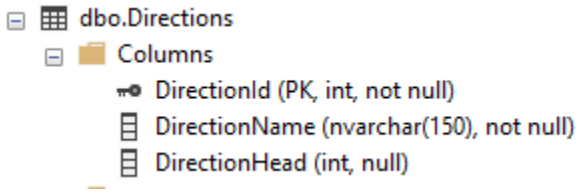
ლაბორატორია 5

მონაცემების შეტანა ცხრილებში INSERT ბრძანების გამოყენებით

ბრძანება INSERT INTO გამოიყენება SQL Server-ის მონაცემთა ბაზის ცხრილში ერთი ან რამდენიმე სტრიქონის ჩასაწერად.
სინტაქსი:

```
INSERT INTO  table_name (column_name1, column_name2...)
VALUES (column1_value, column2_value...);
```

მოდი ჩავსვათ მონაცემები ცხრილში Directions, რომელიც შევქმენით წინა ლაბორატორიაზე.



(საჭიროების შემთხვევაში გამოვიყენოთ ჩვენს მიერ უკვე შემქნილი ბაზის სკრიპტი, რომელიც მოცემულია თავის ბოლოს)
შემდეგი INSERT INTO ბრძანება ჩასვამს ერთ რიგს ზემოთ მოყვანილი ცხრილის ყოველ სვეტში:

```
INSERT INTO [dbo].[Directions]
(
    [DirectionName], [DirectionHead]
)
VALUES
(
N'კომპიუტერული მეცნიერება', NULL
 )
```

ჩასმის ბრძანების ტანი გამოიყურება შემდეგნაირად:

1. `INSERT INTO` საბრძანებო სიტყვა რომესაც მოსდევს ცხრილის დასახელება რომელშიც შედის მონაცემი. ჩვენს შემთხვევაში `[dbo].[Directions]`
2. მრგვალ ფრჩხილებში მოთავსებული იმ სვეტების ჩამონათვალი რომლებიც უნდა შეისვოს შესაბამისი მიმდევრობით. ჩვენს შემთხვევაში DirectionId სვეტი არის აიდენთითი ავტოგადანომვრადი სვეტი, რომლის მონაცემიც გენერირდება ავტომატურად და შესაბამისად იგი არ მონაწილეობს `INSERT`

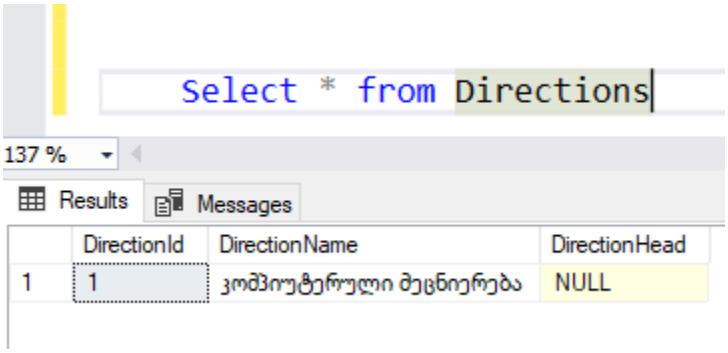
ოპერაციაში. ივსება მხოლოდ DirectionName დაDirectionHead სვეტები, თუმცა ვინდაინდან DirectionHead-ის ნომერი ამ ეტაპზე რჩება ცარიელი და უნდა მივანიჭოთ მოგვიანებით, შესძლებელია მისი გამოტოვებაც.

3. VALUES საბრძანებო სიტყვა და კვლავ მრგვალ ფრჩხილებში მოთავსებული მნიშვნელობები ზემოთ ჩამოთვლილი სტრიქონებისთვის, ჩვენს შემთხვევაში როგორც უკვე ვთქვით ივსება DirectionName დაDirectionHead სვეტები და მრგვალ ფრჩხილებში უნდა გადავცეთ მნიშვნელობები იმავე თანმიმდევრობით. (N'კომპიუტერული მეცნიერება',NULL)

მნიშვნელოვანია რომ სვეტების რაოდენობა და მნიშვნელობების რაოდენობა ასევე ტიპი და თანიმდევრობა იყოს იდენტური, თუ ვავსებთ ორ სვეტს და პირველი არის სტრიქონი ხოლო მეორე მთელი რიცხვი საჭიროა პარამეტრიც გადავცეთ ორი პირველი სტრიქონი და მეორე მთელი რიცხვი.

შეუძლებელია შევსების დროს გამოვტოვოთ სავალდებულოდ შესავსები სვეტები თუ ისინი არ ივსება ავტოგადანომვრით ან გაჩუმების პრინციპით.

ჩასმული მონაცემების სანახავად შეასრულეთ Select * from Directions; მოთხოვნა შეკითხვის რედაქტორში, როგორც ეს ნაჩვენებია ქვემოთ.



როგორც ხედავთ მოხდა Directions ცხრილში პირველი სტრიქონის შეტანა.

შესაძლებელი იყო ზემოთ მოცემულ კოდში გამოგვეტოვებინა სვეტი რომელიც არ იყო სავალდებულოდ შესავსები, მასში ავტომატურად ჩაიწერებოდა NULL ცარიელი მნიშვნელობა:

```
INSERT INTO [dbo].[Directions]
(
 [DirectionName]
)
VALUES
(
N'კომპიუტერული მეცნიერება'
```

```
 )
```

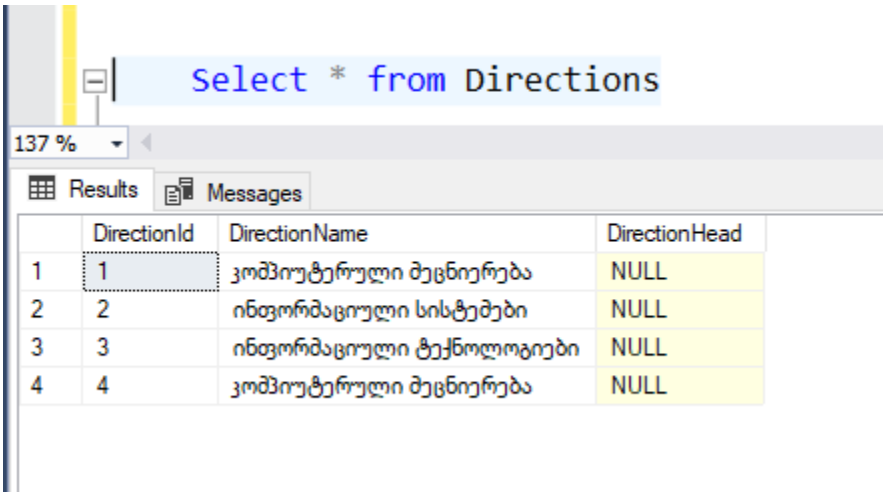
თუ VALUES საბრძანებო სიტყვის შემდგომ მრგვალი მნიშვნელობების შემცველი მრგვალი ფრჩხილების რამოდენიმე ბლოკს ჩავსვავთ და ერთმანეთისგან მძიმით გამოვყოფთ მოხდება ერთი ბრძანებით რამოდენიმე ჩანაწერის შეტანა:

```
INSERT INTO [dbo].[Directions]
(
 [DirectionName]
)
VALUES
(N'ინფორმაციული სისტემები'),
(N'ინფორმაციული ტექნოლოგიები')
```

თუ ცხრილის ყველა სვეტი ივსება, (გარდა IDENTITY-სა) შესაძლებელია INSERT INTO სიტყვის შემდეგ არ მივუთითოთ შესავსები სვეტების ჩამონათვალი. ასეთ შემთხვევაში ვალდებულები ვართ შევსების დროს დავიცვათ მონაცემთა ცხრილში არსებული მიმდევრობა:

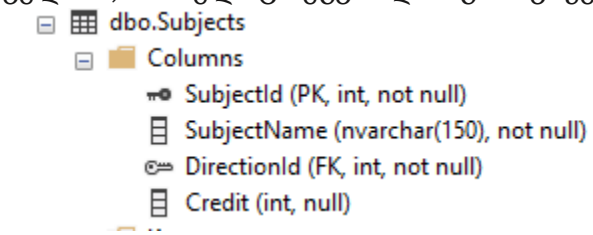
```
INSERT INTO [dbo].[Directions]
VALUES
(
N'კომპიუტერული მეცნიერება', NULL
)
```

შეგვიძლია კვლავ დავათვალიეროთ ცხრილი Select * from Directions ბრძანებით.



	DirectionId	DirectionName	DirectionHead
1	1	კომპიუტერული მეცნიერება	NULL
2	2	ინფორმაციული სისტემები	NULL
3	3	ინფორმაციული ტექნოლოგიები	NULL
4	4	კომპიუტერული მეცნიერება	NULL

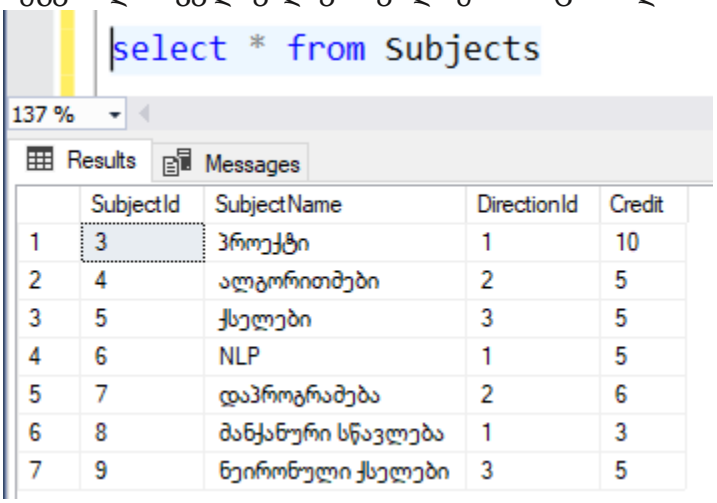
ბაზაში დაგვემატა მიმართულებები, თავისი შესაბამისი რიგითი ნომრით (აიდი ველით) რომელიც შეგვიძლია გამოვიყენოთ და შევავსოთ საგნების ცხრილი:



ვინაიდან ცხრილის სრულად შევსება ხდება და აიდი ველის გარდა არცერთი სვეტის გამოტოვება არ ხდება შესაძლებელია მრგვალ ფრჩხილებში არ მივუთითოთ სვეტების ჩამონათვალი, მაგრამ დავიცვათ საგნების ცხრილში არსებული სტვეტების მიმდევრობა და ჯერ შევავსოთ დასახელება, შემდეგ მიმართულების ნომერი (რომელსაც ვიღებთ წინა ცხრილიდან)და ბოლოს კრედიტების რაოდენობა:

```
insert into Subjects
values
 (N'პროექტი',1, 10)
,(N'ალგორითმები',2,5)
,(N'ქსელები',3,5)
,(N'NLP',1,5)
,(N'დაპროგრამება',2,6)
,(N'მანქანური სწავლება',1,3)
,(N'ნეირონული ქსელები',3, 5)
```

შეგვიძლია კვლავ დავათვალიეროთ ცხრილი Select * from Subjectsბრძანებით.



	SubjectId	SubjectName	DirectionId	Credit
1	3	პროექტი	1	10
2	4	ალგორითმები	2	5
3	5	ქსელები	3	5
4	6	NLP	1	5
5	7	დაპროგრამება	2	6
6	8	მანქანური სწავლება	1	3
7	9	ნეირონული ქსელები	3	5

ახლა შევავსოთ სტუდენტების ცხრილი:

- dbo.Students
  - Columns
    - StudentId (PK, int, not null)
    - StudentName (nvarchar(30), not null)
    - StudentLastName (nvarchar(30), not null)
    - Email (varchar(30), null)
    - DirectionId (FK, int, not null)
    - Password (varbinary(250), null)

გამოვტოვოთ პაროლის ველი ამიტომ სავალდებულოა მრგვალ ფრჩხილებში მივუთითოთ შესავსები ველების ჩამონათვალი:

```
INSERT INTO [dbo].[Students]
(
    [StudentName],
    [StudentLastName],
    [Email],
    [DirectionId]
)
VALUES
(N'ერეკლე', N'ბურკაძე', 'aaa@gmail.com', 1),
(N'ნინო', N'ქურდაძე', 'aaa@gmail.com', 1),
(N'ანნა', N'გაგუა', 'aaa@gmail.com', 2),
(N'გიგო', N'პაიკიძე', 'aaa@gmail.com', 3),
(N'ნუცა', N'ურიდია', 'aaa@gmail.com', 1),
(N'მაკა', N'ჯოჯუა', 'aaa@gmail.com', 2),
(N'გიორგი', N'სალია', 'aaa@gmail.com', 2),
(N'დავითი', N'ზვიადაძე', 'aaa@gmail.com', 3),
(N'ნიკა', N'ცინცაძე', 'aaa@gmail.com', 1),
(N'ლელა', N'დანელია', 'aaa@gmail.com', 1);
```

დავათვალიეროთ ცხრილი Select * from [Students]ბრძანებით.

```
Select * from [Students]
```

137 %

Results | Messages

	StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1	1000	ერეკლე	ბურკაძე	aaa@gmail.com	1	NULL
2	1001	ნინო	ქურდაძე	aaa@gmail.com	1	NULL
3	1002	ანნა	გაგუა	aaa@gmail.com	2	NULL
4	1003	გიგო	პაიკიძე	aaa@gmail.com	3	NULL
5	1004	ნუცა	ურიდია	aaa@gmail.com	1	NULL
6	1005	მაკა	ჯოჯუა	aaa@gmail.com	2	NULL
7	1006	გიორგი	სალია	aaa@gmail.com	2	NULL
8	1007	დავითი	ზვიადაძე	aaa@gmail.com	3	NULL
9	1008	ნიკა	ცინცაძე	aaa@gmail.com	1	NULL
10	1009	ლელა	დანელია	aaa@gmail.com	1	NULL

ახლა კი ზოგიერთი სტუდენტისთვის შევავსოთ შესაბამისი დეტალურის ცხრილი:

```
INSERT INTO [dbo].[StudentDetails]
(
[StudentId],
[DateOfBirth],
[PersonalId],
[Phone],
[City]
)
VALUES

(1001,'3/4/2000', '02020104340','577324543',N'ბათუმი'),
(1000,'3/4/2001', '01212010434','578464543',N'თელავი'),
(1002,'1/5/1998','01110104311',N'578324543',N'ქუთაისი'),
(1003,'5/7/2002','02020111240',N'57731243',N'თელავი'),
(1005,'7/8/2001','01320111233',N'577324543',N'ბათუმი'),
(1007,'10/2/2003','60000000000',N'571224543',DEFAULT),
(1009,'1/3/1997','70000000000',N'599324543',DEFAULT)
```

დავათვალიეროთ ცხრილი Select * from [StudentDetails]ბრძანებით.

```
Select * from [dbo].[StudentDetails]
```

137 %

Results | Messages

	StudentDetailId	StudentId	PersonalId	Photo	DateOfBirth	Phone	Email	City
1	3	1001	02020104340	NULL	2000-03-04	577324543	NULL	ბათუმი
2	4	1000	01212010434	NULL	2001-03-04	578464543	NULL	თელავი
3	5	1002	01110104311	NULL	1998-01-05	578324543	NULL	ქუთაისი
4	6	1003	02020111240	NULL	2002-05-07	57731243	NULL	თელავი
5	7	1005	01320111233	NULL	2001-07-08	577324543	NULL	ბათუმი
6	8	1007	60000000000	NULL	2003-10-02	571224543	NULL	თბილისი
7	9	1009	70000000000	NULL	1997-01-03	599324543	NULL	თბილისი

თუ კი სკრიპტში რომელიმე სტრიქონში გვინდა რომ სვეტი შეივსოს გაჩუმების პრინციპით მინიჭებული მნიშვნელობით უნდა შესაბამისი მნიშვნელობის ადგილას მივუთითოთ სიტყვა `DEFAULT`.

მონაცემების განახლება ცხრილებში Update ბრძანების გამოყენებით

ცხრილში ჩანაწერების განახლებისთვის გამოიყენება ბრძანება UPDATE სინტაქსი:

```
UPDATE table_name
SET column_name1 = new_value,
column_name2 = new_value,
...
[WHERE Condition];
```

იმისათვის რომ განვაახლოთ ერთი ან რამოდენიმე სტრიქიონი Students ცხრილში საჭიროა მივუთითოთ

1. `UPDATE` საბრძანებო სიტყვა, შემდეგ კი ცხრილის სახელი რომელშიც ხდება ცვლილება, ჩვენს შემთხვევაში `UPDATE [dbo].[Students]`
2. `SET` საბრძანებო სიტყვა, შესაცვლელი სვეტებისა და მათი ახალი მნიშვნელობებით, რამოდენიმე შესაცვლელი სვეტის არსებობის შემთხვევაში თითოეული გამოვყოთ მძიმით. ჩვენს შემთხვევაში მეილში ჩავწეროთ ახალი მნიშვნელობა `SET [Email]='Nino123@gmail.com'`
3. `WHERE` პირობა რომელშიც მივუთითებთ რომელ სტრიქონზე ხდება ცვლილება, შეიძლება განახლების ოპერაციას საერთოდ არ ქონდეს ლოგიკური პირობა, მაგრამ უნდა გავითვალისწინოთ რომ ამ შემთხვევაში განახლება მთელი სვეტი. ჩვენს შემთხვევაში მეილი განვაახლოთ სტუდენ ნომერ 1001 სთვის.`WHERE [StudentId]=1001`

მივიღებთ კოდს:

```
UPDATE [dbo].[Students]
SET [Email]='Nino123@gmail.com'
WHERE [StudentId]=1001
```

დავათვალიეროთ ცხრილი `Select * from [Students]` ბრძანებით.

```
Select * from [Students]
```

137 %

Results | Messages

	StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1	1000	ერეკლე	ბურჯაძე	aaa@gmail.com	1	NULL
2	1001	ნინო	ჭურდაძე	Nino123@gmail.com	1	NULL
3	1002	ანნა	გაგუა	aaa@gmail.com	2	NULL
4	1003	გიგო	ჰაციძე	aaa@gmail.com	3	NULL
5	1004	ნუცა	ურიდია	aaa@gmail.com	1	NULL
6	1005	მაკა	ჯოჯუა	aaa@gmail.com	2	NULL
7	1006	გიორგი	სალია	aaa@gmail.com	2	NULL
8	1007	დავითი	ზვიადაძე	aaa@gmail.com	3	NULL
9	1008	ნიკა	ცინცაძე	aaa@gmail.com	1	NULL
10	1009	ლელა	დანელია	aaa@gmail.com	1	NULL

სტუდენტ ნომერ 1001-ს განუახლდა მეილის ველი.

ახლა განვაახლოთ რამოდენიმე სვეტი ერთად, მაგალითად სტუდენტ ნომერ 1005-ს დეტალურ ინფორმაციაში შევუცვალოთ ქალაქი და დაბადების თარიღი:

```
UPDATE [dbo].[StudentDetails]
SET [City]=N'რუსთავი',
[DateOfBirth]='2000-03-07'
WHERE [StudentId]=1005
```

ასევე თუ ლოგიკურ (WHERE) პირობას ერთდროულად რამოდენიმე ჩანაწერი აკმაყოფილებს, შესაძლებელია რამოდენიმე სტრიქონის ერთდროულად განახლება. მაგალითად იმ სტუდენტებს ვისაც აქამდე მეილში ეწერათ 'aaa@gmail.com' ჩავუწეროთ 'student@tsu.edu.ge'

```
UPDATE [dbo].[Students]
SET [Email]='student@tsu.edu.ge'
WHERE [Email]='aaa@gmail.com'
```

კვლავ დავათვალიეროთ ცხრილი `Select * from [Students]` ბრძანებით.

```
Select * from [Students]
```

137 %

Results | Messages

	StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1	1000	ერეკლე	ბურჯაძე	student@tsu.edu.ge	1	NULL
2	1001	ნინო	ჭურდაძე	Nino123@gmail.com	1	NULL
3	1002	ანნა	გაგუა	student@tsu.edu.ge	2	NULL
4	1003	გიგო	ჰაიკიძე	student@tsu.edu.ge	3	NULL
5	1004	ნუცა	ურიდია	student@tsu.edu.ge	1	NULL
6	1005	მაკა	ჯოჯუა	student@tsu.edu.ge	2	NULL
7	1006	გიორგი	სალია	student@tsu.edu.ge	2	NULL
8	1007	დავითი	ზვიადაძე	student@tsu.edu.ge	3	NULL
9	1008	ნიკა	ცინცაძე	student@tsu.edu.ge	1	NULL
10	1009	ლელა	დანელია	student@tsu.edu.ge	1	NULL

განახლება სხვა ცხრილიდან: *(ბონუსი)
შესაძლებელია ჩანაწერების განახლება სხვა ცხრილიდან, მაგალითად StudentDetails ცხრილში გვაქვს ცარიელი იმეილის ველი, თუმცა აღნიშნული ველი შევსებულია ცხრილ Students -ში. შესაძლებელია დაიწეროს პირობა რომელიც შეავსებს შესაბამისი ნომრის მქონე სტუდენტის მეილს დეტალურის ცხრილში საწყისი Students ცხრილის გამოყენებით:

```
UPDATE [dbo].[StudentDetails]
SET [Email]=[s].email
FROM [dbo].[Students] AS [s]
WHERE s.[StudentId]=[StudentDetails].[StudentId]
```

დავათვალიეროთ ცხრილი Select * from [StudentDetails]ბრძანებით.

```
Select * from [StudentDetails]
```

137 %

Results | Messages

	StudentDetailId	StudentId	PersonalId	Photo	DateOfBirth	Phone	Email	City
1	3	1001	02020104340	NULL	2000-03-04	577324543	Nino123@gmail.com	ბათუმი
2	4	1000	01212010434	NULL	2001-03-04	578464543	student@tsu.edu.ge	თელავი
3	5	1002	01110104311	NULL	1998-01-05	578324543	student@tsu.edu.ge	ჭუთაისი
4	6	1003	02020111240	NULL	2002-05-07	57731243	student@tsu.edu.ge	თელავი
5	7	1005	01320111233	NULL	2000-03-07	577324543	student@tsu.edu.ge	რუსთავი
6	8	1007	60000000000	NULL	2003-10-02	571224543	student@tsu.edu.ge	თბილისი
7	9	1009	70000000000	NULL	1997-01-03	599324543	student@tsu.edu.ge	თბილისი

მონაცემების წაშლა ცხრილებში Delete ბრძანების გამოყენებით

ცხრილში ჩანაწერების წაშლისთვის გამოიყენება ბრძანება Delete
სინტაქსი:

```
DELETE FROM table_name
[WHERECondition];
```

იმისათვის რომ წავშალოთ ერთი ან რამოდენიმე სტრიქიონი Students ცხრილში საჭიროა მივუთითოთ

1. `DELETE FROM` საბრძანებო სიტყვა, შემდეგ კი ცხრილის სახელი რომელშიც ხდება ცვლილება, ჩვენს შემთხვევაში `DELETE FROM [dbo].[Students]`
2. `WHERE` პირობა რომელშიც მივუთითებთ რომელი სტრიქონის წაშლა ხდება. შეიძლება წაშლის ოპერაციას საერთოდ არ ქონდეს ლოგიკური პირობა, მაგრამ უნდა გავითვალისწინოთ რომ ამ შემთხვევაში წაიშლება მთელი ცხრილის ჩანაწერები. ჩვენს შემთხვევაში წავშალოთ სტუდენტი ნომერი 1006.`WHERE [StudentId]=1006`

მივიღებთ კოდს:

```
DELETE FROM [dbo].[Students]
WHERE [StudentId]=1006
```

ჩანაწერი `DELETE FROM [dbo].[Students]` წაშლის ყველა ჩანაწერს სტუდენტების ცხრილიდან.

აგრეთვე მნიშვნელოვანია რომ უკვე არსებული გარე გასაღებების გათვალისწინებით შეუძლებელია ისეთი ველების წაშლა რომელსაც აქვს კავშირი. მაგალითად

Directions ცხრილიდან შეუძლებელია წავშალოთ მიმართულება ნომერი 1 ვინაიდან მასზე დაკავშირებულია საგნები და სტუდენტები, მაგრამ შეგვიძლია წავშალოთ მიმართულება ნომერი 4 რომელსაც არ აქვს კავშრები:

```
DELETE FROM [dbo].[Directions] WHERE [DirectionId]=1 -მივიღებთ შეცდომას.
DELETE FROM [dbo].[Directions] WHERE [DirectionId]=4 - არ მივიღებთ
შეცდომას.
```

თუ გვინდა რომ სტუდენტს წავუშალოთ დეტალურიდან მეილი, ამ შემთხვევაში არ უნდა გამოვიყეონთ DELETE ბრძანება, უნდა გადავაწეროთ ცარიელი ჩანაწერი UPDATE ბრანებით:

```
UPDATE [dbo].[StudentDetails]
SET [Email]=NULL
WHERE [StudentDetailId]=3
```

SELECT მოთხოვნა

SQL Server-ში SELECT ბრძანება გამოიყენება სტრიქონების/სვეტების მონაცემების მისაღებად ერთი ან რამოდენიმე არსებული ცხრილიდან. იგი იცავს SQL (Structured Query Language) სტანდარტებს.

სინტაქსი:

```
SELECT column1, column2,...columnN
FROM table_name
```

ყველა სვეტის არჩევა:

ოპერატორი * წარმოადგენს ცხრილის ყველა სვეტს. თუ გვსურს ცხრილის ყველა სვეტის ნახვა, თქვენ არ გჭირდებათ თითოეული სვეტის სახელის მითითება SELECT მოთხოვნაში:

```
Select * from Students
```

მოთხოვნა აბრუნებს ცხრილიდან ყველა მწკრივისა და სვეტის მონაცემებს Students, როგორც ეს ნაჩვენებია ქვემოთ:

```
Select * from [Students]
```

137 %

Results | Messages

	StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1	1000	ერეკლე	ბურჯაძე	aaa@gmail.com	1	NULL
2	1001	ნინო	ჭურდაძე	Nino123@gmail.com	1	NULL
3	1002	ანნა	გაგუა	aaa@gmail.com	2	NULL
4	1003	გიგო	პაიციძე	aaa@gmail.com	3	NULL
5	1004	ნუცა	ურიდია	aaa@gmail.com	1	NULL
6	1005	მაკა	ჯოჯუა	aaa@gmail.com	2	NULL
7	1006	გიორგი	სალია	aaa@gmail.com	2	NULL
8	1007	დავითი	ზვიადაძე	aaa@gmail.com	3	NULL
9	1008	ნიკა	ციცაძე	aaa@gmail.com	1	NULL
10	1009	ლელა	დანელია	aaa@gmail.com	1	NULL

კონკრეტული სვეტების მონაცემების მოთხოვნა:

იმისათვის რომ მივიღოთ მონაცემები მხოლოდ კონკრეტული სვეტებიდან, საჭურია მიუთითოთ სვეტების სახელები SELECT განცხადებაში:

```
SELECT StudentId
      ,StudentName
      ,StudentLastName
  FROM Students
```

მივიღებთ შედეგს:

```
SELECT StudentId
      ,StudentName
      ,StudentLastName
  FROM Students
```

137 %

Results | Messages

	StudentId	StudentName	StudentLastName
1	1000	ერეკლე	ბურჯაძე
2	1001	ნინო	ჭურდაძე
3	1002	ანნა	გაგუა
4	1003	გიგო	პაიციძე
5	1004	ნუცა	ურიდია
6	1005	მაკა	ჯოჯუა
7	1007	დავითი	ზვიადაძე
8	1008	ნიკა	ციცაძე
9	1009	ლელა	დანელია

მეტსახელი სვეტებისა და ცხრილებისთვის

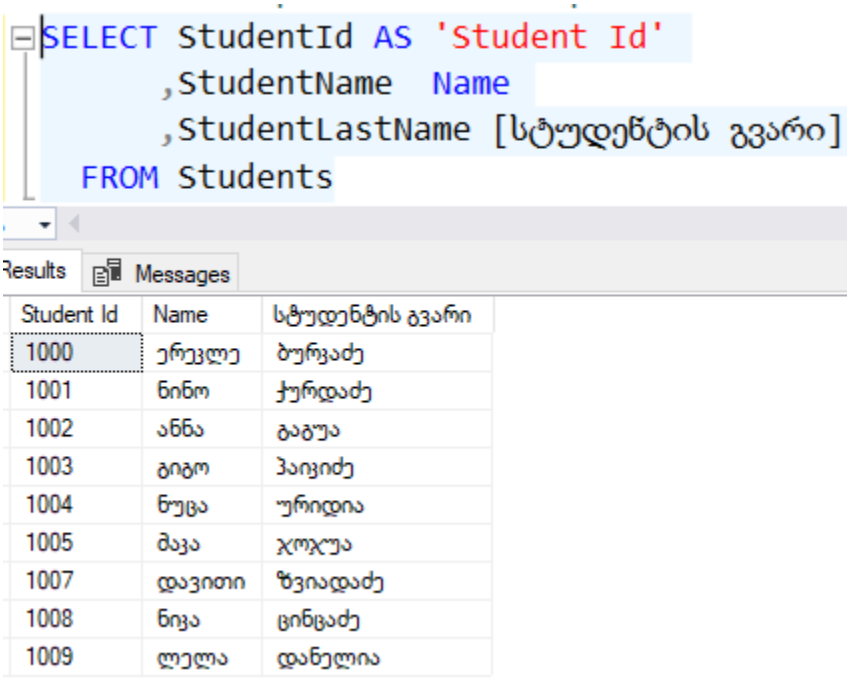
სურვილისამებრ შესაძლებელია მივუთითოთ მეტსახელი (Alias) ერთი ან მეტი სვეტისთვის SELECT მოთხოვნაში. მეტსახელი არის მოთხოვნის ცხრილის ან სვეტის დროებითი სახელი.

Alias-ის უპირატესობა:

- Alias ხდის სვეტს უკეთესად კითხვადს შედეგებში.
- ალიასი გამოიყენება შეკითხვისას ცხრილებისთვის მცირე, შემოკლებული და მნიშვნელოვანი სახელების მისანიჭებლად, რათა ადვილი იყოს ცხრილების მითითება მრავალი ცხრილის შეერთებისას.
- ალიასი გვეხმარება ამოვიცნოთ, რომელი სვეტი რომელ ცხრილს ეკუთვნის, მრავალი ცხრილიდან მონაცემების მიღების შემთხვევაში.

შემდეგი მოთხოვნა ქმნის მეტსახელებს წინა მთხოვნაში არსებული სვეტებისთვის. მიუთითეთ მეტსახელი ბრჭყალებში ან ოთხკუთხა ფრჩხილები, თუ გსურთ მასში ჰარის ან სხვა სიმბოლოს გამოყენება, ასევე შესაძლებელია ალიასის წინ `AS` სიტყვის გამოყენება :

```
SELECT StudentId AS 'Student Id'
      ,StudentName  Name
      ,StudentLastName [სტუდენტის  გვარი]
  FROM Students
```



+ ოპერატორი SELECT განცხადებაში

ოპერატორი +MS SQL Server-ში აერთიანებს სტრიქონულ მნიშვნელობებს ან კრებს ციფრულ მნიშვნელობებს. შემდეგი კოდი აერთიანებს ორ varchar სვეტს შედეგში.

```
SELECT StudentId AS 'Student Id'
      ,StudentName+' '+StudentLastName
  FROM Students
```


```
SELECT StudentId AS 'Student Id'
      ,StudentName+' '+StudentLastName
  FROM Students
```


sults | Messages

Student Id	(No column name)
1000	ერეკლე ბურჯაძე
1001	ნინო ჭურდაძე
1002	ანნა გაგუა
1003	გიგო პაიკიძე
1004	ნუცა ურიდია
1005	მაკა ჯოჯუა
1007	დავითი ზვიადაძე
1008	ნიკა ცინცაძე
1009	ლელა დანელია

თუმცა ასეთი ტიპის შედგენილი სვეტები კარგავენ სახელს და აღნიშნულ შემთხვევებში აუცილებელია ალიასის გამოყენება:

```
SELECT StudentId AS 'Student Id'
      ,StudentName+' '+StudentLastName  AS FullName
  FROM Students
```

esults | Messages

Student Id	FullName
1000	ერეკლე ბურჯაძე
1001	ნინო ჭურდაძე
1002	ანნა გაგუა
1003	გიგო პაიკიძე
1004	ნუცა ურიდია
1005	მაკა ჯოჯუა
1007	დავითი ზვიადაძე
1008	ნიკა ცინცაძე
1009	ლელა დანელია

შემდეგი სკრიპტი უბრალოდ აჯამებს ნომრებს შერჩეულ მოთხოვნაში.

```
SELECT 10 + 15; --აბრუნებს 25
SELECT 10.5 + 15; --აბრუნებს 25.5
```

FROM ოპერატორი

SELECT განცხადებას რომელიც კითხულობს მონაცემებს ცხრილებიდან აუცილებად უნდა ჰქონდეს FROM საბრძანებო სიტყვა. იგი გამოიყენება ცხრილების სახელების ჩამოსაწერად, საიდანაც გვინდა გამოვიტაონოთ მონაცემები და დავაკონკრეტოთ შეერთება ამ ცხრილებს შორის.

FROM პუნქტში შესაძლებელია მივუთითოთ რამოდენიმე ცხრილი. თუმცა, თუ ცხრილებს აქვთ იდენტური დასახელების მქონე სვეტები, მაშინ უნდა დავაკონკრეტოთ სვეტების სრული სახელები ცხრილის სახელის მითითებით, `table_name.column_name` სტილში:

შემდეგი სკრიპტი ირჩევს სვეტებს ორი ცხრილიდან:

```
SELECT * FROM Students, Directions;

SELECT Students.*, Directions.* FROM Students, Directions;

SELECT S.*, D.* FROM Students S, Directions D;

SELECT  [S].[StudentName], [D].[DirectionName]

FROM Students [S], Directions [D];
```

FROM პუნქტში რამოდენიმე ცხრილის არსებობა WHERE პირობისა და JOIN-ის გარეშე დააბრუნებს დუბლირებულ მონაცემებს თითოეული ცხრილიდან. მაგალითად, თუ `Students` ცხრილს აქვს ორი სტრიქონი და `Directions` ცხრილს აქვს ორი მწკრივი, მაშინ ზემოაღნიშნული მოთხოვნა დააბრუნებს 2X2, ოთხ რიგს, სადაც ერთი ცხრილის სვეტები მეორდება მეორე ცხრილის სვეტებისთვის. ამის შესახებ მეტს შეიტყობთ JOIN გაკვეთილიდან.

WHERE პირობა

SQL Server-ში SELECT მოთხოვნას შეიძლება საჭიროებისამებრ ჰქონდეს WHERE პუნქტი მონაცემების გასაფილტრად. WHERE პუნქტი შეიძლება შეიცავდეს ერთ ან მეტ ლოგიკურ პირობას ცხრილების მონაცემების გასაფილტრად.
WHERE პირობა ყოველთვის მოდის FROM პუნქტის შემდეგ და GROUP BY, HAVING და ORDER BY ოპერატორების წინ.
სინტაქსი:

```
SELECT column1, column2,...columnN
FROM table_name
WHERE boolean_expression;
```

WHERE პუნქტი შეიძლება შეიცავდეს ერთ ან მეტ პირობას, რომლებსაც პირობითი ოპერატორების გომოყენებით ფილტრავს მონაცემებს. განიხილეთ შემდეგი მოთხოვნა:

```
SELECT * FROM Subjects
 WHERE Credit>5
```

ზემოხსენებულ მოთხოვნაში, პირობა `Credit>5` აბრუნებს სტრიქონებს, სადაც `Credit` სვეტის მნიშვნელობა 5-ზე მეტია.

```
SELECT * FROM Subjects
 WHERE Credit>5
```

Results | Messages

SubjectId	SubjectName	DirectionId	Credit
3	პროექტი	1	10
7	დაპროგრამება	2	6

შემდეგი მოთხოვნა იყენებს BETWEEN ოპერატორს WHERE პირობაში:

```
SELECT * FROM StudentDetails
 WHERE [DateOfBirth] BETWEEN '2000-03-04' AND '2002-05-07'
```

პირობა `[DateOfBirth] BETWEEN '2000-03-04' AND '2002-05-07'`აბრუნებს რიგებს, სადაც სვეტის მნიშვნელობა `DateOfBirth` არის `'2000-03-04'` -დან `'2002-05-07'`-მდე (ორივე მნიშვნელობის ჩათვლით):

```
SELECT * FROM StudentDetails
WHERE [DateOfBirth] BETWEEN '2000-03-04' AND '2002-05-07'
```

Results | Messages

StudentDetailId	StudentId	PersonalId	Photo	DateOfBirth	Phone	Email	City
3	1001	02020104340	NULL	2000-03-04	577324543	Nino123@gmail.com	ბათუმი
4	1000	01212010434	NULL	2001-03-04	578464543	student@tsu.edu.ge	თელავი
6	1003	02020111240	NULL	2002-05-07	57731243	student@tsu.edu.ge	თელავი
7	1005	01320111233	NULL	2000-03-07	577324543	student@tsu.edu.ge	რუსთავი

პირობა WHERE მრავალი პირობით

WHERE პირობა შეიძლება შეიცავდეს მრავალ პირობას და აერთიანებდეს მათ AND და OR ოპერატორების გამოყენებით. შემდეგი მოთხოვნა იყენებს ლოგიკურ ოპერატორს AND-ს მონაცემების გასაფილტრად ორი პირობის საშულებით:

```
SELECT * FROM Subjects
 WHERE Credit<=5 AND [DirectionId]=1
```

ზემოთ მოყვანილ მოთხოვნაში WHERE პირობა Credit<=5 AND [DirectionId]=1 განსაზღვრავს ორ პირობას, რომლებიც გაერთიანებულია AND ოპერატორით. შედეგად მივიღებთ რიგებს ცხრილიდან Subjects, სადაც მნიშვნელობა DirectionId არის 1 და Credit ნაკლებია ან ტოლია 5-ის.

```
SELECT * FROM Subjects
 WHERE Credit<=5 AND [DirectionId]=1
```

37 %

Results | Messages

	SubjectId	SubjectName	DirectionId	Credit
1	6	NLP	1	5
2	8	მანქანური სწავლება	1	3

ქვემოთხოვნა WHERE პირობაში

WHERE პირობას ასევე შეუძლია გამოიყენოს სხვა მოთხოვნის შედეგად მიღებული მნიშვნელობა:

```
SELECT * FROM Students
 WHERE DirectionId =(SELECT DirectionId FROM Directions WHERE
DirectionName=N'კომპიუტერული მეცნიერება')
```

ზემოთ მოყვანილ სკრიპტში WHERE პირობა გამოიყურება შემდეგნაირად: WHERE DirectionId =(SELECT DirectionId FROM Directions WHERE DirectionName=N'კომპიუტერული მეცნიერება'). ამიტომ, პირველ რიგში შესრულდება ქვემოთხოვნა SELECT DirectionId FROM Directions WHERE DirectionName=N'კომპიუტერული მეცნიერება' და მიღებული DirectionId მნიშვნელობა (ამ შემტხვევაში-1) გამოყენებული იქნება რიგების გასაფილტრად. და აღუშნული პირობა იდენტური იქნება WHERE DirectionId =1 პირობის.

```
SELECT * FROM Students
 WHERE DirectionId =(SELECT DirectionId FROM Directions WHERE DirectionName=N'კომპიუტერული მეცნიერება')
```

sults | Messages

StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1000	ერეკლე	ბურკაძე	student@tsu.edu.ge	1	NULL
1001	ნინო	ჭურდაძე	Nino123@gmail.com	1	NULL
1004	ნუცა	ურიდია	student@tsu.edu.ge	1	NULL
1008	ნიკა	ცინცაძე	student@tsu.edu.ge	1	NULL
1009	ლელა	დანელია	student@tsu.edu.ge	1	NULL

პირობითი ოპერატორები

შემდეგი ოპერატორების გამოყენება შესაძლებელია WHERE პირობაში:

ოპერატორი	აღწერა
=	ტოლია
>	მეტია
<	ნალებია
>=	მეტი ან ტოლი
<=	ნაკლები ან ტოლი
<> ან !=	არ უდრის. ზოგიერთ მონაცემთა ბაზაში !=გამოიყენება არა ტოლი მნიშვნელობების შესადარებლად.
BETWEEN	დიაპაზონს შორის
LIKE	ნიმუშის მსგავსი: პროცენტის სიმბოლო (%): გამოსახავს სიმბოლოთა ნებისმიერ რაოდენობას. ქვედატირე (_): ანაცვლებს ერთ სიმბოლოს. [სიმბოლოების თანმიმდევრობა]: ემთხვეოდეს ამ სიმრავლიდან ნებისმიერ სიმბოლოს. [სიმბოლო-სიმბოლო]: ნებისმიერი სიმბოლო მოცემული დიაპაზონიდან [^]: ნებისმიეროი სიმბოლო, რომელიც არ ეკუთვნის მოცემულ დიაპაზონს.
IN	სიის ელემენტებთან გატოლება

GROUP BY ოპერატორი

SQL Server-ში GROUP BY ოპერატორი გამოიყენება შემაჯამებული მონაცემების მისაღებად ერთი ან მეტი დაჯგუფების პირობის საფუძველზე. ჯგუფები შეიძლება ჩამოყალიბდეს ერთ ან მეტ სვეტზე. მაგალითად, მოთხოვნა GROUP BY გამოყენებული იქნება თითოეულ მიმართულებაზე სტუდენტების რაოდენობის დასათვლელად, ან სტუდენტთა ჯამური რაოდენობის მისაღებად. ამისათვის ჩვენ უნდა გამოვიყენოთ აგრეგატული ფუნქციები, როგორიცაა COUNT(), MAX(), და ა.შ. SELECT მოთხოვნაში GROUP BY პირობის შედეგი აბრუნებს ერთ მწკრივს GROUP BY სვეტის თითოეული მნიშვნელობისთვის.

სინტაქსი:

```
SELECT column1, column2,...columnN FROM table_name
[WHERE]
[GROUP BY column1, column2...columnN]
[HAVING]
[ORDER BY]
```

SELECT მოთხოვნა შეიძლება შეიცავდეს მხოლოდ იმ სვეტებს, რომლებიც გამოიყენება GROUP BY პირობაში. SELECT მოთხოვნაში სხვა სვეტების ჩასართავად

უნდა გამოვიყენოთ აგრეგატული ფუნქციები, როგორიცაა COUNT(), MAX(), MIN(), SUM(), AVG().

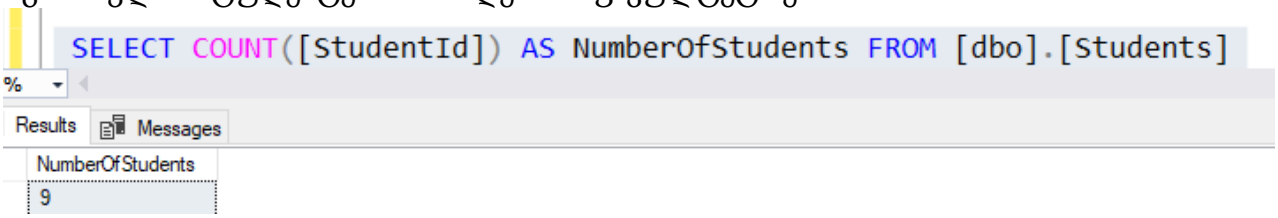
GROUP BY მახასიათებლები:

- GROUP BY პუნქტი გამოიყენება ჩანაწერების ჯგუფების შესაქმნელად.
- GROUP BY პუნქტი უნდა იჯდეს WHERE პუნქტის შემდეგ, მისი არსებობის შემთხვევაში და HAVING პუნქტამდე.
- GROUP BY პუნქტი შეიძლება შეიცავდეს ერთ ან მეტ სვეტს ამ სვეტების საფუძველზე ერთი ან მეტი ჯგუფის შესაქმნელად.
- მხოლოდ GROUP BY სვეტები შეიძლება შევიდეს SELECT პუნქტში. SELECT პუნქტში სხვა სვეტების გამოსაყენებლად გამოიყენეთ მათთან ერთად აგრეგატული ფუნქციები.

განვიხილოთ მოთხოვნა:

```
SELECT COUNT([StudentId]) AS NumberOfStudents FROM [dbo].[Students]
```

იგი ითვლის სტუდენტების რაოდენობა ფაკულტეტზე



თუ გვსურს მიმართულებების მიხედვით სტუდენტების რაოდენობის დათვლა საჭიროა ცხრილი დავაჯგუფოთ მიმართულებების მიხედვით და შემდგომ დავთვალოთ რაოდენობა თითოეულ ჯგუფში:

```
SELECT COUNT([StudentId]) AS NumberOfStudents FROM [dbo].[Students]
 GROUP BY [DirectionId]
```

```sql
CREATE DATABASE [Faculty];
GO

USE [Faculty1];

CREATE TABLE [Directions]
(
    [DirectionId] INT NOT NULL PRIMARY KEY IDENTITY,
    [DirectionName] NVARCHAR(150) NOT NULL
        UNIQUE,
    [DirectionHead] INT
        NULL
);



CREATE TABLE [Students]
(
    [StudentId] INT IDENTITY(1000,1)
        CONSTRAINT [PK_Students] PRIMARY KEY,
    [StudentName] NVARCHAR(30) NOT NULL,
    [StudentLastName] NVARCHAR(30) NOT NULL,
    [Email] VARCHAR(30) CHECK ([Email] LIKE '%@%'),
    [DirectionId] INT NOT NULL
        FOREIGN KEY REFERENCES [dbo].[Directions] ([DirectionId]),
    [Password] VARBINARY(250)
);

CREATE TABLE [StudentDetails]
(
    [StudentDetailId] INT IDENTITY PRIMARY KEY,
    [StudentId] INT
        UNIQUE
        FOREIGN KEY REFERENCES [dbo].[Students] ([StudentId]),
    [PersonalId] VARCHAR(11)
        UNIQUE,
    [Photo] VARBINARY(MAX),
    [DateOfBirth] DATE,
    [Phone] VARCHAR(20),
    [City] NVARCHAR(50)
        DEFAULT (N'თბილისი'),
);

CREATE TABLE [Subjects]
(
    [SubjectId] INT IDENTITY(1, 1) PRIMARY KEY,
    [SubjectName] NVARCHAR(150) NOT NULL,
    [DirectionId] INT NOT NULL
        FOREIGN KEY REFERENCES [dbo].[Directions] ([DirectionId]),
    [Credit] INT
);

CREATE TABLE [StudentSubjects]
(
    [StudentSubjectId] INT IDENTITY PRIMARY KEY,
    [StudentId] INT NOT NULL
```

```
        FOREIGN KEY REFERENCES [dbo].[Students] ([StudentId]),
    [SubjectId] INT NOT NULL
        FOREIGN KEY REFERENCES [dbo].[Subjects] ([SubjectId]),
    [RegisterDate] DATETIME NOT NULL
        DEFAULT (GETDATE()),
    [IsPassed] BIT NOT NULL
        DEFAULT (0)
);


CREATE TABLE [Lecturers]
(
    [LecturerId] INT IDENTITY PRIMARY KEY,
    [LecturerName] NVARCHAR(30) NOT NULL,
    [LecturerLastName] NVARCHAR(30) NOT NULL,
    [Phone] VARCHAR(20),
    [Email] VARCHAR(30) CHECK ([Email] LIKE '%@%')
);


CREATE TABLE [SubjectLecturers]
(
    [SubjectLecturerId] INT IDENTITY PRIMARY KEY,
    [SubjectId] INT NOT NULL
        FOREIGN KEY REFERENCES [dbo].[Subjects] ([SubjectId]) ON UPDATE CASCADE ON DELETE
CASCADE,
    [LecturerId] INT NOT NULL
        FOREIGN KEY REFERENCES [dbo].[Lecturers] ([LecturerId]) ON UPDATE CASCADE ON DELETE
CASCADE,
    CONSTRAINT [UQ_SubjectLecturers]
        UNIQUE (
                    [SubjectId],
                    [LecturerId]
                )
);
```