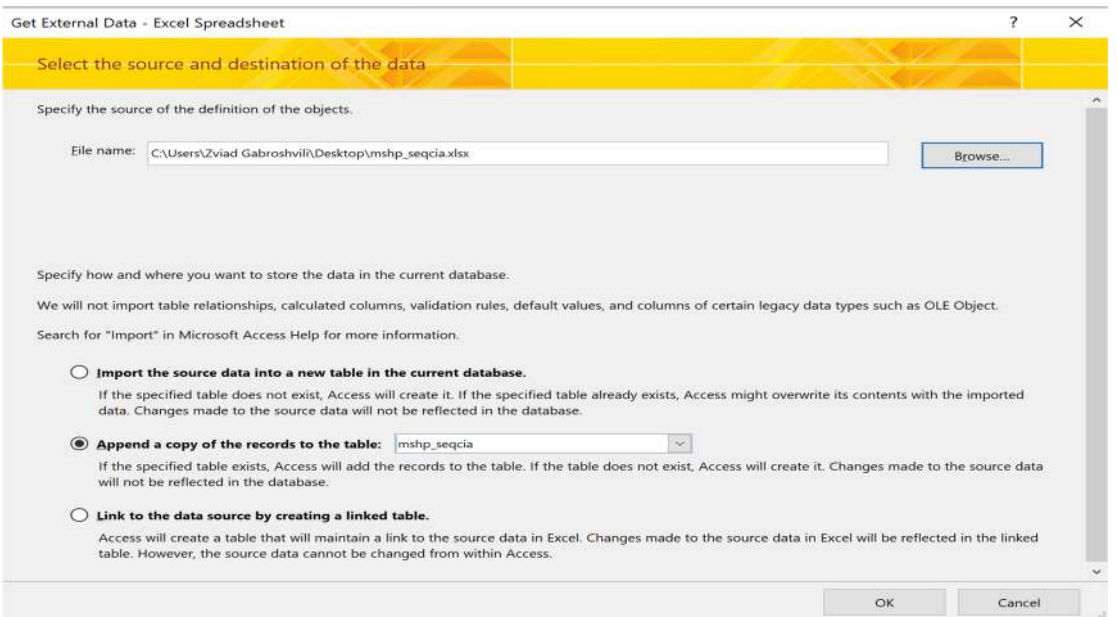

ფაილის შენახვის შემდეგ ვცადოთ მისი შემოტანა Access-ში არსებულ შესაბამის ცხრილში. ამისათვის ჯერ დავხუროთ ცხრილი, შემდეგ All Access Objects პანელში არ-

სებულ ცხრილზე დავაწკაპუნოთ მაუსის მარჯვენა ღილაკით; გამოვა კონტექსტური მენიუ და ავირჩიოთ Import>Excel, როგორც ეს ნაჩვენებია სურათზე 10.14. ამის შემდეგ გამოვა ფანჯარა, როგორიც ნაჩვენებია სურათზე 10.15 და ჩვენს ფაილს მოვიძიებთ Browse ღილაკით; ქვემოთ მოვნიშნავთ Append a copy of the records to the table და ავირჩევთ შესაბამის ცხრილს, როგორიც ნაჩვენებია სურათზე და დავაწკაპუნებთ OK ღილაკზე.

**სურათი 10.14. ცხრილის კონტექსტური მენიუ**

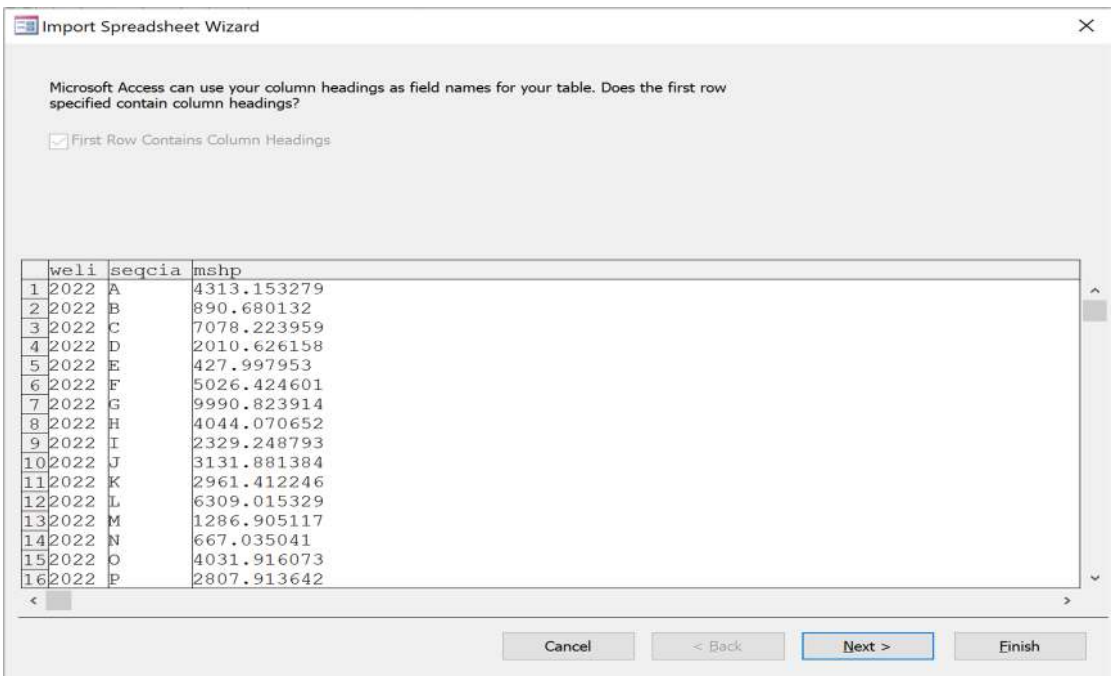



**სურათი 10.15. ფაილისა და ცხრილის არჩევა**



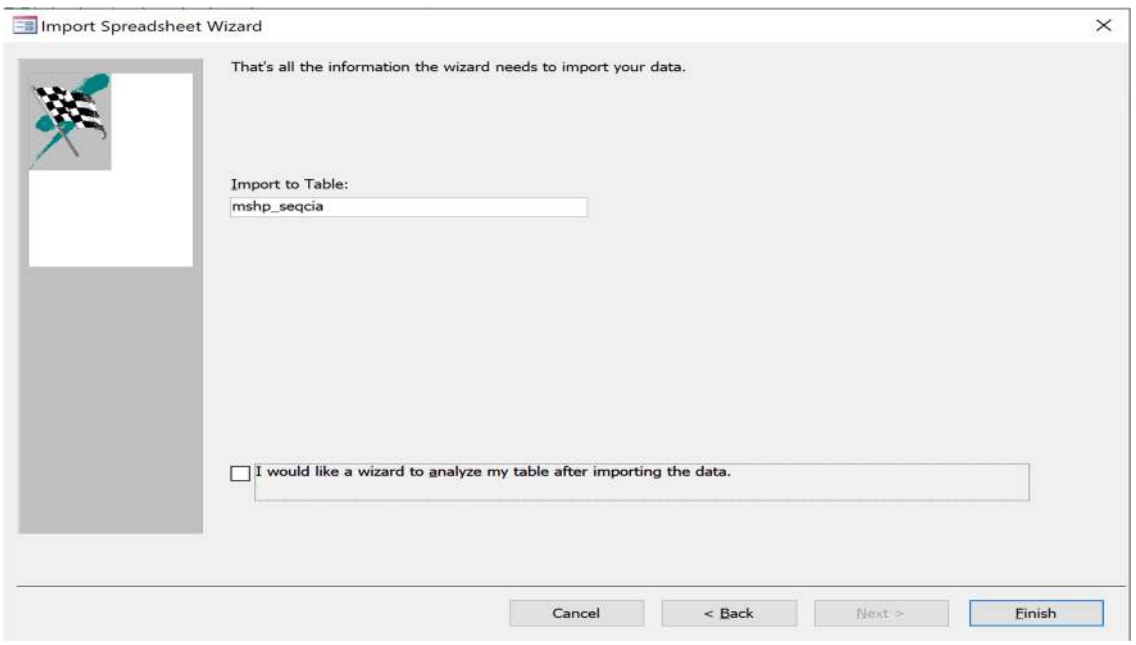
ამის შემდეგ გამოვა ახალი ფანჯარა, როგორიც ნაჩვენებია სურათზე 10.16 და დავაწკაპუნებთ Next ღილაკზე, რის შემდეგაც გამოვა ისევ ახალი ფანჯარა, რომელიც უნდა ემთხვეოდეს სურათზე 10.17 გამოსახულს. დავაწკაპუნოთ Finish ღილაკზე, შემდეგ გამოსულ ფანჯარაზე კი Close ღილაკზე.

**სურათი 10.16. მონაცემების შემოტანა (I)**




| | weli | seqcia | mshp |
|---|---|---|---|
| 1 | 2022 | A | 4313.153279 |
| 2 | 2022 | B | 890.680132 |
| 3 | 2022 | C | 7078.223959 |
| 4 | 2022 | D | 2010.626158 |
| 5 | 2022 | E | 427.997953 |
| 6 | 2022 | F | 5026.424601 |
| 7 | 2022 | G | 9990.823914 |
| 8 | 2022 | H | 4044.070652 |
| 9 | 2022 | I | 2329.248793 |
| 10 | 2022 | J | 3131.881384 |
| 11 | 2022 | K | 2961.412246 |
| 12 | 2022 | L | 6309.015329 |
| 13 | 2022 | M | 1286.905117 |
| 14 | 2022 | N | 667.035041 |
| 15 | 2022 | O | 4031.916073 |
| 16 | 2022 | P | 2807.913642 |

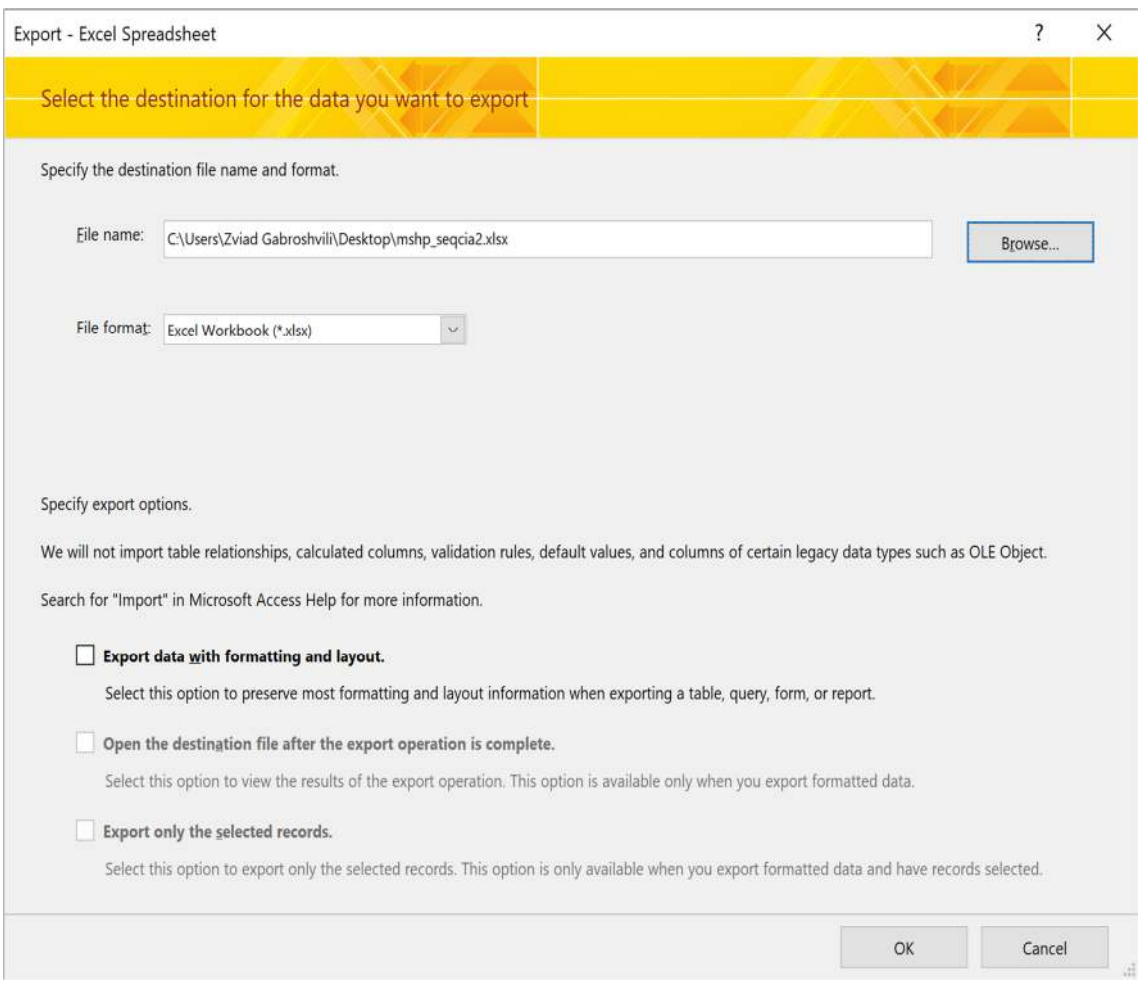
**სურათი 10.17. მონაცემების შემოტანა (II)**



თუ გადავამოწმებთ mshp_seqcia ცხრილს Access-ში Datasheet View ინტერფეისიდან, ვნახავთ, რომ მონაცემები შემოტანილი იქნება. შევინახოთ მონაცემთა ბაზა Ctrl+S ღილაკების გამოყენებით.

მონაცემთა ცხრილიდან გასატანად შესაბამისი ცხრილის კონტექსტურ მენიუში გამოვიყენოთ Export>Excel ბრძანება. გამოსულ ფანჯარაში, რომელიც ნაჩვენებია სურათზე 10.18, ავირჩიოთ ადგილმდებარეობა, თუ სად გვსურს გატანა Browse ღილაკით შესაბამისი ფაილის ფორმატი და დავაწკაპუნოთ OK ღილაკზე; ამის შემდეგ გამოსული ფანჯარა დავხუროთ Close ღილაკით. შედეგად, მონაცემები უკვე გატანილი იქნება.

**სურათი 10.18. მონაცემების გატანა Excel-ის ფორმატში**




უნდა აღინიშნოს, რომ Excel-ის გარდა, შესაძლებელია მონაცემთა გატანა და შემოტანა სხვა ფაილებიდან და სერვისებიდან.

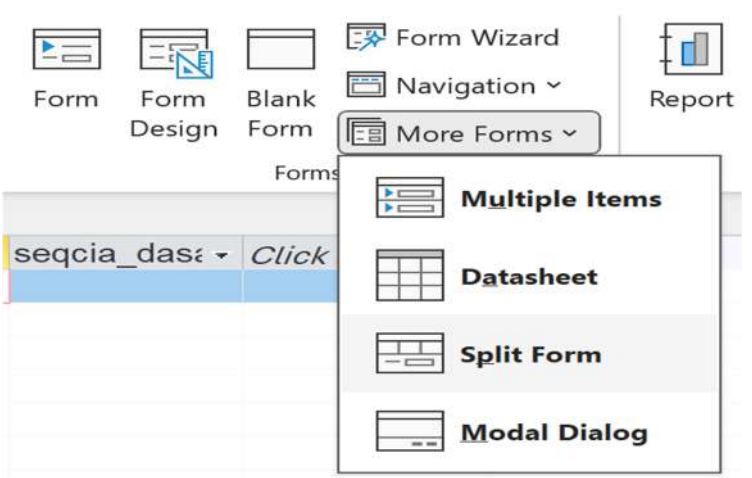
## 10.2. ფორმების შექმნა და გამოყენება

წინა ქვეთავში განვიხილეთ, თუ როგორ ხდება Access-ის ცხრილების შექმნა და მათში მონაცემების ჩაწერა. მონაცემთა ჩაწერისა და შეცვლის ერთ-ერთი ვარიანტი არის ფორმები. Access-ში შესაძლებელია მონაცემთა შეტანისათვის საჭირო ფორმების შედგენა.

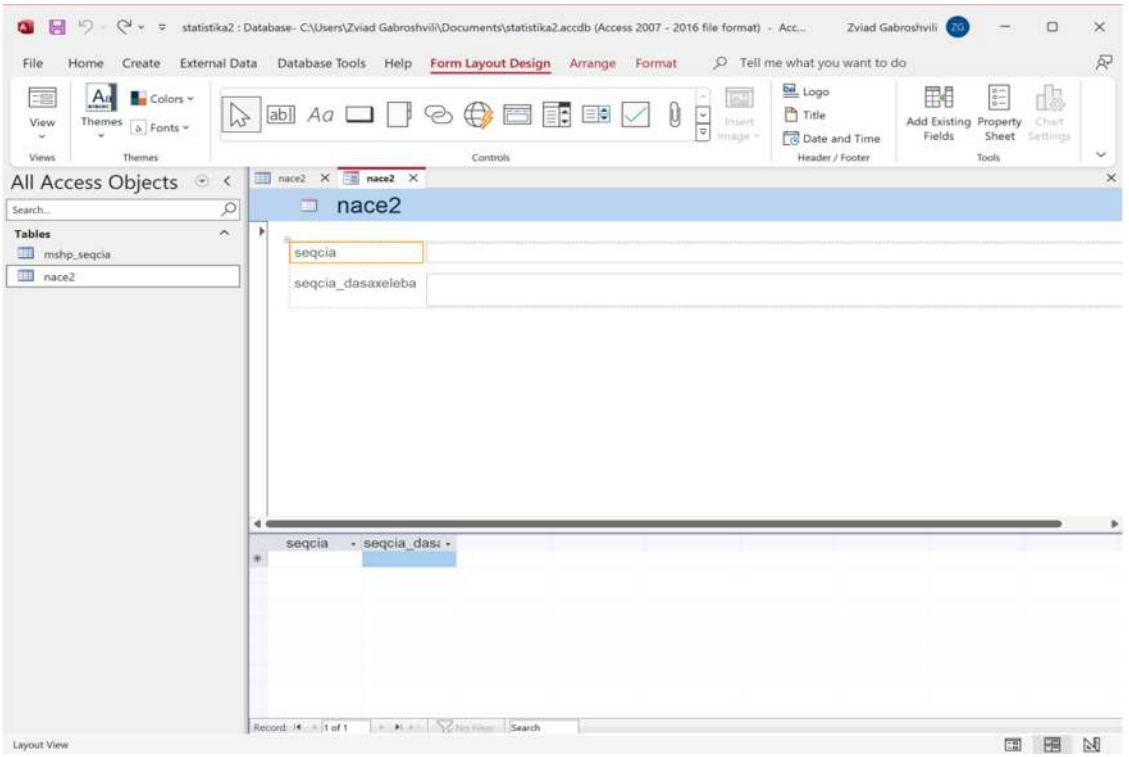
მონაცემთა შეტანისთვის შევქმნათ ფორმა ჩვენს Access-ის ბაზაში არსებული nace2 ცხრილისათვის. ამისათვის მოვნიშნოთ შესაბამისი ცხრილი, შემდეგ გადავიდეთ Create ჩანართში და შევქმნათ Split Form ტიპის ფორმა More Forms>Split Form ღილაკზე დაწკაპუნებით, როგორც ეს ნაჩვენებია სურათზე 10.19. შედეგად გაიხსნება Split Form-ის Layout View ინტერფეისი, საიდანაც ხდება ფორმის სახის ცვლილება. შეიქმნება ერთგვარი გაყოფილი ტიპის ფორმა, რომლის ზედა ნაწილში იქნება შესაბამისი მონაცემების შემტანი ველები, ხოლო ქვედა ნაწილში კი იქნება ცხრილში არსებული ჩანაწერები. თუ ჩანაწერს მოვნიშნავთ, მაშინ ფორმის შესაბამის ველებში გამოჩნდება ჩანაწერში არსებული ინფორმაცია. ფორმის შესანახად კლავიატურაზე გამოვიყენოთ Ctrl+Sღილაკები, დავარქვათ, მაგალითად, nace2_forma და შევინახოთ.

Layout View ინტერფეისში შეგვიძლია ფორმა შევცვალოთ, გადავაადგილოთ ველები, შევცვალოთ ტექსტი კონკრეტულ ფორმის ელემენტზე მოქმედებით და ა. შ. შევცვალოთ ფორმის ზედა ნაწილი, როგორც ეს ნაჩვენებია სურათზე 10.21.

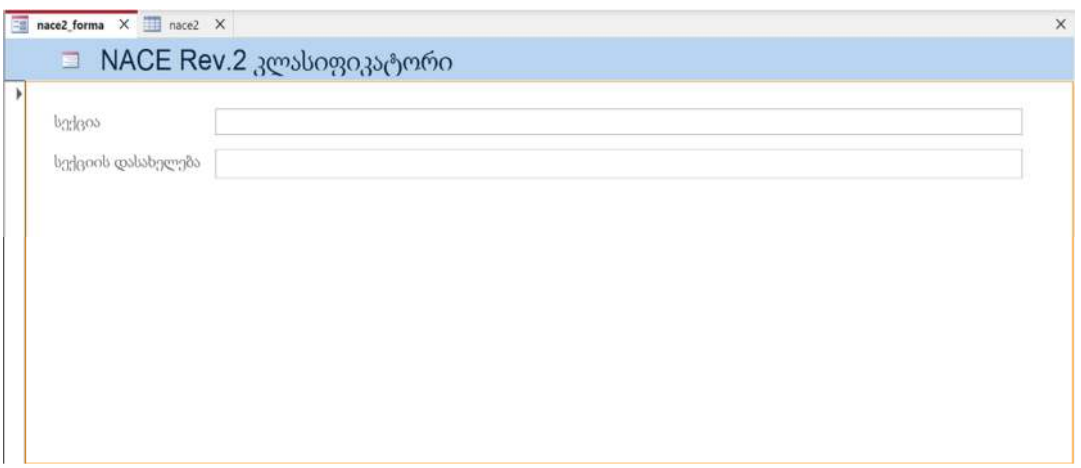
სურათი 10.19. Split-Form-ის შექმნა



სურათი 10.20. Split Form-ის Layout View ინტერფეისი

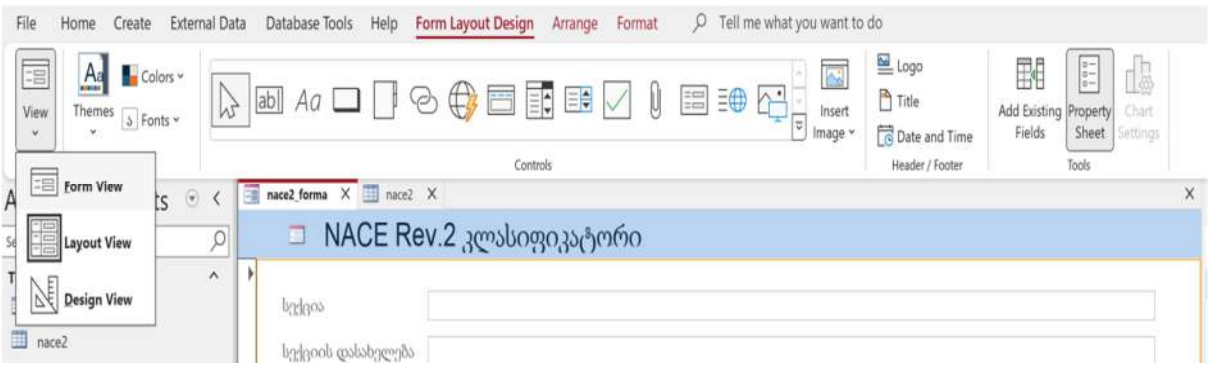



სურათი 10.21. nace2_forma ფორმის შეცვლა



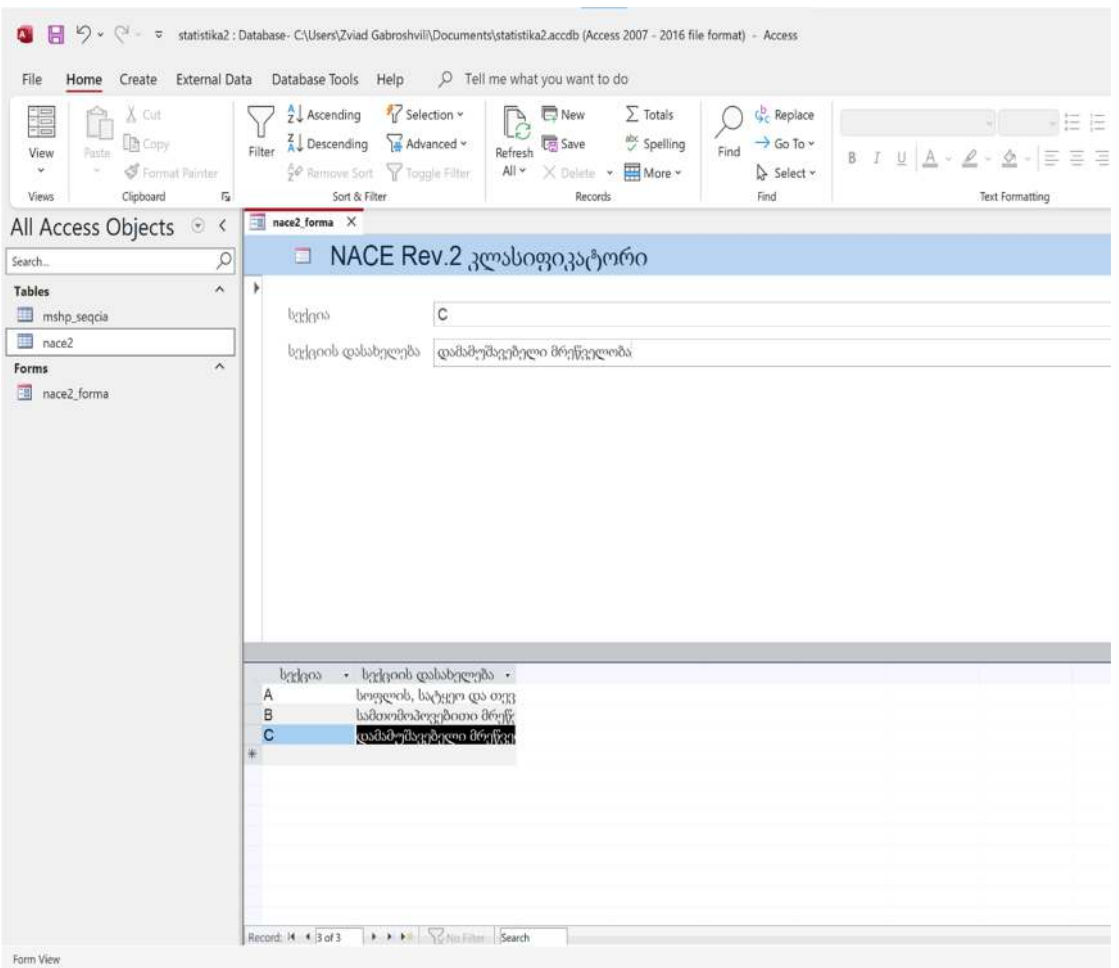

ახლა შეგვიძლია ფორმა გამოვიყენოთ მონაცემთა შეტანისთვის. ამისათვის უნდა გადავიდეთ ფორმის Form View ინტერფეისზე, Form Layout Design ჩანართის View ღილაკის დახმარებით, როგორც ეს ნაჩვენებია სურათზე 10.22.

**სურათი 10.22. ფორმის Form View ინტერფეისზე გადასვლა**




ახლა შეგვეძლება მონაცემების შეტანა, რისთვისაც თითოეულ ტექსტურ ველში ჩავწერთ შესაბამის მონაცემს და კლავიატურაზე გამოვიყენებთ Enter ღილაკს. მაგალითი იხილეთ სურათზე 10.23. შედეგად, მონაცემები შეტანილი იქნება ცხრილში. შეიტანეთ კლასიფიკატორის სხვა ჩანაწერები დამოუკიდებლად.

**სურათი 10.23. nace2 ცხრილში მონაცემების შეტანა ფორმით**



***კითხვები თვითშემოწმებისთვის:***

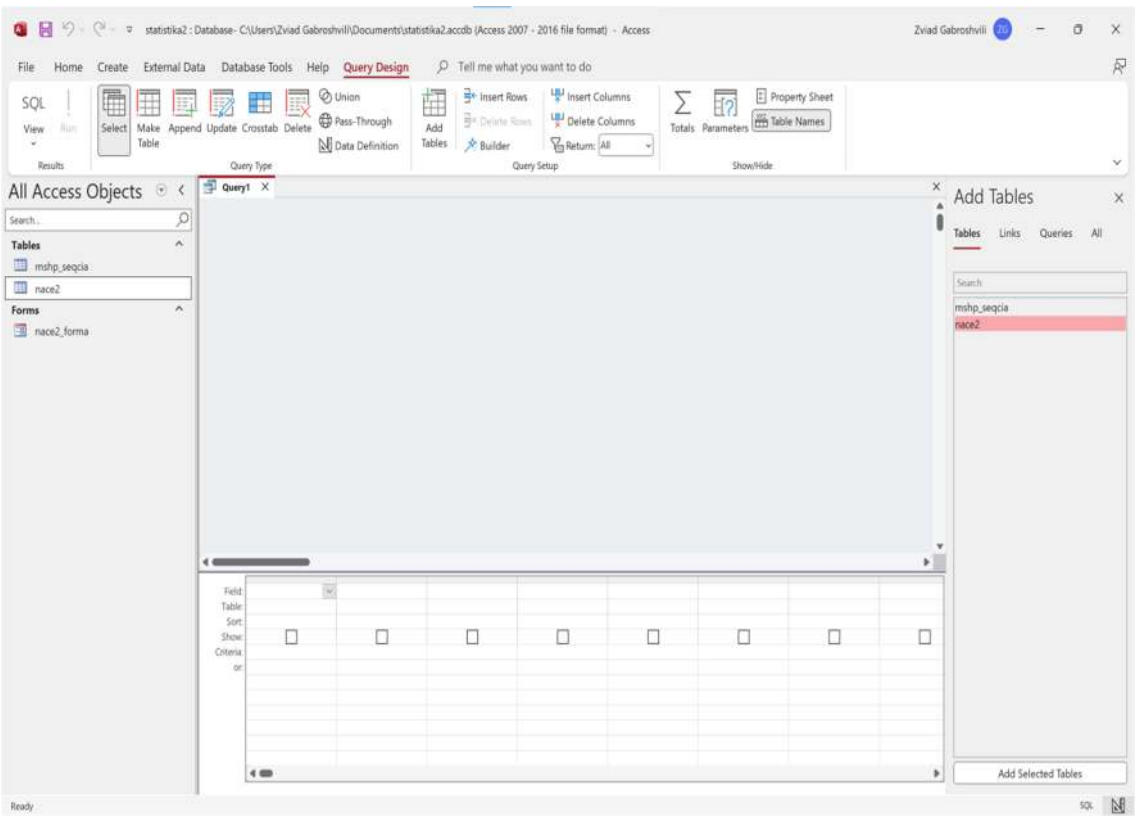
1. რაში შეგვიძლია გამოვიყენოთ Microsoft Office Access?
2. აღწერეთ ცხრილების შექმნის პროცესი MS Access-ში.
3. აღწერეთ ცხრილებში მონაცემების შეტანის სხვადასხვა წესი.
4. როგორ ხდება მონაცემების შემტანი ფორმების შექმნა Access-ში?

# თავი 11. მოთხოვნები MS Access-ში

## 11.1. მოთხოვნების შექმნა და გამოყენება MS Access-ში

MS Access-ში შესაძლებელია შეიქმნას სხვადასხვა ტიპის მოთხოვნა. დავიწყოთ ისეთი მოთხოვნის შექმნით, რომელიც მონაცემებს ამოიღებს ბაზიდან, როგორც SQL-ის შემთხვევაში ვაკეთებდით SELECT ბრძანების გამოყენებით. მაგალითისთვის, შევქმნათ მოთხოვნა, რომელიც Access-ში mshp_seqcia და nace2 ცხრილს დააკავშირებს და ამოიღებს მონაცემებს. ამისათვის შევდივართ Create ჩანართში და ვირჩევთ Query Design-ს. გამოვა მოთხოვნის შემუშავებისათვის საჭირო ინტერფეისი, როგორიც გამოსახულია სურათზე 11.1.

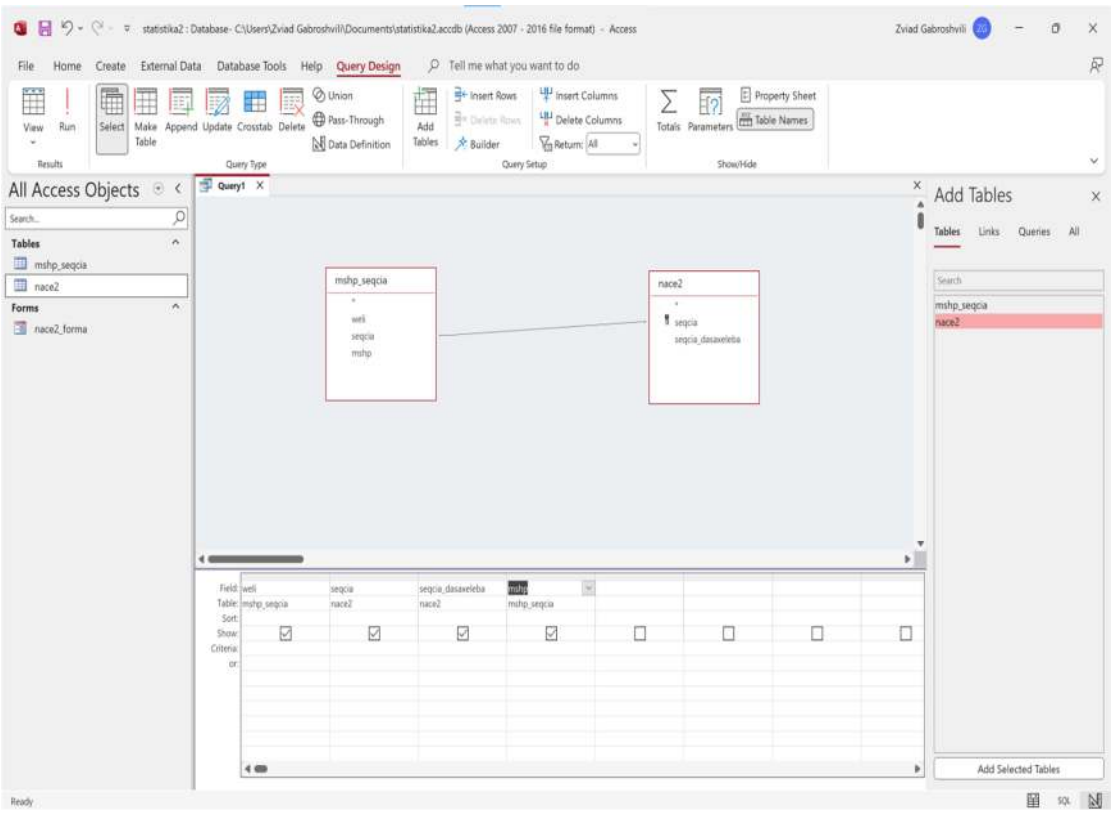
**სურათი 11.1. მოთხოვნის დიზაინის ინტერფეისი**




ამ ინტერფეისში მარჯვნივ Add Tables პანელში ჩამოთვლილია ცხრილები, რომლებიც შეგვიძლია გადმოვიტანოთ ამ ინტერფეისის ცენტრალურ ნაწილში, ხოლო ქვედა ნაწილში შეგვიძლია ავირჩიოთ ველები შესაბამისი ცხრილის ველზე ორჯერ დაწკაპუნებით და ასევე, შესაბამისი კრიტერიუმები. სურათზე 11.2 გამოსახულია უკვე

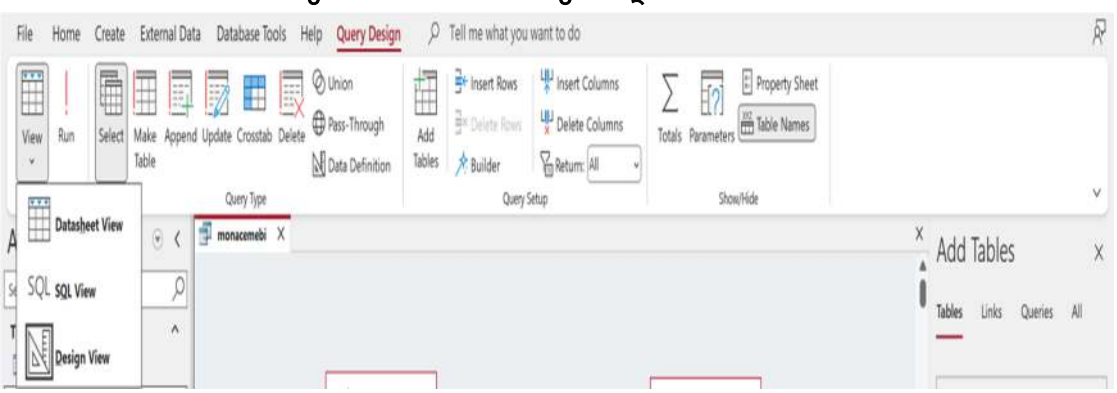
შექმნილი მოთხოვნა, გადმოტანილია ცენტრში ცხრილები, ხოლო ინტერფეისის ქვედა ნაწილში მოცემულია არჩეული ველები.

**სურათი 11.2. შექმნილი მოთხოვნა**




შეგვიძლია მოთხოვნა შევინახოთ Ctrl+S ღილაკებით კლავიატურაზე და დავარქვათ monacemebi. ახლა ჩვენ ვიმყოფებით მოთხოვნის დიზაინის ინტერფეისზე; თუ გვსურს გავუშვათ მოთხოვნა, ამისათვის დავაწკაპუნოთ Query Design ჩანართში Run ღილაკზე, რომელიც გამოსახულია სურათზე 11.3.

**სურათი 11.3. მოთხოვნის დიზაინის ჩანართი**



არსებული მოთხოვნიდან შეგვიძლია გადავიდეთ შესაბამის SQL კოდზე. ამისათვის Query Design ჩანართში ვირჩევთ View>SQL View-ს. SQL View ინტერფეისი გამოსახულია სურათზე 11.4. ამავე ჩანართში შეგვიძლია კოდის შეცვლაც.

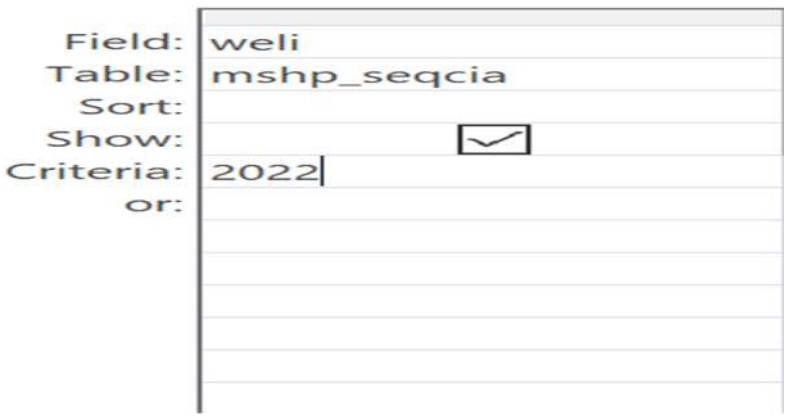
**სურათი 11.4. SQL View ჩანართი**


Design View ჩანართიდან შეგვიძლია გარკვეული პირობების დადება მონაცემთა ამოღებისთვის, მაგალითად, ეს შეიძლება იყოს მონაცემების გაფილტვრა წლის მიხედვით. პირობითად ავიღოთ 2022 წელი. ამისათვის weli ველის ქვემოთ ჩავწეროთ

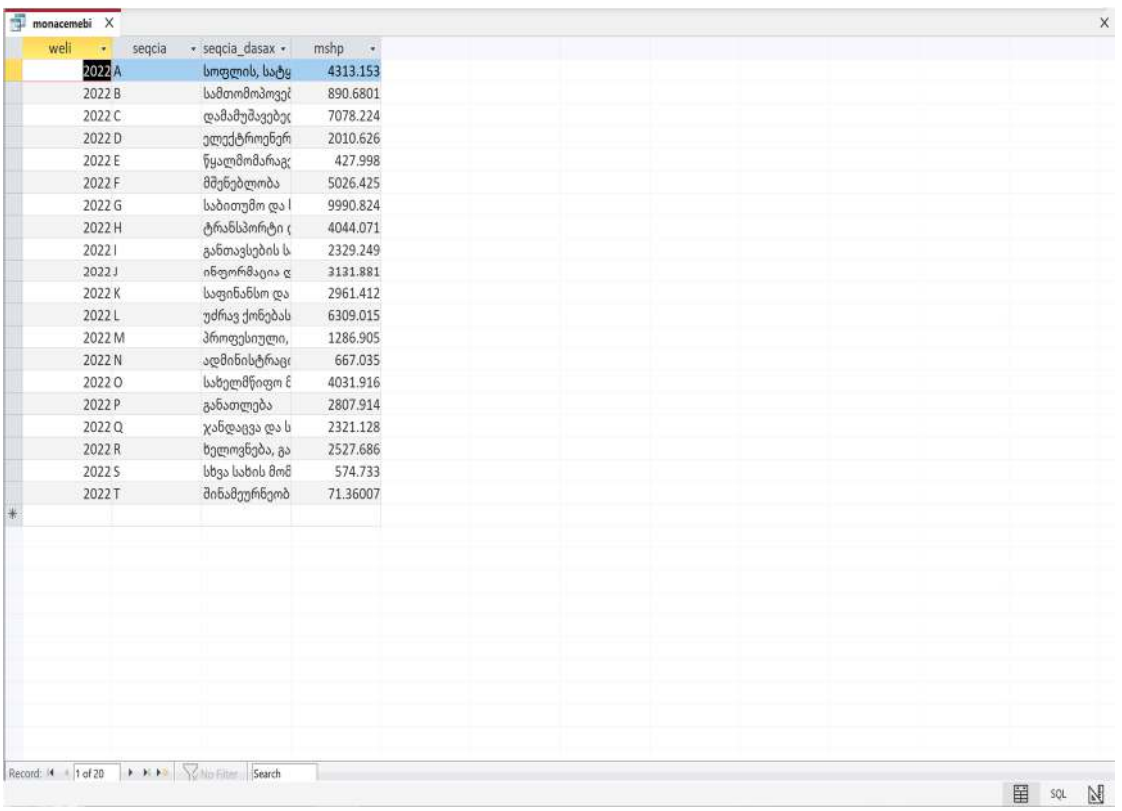
2022 Criteria ველში, როგორც ნაჩვენებია სურათზე 11.5. შედეგი შეგვიძლია ვნახოთ მოთხოვნის Datasheet View ინტერფეისში, როგორც ეს ნაჩვენებია სურათზე 11.6.

Criteria ველში შეგვიძლია მივუთითოთ AND და OR ფილტრაციაც იგივე შინაარსით, როგორც SQL სინტაქსის დროს. მაგალითად, თუ გვსურს, რომ ამოვიღოთ 2021 და 2022 წელი, მაშინ ამ ველში მივუთითებთ შემდეგ კოდს: „2022 or 2021“.

**სურათი 11.5. კრიტერიუმის გამოყენება**




**სურათი 11.6. მოთხოვნის Datasheet View ინტერფეისი**

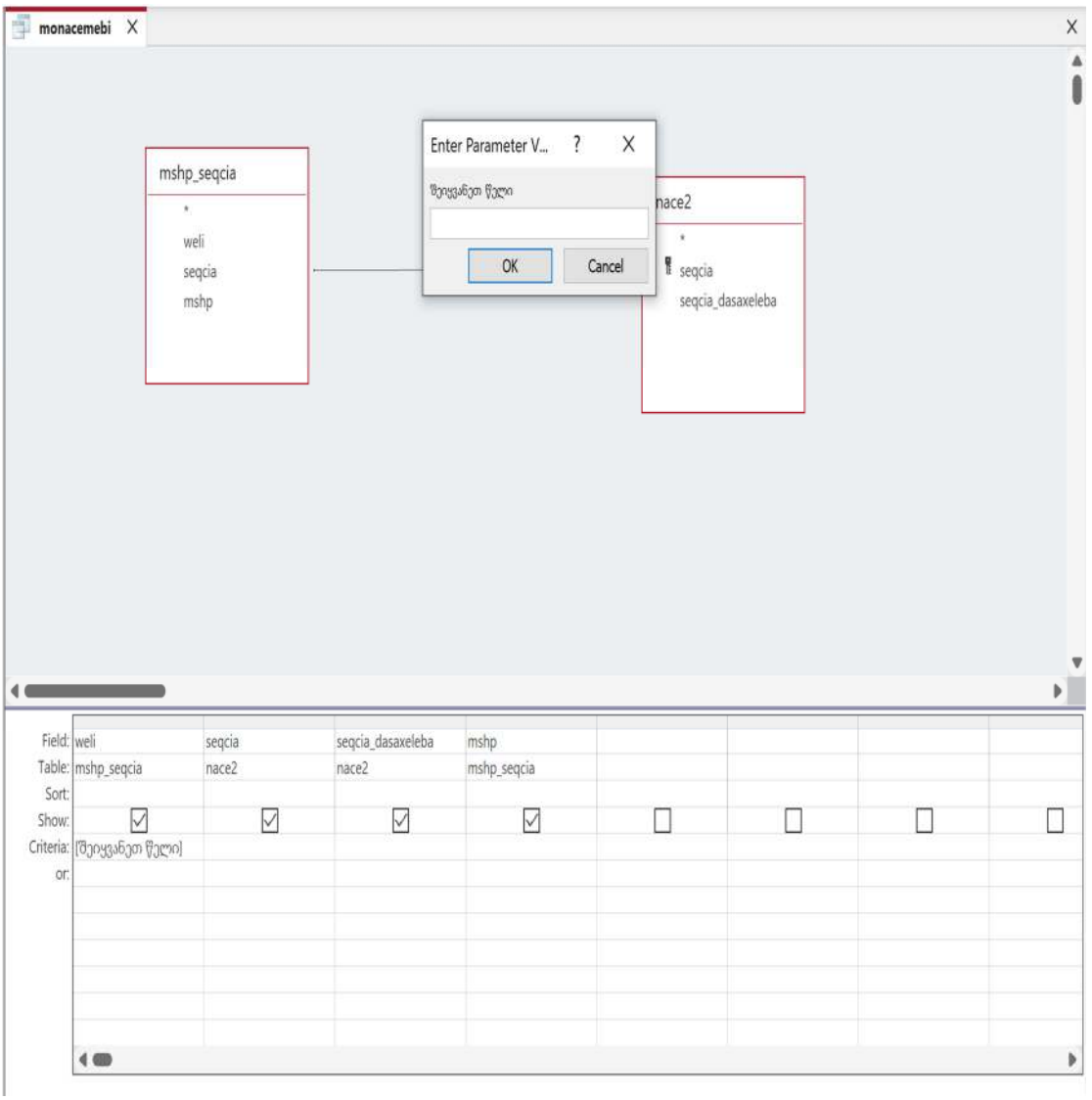


MS Access-ში შესაძლებელია მოთხოვნებში პარამეტრების მიწოდებაც. მაგალითად, weli ველის Criteria ველში ჩავწეროთ შემდეგი ტექსტი და გამოვიყენოთ კვადრატული ფრჩხილები:

| [შეიყვანეთ წელი] |
|---|

შედეგად, როდესაც გაეშვება მოთხოვნა, გამოვა დიალოგური ფანჯარა, როგორიც გამოსახულია სურათზე 11.7 და მოგვთხოვს წლის შეყვანას; მისი შეყვანის შემდეგ დავაწკაპუნებთ OK ღილაკზე. ამის შემდეგ მონაცემთა ბაზა ამოიღებს ისეთ მონაცემებს, რომლებიც აკმაყოფილებენ შეყვანილ პარამეტრს შესაბამისი ველისთვის.

**სურათი 11.7. პარამეტრის მოთხოვნა**

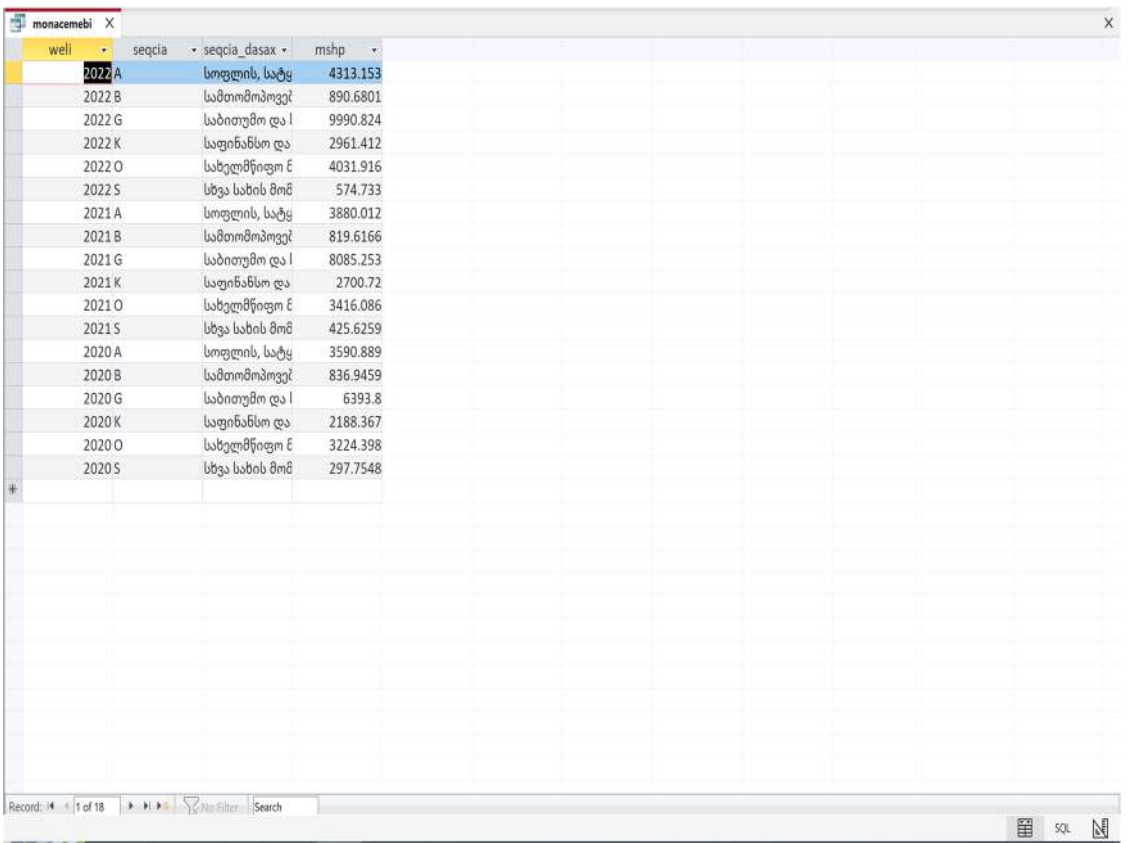


Access-ში შესაძლებელია გამოვიყენოთ LIKE ოპერატორიც, მაგალითად,T-SQL-ში თუ გვსურდა მოგვეძებნა ისეთი მონაცემები, რომლებიც დაიწყებოდნენ „ს“ ასოზე, მაშინ შესაბამისი LIKE პირობა იქნებოდა: „LIKE 'ს%'“, მაგრამ Access-ის შემთხვევაში Criteria ველში ჩასაწერი პირობა ასეთი იქნება:

```
Like "ს*"
```

თუ ამ პირობით შედგენილ მოთხოვნას, რომელიც ფილტრავს seqcia_dasaxeleba ველს, გავუშვებთ, მივიღებთ შედეგს, როგორიც გამოსახულია სურათზე 11.8.

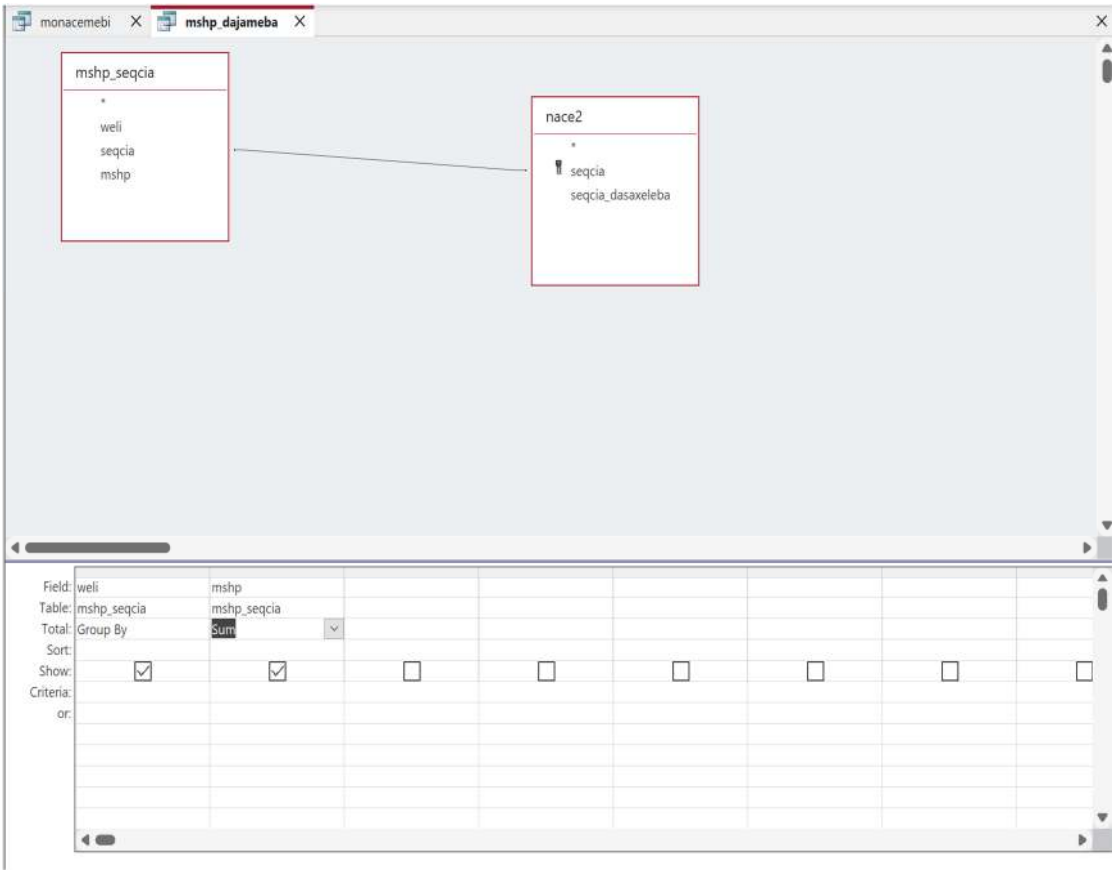
**სურათი 11.8. Like ტიპის ფილტრაციით მიღებული შედეგი**




შესაძლებელია გამოვიყენოთ აგრეგირებული ფუნქციებიც. დავუშვათ, გვჭირდება მშპ-ის დაჯამება წლების მიხედვით. ამისთვის შევქმნათ ახალი მოთხოვნა, დავარქვათ მას mshp_dajameba, გამოვიყენოთ მხოლოდ ორი ველი - weli და mshp; Query Design ჩანართში გავააქტიუროთ Totals ღილაკი, ხოლო mshp ველის ქვემოთ გამოჩენილ Total ველში ავირჩიოთ Sum, როგორც ნაჩენებია სურათზე 11.9.

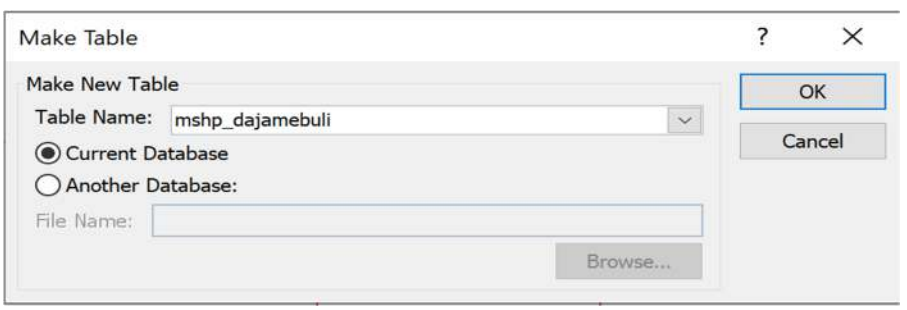
გარდა დაჯამებისა, შესაძლებელია სხვადასხვა ტიპის აგრეგირებული ფუნქციის გამოყენებაც, როგორიც შეიძლება იყოს, მაგალითად, avg (საშუალო არითმეტიკული), min (მინიმალური), max (მაქსიმალური), stdev (სტანდარტული გადახრა), count (დათვლა) და ა. შ.

სურათი 11.9. აგრეგირებული ფუნქციის გამოყენება



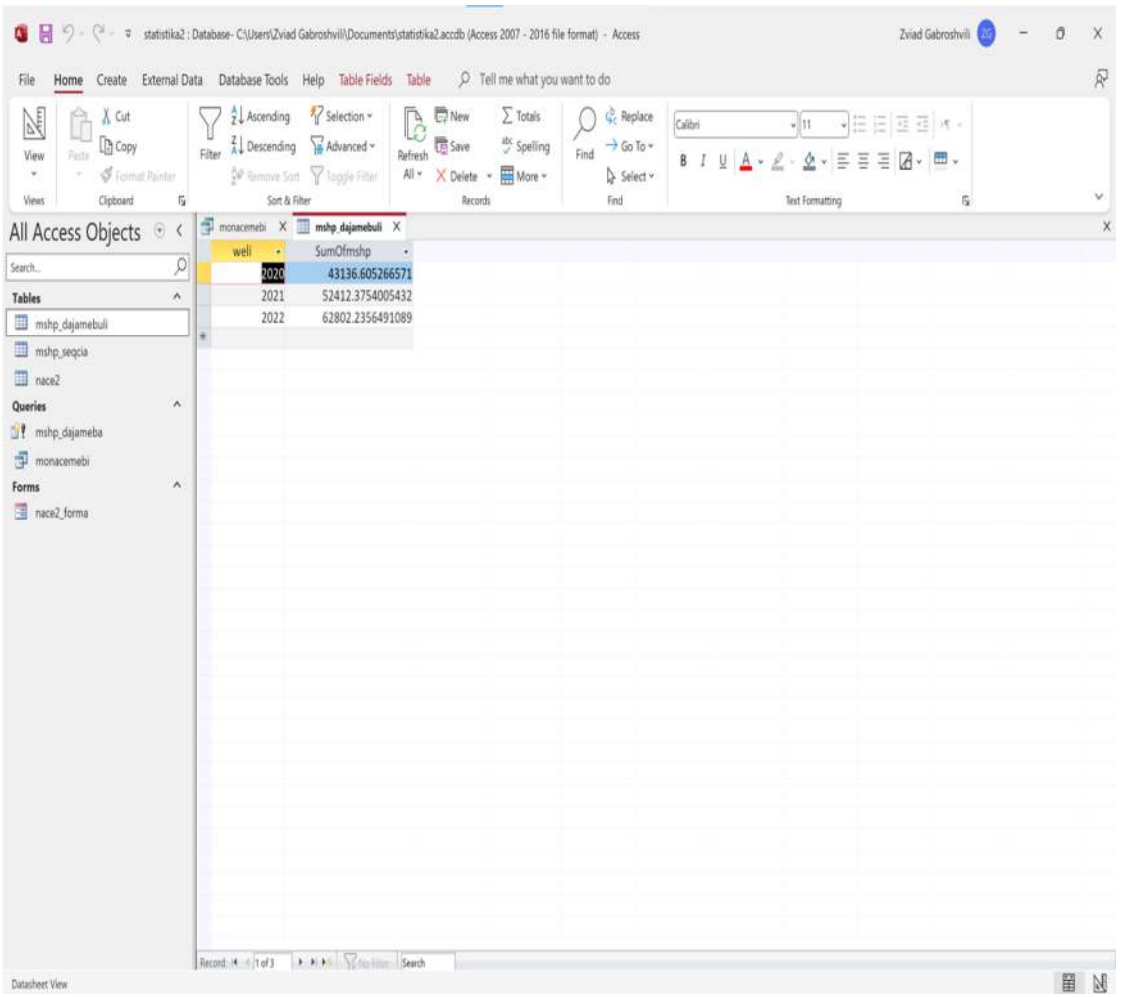

Access-ში მხარდაჭერილია სხვა ტიპის მოთხოვნებიც. მაგალითად, განვიხილოთ Make Table მოთხოვნა. ამისათვის შეგვიძლია შევქმნათ ახალი მოთხოვნა Create ჩანართიდან და მერე ავირჩიოთ Make Table ღილაკი Query Design ჩანართში, ან შევცვალოთ არსებული მოთხოვნა, მაგალითად, mshp_dajameba, Query Design ჩანართიდან და გავააქტიუროთ Make Table ღილაკი. ამ ღილაკის გააქტიურების შემდეგ გამოვა ფანჯარა, სადაც ავირჩევთ ახალი ცხრილის სახელს, როგორც ეს ნაჩვენებია სურათზე 11.10 და დავაწკაპუნებთ OK ღილაკზე.

სურათი 11.10. Make Table მოთხოვნა



თუ გავუშებთ ამ მოთხოვნას, მაშინ შეიქმნება ახალი ცხრილი, რომელშიც იქნება მონაცემები შესაბამისი მოთხოვნიდან. სურათზე 11.11 ჩანს ხსენებული მოთხოვნის გაშვების შედეგი. როგორც ვხედავთ, All Access Objects პანელში გამოჩნდა ახალი ცხრილი, რომლის გახსნის შედეგად დავინახავთ შესაბამის მონაცემებს.

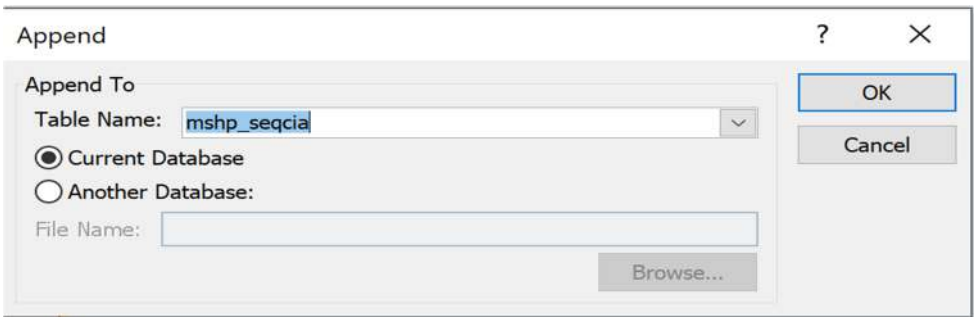
**სურათი 11.11. Make Table მოთხოვნის გაშვების შედეგი**



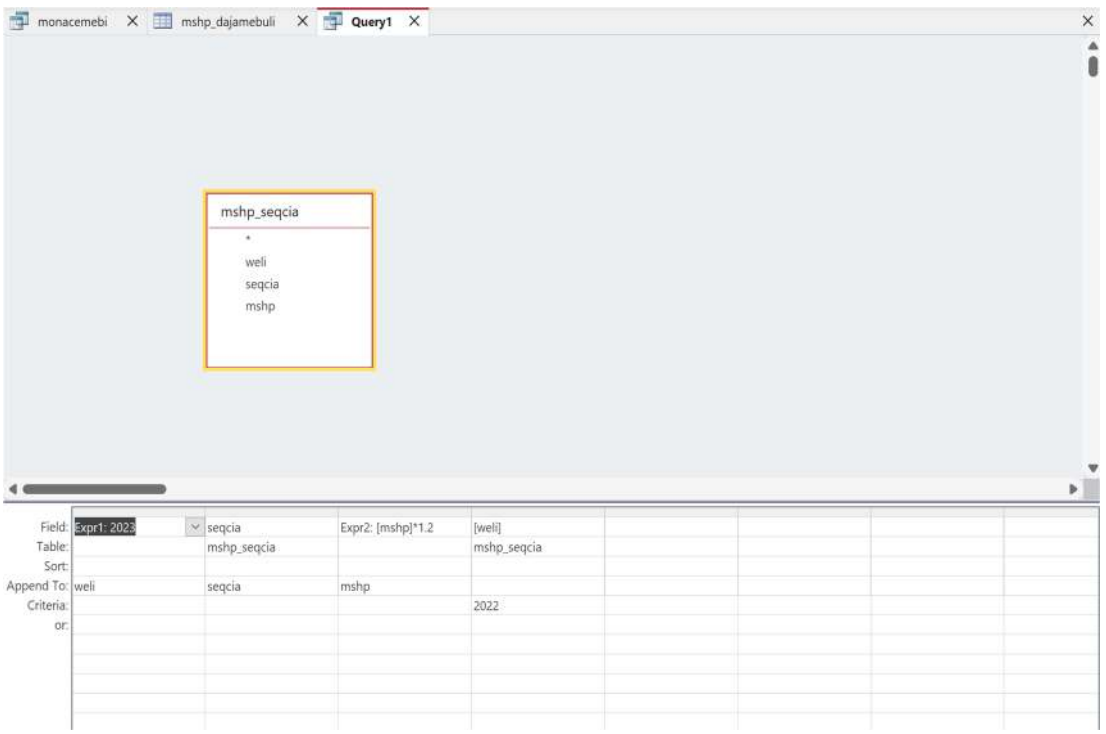

ახლა განვიხილოთ Append ტიპის მოთხოვნა და არითმეტიკული ოპერატორების მხარდაჭერა. ამისათვის შევქმნათ მოთხოვნა, მას დავარქვათ mshp_damateba და გადავიტანოთ mshp_seqcia ცხრილი. ამის შემდეგ გადავრთოთ Query Design ჩანართიდან Append ღილაკზე. გამოვა ფანჯარა, როგორიც გამოსახულია სურათზე 11.12 და ჩავწეროთ Table Name ველში შესაბამისი ცხრილის სახელი. შემდეგ პირველი ველის Field ველში ჩავწეროთ 2023, ხოლო მესამე ველის Field-ის გასწვრივ ჩავწერთ mshp*1.2 და მეოთხე სვეტად ავირჩიოთ weli, მაგრამ Append To ველი დავუტოვოთ ცარიელი, ხოლო Criteria ველში 2022-ს ჩავუწერთ ისე, როგორც ნაჩვენებია სურათზე 11.13. ამ მოთხოვნას დავარქვათ mshp_damateba და შევინახოთ. მოთხოვნის გაშვების შემდეგ mshp_seqcia

ცხრილში ჩაიწერება 2022 წლის მშპ-ის ნამრავლი 1.2-ზე, ხოლო weli ველში იქნება მოთავსებული 2023.

**სურათი 11.12. ცხრილის სახელის არჩევა**

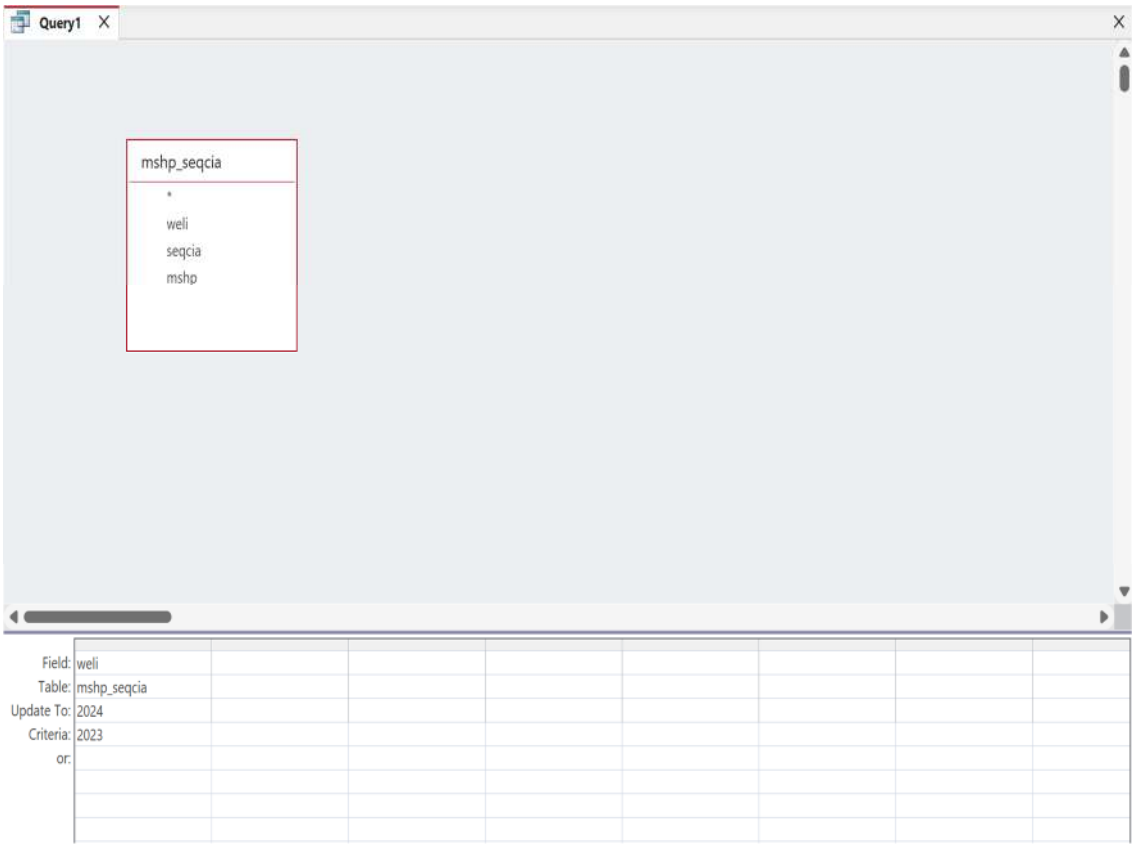



**სურათი 11.13. Append ტიპის მოთხოვნის შემუშავება**



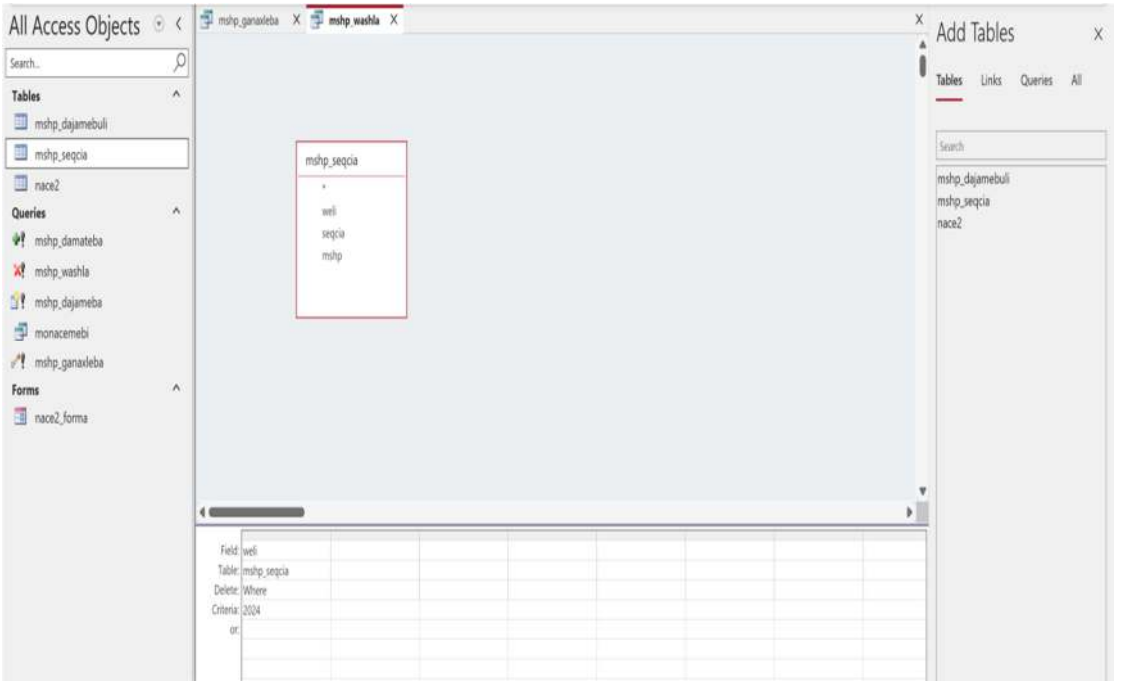

შემდეგ განვიხილოთ Update ტიპის მოთხოვნა. დავუშვათ, გვსურს ჩვენ მიერ mshp_seqcia-ში შეტანილ ახალ მონაცემებში შევცვალოთ weli ველში ჩაწერილი მონაცემი და ვაქციოთ ის 2024 წლის მონაცემად, თუ ამავე ველში არის ისეთი ჩანაწერი, რომელშიც წერია 2023. შევქმნათ მოთხოვნა და გადავიტანოთ weli სვეტი განახლებისთვის, როგორც ნაჩვენებია სურათზე 11.14. აქ Update To ველში ჩავწეროთ 2024, ხოლო Criteria ველში - 2023; შევინახოთ და დავარქვათ mshp_ganaxleba და გავუშვათ მოთხოვნა. შედეგად, mshp_seqcia ცხრილში მონაცემები განახლდება ზემოხსენებული პირობის მიხედვით.

სურათი 11.14. Update ტიპის მოთხოვნა




შემდეგ მოთხოვნას, რომელსაც განვიხილავთ, არის Delete ტიპის მოთხოვნა. ამჯერად mshp_seqcia ცხრილიდან წავშალოთ ისეთი მონაცემები, რომლებსაც weli ველში უწერია 2024. ამისათვის შევქმნათ მოთხოვნა და დავარქვათ მას mshp_washla. გადმოვიტანოთ mshp_seqcia ცხრილი და weli ველი, შემდეგ weli ველს ჩავუწეროთ Criteria ველში 2024. თუ გავუშვებთ ამ მოთხოვნას, მაშინ mshp_seqcia ცხრილიდან წაიშლება ის მონაცემები, რომლებსაც უწერიათ weli ველში 2024.

სურათი 11.15. Delete ტიპის მოთხოვნა



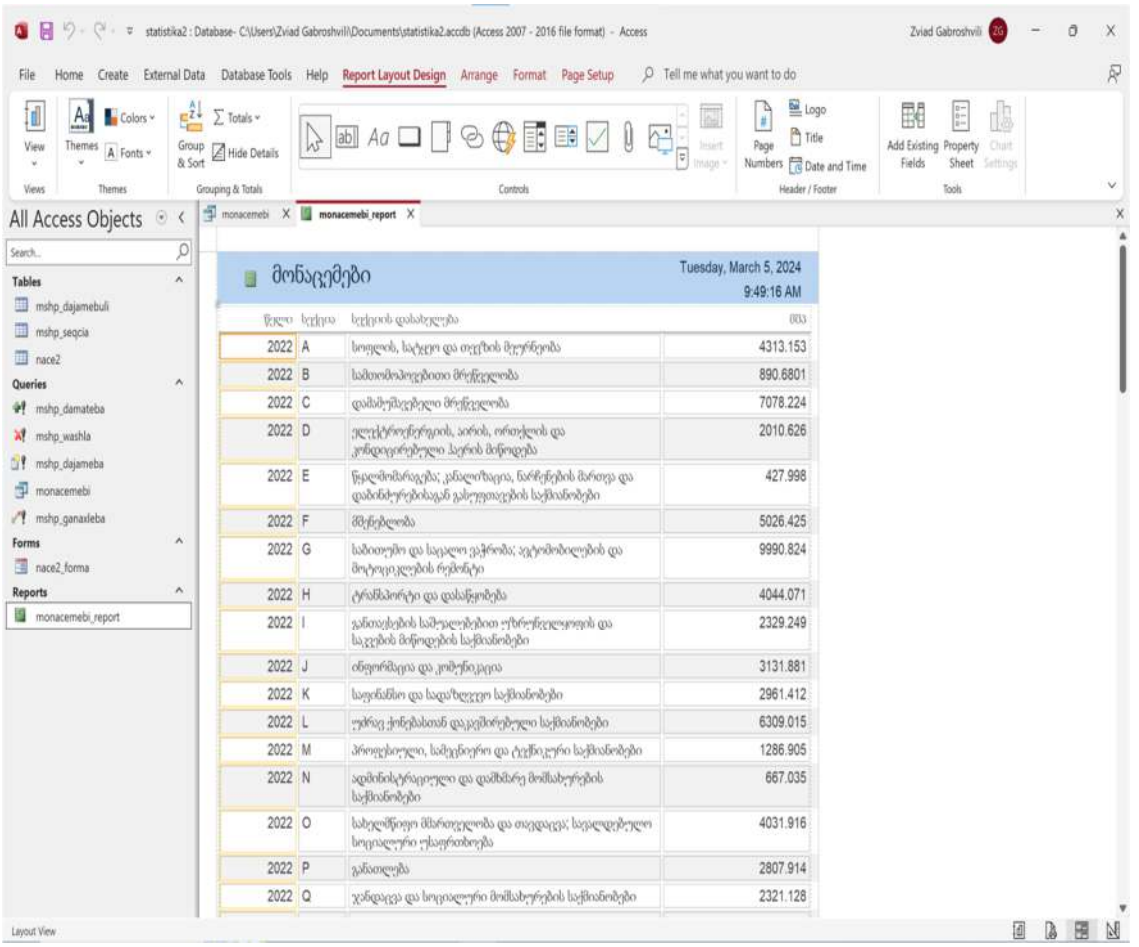

როგორც ვნახეთ, შესაძლებელია სხვადასხვა ტიპის მოთხოვნის შექმნა და გამოყენება მონაცემთა ბაზაში გრაფიკულად, რაც მნიშვნელოვნად ამარტივებს მონაცემებთან მუშაობის პროცესს.

## 11.2. რეპორტები

Access-ის ერთ-ერთი ობიექტი არის რეპორტი. რეპორტების დახმარებით შეგვიძლია მონაცემები ამოვიღოთ კონკრეტული მოთხოვნის საფუძველზე. ამისათვის მოვნიშნოთ შესაბამისი მოთხოვნა და Create ჩანართში დავაწკაპუნოთ Report ღილაკზე. ავაგოთ რეპორტი, როგორც ის გამოსახულია სურათზე 11.16 და შევინახოთ, როგორც monacemebi_report.

მოცემული რეპორტი შეგვიძლია ამოვბეჭდოთ და დავუმატოთ სხვადასხვა სასურველი ველი. რეპორტების დახმარებით შესაძლებელია ამა თუ იმ ორგანიზაციაში მნიშვნელოვნად გამარტივდეს სხვადასხვა ტიპის ანალიტიკური სამუშაო.

**სურათი 11.16. რეპორტი**




Report Layout View ჩანართიდან View ღილაკზე მოქმედებით შეგვიძლია შევცვალოთ ინტერფეისი რეპორტების ნახვისა და დიზაინისთვის.

***კითხვები თვითშემოწმებისთვის:***

1. როგორ შევქმნათ მოთხოვნები Access-ში?
2. მიმოიხილეთ სხვადასხვა ტიპის მოთხოვნა Access-ში.
3. ახსენით კრიტერიუმების გამოყენების გზა მონაცემთა ფილტრაციისთვის.
4. როგორ იქმნება რეპორტები?

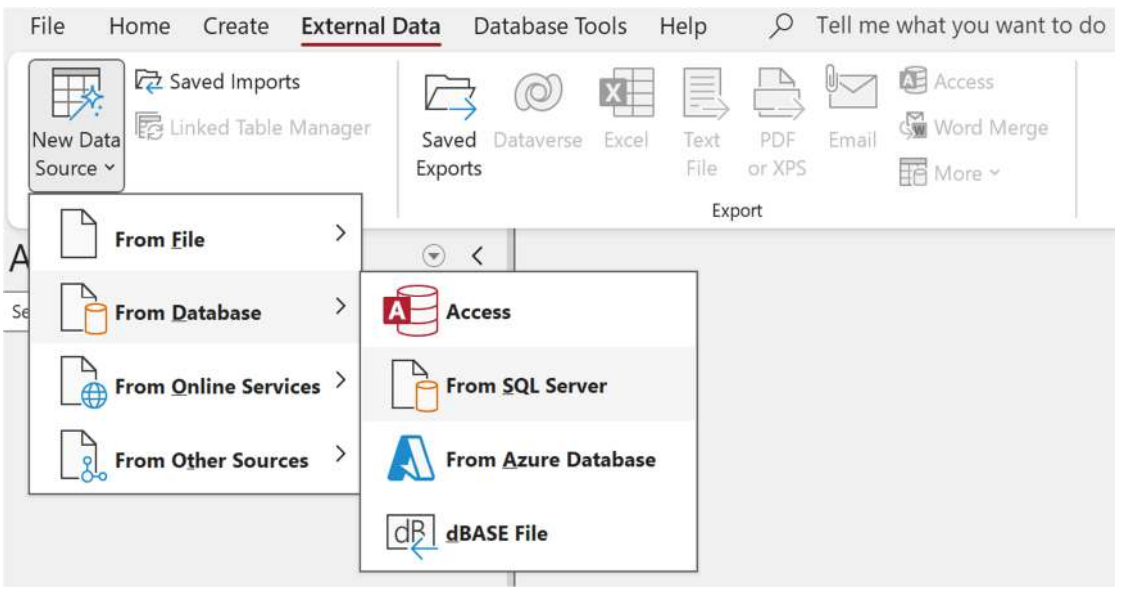
# თავი 12. Microsoft Office Access-ის კავშირი SQL Server 2022-თან

## 12.1. მონაცემთა ბაზასთან დაკავშირება

Microsoft Office Access შეუძლია დაუკავშირდეს SQL Server 2022-ში განთავსებულ მონაცემთა ბაზას, რითაც შესაძლებელია მასთან მუშაობა გახდეს მნიშვნელოვნად მარტივი.

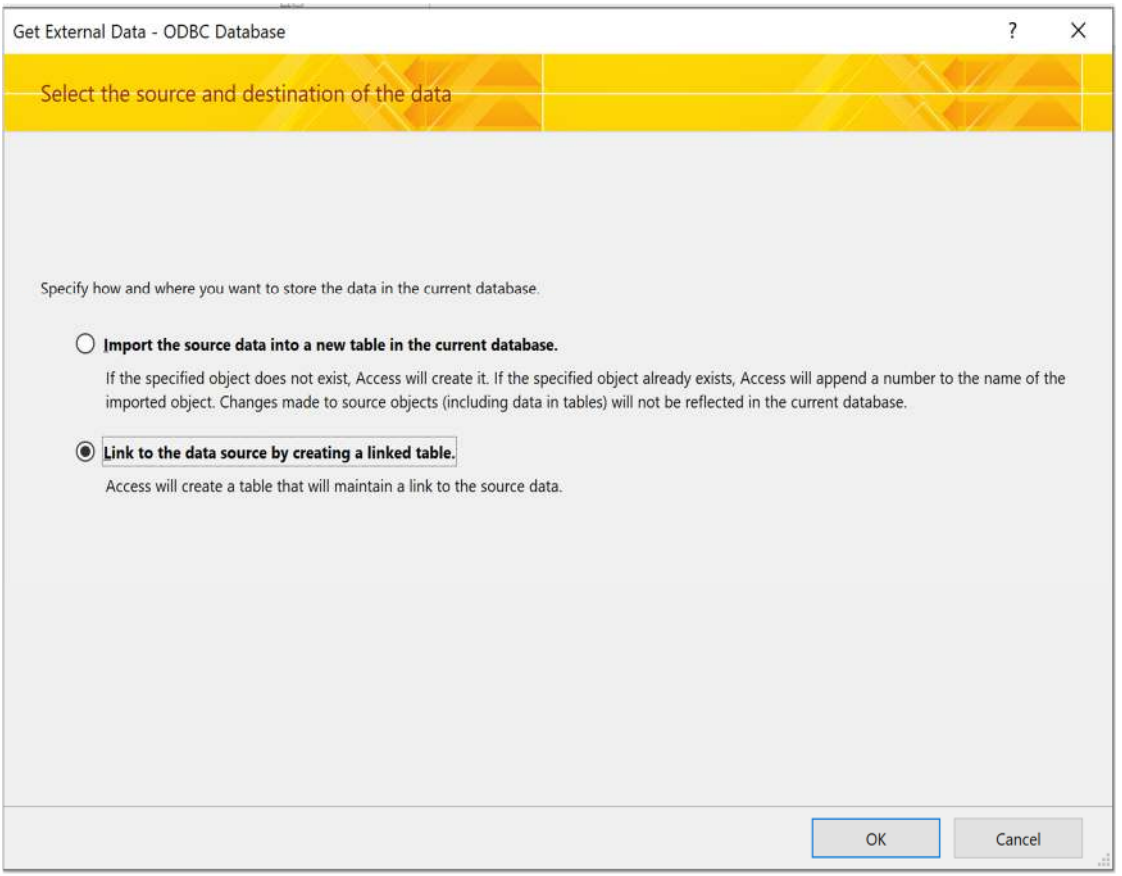
გავხსნათ Access და შევქმნათ ახალი მონაცემთა ბაზა, დავარქვათ მას statistika3. შემდეგ საჭიროა შევქმნათ მონაცემთა ახალი წყარო. ამისათვის External Data ჩანართში ავირჩიოთ New Data Source>From Database>From SQL Server ბრძანება, როგორც ნაჩვენებია შემდეგ სურათზე 12.1:

**სურათი 12.1. მონაცემთა ახალი წყაროს დაზუსტება**



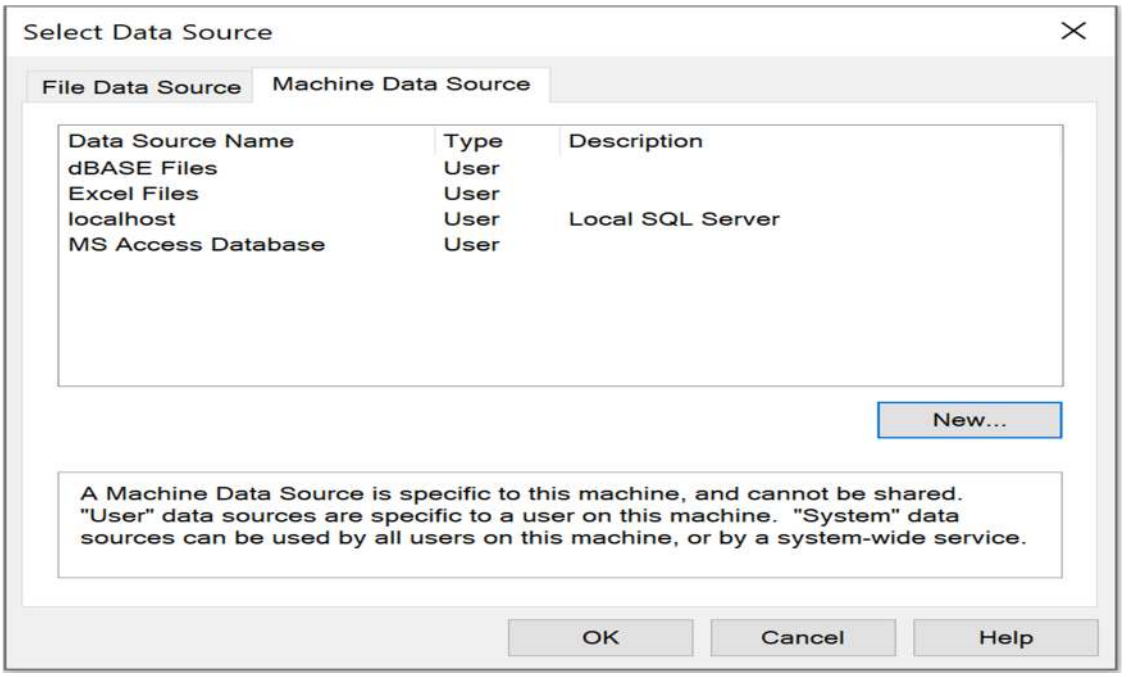

შემდეგ გამოვა ფანჯარა, როგორიც ეს გამოსახულია სურათზე 12.2. აქ უნდა ავირჩიოთ Link to the data source by creating a linked table ვარიანტი, რადგან გვსურს, რომ კავშირი შევინარჩუნოთ ბაზასთან ოპერაციების განხორციელების დროს. შედეგად, სერვერზე არსებულ ბაზაშიც აისახება ცვლილებები და უნდა დავაწკაპუნოთ OK ღილაკზე.

სურათი 12.2. მონაცემთა წყაროსა და დანიშნულების ადგილის დაზუსტება

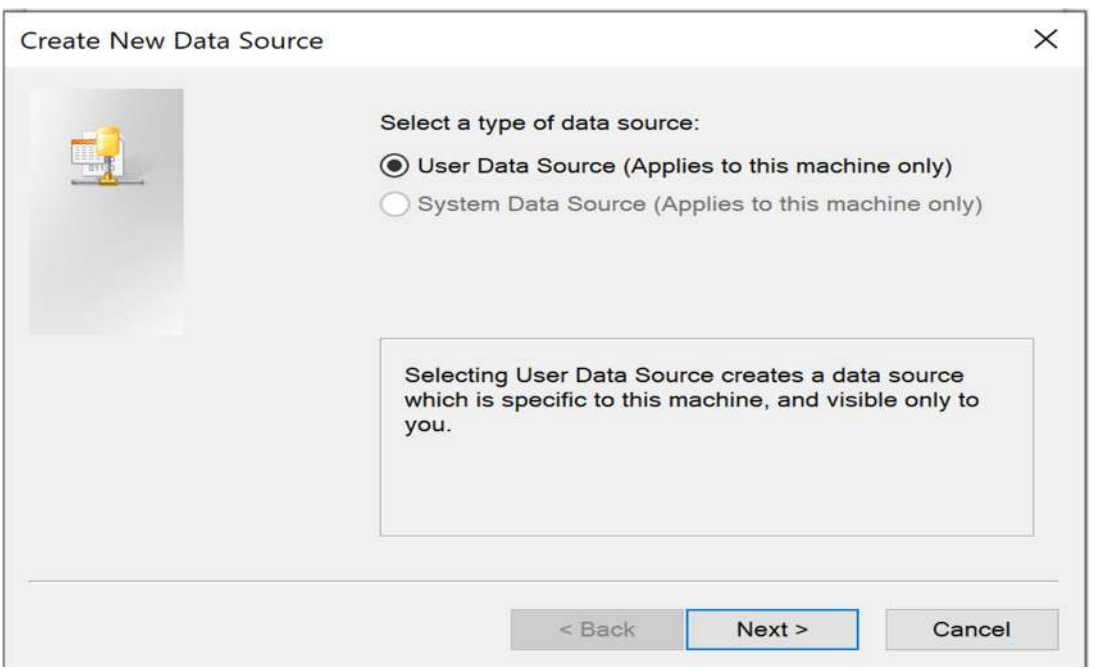



გამოვა ფანჯარა, რომელშიც უნდა გადავიდეთ Machine Data Source ჩანართზე, როგორც ეს გამოსახულია სურათზე 12.3 და დავაწკაპუნოთ New... ღილაკზე. გამოჩნდება ახალი ფანჯარა, როგორიც ნაჩვენებია სურათზე 12.4, სადაც ავირჩევთ User Data Source ვარიანტს და Next ღილაკით გადავალთ შემდეგ ეტაპზე; აქ ავირჩევთ ODBC Driver 17 for SQL Server-ს, როგორც ეს გამოსახულია სურათზე 12.5; ვაგრძელებთ Next ღილაკით და შემდეგ ეტაპზე ვასრულებთ Finish ღილაკით. ამის შემდეგ უკვე გამოდის ფანჯარა, როგორსაც ვხედავთ სურათზე 12.6 და ვუთითებთ იმ სახელს, რაც გვინდა, რომ ერქვას Name ველში, აღწერას Description ველში, ხოლო Server ველში სერვერის ადგილმდებარეობას. ჩვენ შემთხვევაში ვირჩევთ localhost-ს, სხვა ვარიანტში ავირჩევდით შესაბამისი სერვერის მისამართს ორგანიზაციაში. ვაგრძელებთ Next ღილაკით.

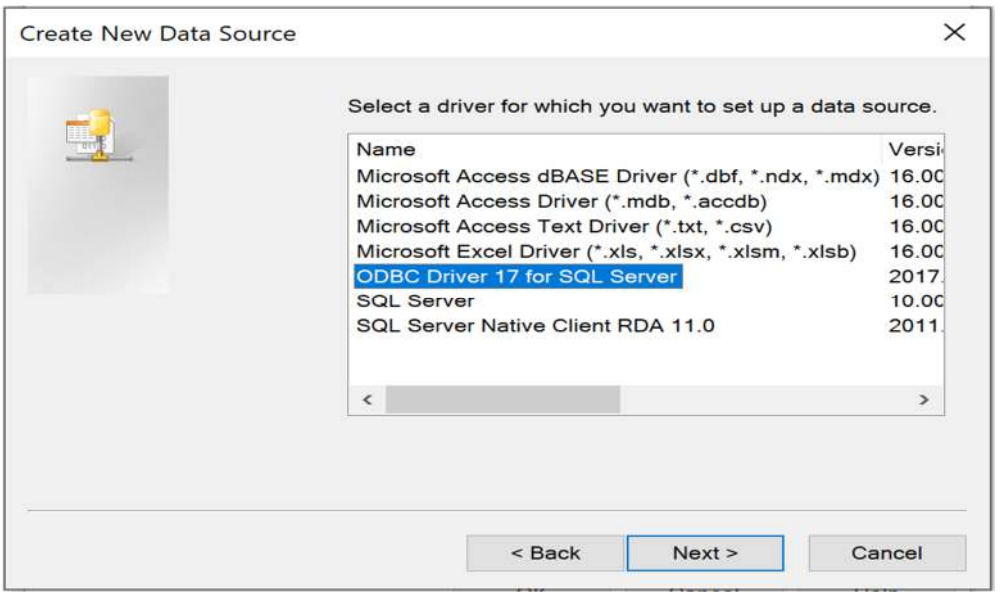
სურათი 12.3. მონაცემთა წყაროს არჩევა



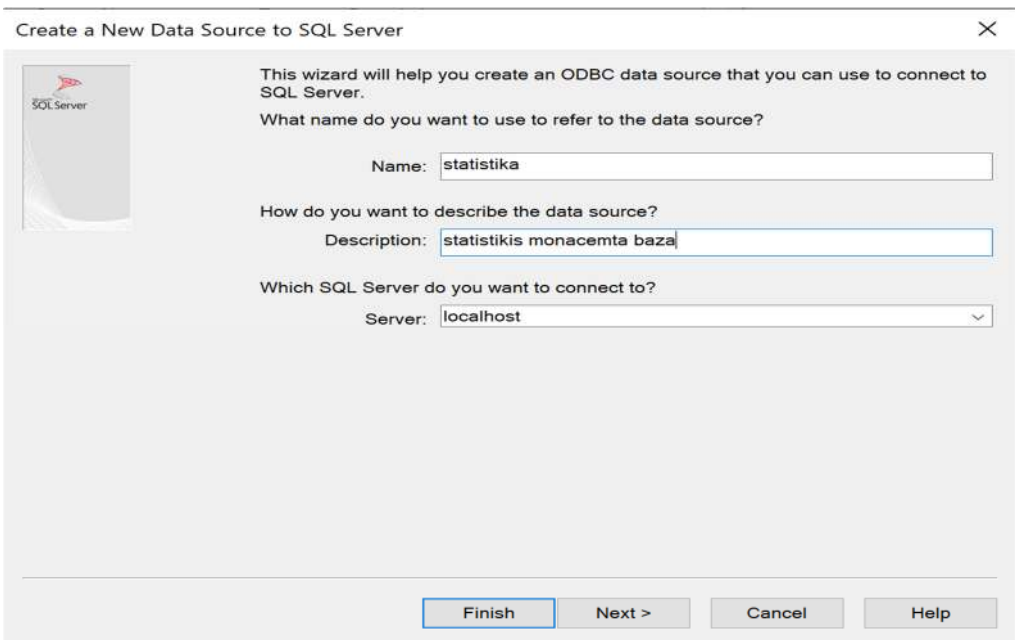

სურათი 12.4. მონაცემთა წყაროს ტიპის არჩევა



სურათი 12.5. დრაივერის არჩევა

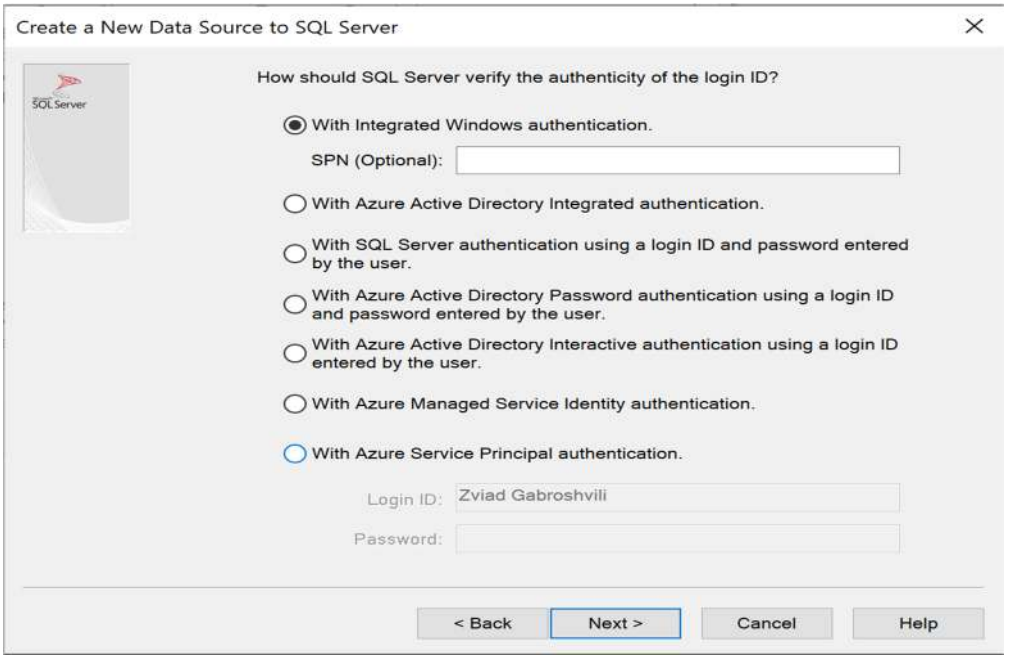



სურათი 12.6. მონაცემთა ახალი წყაროს შექმნა

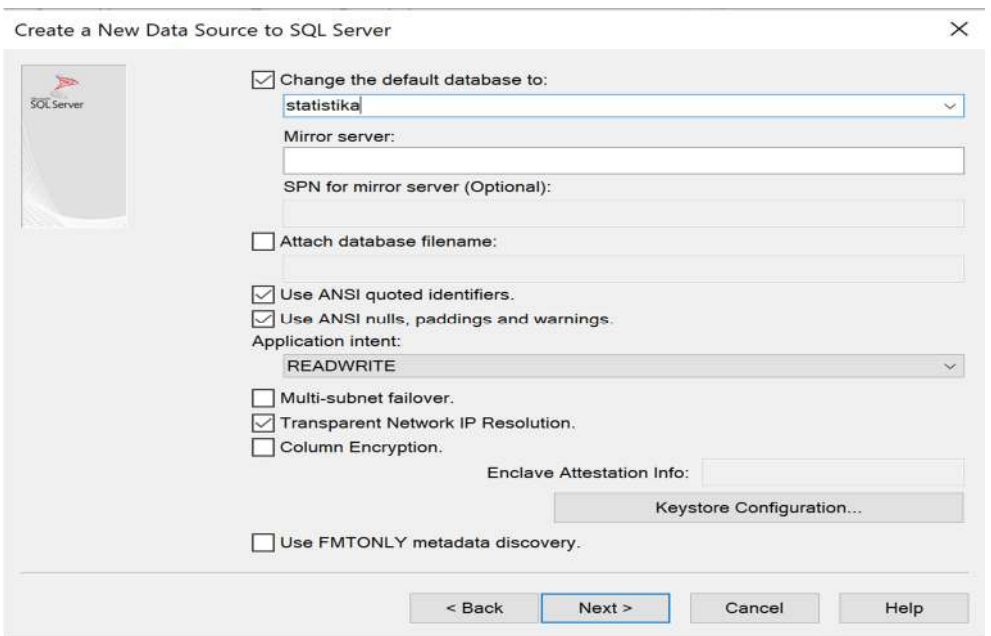



შემდეგ ეტაპზე დავინახავთ ფანჯარას, რომელიც მოცემულია სურათზე 12.7. აქ ვირჩევთ აუთენთიკაციის მექანიზმს და ვაგრძელებთ Next ღილაკით. ამ ეტაპზე ვირჩევთ მონაცემთა ბაზას, როგორც ეს ნაჩვენებია სურათზე 12.8. გავიხსენოთ, რომ სერვერზე ჩვენს ბაზას ერქვა statistika. ამის შემდეგ ვაწკაპუნებთ Next ღილაკზე. შემდეგ ეტაპზე ვასრულებთ პარამეტრების მითითებას, ვაწკაპუნებთ Finish ღილაკზე იმ ფანჯარაზე, როგორიც ნაჩვენებია სურათზე 12.9. მივიღებთ 12.10 სურათზე გამოსახულ ფანჯარას, სადაც გამოჩნდება შესაბამისი პარამეტრები და დავაწკაპუნებთ OK ღილაკზე.

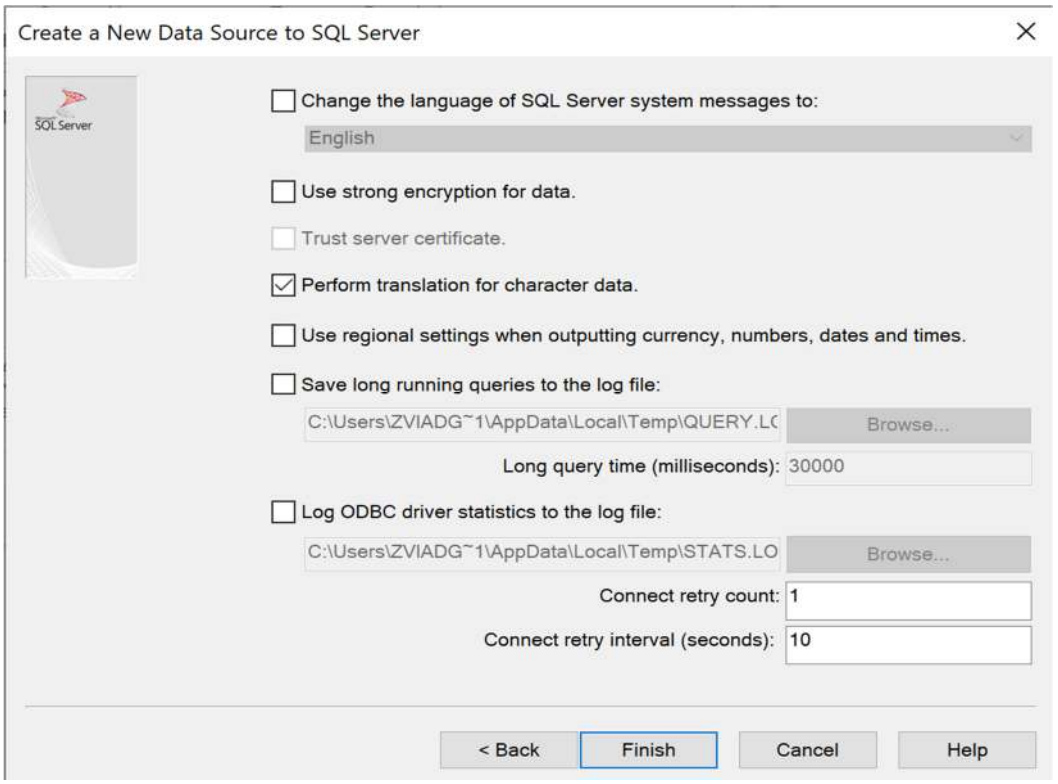
სურათი 12.7. აუთენთიკაციის მექანიზმის არჩევა



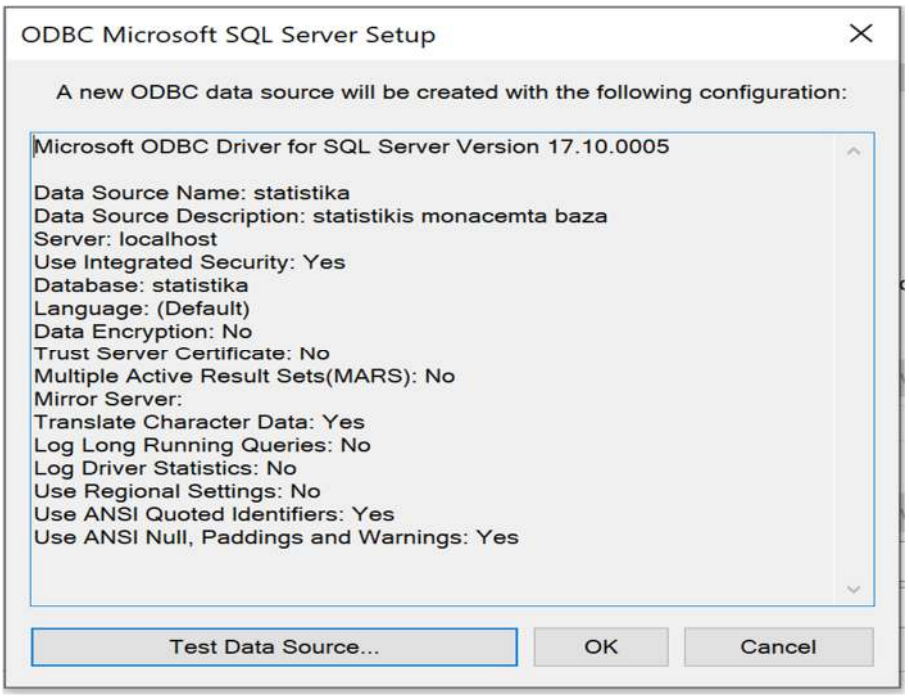

სურათი 12.8. მონაცემთა ბაზის შეცვლა



სურათი 12.9. მონაცემთა ახალი წყაროს შექმნის პროცესისთვის პარამეტრების შეყვანის დასრულება

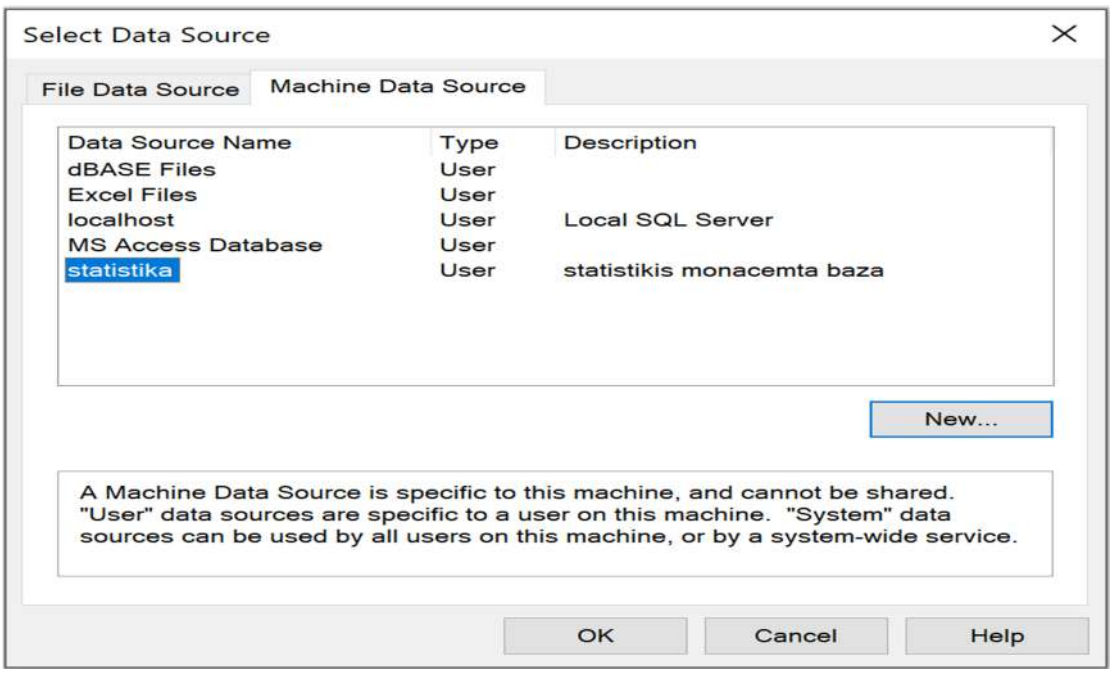



სურათი 12.10. მონაცემთა წყაროს პარამეტრები



რადგან მონაცემთა ახალი წყარო შეიქმნა, შეგვიძლია გამოვიყენოთ ის მონაცემების ამოსაღებად. ამისათვის ავირჩევთ მონაცემთა წყაროს, ჩვენ შემთხვევაში statistika-ს Select Data Source ფანჯარაში, როგორ ნაჩვენებია სურათზე 12.11 და დავაწკაპუნებთ OK ღილაკზე.

**სურათი 12.11. მონაცემთა წყაროს არჩევა**

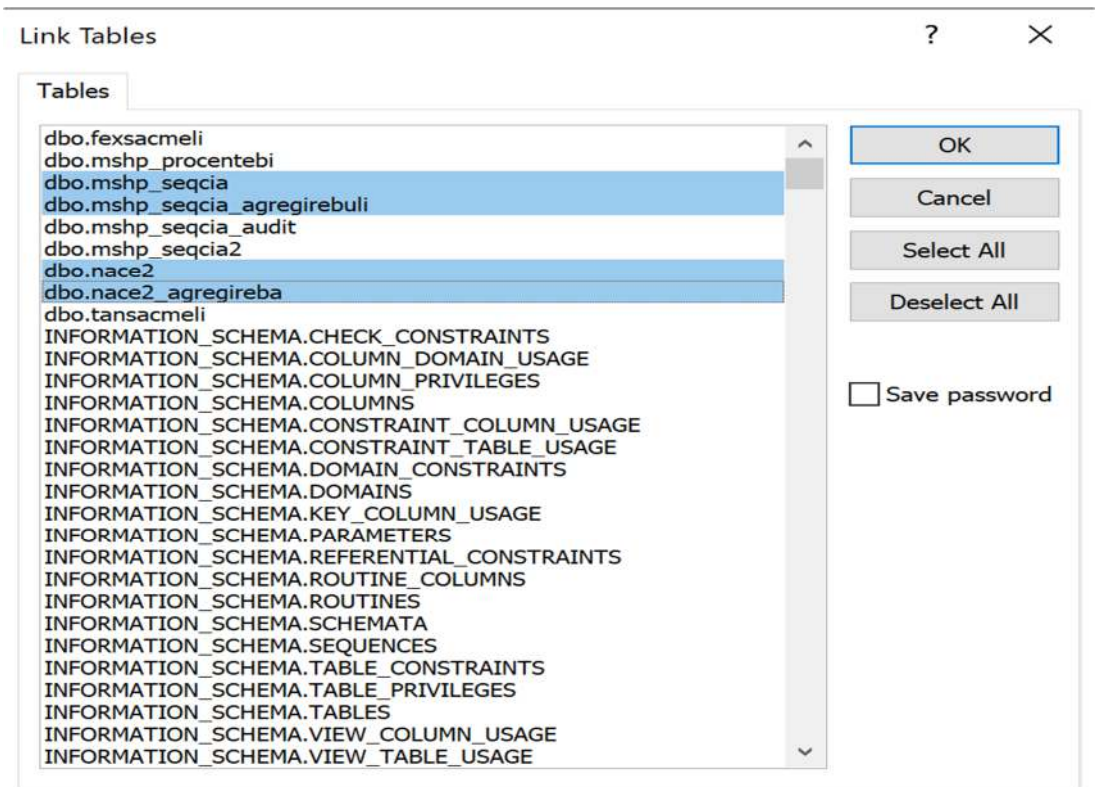



გამოვა ფანჯარა, რომელიც ჰგავს იმ სურათს, რომელიც გვიჩვენებს დასაკავშირებელ ცხრილებს, რომელიც ადრე შევქმენით SQL Server-ზე. აქვე ჩანს ჩვენ მიერ შექმნილი წარმოდგენებიც. შეგვიძლია აქ მოვნიშნოთ რამდენიმე ცხრილი. ავიღოთ შემდეგი ცხრილები:

1. mshp_seqcia;
2. mshp_seqcia_agregirebuli;
3. nace2;
4. nace2_agregireba.

ამის შემდეგ დავაწკაპუნოთ OK ღილაკზე.

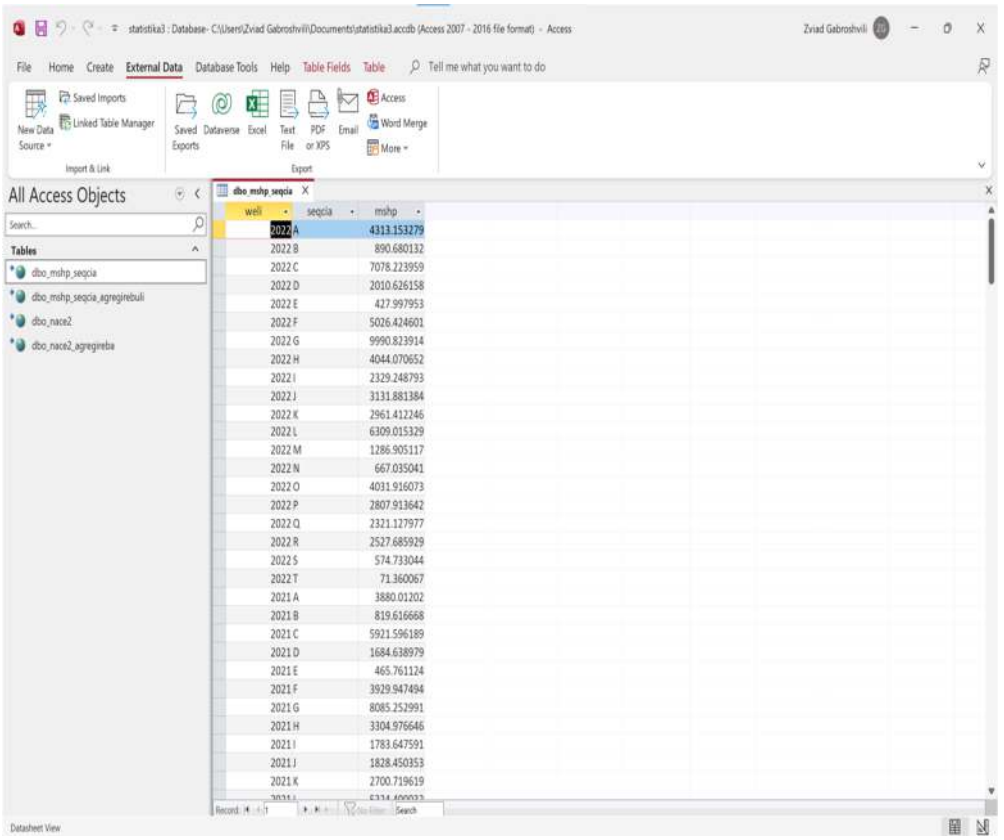
სურათი 12.12. ცხრილების დაკავშირება




ზემოხსენებული ოპერაციების განხორციელების შემდეგ, აღნიშნული ცხრილები დაუკავშირდება ჩვენს Access ბაზას და, შესაბამისად, გამოჩნდება All Access Objects პანელში. თუ გავხსნით ერთ-ერთ ცხრილს, ვნახავთ შესაბამის მონაცემებს. ეს ყველაფერი ნაჩვენებია სურათზე 12.13.

ზოგადად, რადგან დაკავშირებულია ცხრილები და გვაქვს შესაბამისი პრივილეგიები სერვერზე, უკვე შეგვიძლია მონაცემების არა მარტო წაკითხვა, არამედ ცვლილებების მოხდენაც, რომელსაც შემდეგ ქვეთავში გავეცნობით.

სურათი 12.13. დაკავშირებული ცხრილიდან მონაცემების წაკითხვა



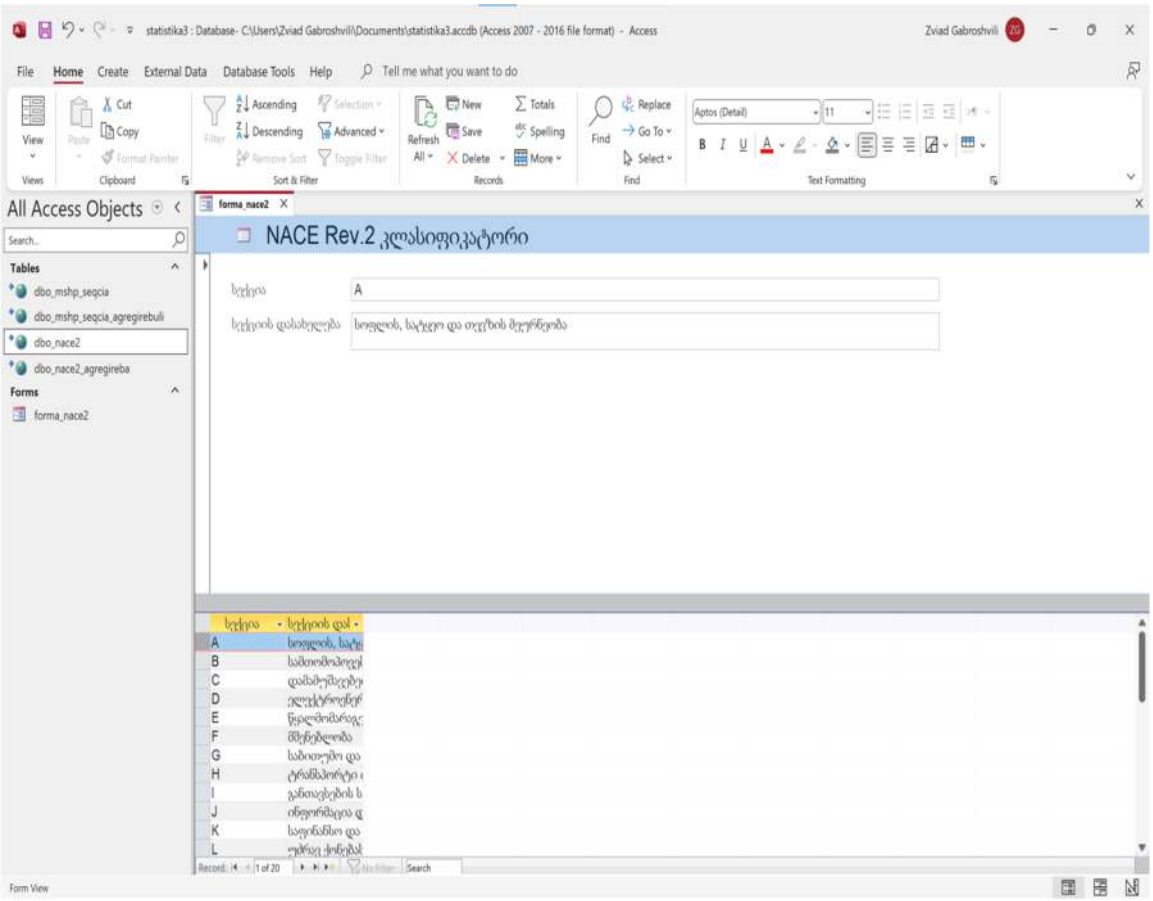

## 12.2. CRUD ოპერაციები

როგორც აღვნიშნეთ, შეგვიძლია განვახორციელოთ CRUD ოპერაციები SQL Server-ზე Microsoft Access-დან. ამ ქვეთავში განვიხილავთ მონაცემების შემყვანი ფორმების შექმნას, როგორც ეს განვახორციელეთ ამ წიგნის მე-10 თავში.

დავუშვათ, გვსურს შევქმნათ მონაცემთა შემყვანი ფორმა nace2 ცხრილისთვის. ამისათვის მოვნიშნავთ ცხრილს და Create ჩანართიდან შევქმნით Split ტიპის ფორმას, შევცვლით ტექსტს დაახლოებით ისე, როგორც გამოსახულია სურათზე 12.14 და შევინახავთ, როგორც forma_nace2.

აქედან გამომდინარე, შეგვიძლია მონაცემთა ბაზა ავაწყოთ SQL Server-ზე, რომელიც განთავსებული იქნება კონკრეტულ სერვერზე, დავუკავშიროთ მას მომხმარებლები Access-ის საშუალებით, რომელსაც ექნება სხვადასხვა ტიპის ფორმა მონაცემთა შეყვანისთვის, წაშლისთვის და რედაქტირებისთვის და შემდეგ განვახორციელოთ შესაბამისი ოპერაციები. აღნიშნული ხშირად გამოიყენება პრაქტიკაში სხვადასხვა ორგანიზაციის მიერ. შედეგად Access-ში შექმნილი ფაილი იქნება ერთგვარი ინტერფეისი მონაცემთა ბაზის სერვერსა და მომხმარებელს შორის.

სურათი 12.14. მონაცემთა შემყვანი ფორმა სერვერზე არსებული nace2 ცხრილისთვის

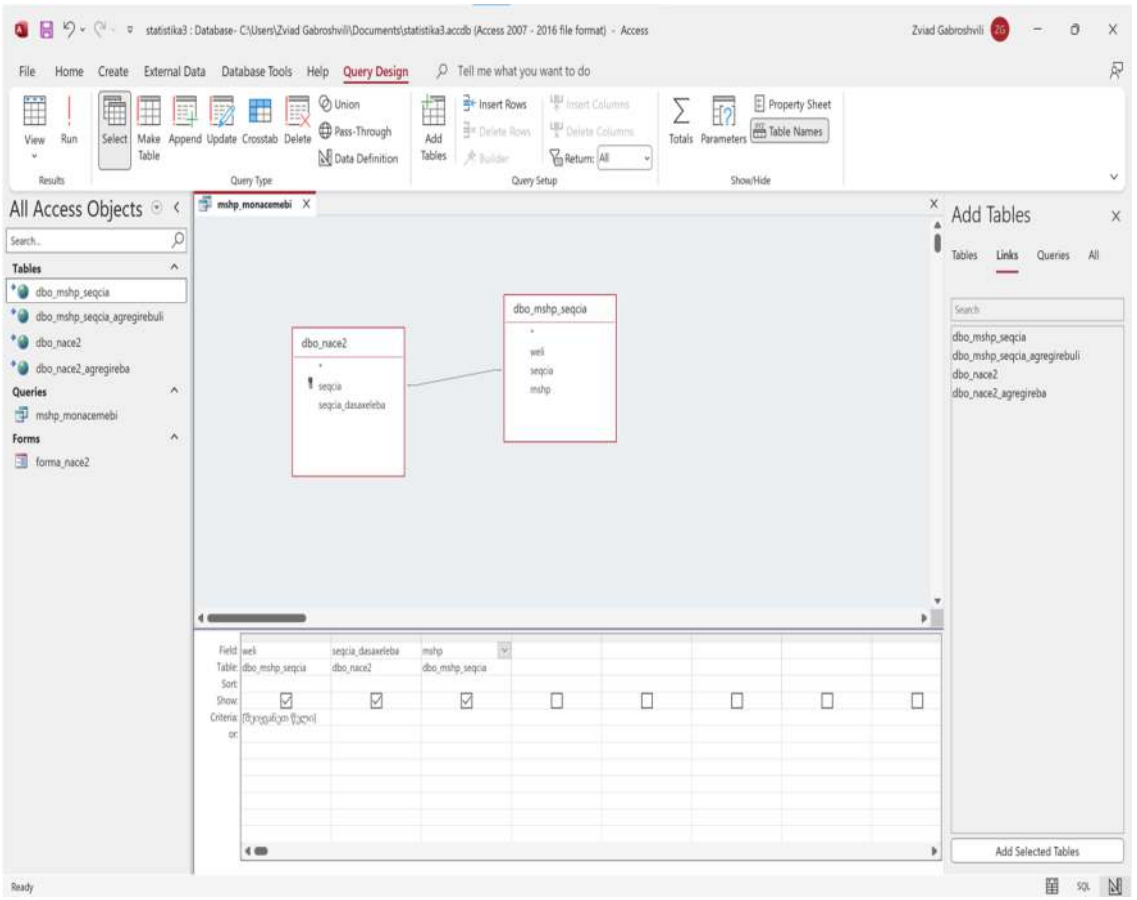



გარდა ფორმებისა, ჩვენ შეგვიძლია Access-დან შევქმნათ მოთხოვნები SQL Server-ზე განთავსებული მონაცემთა ბაზისთვის. დავუშვათ, გვსურს შევქმნათ მოთხოვნა, რომელიც მომხმარებელს შეეკითხება, თუ რომელი წლის მონაცემების ამოღება სურს mshp_seqcia ცხრილიდან და ამის შემდეგ ამოიღებს შესაბამის მონაცემებს. ამასთან ერთად, ეს ცხრილი უნდა დაუკავშირდეს nace2 ცხრილს და გვიჩვენოს შემდეგი სამი სვეტი შედეგში:

1. წელი;
2. სექციის დასახელება;
3. მშპ.

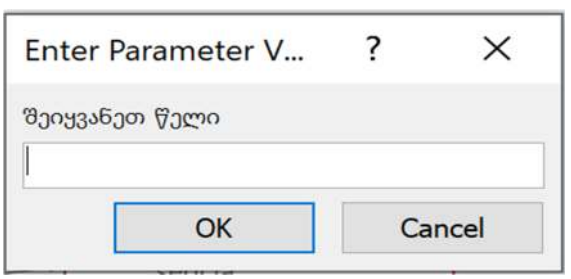
ამისათვის შევქმნათ მოთხოვნა, მას პირობითად დავარქვათ mshp_monacemebi და ავაგოთ ისე, როგორც ეს გამოსახულია სურათზე 12.15.

**სურათი 12.15. mshp_monacemebi მოთხოვნა**



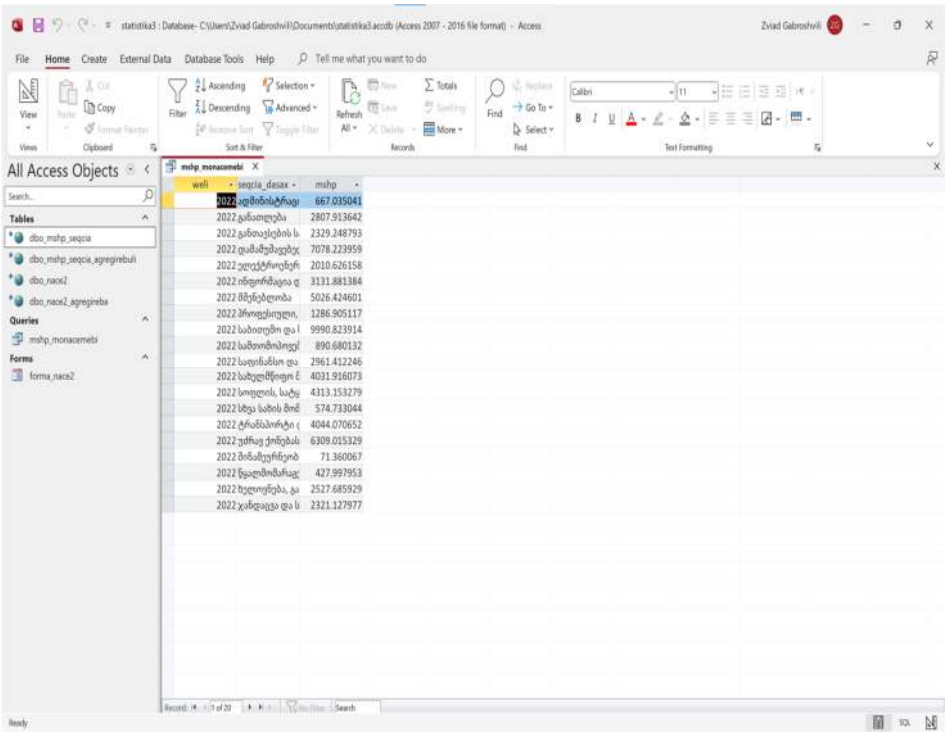

თუ გავუშვებთ ამ მოთხოვნას, მაშინ პროგრამა შეგვეკითხება, თუ რომელი წლის მონაცემები გვინდა, როგორც ეს ნაჩვენებია სურათზე 12.16, შევიყვანოთ 2022 და დავაწკაპუნოთ OK ღილაკზე.

**სურათი 12.16. პარამეტრული მოთხოვნა მონაცემთა ბაზისთვის**




შედეგად, მივიღებთ მონაცემებს, რომელიც გამოსახულია მომდევნო სურათზე 12.17.

სურათი 12.17. შედეგები სერვერიდან პარამეტრული მოთხოვნის საფუძველზე




ანალოგიურად შესაძლებელია სხვა ტიპის მოთხოვნების შექმნა და ამ მოთხოვნებით CRUD ოპერაციების განხორციელება SQL Server-ზე, Microsoft Office Access-ის გამოყენებით. შეგვიძლია შევინახოთ Access-ში შექმნილი ფაილი და მოგვიანებით გამოვიყენოთ მონაცემთა ბაზის მართვისათვის.

***კითხვები თვითშემოწმებისთვის:***

1. როგორ ხდება SQL Server-თან დაკავშირება Access-ის საშუალებით?
2. როგორ ხდება მონაცემთა ბაზაში ინფორმაციის შემყვანი ფორმების შექმნა?
3. როგორ გამოიყენება პარამეტრული მოთხოვნები Access-ში მონაცემთა ბაზებთან სამუშაოდ?

# თავი 13. მონაცემები თანამედროვე მსოფლიოში

## 13.1. მონაცემები გადაწყვეტილებების მიღებაში

თანამედროვე მსოფლიოში მონაცემების შეგროვებასა და მათ გამოყენებას მნიშვნელოვანი ადგილი უკავია. აქამდე თუ მონაცემები უფრო მეტად გამოიყენებოდა საბუნებისმეტყველო და, ზოგადად, სამედიცინო მეცნიერებების მიმართულებით, ციფრული ტექნოლოგიების მნიშვნელოვან განვითარებასთან ერთად გაიზარდა მოთხოვნა მონაცემებზე განსაკუთრებით ისეთ სფეროებში, როგორიცაა: ეკონომიკა, ბიზნესი, სამართალი, ეკოლოგია და სხვ.

განვიხილოთ მონაცემების გამოყენების მნიშვნელობა ეკონომიკაში. ზოგადად, თუ დავყოფთ ეკონომიკას მიკროეკონომიკად და მაკროეკონომიკად, ორივე ნაწილში მოიძებნება ისეთი ხშირი საჭიროებები, რომლებიც მოითხოვენ მონაცემების გამოყენებას. მაკროეკონომიკაში მონეტარული, ფისკალური თუ სტრუქტურული პოლიტიკის გატარება მოითხოვს ადეკვატური მოდელების შემუშავებას. ამასთან ერთად, მოდელებში შესაბამისი პარამეტრების გასაზომად საჭიროა მონაცემთა ასევე ადეკვატური რაოდენობა, თანაც თანამედროვე მსოფლიოში სწრაფი ცვლილებები მნიშვნელოვნად ზრდის რისკებს ეკონომიკურ საქმიანობებში. თუ მონაცემების არასაკმარისი რაოდენობის გამო, ან რიგი ფაქტორების არასაკმარისად გათვალისწინების გამო მოდელი აღმოჩნდა სუსტი პროგნოზირების მხრივ, მაშინ ამ მოდელზე დაფუძნებული გასატარებელი პოლიტიკა შესაძლებელია აღმოჩნდეს სახიფათო ეკონომიკური მდგრადობის კუთხით. ამის გააზრება მარტივად შეიძლება კონკრეტულ მაგალითებზე დაყრდნობით: მაგალითად, რა მოხდება, თუ მონეტარული პოლიტიკა გატარდება არასწორად, რადგან მოდელი არ იყო ადეკვატური? რა მოხდება, თუ გადასახადები შეიცვლება არასაკმარისად კარგი მოდელის საფუძველზე? რა მოხდება, თუ სუბსიდიები განხორციელდება ისეთ დარგებში, რომლებიდანაც უკუგება იქნება ბევრად ნაკლები, ვიდრე სხვა დარგებიდან და ეს გადაწყვეტილება განაპირობა არაობიექტურმა და არასრულმა მონაცემებმა? ამ მაგალითებზე დაფიქრებითა და პასუხის გაცემით მივხვდებით, თუ რამდენად დიდი პრობლემები შეიძლება მოიტანოს არასათანადო მონაცემებზე დაყრდნობით მიღებულმა გადაწყვეტილებებმა.

გავიაზროთ მონაცემებზე დაყრდნობით მიღებული ზოგადი გადაწყვეტილებები.

ამჯერად დავუშვათ, რომ მონაცემები შესაბამისი და, ამავდროულად, მაღალი ხარისხისაა. ამ შემთხვევაში მოდელების წარმადობა უფრო მაღალი იქნება და მიღებული გადაწყვეტილებებიც უფრო ადეკვატური. აქედან გამომდინარე, სწორად გატარდება შესაბამისი მონეტარული, ფისკალური თუ სტრუქტურული პოლიტიკა.

რაც შეეხება კერძო სექტორს, ბიზნესში მიღებული გადაწყვეტილებები სულ უფრო მეტად ხდება დაფუძნებული მონაცემებზე. შესაბამისად, იზრდება მოთხოვნა ისეთ კადრებზე, როგორიცაა: მონაცემთა ანალიტიკოსები, სტატისტიკოსები, ეკონომეტრიკოსები, მონაცემთა ინჟინრები, ზოგადად მონაცემთა მეცნიერები და სხვა.

დღეს ინდუსტრიაში მონაცემთა მეცნიერები ფლობენ სტატისტიკურ უნარებს, პროგრამირებას და იმ კონკრეტულ სფეროში გააჩნიათ ცოდნა, რომლის მიმართულებითაც ახორციელებენ საქმიანობას, როგორც მონაცემთა მეცნიერები. მაგალითისთვის: მონაცემთა მეცნიერი შეიძლება მუშაობდეს კონკრეტულ სფეროში მარკეტინგის მიმართულებით. რა სახის დავალებები შეიძლება შეასრულოს მან? ასეთი დავალებები შეიძლება იყოს არსებული მონაცემების საფუძველზე მომხმარებლების კლასტერიზაცია, სხვადასხვა ფაქტორის გავლენის შეფასება და შესაბამისი ფასდადება რეგრესიული ანალიზის გზით, მომხმარებლების კლასიფიკაცია, სხვადასხვა სამიზნე აუდიტორიის ანალიზი და სხვა მრავალი.

როგორც ვნახეთ, მონაცემების გამოყენება საკმაოდ მნიშვნელოვანი შეიძლება აღმოჩნდეს მთელი რიგი საქმიანობების განხორციელებაში. შესაბამისად, მნიშვნელოვანია მონაცემთა მოძიება და ინჟინერიაც. მონაცემთა მეცნიერებს ხშირად ეხმარება შესაბამისი მონაცემების საჭირო სახით მიწოდება, მაგალითად, ორი მონაცემის საფუძველზე ერთი მთავარი მუშაობის ინდიკატორის მიღება, მათთვის მიწოდება და სხვა.

ზემოხსენებულიდან გამომდინარე, საკმაოდ მნიშვნელოვანია მონაცემთა უსაფრთხოებაც. აქედან გამომდინარე, დღეს აქტიურად ინერგება ისეთი გადაწყვეტილებები, რომლებიც ხელს უწყობენ მონაცემების უსაფრთხო გადაცემას და სხვა. დღევანდელ მსოფლიოში მონაცემებისა და მოდელების მიმართულებით სულ უფრო და უფრო დიდ მნიშვნელობას იძენს კიბერუსაფრთხოება. კიბერუსაფრთხოების დაჯგუფება ხშირად ხდება ორ კლასად. ესენია: დაცვითი და შეტევითი კიბერუსაფრთხოება. დაცვითი კიბერუსაფრთხოების დროს ხდება დაცვითი მექანიზმების შემუშავება, სხვადასხვა სტანდარტების დანერგვა და სხვა, ხოლო შეტევითი კიბერუსაფრთხოების დროს ხდება უკვე არსებული ინფრასტრუქტურის შემოწმება, სიმულაციური შეტევის განხორციელება და ა. შ. იმ შემთხვევაში, თუ მოხდა მონაცემების მოწამვლა ან მოდელის გაფუჭება, მაშინ საფრთხე საკმაოდ მაღალია იმისა, რომ შესაბამის მოდელსა და მონაცემებზე დაყრდნობით მიღებული გადაწყვეტილებები იქნება არასასურველი. მონაცემთა გადაცემისას, მათი უსაფრთხოების უზრუნველყოფის მიზნით, გამოიყენება სხვადასხვა პროტოკოლი, რომელიც უზურუნველყოფს უსაფრთხოებას. ხშირია მონაცემთა მიმოცვლა დაცული არხებით. დაცვის ერთ-ერთ ინსტრუმენტს ამ შემთხვევაში წარმოადგენს კრიპტოგრაფია. კრიპტოგრაფიის გამოყენებისას მონაცემთა მიმოცვლის დროს ხდება გადასაცემი ინფორმაციის დაშიფვრა. გამოიყენება სიმეტრიული და ასიმეტრიული კრიპტოგრაფია. სიმეტრიულის დროს მონაცემთა გადამცემი და მიმღები წინასწარ არიან შეთანხმებულები ერთ გასაღებზე, რომელსაც შეუძლია გახსნას შიფრი და, შესაბამისად, მონაცემთა დაშიფვრისას და გაშიფვრისას გამოიყენება ეს ერთი გასაღები. დღეს საკმაოდ წარმატებულად გამოიყენება კრიპტოგრაფიის AES-256 ალგორითმი. რაც შეეხება ასიმეტრიულს, შესაძლებელია იყოს შემთხვევა, რომ მონაცემთა მიმღები და გადამცემი ერთმანეთთან არ იყვნენ წინასწარ შეთანხმებული გასაღებზე. ამ შემთხვევაში მონაცემთა მიმღებმა შესაძლებელია დააგენერიროს ორი გასაღები, საჯარო და პირადი. საჯარო გასაღებით მათემატიკურად შესაძლებელია მხოლოდ მონაცემთა დაშიფვრა და ეს საჯარო გასაღები შეიძლება იყოს გა-

მოქვეყნებული საზოგადოებისთვის მისაწვდომ ადგილზე, ხოლო უკვე დაშიფრული მონაცემების გასაშიფრად კი იყენებენ პირად გასაღებს, რომელსაც მონაცემთა მიმღები ინახავს სხვა პირებისთვის მიუწვდომელ ადგილას. ამგვარად ხდება ინფორმაციის შედარებით უსაფრთხო გადაცემის უზრუნველყოფა. დღეს წარმატებით გამოიყენება ასიმეტრიული კრიპტოგრაფიის RSA და ელიფსური მრუდების ალგორითმები. არის ვარიანტები, როდესაც იყენებენ ერთდროულად ორივე მიდგომას. ამ შემთხვევაში ხდება ასიმეტრიული დაშიფვრით უსაფრთხოება უზრუნველყოფილი არხით სიმეტრიულ ალგორითმზე გადასვლა, რამაც შეიძლება უზრუნველყოს უფრო მაღალი უსაფრთხოება და მონაცემთა გაცვლის სისწრაფე.

გარდა კრიპტოგრაფიისა, მონაცემთა უსაფრთხოებისთვის საჭიროა მონაცემთა შენახვა იმგვარად, რომ ტექნიკური თუ პროგრამული ხარვეზის შემთხვევაში მოხდეს მონაცემთა აღდგენა. ამისათვის გამოიყენება Backup სისტემები. ზოგჯერ ხდება მონაცემების განთავსება სხვა ადგილზეც. მაგალითისათვის, თუ მონაცემები დაზიანდა ერთ სერვერზე, შესაძლებელი უნდა იყოს სხვა სერვერიდან ან მონაცემთა შენახვის მოწყობილობიდან მისი აღდგენა. ამგვარად, შესაძლებელია მოხდეს მონაცემთა დანაკარგებთან და დაზიანებასთან გამკლავება სხვადასხვა შემთხვევაში, იქნება ეს კიბერთავდასხმა თუ სხვა ტიპის შემთხვევა.

## 13.2. სხვადასხვა პროგრამული ინსტრუმენტი მონაცემთა დამუშავებისა და ანალიზისათვის

ამ ქვეთავში განვიხილავთ სხვადასხვა ინსტრუმენტს, რომელიც გამოიყენება მონაცემთა ანალიზისათვის. ზოგადად, ამ წიგნის წაკითხვის შემდეგ შეიძლება მკითხველს გაუჩნდეს სურვილი უფრო მეტად გაიღრმაოს ცოდნა მონაცემთა ანალიზის მიმართულებით. შესაბამისად, მას შეიძლება დასჭირდეს ისეთი ინსტრუმენტები, რომლებიც დაეხმარება სხვადასხვა ტიპის სტატისტიკური ანალიზის განხორციელებაში. აქედან გამომდინარე, ამ ქვეთავის მეორე სათაური შეიძლება ყოფილიყო „რა ვისწავლოთ“.

ჩვენ უკვე განვიხილეთ Microsoft SQL Server და,ზოგადად, SQL პროგრამული ენა მონაცემთა ანალიზისათვის, ვნახეთ MS Access-ის გამოყენების მეთოდებიც. ხშირია მონაცემთა ანალიზისთვის Microsoft Excel-ის გამოყენებაც, მაგრამ ამჯერად განვიხილავთ სხვა პროგრამებს.

**Eviews** - ბაზარზე ერთ-ერთი პროგრამაა, რომელიც გამოიყენება სხვადასხვა ტიპის ეკონომეტრიკული ანალიზის განხორციელებისთვის. პროგრამას გააჩნია გრაფიკული ინტერფეისი და, ამავდროულად, პროგრამირების საშუალებაც. ძირითადად, იგი გამოიყენება სხვადასხვა ტიპის რეგრესიული და, ზოგადად, დროითი მწკრივების ანალიზისათვის. მას შეუძლია წაიკითხოს Excel-ის ფორმატში არსებული ფაილები, დაუკავშირდეს მონაცემთა სხვადასხვა ბაზას და განახორციელოს დროითი მწკრივების რთული ანალიზი. პროგრამაში ჩაშენებულია დროითი მწკრივების ანალიზის ისეთი მეთოდები, როგორიცაა მათი სტაციონალურობის, ჰეტეროსკედასტურობის, ავტოკორე-

ლაციის შემოწმება, ავტორეგრესიული, ავტორეგრესიული ინტეგრირებული მოძრავი საშუალოს, სტანდარტული ვექტორული ავტორეგრესიის, სტრუქტურული ვექტორული ავტორეგრესიის, ბაიესიანური ვექტორული ავტორეგრესიის, რეჟიმის ცვლილების მოდელების, პანელური მონაცემების მოდელების, ვექტორული ხარვეზის შესწორების მოდელების გამოყენება და სხვა. პოპულარულია მისი გამოყენება ნაუკასტინგის მიმართულებით. გარდა ამისა, ანალიზის საფუძველზე შეუძლია როგორც გრაფიკების, ასევე სხვადასხვა საშედეგო მონაცემის გამოტანაც ადვილად აღქმად ფორმატში. პროგრამაში გამარტივებულია ახალი მწკრივების შექმნა არსებული მწკრივებიდან, მათი წინასწარი დამუშავება, ფილტრაცია, სეზონური მოსწორება და, ასევე, სტატისტიკისთვის მნიშვნელოვანი ტესტების განხორციელება. საჭიროების შემთხვევაში პროგრამით შესაძლებელია იმპუტაციის განხორციელება სხვადასხვა მეთოდით. ამჟამად Eviews-ის სხვადასხვა ვერსიაა ჩაშვებული გაყიდვაში, რომლებიც განსხვავდება ფასის და, შესაბამისად, შესაძლებლობების მიხედვით.

**Power BI** - არის მონაცემთა ანალიტიკისთვის საჭირო პროდუქტი კორპორაცია Microsoft-ისგან. მისი საშუალებით შესაძლებელია სხვადასხვა ტიპის დიაგრამის აგება, მონაცემთა გავრცელება და სხვა. მისი საშუალებით შესაძლებელია დამყარდეს კავშირი მონაცემთა ბაზებთან, მათ შორის Microsoft SQL Server 2022-თან. თუმცა, მონაცემთა შემოტანა პროგრამაში სხვადასხვა მეთოდითაა შესაძლებელი. ხშირად იყენებენ „კომპლექსური დაფების“ ასაგებად, რომლებიც მარტივად აღსაქმელია და მონაცემებით მართული გადაწყვეტილებების მისაღებად საჭირო ინსტრუმენტებია. Power BI-ის შეუძლია მოწინავე მონაცემთა ანალიტიკის ინსტრუმენტების გამოყენება ვიზუალიზაციისთვის და მონაცემთა სხვადასხვა ტიპის დამუშავება. მას გააჩნია ხელოვნური ინტელექტის შესაძლებლობები და მარტივი ინტერფეისი ვიზუალიზაციისათვის, რადგან საკმაოდ კარგად არის თავსებადი Microsoft 365 პროდუქტებთან, შესაბამისად გამოიყენება სხვა აპლიკაციებთან ერთად. მონაცემებისა და რეპორტების გატანის საშუალება ისეთ Microsoft 365 პროდუქტებში, როგორიციაა Excel, PowerPoint, Teams და Sharepoint, ქმნის შესაძლებლობას უფრო მეტად გაიზარდოს ნებისმიერ ორგანიზაციაში თანამშრომლობა და მონაცემებით მართული გადაწყვეტილებები. რაც შეეხება გამოყენების ქეისებს, პოპულარულია მისი გამოყენება რეპორტების ასაგებად, სხვადასხვა წყაროებიდან მონაცემთა თავმოყრისთვის და ამ მონაცემებში კანონზომიერებების დასადგენად. კიდევ ერთი მნიშვნელოვანი შესაძლებლობა, რაც გააჩნია ამ აპლიკაციას, არის ის, რომ თავსებადია Azure ღრუბლოვან ტექნოლოგიებთან. შესაბამისად, მისი გამოყენება იძლევა საშუალებას დამუშავდეს პეტაბაიტობით ინფორმაცია და რეპორტები ჩაშენდეს Azure აპლიკაციებში. გარდა Power BI Desktop ვერსიისა, არსებობს მისი მობილური ვერსიებიც, რომლის საშუალებითაც შეიძლება დესკტოპ აპლიკაციით შემუშავებული რეპორტების ნახვა მობილურებში, Android და iOS ოპერაციულ სისტემებზე. პროგრამას გააჩნია DAX-ის მხარდაჭერაც. პროგრამის სხვადასხვა შესაძლებლობის ვერსია არსებობს და, შესაბამისად, განსხვავდება ფასიც, თუმცა, ამ დროისთვის მისი გადმოწერა უფასოდაც არის შესაძლებელი. უნდა აღინიშნოს, რომ პროგრამა გამოიყენება მსოფლიო მასშტაბით სხვადასხვა სახელ-

მწიფო და კერძო ორგანიზაციების მიერ. საქართველოში კი მასზე მოთხოვნა იზრდება და უკვე აქტიურად გამოიყენება არაერთი ორგანიზაციის მიერ.

**Matlab** - კიდევ ერთი გავრცელებული პროგრამა და, ზოგადად, პროგრამული ენაა, რომელიც გამოიყენება სხვადასხვა ტიპის ანალიზისთვის. მას გააჩნია მარტივი გრაფიკული ინტერფეისი და, ამავდროულად, შესაძლებელია მისი პროგრამირებაც. სტატისტიკური, ასევე მათემატიკური და სიმულაციური ანალიზური შესაძლებელობების გამო, გამოიყენება სხვადასხვა მეცნიერებაში. იგი ფასიანია და არსებობს მისი სხვადასხვა ვერსია, რომელთაც საკუთარი შესაძლებლობები და ფასი აქვთ. იგი მოიცავს სხვადასხვა ინსტრუმენტს, რომლის შეძენაც, ასევე, შესაძლებელია ცალკე. ზოგიერთი ინსტრუმენტი უფასოა და დაწერილია სხვადასხვა ორგანიზაციის მიერ, რომლებიც პირდაპირ არ უკავშირდებიან Mathworks. ეკონომიკაში ხშირად გამოიყენება Dynare და IRIS Toolbox სხვადასხვა ორგანიზაციის მიერ. მათი საშუალებით მნიშვნელოვნად მარტივდება მთელი რიგი ეკონომიკური მოდელების აგება, როგორებიცაა ზოგადი წონასწორობა, გადაფარვადი თაობები, სხვადასხვა სახის ეკონომეტრიკული ანალიზი და სხვა. ზოგადად, Matlab-ის შესაძლებლობებში შედის:

- მონაცემთა ანალიზი;
- ვიზუალიზაცია;
- პროგრამირება;
- დესკტოპ და ვებაპლიკაციების შექმნა;
- სხვა ენებთან დაკავშირება, როგორებიცაა Python, C++ და ა. შ.;
- ტექნიკასთან დაკავშირება;
- პარალელური კომპიუტინგი, რომელიც საშუალებას იძლევა მსხვილმასშტაბიანი გაანგარიშებებისთვის მრავალი ბირთვის, გრაფიკული დამუშავების ერთეულებით და კლასტერების გამოყენებით;
- ღრუბლოვანი ტექნოლოგიბის გამოყენება, როგორებიცაა Amazon Web Services (AWS) და Azure.

Matlab-ს ხშირად იყენებენ ისეთ სფეროებში, როგორიცაა:

- მართვის სისტემების დიზაინი, ტესტირება და დანერგვა;
- მანქანური დასწავლის მოდელების შემუშავება და დანერგვა; აქვე მოიაზრება ღრმა ნეირონული ქსელების გამოყენება;
- სიგნალების დამუშავება, დროითი მწკრივების ანალიზი, მოდელირება და სიგნალების დამუშავების სისტემების სიმულაცია;
- მანქანური ხედვის მიმართულება;
- რობოტიკა;
- უკაბელო კომუნიკაციები;
- აეროკოსმოსური, ავტომობილების და სამედიციონო მოწყობილოების დიზაინი;
- ფიზიკა, ქიმია, ქიმიური ინჟინერია, ბიოლოგია და სხვა.

რაც შეეხება დამატებით ფასიან ინსტრუმენტებს, პოპულარულია შემდეგი:

- პარალელური კომპიუტინგის;
- სტატისტიკისა და მანქანური დასწავლის;

- ოპტიმიზაციის;
- გლობალური ოპტიმიზაციის;
- ღრმა დასწავლის;
- კერძო დიფერენციალური განტოლებების;
- მონაცემთა ბაზების;
- რეპორტის გენერატორის;
- ეკონომეტრიკის და სხვა.

**IBM SPSS** - არის IBM-ის პროგრამა, რომელიც აქტიურად გამოიყენება მონაცემთა ანალიზისთვის, გააჩნია გრაფიკული ინტერფეისი და შეუძლია სხვადასხვა ტიპის სტატისტიკური ანალიზის შესრულება. ანალიზის საფუძველზე შესაძლებელია გრაფიკების აგება და მათი შესაბამის ფორმატში გატანაც. შეუძლია მონაცემების წაკითხვა სხვადასხვა ფორმატიდან და მონაცემთა ბაზებთან დაკავშირება, მათ შორის SQL Server 2022-თან, ფასიანია, წარმატებით გამოიყენება არა მარტო სხვადასხვა სტატისტიკური და კვლევითი ორგანიზაციის მიერ, არამედ კერძო ბიზნესშიც, რომელიც შეიძლება არ იყოს ორიენტირებული სტატისტიკური საქმიანობის მიმართულებით. მისი გამოყენება შესაძლებელია დიდ მონაცემებთან სამუშაოდ. მის ზოგიერთ ვერსიას გააჩნია მანქანური დასწავლის შესაძლებლობებიც. პროგრამას გააჩნია სხვადასხვა მოდული, რომელიც შეიძლება იყოს საბაზო ვერსიაში ან საჭიროებდეს ცალკე შესყიდვას, ესენია:

- მონაცემთა მომზადების;
- ბუთსტრეპინგის;
- საბაზო სტატისტიკური;
- რეგრესიის;
- მოწინავე სტატისტიკის;
- კერძო ცხრილების;
- პროგნოზირების;
- გადაწყვეტილების ხეების;
- პირდაპირი მარკეტინგის;
- ნეირონული ქსელების;
- კომპლექსური შერჩევის;
- კატეგორიების;
- გამოტოვებული მნიშვნელობების და სხვ.

**R** - ეს არის ერთ-ერთი თანამედროვე პროგრამული ენა, რომელიც აქტიურად გამოიყენება სხვადასხვა სახის სტატისტიკური ანალიზისთვის და ვიზუალიზაციისთვის. მისი გამოყენება მნიშვნელოვნად ამარტივებს მონაცემებთან მუშაობას. შესაძლებელია სიმულაციების აგება, მანქანური დასწავლის მოდელების სრულყოფა და გამოყენება. მას ხშირად იყენებენ დროითი მწკრივების და ფინანსური ანალიზისთვისაც. შესაბამისი პროგრამული ბიბლიოთეკების გამოყენებით მასაც შეუძლია მონაცემთა ბაზებთან დაკავშირება და შესაბამისი ოპერაციების განხორციელება. ხშირად იყენებენ R Studio აპლიკაციას, როგორც ინტეგრირებული დეველოპმენტის გარემოს პროგრამული ენა R-ისთის, რომელშიც შესაძლებელია როგორც კოდების წერა, ისე შესაბამი-

სი შედეგების მიღება. აქვე უნდა აღვნიშნოთ, რომ ეკონომიკური მოვლენებისა და პროცესების სტატისტიკური ანალიზისა და პროგნოზირების პროცესში ამჟამად უკვე ძლიერ გაიზარდა R-პროგრამის გამოყენება, რომელსაც მეტად ფართო შესაძლებლობები გააჩნია. R-პროგრამის შემქმნელებად მოიაზრებიან ოკლენდის უნივერსიტეტის სტატისტიკის დეპარტამენტის მეცნიერები **რობერტ ჯენთელმენი** და **როისი კაჰა.** მე-20 საუკუნის ბოლოდან, კონკრეტულად კი 1998 წლიდან დღემდე, R-პროგრამის სრულყოფაში თავისი წვლილი შეიტანა მრავალმა სხვა მეცნიერმა, როგორიცაა: **ჯონ ჩემბერსი, კურტ ჰორნიკი, მიშელ ლეურენცი, მარტინ მორგანი, პაულ მურელი, მარტინ პლუმერი, ბრაიან რიპლი, საიმონ ურბანეკი, სეთ ფალკონი, რიკ ბეიკერი, ალან უილკსი** და კიდევ ბევრი სხვა [5, გვ. 94; აგრეთვე 9]. R-პროგრამის საწყისი ვერსია აღნიშნულმა პირებმა განავითარეს სხვადასხვა მიმართულებით (კოდებით, დოკუმენტირებითა და ცალკეული პრობლემების აღმოფხვრით), რის შედეგადაც იგი თანამედროვე სახით არსებობს და, რა თქმა უნდა, მომავალში კიდევ განხორციელდება მისი სრულყოფა.

თანამედროვე სტატისტიკური ანალიზისა და პროგნოზირების პროცესში ამჟამად უკვე მრავალ ქვეყანაში ფართოდ გამოიყენება R-პროგრამა მისი ფართო შესაძლებლობების გამო. იგი რეალურად არის პროგრამების კომპლექსი, რომელსაც შეუძლია მანიპულირება მონაცემთა დიდ მასივებზე. ზოგჯერ ამ პროგრამას მოიხსენიებენ **R&R**–ის სახელწოდებითაც [9]. ამ პროგრამას, მის შექმნამდე არსებულ კომპიუტერულ სტატისტიკურ პროგრამებთან შედარებით, როგორიცაა, მაგალითად, SPSS, STATA და სხვა, დამატებული აქვს სხვადასხვა მნიშვნელოვანი ფუნქცია, რაც აფართოებს მისი პრაქტიკული გამოყენების არეალს და მოიცავს სხვადასხვა სფეროს პროცესების ანალიზსა და პროგნოზირებას [გელაშვილი, 2015, გვ. 47].

R - პროგრამის გამოყენება, პირველ რიგში, გულისხმობს შემდეგს:

❖ დიდი მოცულობის სტატისტიკური მონაცემების არსებობას;

❖ ფართო სპექტრის გაანგარიშებების განხორციელების შესაძლებლობებს;

❖ მრავალგვარი გრაფიკების აგებას და გამოსახვას სხვადასხვა ფორმით;

❖ მონაცემთა დაჯგუფებისა და სტრუქტურიზაციის, შიგაჯგუფური და ჯგუფთაშორისი სტატისტიკური მაჩვენებლების გაადვილებული გამოთვლების

შესაძლებლობას;

❖ განვითარებულ და ეფექტიან პროგრამულ ენას, რაც აადვილებს მის გამოყენებას;

❖ გაანგარიშებული მაჩვენებლების ვარგისიანობის შემოწმების შესაძლებლობას [7, გვ. 129]:.

აღნიშნულის გარდა, სხვადასხვა მოვლენისა და პროცესის სტატისტიკური ანალიზისა და პროგნოზირების დროს R-პროგრამის გამოყენება შესაძლებელია, ასევე, წრფივი და არაწრფივი მოდელების ასაგებად და სათანადო ტესტების (კრიტერიუმების) გასაანგარიშებლად, აგებული მოდელების შესაფასებლად და სხვ. იგი შეიცავს ყველა ცნობილ მათემატიკურ სიმბოლოს და ადვილად აიგება ყველა საჭირო მათემატიკურდსტატისტიკური ფორმულა. მაშასადამე, R-პროგრამის გამოყენებით შესაძ-

ლებელია სხვადასხვა მოვლენის სტატისტიკური პროგნოზების გაანგარიშება ერთდროულად რამდენიმე მოდელის საფუძველზე და მიღებული პროგნოზული სიდიდეების ვერიფიკაცია [გელაშვილი, 2015, გვ. 48].

ამ პროგრამის მნიშვნელოვანი დადებითი თვისებაა ისიც, რომ მას ადვილად შეუძლია მისი გამოყენების დროს დაშვებული შეცდომების აღმოჩენა და მონიტორზე გამოტანა, რათა მაშინვე შეამჩნიოს ოპერატორმა და გაასწოროს ისინი. მაშასადამე, მასში კარგად არის შემუშავებული კონტროლის მექანიზმიც [გელაშვილი, 2015, გვ. 48].

Python - ზოგადი გამოყენების გავრცელებული პროგრამული ენაა. იგი არის მრავალპარადიგმული და აქვს ობიექტზე ორიენტირებული პროგრამირების მხარდაჭერა. იგი უფასოა და ხშირად გამოიყენება სხვადასხვა დანიშნულებით. ამჟამად მთელ მსოფლიოში უკვე საკმაოდ პოპულარულია. საქართველოშიც და საზღვარგარეთაც მასზე საკმაოდ მაღალი მოთხოვნაა და მას იყენებს არაერთი საჯარო თუ კერძო ორგანიზაცია. ეს ორგანიზაციები მისთვის ქმნიან სხვადასხვა პროგრამულ ბიბლიოთეკასა და ფრეიმვერკს, რომლებიც აადვილებენ გარკვეული ტიპის საქმიანობების შესრულებას. მაგალითად, Flask და Django გამოიყენება ვებდეველოპმენტის მიმართულებით. Pandas ბიბლიოთეკას მონაცემთა მეცნიერები ხშირად იყენებენ მრავალი სახის მონაცემებთან სამუშაოდ; მას შეუძლია მონაცემების წაკითხვა და ჩაწერა სხვადასხვა ფორმატში და ასევე SQL Server მონაცემთა ბაზასთან მუშაობა თუნდაც ODBC (Open Database Connectivity) სტანდარტის გამოყენებით. აქ ჩანს მისი უპირატესობა R-თან შედარებით იმ მიმართულებით, რომ Python-ის გამოყენება, გარდა სტატისტიკური მიმართულებისა, შესაძლებელია ვებ დეველოპმენტის მიმართულებითაც. რიცხვებთან სამუშაოდ საკმაოდ პოპულარულია numpy ბიბლიოთეკა. Scipy გამოიყენება სამეცნიერო ანალიზისთვის; მისი საშუალებით შესაძლებელია სხვადასხვა ტიპის ანალიზის ჩატარება, ფუნქციების ოპტიმიზაცია და სხვ. ასევე, წარმატებულად გამოიყენება sklearn ბიბლიოთეკა მანქანური დასწავლის მიმართულებით და Google-ის მიერ გამოშვებული Tensorflow, რომელიც საშუალებას იძლევა შეუსაბამდეს მანქანური დასწავლის ალგორითმები როგორც პროცესორისა და გრაფიკული დამუშავების ბარათის გამოყენებით, ისე ტენსორის დამუშავების ერთეულის გამოყენებით, რასაც შეუძლია მნიშვნელოვნად შეამციროს მონაცემთა დამუშავებისა და ანალიზისათვის საჭირო დრო.

Python გამოიყენება მანქანური ხედვის ალგორითმებთან სამუშაოდაც, რისი კარგი მაგალითიცაა OpenCV ფრეიმვერკი და, ასევე, ჩვენ მიერ უკვე ნახსენები Tensorflow-ც. Python-ის გამოყენებით შესაძლებელია ვებსკრეპინგი, რისი დახმარებითაც შესაძლებელია დაწერილი პროგრამა დაუკავშირდეს სხვადასხვა ვებსაიტს და იქიდან წამოიღოს ინფორმაცია. ბოლო პერიოდში მნიშვნელოვნად გაიზარდა ამგვარად მიღებული მონაცემების მნიშვნელობა. ამ გზით შესაძლებელია მონაცემების მოძიებისთვის მნიშვნელოვანი დროის შემცირება. ხშირად იყენებენ Python-ს სხვადასხვა აპლიკაციის პროგრამულ ინტერფეისთან (API) დასაკავშირებლად და მონაცემების წამოსაღებად. აქედან ზოგიერთი API შეიძლება იყოს ფასიანი, ხოლო ზოგიც უფასო, თუმცა არსებობს ჰიბრიდული ვერსიებიც, რომელთა კომერციული და სხვა ტიპის მიზნებისთვის გამოყენებაც შეზღუდულია.

Python-ით სხვადასხვა წყაროდან ავტომატურად მოძიებული მონაცემების ატვირთვა შესაძლებელია SQL Server 2022-ის მონაცემთა ბაზაში. მოგვიანებით ისევ Python-ით შეიძლება დამუშავდეს ეს მონაცემები, მაგალითად, მოხდეს ინფორმაციის წინასწარი დამუშავება და სხვადასხვა მონაცემთა ინჟინერიის მეთოდების გამოყენება; არსებული მონაცემებიდან აიგოს სხვადასხვა ტიპის მოდელი და მათზე დაყრდნობით მოხდეს ოპტიმალური გადაწყვეტილების მიღება.

სხვადასხვა ბიბლიოთეკის საშუალებით ხშირია Python-ის გამოყენება მრავალი პროცესის დროით მწკრივებთან სამუშაოდ, იქნება ეს მოდელების ასაგებად და სხვადასხვა კრიტერიუმის გასაანგარიშებლად, თუ ვიზუალიზაციისთვის, მარტივი დიაგრამები თუ გეოგრაფიული რუკები. Python მხარდაჭერილია ისეთი ინტეგრირებული დეველოპმენტის გარემოში და ტექსტურ რედაქტორებში, როგორიცაა:

- Pycharm;
- Visual Studio Code;
- JupyterNotebook;
- Spyder და სხვ.

რაც შეეხება დესკტოპ აპლიკაციებს, არსებობს სხვადასხვა პროგრამული ბიბლიოთეკა, რომელიც ამ მხრივაც აძლევს მოცემულ პროგრამულ ენას მუშაობის შესაძლებლობას. ასეთები შეიძლება იყოს:

- Tk;
- Kivy;
- PyQTდა სხვ.

Python შექმნა **გაიდო ვან როსუმმა** 1980-იანი წლების ბოლოს, თუმცა, რამდენჯერმე იცვალა სახე და მის განვითარებაში ჩართულია სხვადასხვა ორგანიზაცია. მისი სინტაქსის სიმარტივის, საზოგადოების მხარდაჭერის და სხვა რიგი უპირატესობის გამო, ამჟამადაც მიმდინარეობს მისი გაუმჯობესება და სხვადასხვა პროგრამული ბიბლიოთეკის შემუშავება.

**C++** - საკმაოდ პოპულარული ზოგადი გამოყენების პროგრამული ენაა დეველოპერებში, რომელიც დაფუძნებულია პროგრამულ ენა C-ზე. **C++** პროგრამულ ენაზე მუშაობა დაიწყო **ბიარნ სტრუსტუპმა.** ჯერ კიდევ 1979 წელს ის მუშაობდა პროგრამულ ენა C-ში კლასების ჩამატებაზე, რაც იყო C++-ის წინა ვერსია. 1985 წელს გამოვიდა C++-ის პირველი ვერსია. მოგვიანებით დაიხვეწა და დღეს უკვე მისი სხვადასხვა ვერსია არსებობს.

C++ გამოიყენება სისტემური პროგრამირებისთვის და სხვადასხვა სახის აპლიკაციების შესაქმნელად; მას აქვს ობიექტზე ორიენტირებული პროგრამირების მხარდაჭერა და, ამავდროულად, კომპილირებადია. შესაბამისად, ისეთი ამოცანების შესასრულებლად, რომლებიც მოითხოვენ სისწრაფეს, კარგი არჩევანია. მისთვის შემუშავებულია სხვადასხვა პროგრამული ბიბლიოთეკა. აქედან ზოგი ფასიანია, ხოლო ზოგის გამოყენება შესაძლებელია უფასოდ. თავად უმეტესობა C++ კომპილერების უფასოა. ხშირად გამოიყენება სამეცნიერო კომპიუტინგის მიმართულებით. ამიტომაც მისი გამოყენება მონაცემთა ანალიზის მიმართულებით ხშირია, განსაკუთრებით მაშინ, როდესაც სისწრაფე მნიშვნელოვანი ხდება. მისი დახმარებით შესაძლებელია პარალე-

ლური კომპიუტინგის გამოყენებაც. რაც შეეხება მონაცემთა შემოტანას SQL Server 2022-დან, C++-ს ODBC-ის გამოყენებით შეუძლია ბაზასთან დაკავშირება და შესაბამისი მოთხოვნების გაშვება სათანადო პროგრამული ბიბლიოთეკების დახმარებით.

C++-ის სხვადასხვა გავრცელებული ბიბლიოთეკა და ფრეიმვერკი არსებობს. მაგალითად, შესაძლებელია მასში დესკტოპ აპლიკაციების აწყობა QT ფრეიმვერკის დახმარებით, თუმცა, იგი თავსებადია Net Framework-თან.

C++-ის დახმარებით შეიქმნა მრავალი პროგრამა, რაც ამჟამად გამოიყენება მსოფლიო მასშტაბით სხვადასხვა ორგანიზაციის მიერ. აქედან მხოლოდ რამდენიმე პროგრამის ჩამოთვლა ადასტურებს, თუ რამდენად გავრცელებულია მისი გამოყენება პროფესიული მიმართულებით:

- Microsoft Windows;
- Microsoft Office;
- Microsoft SQL Server;
- Bloomberg Terminal;
- Google Chrome;
- Mozilla Firefox;
- Adobe Photoshop;
- MySQL და სხვა.

გარდა ზემოხსენებულისა, თვით ისეთი ბიბლიოთეკები და ფრეივერკები, რომლებიც გამოიყენება სხვა პროგრამულ ენებში მანქანური დასწავლის მიმართულებით, შექმნილია C++-ში. ასეთია:

- Tensorflow;
- XGBoost;
- CatBoost;
- LightGBM;
- OpenCV;
- Scikit-Learn;
- PyTorchდა სხვ.

რაც შეეხება ინტეგრირებული დეველოპმენტის გარემოების და ტექსტური რედაქტორების მხარდაჭერას, ის ხორციელდება ისეთ პოპულარულ პროგრამებში, როგორიცაა:

- Microsoft Visual Studio;
- Microsoft Visual Studio Code;
- CLion;
- Xcode;
- Code::Blocks;
- Dev-C++;
- Qt Creator და სხვ.

**Java** - მოცემული პროგრამული ენა იყენებს ობიექტზე ორიენტირებულ პროგრამირებას და საკმაოდ პოპულარულია დეველოპერებში სხვადასხვა მიმართულებით.

მას გააჩნია საკუთარი ვირტუალური მანქანა, რომელიც Java Virtual Machine-ის (JVM) სახელით არის ცნობილი. მისი საშუალებით შესაძლებელია როგორც დესკტოპ და ვებ აპლიკაციების შექმნა, ისე მობილური აპლიკაციების შემუშავებაც. Android ოპერაციული სისტემისთვის აპლიკაციების დაწერა შესაძლებელია Java პროგრამული ენის გამოყენებით. თუმცა, ბოლო პერიოდში გამოიყენება სხვა პროგრამული ენებიც. არსებობს ამ პროგრამული ენისთვის განკუთვნილი მრავალი პროგრამული ბიბლიოთეკა და ფრეიმვერკი. მონაცემთა მეცნიერებაში მისი გამოყენება უცხო არ არის. მისი სისწრაფე Python-თან შედარებით ხშირად ხდება მიზეზი, თუ რატომ შეიძლება არჩევანი შეჩერდეს Java-ს გამოყენებაზე. არსებობს Java-ს დეველოპმენტის კიტების როგორც კომერციული, ისე უფასო ვერსიები და, შესაბამისად, განსხვავებულია მათი შესაძლებლობები. მას შეუძლია დაუკავშირდეს სხვადასხვა ტიპის მონაცემთა ბაზას და განახორციელოს მთელი რიგი ოპერაციები. რაც შეეხება SQL Server 2022-თან დაკავშირებას, Java-ს ამის განხორციელება შეუძლია jDBC დრაივერის საშუალებით. როგორც ზემოთ აღვნიშნეთ, Python-ის შემთხვევაში Java-ს გამოყენებითაც სავსებით შესაძლებელი და გამარტივებულია ვებსკრეპინგი და სხვადასხვა API-ებთან დაკავშირება და, შესაბამისად, მონაცემების წამოღება და SQL Server-ის მონაცემთა ბაზაში ჩაწერა. ამგვარად, მონაცემებთან სამუშაოდ, თუ საქმე ეხება ავტომატიზებას, საკმაოდ მნიშვნელოვანი დახმარების გაწევა არის შესაძლებელი Java-ს გამოყენებით. რაც შეეხება ისევ მონაცემების შეგროვებასთან Java-ს კავშირს, როგორც ზემოთ აღვნიშნეთ, Java გამოიყენება მობილური აპლიკაციების შესაქმნელად; შესაბამისად, თუ გამოვიყენებთ მობილურ აპლიკაციებს მონაცემთა შეგროვებისთვის, მნიშვნელოვანი დროითი რესურსების დაზოგვა მოხდება. ამის მაგალითი შეიძლება იყოს სხვადასხვა გამოკითხვა (შერჩევითი დაკვირვება), რაც ტარდება მსოფლიო მასშტაბით, შესაბამისად, ინტერვიუერს მოპასუხის კითხვების ჩაწერა აღარ მოუწევს ფურცელზე, არამედ გამოიყენებს მობილურ აპლიკაციას, რომელიც შეიტანს მონაცემებს სათანადო ბაზაში, იქნება ეს SQL Server-ში არსებული, თუ სხვა. ამასთან ერთად, როგორც უკვე აღვნიშნეთ, Java გამოიყენება ვებაპლიკაციების დეველოპმენტის მიმართულებით და ამისათვის სხვადასხვა ფრეიმვერკი არსებობს. შესაბამისად, პოტენციური გამოკითხვის აპლიკაციის არქიტექტურაში შესაძლებელია გამოვიყენოთ Java. მაგალითად, შესაძლებელია შეიქმნას ვებაპლიკაცია, რომელსაც ექნება ერთგვარი API, მობილური აპლიკაცია, ან გამოკითხვების ვებაპლიკაცია დაუკავშირდება ამ API-ის და ატვირთავს პასუხებს, რის შედეგადაც მონაცემები შეინახება SQL Server-ზე jDBC-ის გამოყენებით. რაც შეეხება უკვე არსებული მონაცემების დამუშავებას, ეს შეიძლება განხორციელდეს როგორც ამ პროგრამული ენის დახმარებით, ასევე სხვა პროგრამული ენებისა თუ პროგრამების გამოყენებით. ამ პროგრამული ენისთვის არსებობს სხვადასხვა პროგრამული ბიბლიოთეკა და ფრეიმვერკი, რომლის დახმარებით შესაძლებელია მონაცემთა ანალიზი, ვიზუალიზაცია, მანქანური დასწავლის ალგორითმების რეალიზება, დიდ მონაცემებთან მუშაობა, ღრუბლოვანი კომპიუტინგის გამოყენება და სხვა. პოპულარული ფრეიმვერკები, რომლებიც გამოიყენება Java-ში სხვადასხვა მიზნით, იქნება ეს ვებდეველოპმენტი თუ მონაცემებთან მუშაობა, შემდეგია:

- Spring;

- Hibernate;
- Apache Spark;
- Apache Hadoop;
- Google Web Toolkitდა სხვ.

რაც შეეხება ინტეგრირებული დეველოპმენტის გარემოს მხარდაჭერას, მისი გამოყენება გამარტივებულია ისეთ გარემოში და ტექსტურ რედაქტორებში, როგორიცა:

- IntellijIDEA;
- Eclipse;
- NetBeans;
- Android Studio;
- Xcode;
- Visual Studio Code და სხვ.

Java-ს შექმნაში მნიშნელოვანი წვლილი შეიტანეს **ჯეიმს გოსლინგმა, მაიკ შერიდანმა** და **პატრიკ ნაუგტონმა.** მოგვიანებით მოხდა მისი დახვეწა სხვადასხვა ორგანიზაციის მიერ. დღეს ის გამოიყენება მრავალი მიმართულებით, იქნება ეს მობილურის სიმ ბარათებში, ჭკვიან ბარათებში, ვებდეველოპმენტში, მონაცემთა დამუშავებაში და ა. შ. საკმაოდ მოთხოვნადი პროგრამული ენაა მსოფლიოს მასშტაბით და საქართველოშიც და მას იყენებს არაერთი ცნობილი ორგანიზაცია (მაგალითად, Google, საქსტატი და ა. შ.) სხვადასხვა დანიშნულებით.

**C#** - ეს პროგრამული ენა ხშირად გამოიყენება დესკტოპ და ვებაპლიკაციებთან სამუშაოდ და აქტიურად არის მხარდაჭერილი მისი ეკოსისტემის განვითარება კორპორაცია Microsoft-ის მიერ. იგი თავსებადია პოპულარულ .NET Framework-თან და ASP.NET-თან, რაც საშუალებას იძლევა შემუშავდეს სხვადასხვა პროგრამული გადაწვეტილება უფრო მარტივად. C#-ის ძირითადი შემქმნელია **ანდერს ჰელსბერგი.** Windows-ის ოპერაციულ სისტემასთან და, ამავდროულად, სხვა Microsoft-ის პროდუქტებთან საკმაოდ კარგი თავსებადობის გამო, მისი არჩევა დეველოპერებში საკმაოდ ხშირია გარკვეული პროგრამული გადაწყვეტილებების დასანერგად. ამიტომაც SQL Server-ის მონაცემთა ბაზებში განთავსებულ მონაცემებთან სამუშაოდ გარკვეულ პირობებში შესაძლებელია ამ პროგრამულ ენაზე არჩევანის გაკეთება კარგი ვარიანტი იყოს. C#-ს შეუძლია დაუკავშირდეს SQL Server-ის მონაცემთა ბაზას ADO.NET-ის გამოყენებით.

C# ძირითადად გამოიყენება ვებ და დესკტოპ აპლიკაციების დეველოპმენტის მიმართულებით, თუმცა, NET MAUI-ის დახმარებით შესაძლებელია, ასევე, მრავალპლატფორმული აპლიკაციების შემუშავება, იქნება ეს Windows, Mac OS, iOS თუ Android ოპერაციული სისტემებისთვის. როგორც Java-ს განხილვისას აღვნიშნეთ, ანალოგიურად არის შესაძლებელი C#-ის დანერგვა მონაცემთა დამუშავების მიმართულებით. მაგალითად, შესაძლებელია შეიქმნას მობილური აპლიკაცია გამოკითხვებისთვის. NET MAUI-ის გამოყენებით, ამავდროულად, შესაძლებელია ცალკე შეიქმნას ვებაპლიკაცია გამოკითხვების მიმართულებით ASP.NET-ის დახმარებით. თუ შეიქმნება API C#-ის დახმარებით ისევ ASP.NET-ის გამოყენებით, ორივე წყაროდან მიღებული მონაცემები შესაძლებელია შეინახოს SQL Server 2022-ის მონაცემთა ბაზაში ADO.NET ფრეიმ-

ვერკის დახმარებით. რაც შეეხება უშუალოდ მონაცემთა დამუშავებას, Microsoft Office-ის აპლიკაციებისთვის C#-ის დახმარებით შეიძლება სხვადასხვა დამატების შექმნა, რომელსაც შეუძლია მონაცემებთან მუშაობა მნიშვნელოვნად გაამარტივოს. მიუხედავად იმისა, რომ მონაცემთა ანალიზის მიმართულებით სხვა პროგრამული ენები და ინსტრუმენტები უფრო პოპულარულია, ვიდრე C#, მისი დახმარებით შესაძლებელია Azure-თან ინტეგრირება და სხვადასხვა სერვისის გამოყენება. ამასთან ერთად, მონაცემთა მეცნიერების, კერძოდ მანქანური დასწავლის მიმართულებით, C#-ისთვის გამოიყენება ML.NET ფრეიმვერკი, რომელსაც ჩაშენებული აქვს სხვადასხვა ალგორითმი, რომელიც ამარტივებს მანქანური დასწავლის გამოყენებას მთელი რიგი მიმართულებებით. ეს ალგორითმები მოცემულია მის ოფიციალურ ვებსაიტზე:

- სენტიმენტების ანალიზი ორობითი კლასიფიკაციების ალგორითმებით;
- პროდუქტების რეკომენდირება მატრიცის ფაქტორიზაციის ალგორითმით;
- ფასების პროგნოზირება სხვადასხვა ტიპის რეგრესიული ალგორითმებით;
- მომხმარებელთა სეგმენტაცია კლასტერიზაციის ალგორითმით;
- ობიექტების დეტექცია სურათებში ღრმა დასწავლის მოდელებით;
- თაღლითობის აღმოჩენა ორობითი კლასიფიკაციის მოდელებით;
- ფასების პიკების აღმოჩენა ანომალიების აღმოჩენის მოდელით;
- სურათების კლასიფიკაცია ღრმა დასწავლის მოდელებით;
- გაყიდვების პროგნოზირება რეგრესიული ანალიზით და სხვ.

აღნიშნულ ფრეიმვერკს აქვს სხვა ისეთი ფრეიმვერკების გამოყენების საშუალება, როგორიცაა, მაგალითად, Tensorflow. C# მხარდაჭერილია ისეთ პოპულარულ ინტეგრირებული დეველოპმენტის გარემოში, როგორიცაა Visual Studio. უკვე ოც წელზე მეტია მიმდინარეობს C#-ის უფრო მეტად განვითარება სხვადასხვა მიმართულებით და წარმატებით გამოიყენება მსოფლიო მასშტაბით და საქართველოშიც, კერძო თუ სახელმწიფო ორგანიზაციების მიერ.

**JavaScript** - არის პოპულარული ზოგადი გამოყენების პროგრამირების ენა, რომელიც, ძირითადად, ვებდეველოპმენტის მიმართულებით გამოიყენება, თუმცა, Electron.js-ის გამოყენებით დესკტოპ აპლიკაციების მიმართულებითაც შესაძლებელია მისი გამოყენება, ხოლო მობილური აპლიკაციების დეველოპმენტისთვის კი ერთ-ერთი პოპულარული არჩევანია React Native. დღეს მრავალი ვებსაიტი იყენებს JavaScript-ს საკუთარ ვებაპლიკაციებში და მხარდაჭერილია ყველა პოპულარულ ვებბრაუზერში. მისი დახმარებით ხდება დინამიკური ვებგვერდების შემუშავება. გარდა ფრონტ-ენდისა, მისი გამოყენება საკმაოდ პოპულარული გახდა ბექ-ენდის მიმართულებით Node.js-ის ფრეიმვერკის გამოყენებით. ამ ენისთვის შემუშავებულია მრავალი პოპულარული ვებფრეიმვერკი, რომელიც გამოიყენება სხვადასხვა კერძო თუ საჯარო ორგანიზაციის მიერ და აქტიურად მიმდინარეობს მათი განვითარება. ეს ფრეიმვერკებია:

- React;
- Angular;
- Vue.js;
- Svelte;
- jQuery;

- Node.js;
- Express.js;
- Next.js;
- Passport.js და სხვა.

რაც შეეხება მონაცემთა დამუშავებასთან დაკავშირებულ საკითხებს, Node.js-ისა და msnodesql-ის დახმარებით SQL Server 2022-ის მონაცემთა ბაზებთან დაკავშირება შესაძლებლია JavaScript-ის გამოყენებით და, შესაბამისად, სხვადასხვა ოპერაციების განხორციელება მონაცემთა ბაზაში. აქვე უნდა აღინიშნოს ისიც, რომ JavaScript-ის გამოყენება შესაძლებელია ისეთ NoSQL მონაცემთა ბაზებში, როგორიცაა MongoDB-ში არსებული მონაცემთა ბაზები. თავის მხრივ, MongoDB დიდ და არაცხრილურ მონაცემთან მუშაობისთვის შეიძლება საკმაოდ კარგი არჩევანი აღმოჩნდეს. ამასთან, მარტივია JavaScript-ის დახმარებით სხვადასხვა API-ების შემუშავება, რომლებიც შემდგომში შეგვიძლია გამოვიყენოთ მონაცემთა შეგროვებისთვის. გარდა ამისა, კითხვარებისთვის საჭირო ვებაპლიკაციების დინამიკური გრაფიკული ინტერფეისების აგებაში მნიშვნელოვანია JavaScript-ის გამოყენება. აღნიშნული პროგრამული ენა მხარდაჭერილია სხვადასხვა ინტეგრირებული დეველოპმენტის გარემოსა და ტექსტურ რედაქტორებში, როგორიცაა:

- Visual Studio;
- Visual Studio Code;
- Sublime Text;
- WebStorm;
- IntellijIDEA;
- NetBeans და სხვ.

მიუხედავად იმისა, რომ მანქანური დასწავლისა და ანალიტიკის მიმართულებით **JavaScript** არც ისე პოპულარულია, როგორც სხვა პროგრამები და პროგრამული ენები, მისთვის შექმნილია სხვადასხვა ფრეიმვერკი, რომლებიც აქტიურად გამოიყენება ამა თუ იმ ორგანიზაციის მიერ ვიზუალიზაციის და მანქანური დასწავლისთვის. მაგალითად, ვიზუალიზაციისთვის გამოიყენება D3.js და Chart.js. მანქანური დასწავლისთვის პოპულარულია Tensorflow.js.

JavaScript თავდაპირველად შეიმუშავა **ბრენდან ეიკმა** 1995 წელს, თუმცა, მისი დახვეწა და ახალი სტანდარტების შემუშავება მუდმივად მიმდინარეობს. საქართველოშიც და საზღვარგარეთაც აქტიურად გამოიყენება სხვადასხვა დანიშნულებით, რაც საინტერესოს და მიზანშეწონილს ხდის მის შესწავლას.

## 13.3. მანქანური დასწავლა

მანქანური დასწავლა ხელოვნური ინტელექტის ერთ-ერთი განშტოებაა, რომელიც შეგვიძლია გამოვიყენოთ მონაცემთა დამუშავებისა და ანალიზისთვის. ხელოვნური ინტელექტი პოპულარული გახდა სხვადასხვა მიმართულებით, იქნება ეს რაოდენობრივ თუ ხარისხობრივ მონაცემთა დაჯგუფებისა და სტრუქტურირებისთვის, სათანადო მაჩვენებლების გაანგარიშებისთვის, პროგნოზირებისთვის და სხვ. ამასთან, ყურადსაღებია ის ფაქტიც, რომ ხელოვნური ინტელექტი ხშირად გამოიყენება სხვადასხვა ორგანიზაციების მიერ საქმიანობის ავტომატიზებისთვის. ასეთი მაგალითი შეიძლება იყოს ხელოვნური ინტელექტის მოდელების საშუალებით დოკუმენტების დამუშავება [თოფურია და სხვ., 2023].

ზოგადად, მანქანურ დასწავლას ყოფენ სამ ნაწილად. გვხვდება მეთოდების კომბინაციებიც, რომლებიც შესაძლებელია დაკლასიფიცირდეს დამატებით ქვეკლასებად, თუმცა, ძირითადი კლასიფიკაცია შემდეგია:

1. ზედამხედველობითი დასწავლა;
2. არაზედამხედველობითი დასწავლა;
3. გაძლიერებული დასწავლა.

თუ განვიხილავთ მაგალითების საფუძველზე ზედამხედველობითი მანქანური დასწავლის ამოცანებს, მაშინ უფრო მარტივად გასაგები გახდება, თუ რასთან გვაქვს საქმე. ამ შემთხვევაში ამოცანები შეგვიძლია გავყოთ ორ ძირითად ნაწილად, ესენია: კლასიფიკაცია და რეგრესია. კლასიფიკაციის შემთხვევაში კლასებსა და შესაბამის დამოუკიდებელ ცვლადებს მიუთითებს ზედამხედველი, ხოლო უკვე გაწვრთნილი ალგორითმი შეეცდება, ადრე მითითებული კლასებიდან ამოირჩიოს შესაბამისი კლასი, რომელიც შეესაბამება არსებულ დამოკიდებულებას. არაზედამხედველობითი დასწავლის შემთხვევაში ზედამხედველი არ მიუთითებს წინასწარ კლასებს, არამედ უშუალოდ მონაცემებიდან ხდება შესაბამისი შედეგის მიღება. პოპულარული ზედამხედველობითი დასწავლის მეთოდებია მარტივი რეგრესიები, გადაწყვეტილების ხეები, ნეირონული ქსელები, შემთხვევითი ტყე და სხვ. ამ მეთოდებით შესაძლებელია ისეთი ამოცანების გადაჭრა, როგორიც შეიძლება იყოს წარმოდგენილი მანქანურ ხედვაში, დროითი მწკრივების პროგნოზირებაში, გეოგრაფიულ მონაცემებთან მუშაობისას და სხვ.

არაზედამხედველობითი დასწავლის მეთოდებში შედის კლასტერიზაცია, ანომალიების აღმოჩენა და სხვ. ანომალიების აღმოჩენისას კონკრეტული დაკვირვება შესაძლებლია რადიკალურად განსხვავდებოდეს არსებული დაკვირვებებისგან, რამაც შესაძლოა კითხვები გააჩინოს მოცემულ დაკვირვებაზე. კლასტერიზაცია პოპულარულია გარკვეული ცვლადების მიხედვით. მაგალითად, წარმოვიდგინოთ, რომ გვაქვს ეკონომიკური დარგები და თითოეულს განვიხილავთ ორი ცვლადით. მაგალითისთვის ავიღოთ დასაქმება და გამოშვება. თუ, დავუშვათ, მივუთითებთ, რომ დარგები გადანაწილდეს ოთხ კლასტერში არსებული მონაცემების მიხედვით, მაშინ შესაძლოა ასეთი სურათი მივიღოთ: არსებობს ოთხი კლასტერი, რომელშიც გადანაწილდა დარგები. ეს კლასტერებია:

1. მაღალი გამოშვება, მაღალი დასაქმება;
2. მაღალი გამოშვება, დაბალი დასაქმება;
3. დაბალი გამოშვება, მაღალი დასაქმება;
4. დაბალი გამოშვება, დაბალი დასაქმება.

გარდა ანალიზისა, ამოცანა შეიძლება დაისვას სხვადასხვა ასპექტით. მაგალითად, როგორ შეიძლება მოხდეს, რომ მოცემულ კლასტერში არის დარგი, რომელსაც აქვს მაღალი დასაქმება და დაბალი გამოშვება, გადავიდეს ისეთ კლასტერში, რომელსაც აქვს მაღალი დასაქმება და მაღალი გამოშვება. ამგვარად შეგვეძლება შევაფასოთ დარგები და, აქედან გამომდინარე, გავაკეთოთ საჭირო დასკვნები, რამაც შესაძლოა გამოკვეთოს პრიორიტეტები სხვადასხვა დარგის განვითარებასთან დაკავშირებით.

არსებობს გაძლიერებული დასწავლის სხვადასხვა ვარიანტი, თუმცა, უმეტეს შემთხვევაში ხდება გარემოს მოდელირება და არსებულ გარემოში აგენტების ჩასმა; შემდეგ ეს აგენტები სწავლობენ მიიღონ ისეთი გადაწყვეტილებები, რომლებიც მაქსიმალური სარგებლის მოტანას უზრუნველყოფს. ინერგება ისეთი მეთოდები, როგორიც შეიძლება იყოს მაგალითად, სიმულირებული აგენტის დასჯა და სხვ.

გაძლიერებულ დასწავლას მჭიდრო კავშირი აქვს მეცნიერების სხვადასხვა დარგთან, მათ შორის ეკონომიკასთან და სტატისტიკასთან. დღევანდელ ეტაპზე აღნიშნული მეთოდები წარმატებით გამოიყენება სხვადასხვა თანამედროვე მეცნიერებასა და ტექნოლოგიებში, მათ შორის რობოტიკაში (ხელოვნურ ინტელექტში).

## 13.4. დიდი მონაცემები

თანამედროვე სტატისტიკაში და მეცნიერების სხვა დარგებში ხშირად ისმის ტერმინი „დიდი მონაცემები“ (ინგლ. Big Data). ამ ქვეთავში მოკლედ განვიხილავთ დიდი მონაცემების ცნებას და შინაარსს, რადგან მათი ფართოდ განხილვა ამ სახელმძღვანელოს მიზანს სცილდება.

ზოგადად, დიდი მონაცემები, რაც იყო წარსულში, თანამედროვე ცხოვრებაში, ტექნოლოგიური განვითარებიდან გამომდინარე, შეიძლება არ ჩაითვალოს დიდ მონაცემებად, რადგან თუ ადრე მონაცემთა დამუშავება რთული იყო გარკვეული ზღვრის შემდეგ, დღეს ამ ზღვარმა უფრო მაღლა აიწია. შესაბამისად, მონაცემებს, რომელთა დასამუშავებლად შეიძლება საჭირო ყოფილიყო რამდენიმე კომპიუტერი, ახლა შესაძლებელია იგივე რაოდენობის მონაცემები ერთ კომპიუტერზეც დამუშავდეს.

დიდი მონაცემები არ გულისხმობს მხოლოდ დიდ ფიზიკურ მოცულობას. იგი რთულად სტრუქტურირებული და ერთმანეთთან ძლიერად თუ სუსტად დაკავშირებული რაოდენობრივი მონაცემების ოკეანეა. თანამედროვე ეტაპზე მთავარი პრობლემა არა მათი მოპოვება, არამედ დამუშავება და ანალიზია. დიდი მონაცემების რეალურ დროში ანალიზი არ არის მარტივი ამოცანა, რადგან მისი პროგრამული უზრუნველყოფა დამოკიდებულია ბევრ კონკრეტულ პირობაზე. მართალია, დღეს ღია სივრცეში არაერთი პროგრამული ტექნოლოგია უფასოდ ხელმისაწვდომია, მრავალი ორგანიზა-

ციისთვის (კომპანიისთვის), მაგრამ საკუთარი ამოცანების გადასაწყვეტად ჯერ კიდევ დიდი ძალისხმევა ესაჭიროება.

IBM-ის მეცნიერების კლასიფიკაციით, იმისათვის რომ მონაცემები ჩაითვალოს დიდად, შემოგვთავაზეს ოთხი მახასიათებელი. ესენია:

**1. მოცულობა**

მონაცემთა მოცულობა იზრდება ყოველდღიურად, მაგალითისთვის: ინტერნეტში მუშაობისათვის ადამიანი ტოვებს ციფრულ კვალს სოციალურ ქსელებში, საძიებო ძრავში ინფორმაციის მოძიებისას, სხვადასხვა ვებსაიტებზე წვდომისას და ა. შ. შესაბამისად, დაგროვებულ მონაცემთა რაოდენობა იზრდება და საჭირო ხდება ისეთი ტექნოლოგიების შემუშავება და გამოყენება, რომლებიც არსებული მოთხოვნის დაკმაყოფილებისთვის იქნება საჭირო.

**2. მრავალფეროვნება**

ამ შემთხვევაში მონაცემები შეიძლება განსხვავდებოდეს ერთმანეთისგან, კერძოდ: ზოგი შეიძლება იყოს სტრუქტურირებული, ზოგი არასტრუქტურირებული. ზოგი შეიძლება იყოს მოცემული ცხრილის სახით, ხოლო ზოგი მონაცემი - ვიდეოს სახით და ა. შ.

**3. სიზუსტე**

ზოგიერთი მონაცემი შეიძლება იყოს ზუსტი, თუმცა არსებობს ისეთი მონაცემებიც, რომელთა ხარისხიც შეიძლება არ იყოს მაღალი. აქედან გამომდინარე, ჩნდება გარკვეული საფიქრალი ასეთ მონაცემთან მუშაობისას.

**4. სისწრაფე**

შესაძლოა, მონაცემები სწრაფად გროვდებოდეს და საჭირო ხდებოდეს მისი სწრაფი დამუშავება. მაგალითისთვის, ასეთი სიტუაცია შეიძლება შეიქმნას საფონდო ბირჟაზე სავაჭრო სესიის განმავლობაში.

ამჟამად დიდ მონაცემთან მუშაობისა და ანალიზისთვის შემუშავებულია სხვადასხვა ტექნოლოგია. მოკლედ განვიხილოთ ზოგიერთი მათგანი:

**Hadoop** არის **Apache-ს** მიერ შექმნილი პროგრამული უზრუნველყოფა, რომელიც დაწერილია Java-ში. იგი არის ღია კოდის და ხშირად გამოიყენება დიდ მონაცემებთან სამუშაოდ. მის ოფიციალურ საიტზე (hadoop.apache.org) ვკითხულობთ:

**„Apache Hadoop** პროგრამული ბიბლიოთეკა არის ფრეიმვერკი, რომელიც იძლევა მონაცემთა დიდი ნაკრების განაწილებული დამუშავების საშუალებას კომპიუტერების კლასტერებში მარტივი პროგრამირების მოდელების გამოყენებით. იგი შექმნილია ერთი სერვერიდან ათასობით მანქანამდე მასშტაბირების გაზრდის მიზნით, როდესაც თითოეული გთავაზობთ ადგილობრივ გამოთვლას და შენახვას. იმის ნაცვლად, რომ დაეყრდნოს აპარატურას მაღალი ხელმისაწვდომობის უზრუნველსაყოფად, თავად ბიბლიოთეკა შექმნილია იმისათვის, რომ აღმოაჩინოს და გაუმკლავდეს წარუმატებლობებს აპლიკაციის ფენაში. ასე რომ, იგი უზრუნველყოფს მაღალი ხელმისაწვდომობის სერვისს კომპიუტერების კლასტერზე, რომელთაგან თითოეული შეიძლება მიდრეკილი იყოს წარუმატებლობისკენ“.

შემდეგი საინტერესო ტექნოლოგია არის **Apache Spark.** მის ოფიციალურ ვებსაიტზე (spark.apache.org) ვკითხულობთ:

Apache Spark™ არის მრავალენოვანი ძრავა მონაცემთა ინჟინერიის, მონაცემთა მეცნიერებისა და მანქანური დასწავლის შესასრულებლად ერთკვანძოვან მანქანებზე ან კლასტერებზე. მისი ძირითადი მახასიათებლებია: მონაცემების დამუშავების გაერთიანება ჯგუფურად და რეალურ დროში სტრიმინგში, სასურველი ენის გამოყენებით, იქნება ეს Python, SQL, Scala, Java თუ R. მას შეუძლია შეასრულოს სწრაფი, განაწილებული ANSI SQL მოთხოვნები ვიზუალიზაციისა და მონაცემების გამოსატანად და, ასევე, ad-hoc რეპორტინგისთვის. იგი მუშაობს უფრო სწრაფად, ვიდრე მონაცემთა საწყობების უმეტესობა. მასში შესაძლებელია შესრულდეს ექსპლორაციული მონაცემების ანალიზი (EDA) პეტაბაიტის მასშტაბის მონაცემებზე შერჩევის შემცირების გარეშე. შესაძლებელია მანქანური დასწავლის ალგორითმების ადაპტირება ლეპტოპზე და იგივე კოდის გამოყენება ხარვეზების მიმართ მდგრად კლასტერებზე და ათასობით მოწყობილობაზე. რაც შეეხება ეკოსისტემას, მის საშუალებით შესაძლებელია ისეთი ტექნოლოგიების გამოყენება, როგორიცაა:

1. Sklearn;
2. Pandas;
3. Tensorflow;
4. Numpy;
5. R;
6. Power BI;
7. Parquet;
8. SQL Server;
9. MongoDB;
10. Elasticsearch;
11. Casandra;
12. Kubernetes და სხვა.

ჩვენთვის ამ მხრივ საკმაოდ საინტერესოა SQL-ის გამოყენების შესაძლებლობა, რასაც შეუძლია მნიშვნელოვნად გაამარტივოს დიდ მონაცემებთან მუშაობა.

**MongoDB,** როგორც **NoSQL** მონაცემთა ბაზა, წარმატებით გამოიყენება დიდ მონაცემებთან სამუშაოდ. მისი საშუალებით შესაძლებელია მონაცემთა გადანაწილება სხვადასხვა სერვერზე და ისეთ მონაცემებთან მუშაობა, რომლებიც შეიძლება არ იყოს ცხრილური ფორმატით. წარმატებით გამოიყენებენ IoT ტექნოლოგიებიდან მონაცემთა შეგროვებისთვის. ზოგადად იყენებს BSON ფორმატს, რომელიც წააგავს JSON ტიპის ჩანაწერებს.

**Docker** არის ერთ-ერთი წარმატებული პროგრამული უზრუნველყოფა დეველოპერებს შორის. მისი საშუალებით მარტივადაა შესაძლებელი პროგრამული უზრუნველყოფის მიწოდება მომხმარებლებისთვის კონტეინერების საშუალებით ისე, რომ მცირდება გარემოს კონფიგურაციისა და მენეჯმენტის საჭიროება.

**Kubernetes** არის კონტეინერების ორკესტრაციის სისტემა, რომელიც გამოიყენება პროგრამული უზრუნველყოფის გამართვის, მასშტაბირებისა და მენეჯმენტისთვის. იგი გამოშვებულია კორპორაცია Google-ის მიერ.

თანამედროვე ეტაპზე დიდი მონაცემების დასამუშავებლად, ასევე, მათი ანალიზისა და, საერთოდ, მონაცემთა მართვის პროცესში უკვე საკმაოდ ხშირად გამოიყენება **ღრუბლოვანი ტექნოლოგიები.** აქვე უნდა აღვნიშნოთ, რომ ღრუბლოვანი ტექნოლოგიები გამოიყენება არა მარტო დიდი მონაცემების დასამუშავებლად, არამედ სხვა ტიპის პროგრამულ აქტიურობებში და გადაწყვეტილებებში.

ღრუბლოვანი გარემო (cloud) - ეს არის სერვერების ქსელი, სადაც ყოველ სერვერს აქვს განსხვავებული ფუნქცია. ღრუბლოვანი ტექნოლოგიების ისტორია სათავეს იღებს 1963 წლიდან, როდესაც DARPA-მ (the Defense Advanced Research Projects Agency) შეიმუშავა პროექტი, რომლის თანახმად ერთი კომპიუტერის გამოყენება უნდა შეძლებოდა ორ ან მეტ მომხმარებელს; ასევე, მათ გამოიყენეს სიტყვა „ვირტუალიზაცია“, რომელიც აღნიშნავდა თანამედროვე ვირტუალიზაციის საწყის იდეებს. პროექტი თანდათან განვითარდა და 21-ე საუკუნეში მიაღწია დიდ წარმატებას. ღრუბლოვანი ტექნოლოგიების ძირითადი იდეა მდგომარეობს იმაში, რომ ზოგიერთ სერვერზე შესაძლებელია გაშვებული იყოს აპლიკაცია, ან რაიმე მომსახურება და სერვისი.

ღრუბლოვანი ტექნოლოგიების გამოყენებით საჭირო აღარ არის ცალკე მდგომი აპლიკაცია ან სერვისი, რომელიც გაშვებულია ყოველი მომხმარებლის კომპიუტერზე ცალ-ცალკე. დამატებით ღრუბლოვანი გარემო საშუალებას იძლევა გავაზიაროთ ტექნოლოგიები და რესურსები ისე, რომ მასზე დაშვება ჰქონდეს რამდენიმე მომხმარებელს. მომხმარებლის თვალსაზრისით, ღრუბლოვანი გარემო არის შავი ყუთი, რომელსაც იგი მიმართავს, მაგრამ არ იცის და არ აინტერესებს, თუ რა ხდება ყუთის შიგნით [ასანიშვილი, 2018].

ღრუბლოვანი ტექნოლოგიების გამოყენებით შესაძლებელია ნებისმიერი ორგანიზაციის თუ კომპანიის მართვის გაუმჯობესება ოპტიმალური გადაწყვეტილებების საფუძველზე. თანამედროვე სტატისტიკური მეცნიერების მიხედვით, მონაცემების მართვისა და დამუშავების მეთოდოლოგიაში ღრუბლოვანი ტექნოლოგიები ერთ-ერთ ყველაზე პოპულარულ გადაწყვეტილებად ითვლება.

ამჟამად უკვე მსოფლიოში არაერთი ტიპის ღრუბლოვანი ტექნოლოგია არსებობს, რომლებიც მნიშვნელოვნად ამარტივებენ სხვადასხვა ტიპის ამოცანას. პოპულარულია ისეთი პროვაიდერები, როგორიცაა: **Google Cloud, Amazon Web Services და Microsoft Azure.**

თანამედროვე კომპიუტერულ სისტემებში, ძირითადად, გამოიყენება სამი ტიპის ღრუბლოვანი გარემო. ესენია: ღია, დახურული და პიბრიდული.

- ღია ტიპის ღრუბლოვან გარემოში მომხმარებელს შეუძლია ნებისმიერ დროს და ნებისმიერი ადგილიდან ინფორმაციაზე წვდომა;
- დახურული ტიპის ღრუბლოვან გარემოს აქვს იგივე მახასიათებლები და ბენეფიტები, რაც ღია ტიპისას, მაგრამ იმ განსხვავებით, რომ დახურული ტიპის გარემოში მომხმარებლების წვდომა შეზღუდულია;
- ჰიბრუდული ტიპის ღრუბლოვან გარემოში გაერთიანებულია როგორც ღია, ასევე დახურული ტიპის მახასიათებლები.

ღრუბლოვანი ტექნოლოგიების პოპულარობის მიზეზებად შეძლება ჩაითვალოს შემდეგი უპირატესობები:

❖ ვირტუალური მონაცემთა სანახის ცენტრი მომხმარებლებს საშუალებას აძლევს ჰქონდეთ წვდომა მონაცემებთან ნებისმიერი ადგილიდან, მაშინ როცა, მონაცემთა სანახის ადგილმდებარეობას მნიშვნელობა არ აქვს;

❖ საექსპლოატაციო ხარჯების მაქსიმალურად შემცირების შესაძლებლობა;

❖ ვირტუალური სერვერები ხელს უწყობს მათ კონსოლიდაციას, რაც საშუალებას იძლევა გამოვიყენოთ რამდენიმე პოსტინგი ერთ ვირტუალურ მანქანაზე;

❖ გაუმჯობესებული მდგრადობა და მოქნილობა [ასანიშვილი, 2018].

ღრუბლოვანი ტექნოლოგიების გამოყენებით შესაძლებელია მნიშვნელოვნად შემცირდეს ხარჯები დეველოპმენტის მიმართულებით და, ასევე, დაიზოგოს თანხები, რომელიც შეიძლება მანამდე საჭირო ყოფილიყო შესაბამისი ინფრასტრუქტურის შესაძენად, იქნებოდა ეს სერვერული აპარატურა, საჭირო მარშრუტიზატორები და გადამრთველები თუ სხვა. ღრუბლოვან ტექნოლოგიებში მხარდაჭერილია ისეთი სერვისები, როგორიც შეიძლება იყოს ხელოვნური ინტელექტის გამოყენება სხვადასხვა ტიპის მონაცემთა დამუშავებაში, იქნება ეს მანქანური ხედვის მიმართულებით, ხმოვანი მონაცემების, ტექსტური ინფორმაციის, რაოდენობრივი გამოთვლებისთვის და სხვა. შესაძლებელია ღრუბლოვანი ტექნოლოგიების გამოყენება პროგნოზირების მიმართულებითაც, მაგალითად საპროგნოზო მოდელის დანერგვა ღრუბლოვანი ტექნოლოგიების სერვისების გამოყენებით (Chogovadze et al., 2020). ასევე, შესაძლებელია უკვე შექმნილი აპლიკაციის გაშვება ღრუბლოვან პლატფორმაზე და საჭიროებიდან გამომდინარე, მისი მასშტაბირება. ასეთი საჭიროება შესაძლებელია შეიქმნას აპლიკაციის მომხმარებლების უეცარი ზრდის გამო და სხვა მიზეზებით. ჩვენ შემთხვევაში, საინტერესოა არსებული მონაცემების დამუშავება სხვადასხვა მიმართულებით. მაგალითისთვის მხარდაჭერილია SQL-ის სერვისების შეთავაზება ღრუბლოვანი ტექნოლოგიების პროვაიდერების მხრიდან, ასევე, გარკვეული მოწყობილობების ქირაობა მონაცემების დასამუშავებლად, იქნება ეს გრაფიკული პროცესორები თუ ტენსორის დამუშავების ერთეულები. შესაბამისად, ამ გზით შესაძლებელია ხარჯებისა და დროის შემცირება. ამასთან ერთად, ზოგიერთ შემთხვევაში, მონაცემთა განთავსება ღრუბელში შეიძლება უფრო უსაფრთხო აღმოჩნდეს, ვიდრე ამა თუ იმ კონკრეტული ორგანიზაციის საკუთარ სერვერებზე.

### *კითხვები თვითშემოწმებისთვის*

1. დაახასიათეთ თქვენთვის ნაცნობი მეთოდები მონაცემთა უსაფრთხოების უზრუნველსაყოფად და ახსენით მათი საჭიროება.
2. ჩამოთვალეთ სხვადასხვა ინსტრუმენტი მონაცემთა დამუშავებისთვის.
3. აღწერეთ, რას ნიშნავს დიდი მონაცემები.
4. განიხილეთ რამდენიმე ინსტრუმენტი დიდ მონაცემებთან სამუშაოდ.

# დამატებითი რესურსები

ამ წიგნის გარდა, ინტერნეტში არსებობს დამატებით რესურსები, რომლებიც შესაძლოა დაეხმაროს მკითხველებს მონაცემებთან მუშაობაში. პირველ რიგში, საინტერესოა მონაცემთა წყაროები საქართველოს მაგალითიდან გამომდინარე. აქედან უნდა გამოვარჩიოთ საქართველოს სტატისტიკის ეროვნული სამსახურის ვებსაიტი https://www.geostat.ge/ka და საქართველოს ეროვნული ბანკის სტატისტიკური მონაცემების ვებგვერდი https://nbg.gov.ge/statistics/statistics-data.

SQL პროგრამირების შესასწავლად გამოგვადგება https://w3schools.com და თავად Microsoft-ის ონალაინრესურსები https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16.

დამატებითი ინსტრუმენტების შესასწავლად და, ასევე, მონაცემების მოსაძიებლად კარგი რესურსია https://kaggle.com, რომელზეც განთავსებულია სასწავლო რესურსები, რომლებიც დაეხმარება ადამიანებს მონაცემთა მეცნიერების შესწავლაში. აქვე მოთავსებულია სხვადასხვა მონაცემთა ბაზა, რომელთა საშუალებით შესაძლებელია დაიხვეწოს მონაცემთა ინჟინერიის, ანალიტიკის და, ზოგადად, მონაცემთა მეცნიერების უნარები. აქვე მკითხველს შეუძლია გაეცნოს მანქანური დასწავლის გაკვეთილებს და უშუალოდ პლატფორმაზე შეასრულოს შესაბამისი დავალებები.

გარდა ამისა, SQL-ის შესასწავლად დიდ დახმარებას გაგიწევთ ქართულენოვანი სახელმძღვანელო „SQL სერვერი“, 2016, რომლის ავტორები არიან რ. სამხარაძე და ლ. გაჩეჩილაძე.

არარელაციურ ბაზებში საკმაოდ პოპულარული გახდა MongoDB, რომლის გამოყენებაც შესაძლებელია დიდ მონაცემებთან სამუშაოდ (https://www.mongodb.com). აქედან გამომდინარე, ამ ტექნოლოგიის შესწავლა მომავალში დამატებით უპირატესობას მისცემს ამა თუ იმ ადამიანს მონაცემებთან მუშაობაში.

# გამოყენებული ლიტერატურა


1. ასანიშვილი გ. (2018). ღრუბლოვანი ტექნოლოგიები და უსაფრთხოების პრობლემები. თსუ, თბილისი.
2. გელაშვილი ს. (2015). სტატისტიკური ინერციულობა და R-პროგრამის გამოყენების შესაძლებლობა ეკონომიკურ პროგნოზირებაში. ივ. ჯავახიშვილის სახ. თსუ-ს საერთაშორისო სამეცნიერო კონფერენციის - „თანამედროვე ინფორმაციული ტექნოლოგიები ეკონომიკური გლობალიზაციის პირობებში“ - შრომების კრებული. თსუ, თბ., 2015, გვ. 45-49. https://www.tsu.ge/data/file_db/economist_faculty/inform.pdf
3. თოფურია ნ. (2020). ღრუბლოვან პლატფორმაზე ბიზნეს გადაწყვეტილებების იმპლემენტაცია, პრაქტიკული სამუშაოს მეთოდური მითითებანი, სტუ, თბ., 2020, გვ. 32.
4. თოფურია, ნ., ზაკაშვილი, შ., კაშიბაძე, მ., (2023). დოკუმენტ-პროცესინგი ხელოვნური ინტელექტის მოდელების საშუალებით. საქართველოს ტექნიკური უნივერსიტეტის საერთაშორისო სამეცნიერო-პრაქტიკული კონფერენცია „თანამედროვე გამოწვევები და მიღწევები ინფორმაციულ და საკომუნიკაციო ტექნოლოგიებში - 2023“.
5. სამხარაძე რ., გაჩეჩილაძე ლ. (2016). SQL სერვერი. გამომც. სტუ, თბილისი, 2016, გვ. 450.
6. Ben-Gan, I. (2023). *T-SQL Fundamentals (Developer Reference), Fourth Edition. Microsoft Press.*
7. Chogovadze, G., Surguladze, G., Topuria, N., Archvadze, N. (2020). Implementation of a Prediction Model with Cloud Services, *Bulletin of the Georgian National Academy of Sciences*
8. Codd, E. (1970). A relational model of data for large shared data banks. *Commun. ACM, 13(6), 377-387.*
9. Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to Algorithms, Third Edition. The MIT Press.*
10. European Commission, Eurostat, NACE Rev. 2 – *Statistical classification of economic activities in the European Community, Publications Office, 2008.*
11. Knuth, D. (1988). Sorting and Searching. The Art of Computer Programming. *Vol. 3 (2nd ed.).*
12. Rankins, R., Bertucci, P., Gallelli, Silverstein, T. A. (2015). Microsoft SQL Server. Pearson Education, Inc.
13. https://www.datacamp.com/tutorial/normalization-in-sql
14. https://geostat.ge
15. https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16
16. https://medium.com/@sukeshgowdakr54/normalization-in-sql-1828a21eb785
17. https://www.w3schools.com/sql/sql_datatypes.asp
18. https://eviews.com/home.html

19. https://www.ibm.com/spss
20. https://www.microsoft.com/en-us/power-platform/products/power-bi
21. https://www.mathworks.com/products/matlab.html
22. https://www.java.com/en/
23. https://dotnet.microsoft.com/en-us/apps/ai/ml-dotnet
24. https://developer.mozilla.org/en-US/docs/Web/JavaScript

გამომცემლობის რედაქტორი – მაია ეჯიბია

გამოცემაზე მუშაობდნენ ნათია დვალი, ლელა წიკლაური და მარიამ ებრალიძე

ივანე ჯავახიშვილის სახელობის თბილისის სახელმწიფო უნივერსიტეტის გამომცემლობა

თბილისი, 2025

---

0128 თბილისი, ილია ჭავჭავაძის გამზ. 1
1 Ilia Tchavtchavadze Avenue, Tbilisi 0128
Tel +995 (32) 225 04 84, 6284, 6278
https://www.tsu.ge/ka/publishing-house