ლაბორატორია 6

(ლაბორატორიის ბოლოს მოცემულია ბაზა, რომელიც დაგჭირდებათ მოთხოვნების შესასრულებლად)

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

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

Results | Messages

NumberOfStudents
9

მნიშვნელოვანია დავიმახსოვროთ, რომ თუ მთხოვნა შეიცავს აგრეგატულ ფუნქციებს (COUNT(), MAX(), MIN(), SUM(), AVG()...) მოთხოვნის პირობაში ზემოთაღნიშნულის გარდა სხვა ველის დამატება ხდება შეუძლებელი. მხოლოდ დაჯგუფების პირობის გამოყენება გვაძლევს საშუალებას დავამატოთ მოთხოვნაში ის ველი რომელიც გამოყენებულია GROUP BY -ს ტანში. მარტივად რომ ვთქვათ მოთხოვნა

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

დააბრუნებს შეცდომას: „Column 'dbo.Students.DirectionId' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.“

„სვეტი „dbo.Students.DirectionId“ არასწორია მოთხოვნის სიაში, რადგან ის არ შეადგენს არც აგრეგატული ფუნქციის პარამეტრს და არც GROUP BY ოპერატორის შიგთავსს“ მარტივი მსახვედრია რომ თუ კი გვინდა სტუდენტთა რაოდენობა დავთვალოთ მიმართულებების მიხედვით ჯერ ცხრილი უნდა დაჯგუფდეს მიმართულებების შესაბამისად და შემდგომ უკვე დაჯგუფებულ დროებით ქვეცხრილებში მოხდეს სტუდენტთა რაოდენობების დათვლა.

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

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

137 %

Results | Messages

	NumberOfStudents
1	5
2	3
3	2

თუმცა აღნიშნული მოთხოვნის შედეგში გაურკვეველია რომელ ფაკულტეტზე რამდენია სტუდენტების რაოდენობა, შესაბამისად საჭიროა რომ გამოვიტანოთ ინფორმაცია ფაკულტეტის შესახებ. ზემოაღნიშნული მოთხოვნა მოიცავს GROUP BY `DirectionId` დირექტივას, ვინდაიდან `DirectionId` გამოყენებულია როგორც დაჯგუფების ბრძანების პარამეტრი მისი გამოყენება შესაძლებელი ხდება SELECT ოპერატორში:

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

შედეგად შესაძლებელი ხდება გარკვევა თუ რომელ ნომერ მიმართულებაზე რამდენი სტუდენტია დარეგისტრირებული:

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

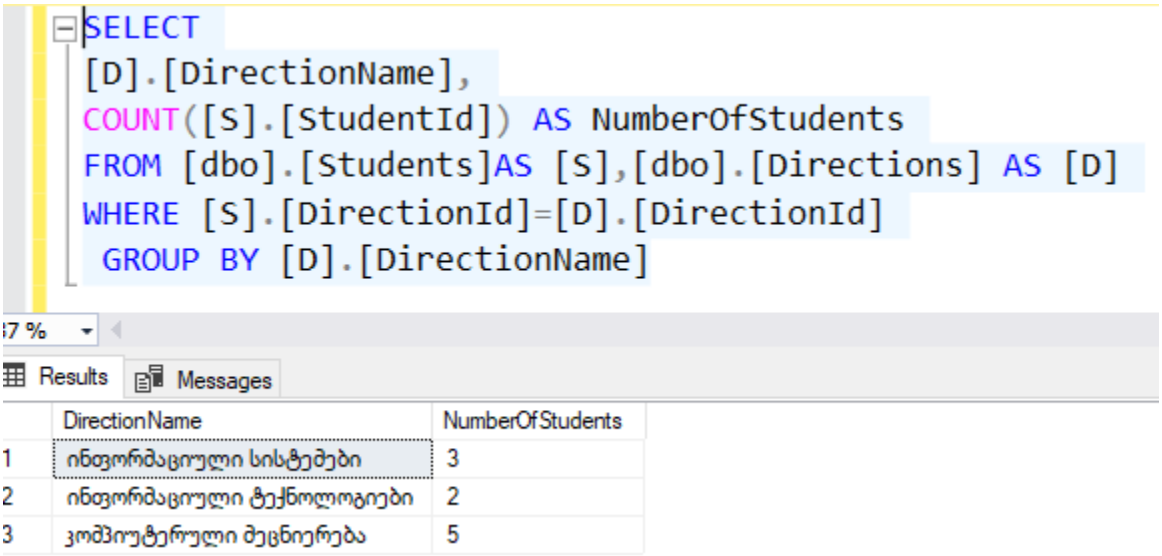
137 %

Results | Messages

	DirectionId	NumberOfStudents
1	1	5
2	2	3
3	3	2

იმისთვის რომ ნომრის მაგიერ მოთხოვნაში მივიღოთ მიმართულების სახელი საჭიროა შევასრულოთ მოთხოვნა:

```sql
SELECT
[D].[DirectionName],
COUNT([S].[StudentId]) AS NumberOfStudents
FROM [dbo].[Students]AS [S],[dbo].[Directions] AS [D]
WHERE [S].[DirectionId]=[D].[DirectionId]
 GROUP BY [D].[DirectionName]
```



HAVING პირობა

როგორც უკვე ვნახეთ დაჯგუფების GROUP BY პირობა სრულდება WHERE-ის შემდგომ, შესაბამისად ჯერ ხდება where პირობის მიხედვით საწყისი ცხრილის გაფილტვრა, შემდეგ დაჯგუფება კონკრეტული სვეტის/ სვეტების მიხედვით და შემდგომ მათზე აგრეგატული ფუნციების შესრულება. გამოდის რომ აგრეგატული ფუნქციის შედეგზე ჩვენ აღარ გვაქვს ფილტრი. რომ გვინდოდეს მხოლოდ იმ მიმართულებების გამოტანა სადაც 3-ზე მეტი სტუდენტია დარეგისტრირებული, ზემოთ მოცემული ოპერატორების საშუალებით ნამდვილად ვერ შევძლებთ. ამისათვის შემოდის დაჯგუფების ლოგიკური პირობა HAVING.

სინტაქსი:

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

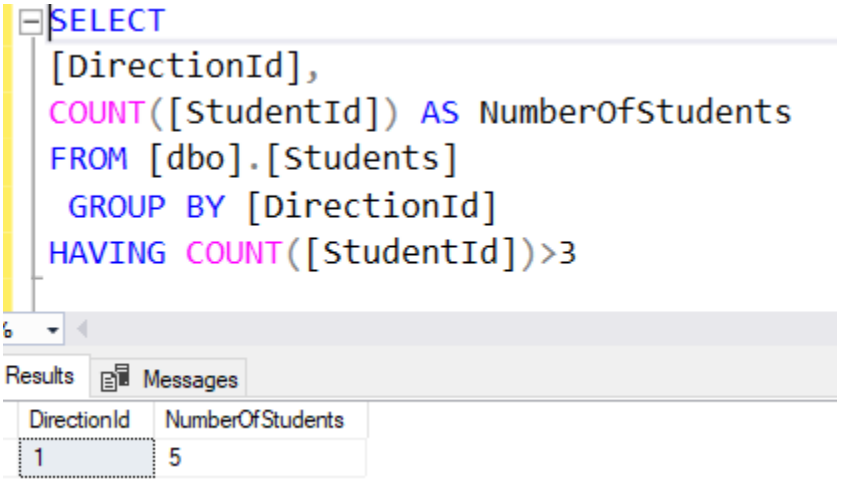
- HAVING პუნქტი გამოიყენება დაჯგუფებული ჩანაწერების გასაფილტრად.
- HAVING პირობა უნდა მოდიოდეს GROUP BY ოპერატორის შემდეგ და ORDER BY პუნქტის წინ.

- HAVING ბრძანება შეიძლება შეიცავდეს ერთ ან მეტ ლოგიკურ პირობას.
- HAVING პირობა შეიძლება შეიცავდეს მხოლოდ სვეტებს, რომლებიც გამოიყენება GROUP BY ბრძანებასთან ერთად სელექთის მოთხოვნაში. მატივად რომ ვთქვათ HAVING პირობა ედება აგრეგატული ფუნქციის შედეგს და შესაბამისად მის ტანში ვიყენებთ ზემოთაღნიშნულ აგრეგატულ ფუნქციას:

```
SELECT
[DirectionId],
COUNT([StudentId]) AS NumberOfStudents
FROM [dbo].[Students]
GROUP BY [DirectionId]
HAVING COUNT([StudentId])>3
```

ზემოთ განხილულ კოდს დავამატოთ ფილტრი დაჯგუფების შედეგზე, რომ მხოლოდ ის მიმართულებები გამოვიტანოთ სადაც 3-ზე მეტი სტუდენტია დარეგისტრირებული.

დავაკვირდეთ მოთხოვნის ტანს, აგრეგატული ფუნქცია ჩადის HAVING პირობაში, ხოლო ყველა სვეტი რაც აგრეგატულის გარდა გვაქვს მოთხოვნის ჩამონათვალში უნდა ჩავიტანოთ GROUP BY ოპერატორში:



ORDER BY პირობა

SQL Server-ში, ORDER BY პუნქტი გამოიყენება SELECT მოთხოვნაში, რათა დაალაგოს შედეგი ერთი ან მეტი სვეტის ზრდადობის ან კლებადობის მიხედვით. სინტაქსი:

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

```
[WHERE]
[GROUP BY]
[HAVING]
[ORDER BY column(s) [ASC|DESC]]
```

- ORDER BY პირობა გამოიყენება მოთხოვნის შედეგის ერთ ან რამდენიმე სვეტის მნიშვნელობების მიხედვით დასაგებლად. მაგაგალითად სტუდენტების ცხრილი დალაგდეს სტუდენტების გვარების მიხედვით, ხოლო იმ შემთხვევაში თუ გვარი განმეორდება გამოვიყენოთ მრავალდონიანი სორტირება და იდენტური ვერის მქონე სტუდენტები დავალაგოთ სახელის მიხედვით.
- ORDER BY პუნქტი უნდა მოდიოდეს WHERE, GROUP BY და HAVING პუნქტის შემდეგ, მოთხოვნაში მათი არსებობის შემთხვევაში.
- გამოიყენეთ ASC ან DESC სვეტის სახელის შემდეგ დალაგების თანმიმდევრობის შესაბამისად ზრდადობით ან კლებადობით განსასაზღვრად. ნაგულისხმევად, ORDER BY პუნქტი ახარისხებს ჩანაწერებს ზრდადობის მიხედვით.

```
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [S].[DirectionId]
        FROM [dbo].[Students] AS [S]
         ORDER BY [S].[StudentLastName]
```

დაალაგებს სტუდენტების ცხრილს, სტუდენტთა გვარის სვეტის მიხედვით, ზრდადობით. ხოლო

```
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [S].[DirectionId]
        FROM [dbo].[Students] AS [S]
         ORDER BY [S].[StudentLastName] desc
```

მოახდენს ცხრილის დალაგებას სტუდენტთა გვარის სვეტის მიხედვით კლებადობით.
დალაგება შეიძლება მოხდეს რამოდენიმე სვეტის მიხედვით, დავალაგოთ ჯერ გვარის ხოლო შემდეგ კი სახელის სვეტის მიხედვით (მრავალდონიანი სორტირება)

```
SELECT [S].[StudentId],
       [S].[StudentName],
```

```
        [S].[StudentLastName],
        [S].[Email],
        [S].[DirectionId]
         FROM [dbo].[Students] AS [S]
          ORDER BY [S].[StudentLastName] ASC, [S].[StudentName] ASC
```

ასევე დალაგების ფუნქცია შეიძლება გამოყენებული იყოს დაჯგუფებულ მონაცემებზე:

```
SELECT
[D].[DirectionName],
COUNT([S].[StudentId]) AS NumberOfStudents
FROM [dbo].[Students]AS [S],[dbo].[Directions] AS [D]
WHERE [S].[DirectionId]=[D].[DirectionId]
 GROUP BY [D].[DirectionName]
 ORDER BY [D].[DirectionName] asc
```

აღნიშნულ შემთხვევაშიც დალაგების პარამეტრად შეიძლება გამოვიყენოთ მხოლოდ GROUP BY ოპერატორის პარამეტრად მდგომი სვეტი/სვეტები. და დავალაგოთ შედეგი მიმართულებების სახელების ზრდადობის მიხედვით:

```
SELECT
[D].[DirectionName],
COUNT([S].[StudentId]) AS NumberOfStudents
FROM [dbo].[Students]AS [S],[dbo].[Directions] AS [D]
WHERE [S].[DirectionId]=[D].[DirectionId]
 GROUP BY [D].[DirectionName]
 ORDER BY [D].[DirectionName] asc
```

Results | Messages

DirectionName	NumberOfStudents
ინფორმაციული სისტემები	3
ინფორმაციული ტექნოლოგიები	2
კომპიუტერული მეცნიერება	5

INNER JOIN მოთხოვნა

INNER JOIN მოთხოვნა გამოიყენება ორი ან მეტი ცხრილიდან შესატყვისი ჩანაწერების მოსაძიებლად, მითითებული პირობის საფუძველზე.
სინტაქსი:

```
SELECT table1.column_name(s), table2.column_name(s)
FROM table1
INNER JOIN table2
```

```
ON table1.column_name = table2.column_name;
```

განვიხილოთ კვლავ სტუდენტისა და მიმართულების ცხრილები:

	StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1	1000	ერეკლე	ბურჯაძე	aaa@gmail.com	1	NULL
2	1001	ნინო	ჭურდაძე	aaa@gmail.com	1	NULL
3	1002	ანნა	გაგუა	aaa@gmail.com	2	NULL
4	1003	გიგო	ჰაიკიძე	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

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

თითოეულ სტუდენს აქვს მინიჭებული ვალიდური მიმართულების ნომერი. დავწეროთ მოთხოვნა, რომელიც აიღებს ჩანაწერებს ორივე ცხრილიდან , თითოეული სტუდენტისთვის ნახავს რა მიმართულების ნომერია მინიჭებული, შემდეგ მიმართულების ნომრის მიხედვით გადავა და მიმართულებების ცხრილში მოიძიებს შესაბამის მიმართულებას, აიღებს მის დასახელებას და მიუწერს სტუდენტს. ფაქტია რომ მოჭიდების, ანუ გადაბმის ველი ამ ორ ცხრილს შორის იქნებს მიმართულების ნომერი. მივიღებთ მოთხოვნას:

```
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [S].[DirectionId],
       [D].[DirectionName]
FROM [dbo].[Students] AS [S]
    INNER JOIN [dbo].[Directions] AS [D]
        ON [D].[DirectionId] = [S].[DirectionId];
```

ორი ცხრილი შეუერთდა ერთმანეთს [D].[DirectionId] = [S].[DirectionId] პირობით, და ამოიკრიბა შედეგში ჩანაწერები რომლებიც აკმაყოფილებდა აღნიშნულ პირობას. ხოლო სტრიქონები სადაც DirectionId იყო NULL ან არ ჰქონდა შესაბამისი ჩანაწერი ორივე ცხრილში, უბრალოდ გამოიტოვება შედეგში.

```sql
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [S].[DirectionId],
       [D].[DirectionName]
FROM [dbo].[Students] AS [S]
    INNER JOIN [dbo].[Directions] AS [D]
        ON [D].[DirectionId] = [S].[DirectionId];
```

%

Results | Messages

StudentId	StudentName	StudentLastName	Email	DirectionId	DirectionName
1000	ერეკლე	ბურკაძე	aaa@gmail.com	1	კომპიუტერული მეცნიერება
1001	ნინო	ჭურდაძე	aaa@gmail.com	1	კომპიუტერული მეცნიერება
1002	ანნა	გაგუა	aaa@gmail.com	2	ინფორმაციული სისტემები
1003	გიგო	პაიკიძე	aaa@gmail.com	3	ინფორმაციული ტექნოლოგიები
1004	ნუცა	ურიდია	aaa@gmail.com	1	კომპიუტერული მეცნიერება
1005	მაკა	ჯოჯუა	aaa@gmail.com	2	ინფორმაციული სისტემები
1006	გიორგი	სალია	aaa@gmail.com	2	ინფორმაციული სისტემები
1007	დავითი	ზვიადაძე	aaa@gmail.com	3	ინფორმაციული ტექნოლოგიები
1008	ნიკა	ცინცაძე	aaa@gmail.com	1	კომპიუტერული მეცნიერება
1009	ლელა	დანელია	aaa@gmail.com	1	კომპიუტერული მეცნიერება

შიდა INNER შეერთების დროს არ აქვს მნიშვნელობა რომელი ცხრილი იქნება პირველი და რომელი მეორე, შედეგი იქნება უცვლელი ვინაიდან მარტივად რომ ვთქვათ ცხრილებიდან ვიღებთ ჩანაწერების „თანაკვეთას“.
ასევე თუ დავწერთ მხოლოდ JOIN სიტყვას მაინც გაჩმების პრინციპით იგულისხმება INNER JOIN პირობა.

```sql
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [S].[DirectionId],
       [D].[DirectionName]
FROM [dbo].[Directions] AS [D]
     JOIN [dbo].[Students] AS [S]
        ON [D].[DirectionId] = [S].[DirectionId]
```

და მეტიც მსგავში შიდა შეერთების მაგალითები ჩვენ უკვე დავწერეთ WHERE პირობის გამოყენებით:

```sql
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
```

```sql
       [S].[Email],
       [S].[DirectionId],
       [D].[DirectionName]
FROM [dbo].[Directions] AS [D],
     [dbo].[Students] AS [S]
WHERE [D].[DirectionId] = [S].[DirectionId];
```

დავწეროთ მოთხოვნა სამი ცხრილის შეერთებით, დამატებით გაოვიტანოთ სტუდენტის სახცოვრებელი ქალაქი და ტელეფონი:

```sql
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [SD].[Phone],
       [SD].[City],
       [S].[DirectionId],
       [D].[DirectionName]
FROM [dbo].[Directions] AS [D]
    INNER JOIN [dbo].[Students] AS [S]
        ON [D].[DirectionId] = [S].[DirectionId]
    INNER JOIN [dbo].[StudentDetails] AS [SD]
        ON [SD].[StudentId] = [S].[StudentId];
```

```sql
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [SD].[Phone],
       [SD].[City],
       [S].[DirectionId],
       [D].[DirectionName]
FROM [dbo].[Directions] AS [D]
    INNER JOIN [dbo].[Students] AS [S]
        ON [D].[DirectionId] = [S].[DirectionId]
    INNER JOIN [dbo].[StudentDetails] AS [SD]
        ON [SD].[StudentId] = [S].[StudentId]
```

Results | Messages

StudentId	StudentName	StudentLastName	Email	Phone	City	DirectionId	DirectionName
1001	ნინო	ჭურდაძე	aaa@gmail.com	577324543	ბათუმი	1	კომპიუტერული მეცნიერება
1000	ერეკლე	ბურჯაძე	aaa@gmail.com	578464543	თელავი	1	კომპიუტერული მეცნიერება
1002	ანნა	გაგუა	aaa@gmail.com	578324543	ჭუთაისი	2	ინფორმაციული სისტემები
1003	გიგო	პაიჭიძე	aaa@gmail.com	57731243	თელავი	3	ინფორმაციული ტექნოლოგიები
1005	მაკა	ჯოჯუა	aaa@gmail.com	577324543	ბათუმი	2	ინფორმაციული სისტემები
1007	დავითი	ზვიადაძე	aaa@gmail.com	571224543	თბილისი	3	ინფორმაციული ტექნოლოგიები
1009	ლელა	დანელია	aaa@gmail.com	599324543	თბილისი	1	კომპიუტერული მეცნიერება

თუ დავაკვირდებით მოთხოვნას ვნახავთ რომ შემცირდა სტუდენტთა რაოდენობა, და შედეგში აღარ დაბრუნდა 1004, 1006 და სხვა სტუდენტები.
რა შეიძლება იყოს აღნიშნულის გამომწვევი?
როგორც ზემოთ უკვე ვთქვით, შიდა შეერთებების დროს ჩვენ ვიღებთ ცხრილების თანაკვეთას, ანუ ჩანაწერებს რომლებიც საერთოა ორივე ცხრილისთვის. `[dbo].[StudentDetails]` ცხრილის გადაბმა მოხდა `ON [SD].[StudentId] = [S].[StudentId]` პირობით რაც გულისხმობდა რომ ყოველი სტუდენტის ნომრისთვის `Students` ცხრილიდან, ჩვენ უნდა გვქონოდა შესაბამისი ჩანაწერი `StudentDetails` ცხრილში. წინააღმდეგ შემთვევაში მოხდებოდა ჩანაწერის გამოტოვება მთხოვნიდან. თუ დავაკვირდებით ამ ორი ცხრილის ჩანაწერებს აღმოვაჩენთ რომ გამოიტოვა სტუდენტების ზუსტად ის ჩანაწერები სადაც არ გვქონდა შესაბამისობა დეტალურ ცხრილში:

```
SELECT * FROM [dbo].[Students] AS [S]
SELECT * FROM [dbo].[StudentDetails] AS [SD]
```

%

Results | Messages

StudentId	StudentName	StudentLastName	Email	DirectionId	Password
1000	ერეკლე	ბურკაძე	aaa@gmail.com	1	NULL
1001	ნინო	ჭურდაძე	aaa@gmail.com	1	NULL
1002	ანნა	გაგუა	aaa@gmail.com	2	NULL
1003	გიგო	პაიციძე	aaa@gmail.com	3	NULL
1004	ნუცა	ურიდია	aaa@gmail.com	1	NULL
1005	მაკა	ჯოჯუა	aaa@gmail.com	2	NULL
1006	გიორგი	სალია	aaa@gmail.com	2	NULL
1007	დავითი	ზვიადაძე	aaa@gmail.com	3	NULL
1008	ნიკა	ცინცაძე	aaa@gmail.com	1	NULL
1009	ლელა	დანელია	aaa@gmail.com	1	NULL

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

გამოდის „თანაკვეთის“ პირობის არსებობის გამო მოთხოვნიდან დავკარგე სტუდენტები რომლებსაც არ ქონდათ შევსებული დეტალური ინფორმაცია. შესაბამისად მსგავსი შეერთება არ გამოგვადგება, თუკი არ გვინდა გამოვტოვოთ აღნიშნული სტუდენტები.

LEFT JOIN მოთხოვნა

LEFT JOIN არის შეერთების ტიპი, სადაც მნიშვნელოვანია რომელი ცხრილი წერია JOIN სიტყვის მარცხნივ და რომელი მარჯვნივ, ვინდაიდან იგი აბრუნებს ყველა ჩანაწერს მარცხენა ცხრილიდან და შესატყვის ჩანაწერებს მარჯვენა ცხრილიდან. აქ, მარცხენა ცხრილი ნიშნავს ცხრილს, რომელიც დგას მარცხენა მხარეს ან მოთხოვნაში "LEFT JOIN" ფრაზის წინ, ხოლო მარჯვენა ცხრილი მიუთითებს ცხრილს, რომელიც დგას მარჯვენა მხარეს "LEFT JOIN" ფრაზის შემდეგ. აღიშნულ შემთხვევაში მარცხენა ცხრილი არის პრიორიტეტული, იგი მოთხოვნაში მოდის სრულად, ხოლო მარჯვენა ცხრილიდან ივესება შესაბამისობები აღნიშნული ცხრილისთვის. ხოლო შეუსაბამო ჩანაწერისთვის მარჯვენა ცხრილიდან აბრუნებს NULL-ს. ჩვენს შემთხვევაში ზემოთ არსებული პრობლემის საპასუხოდ, თუ შიდა შეერტებას (INNER JOIN) ჩავანაცვლებთ (LEFT JOIN) მარცხენა შეერთებით, გამოვა რომ მარცხენა, სტუდენტების ცხრილი წამოვა შედეგში სრულად, ხოლო მარჯვენა დეტალურის ცხრილიდან შეივსება მხოლოდ შესაბამისი ჩანაწერები. რომელ სტუდენტსაც არ ექნება დეტაურ ცხრილში შესაბამისი ინფორმაცია, საჭირო ველები შეევსება NULL-ებით.

```
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [SD].[Phone],
       [SD].[City],
       [S].[DirectionId]
FROM [dbo].[Students] AS [S]
    LEFT JOIN [dbo].[StudentDetails] AS [SD]
        ON [SD].[StudentId] = [S].[StudentId];
```

კოდის შესრულების შედეგად, მივიღებთ შედეგს:

```
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [SD].[Phone],
       [SD].[City],
       [S].[DirectionId]
FROM [dbo].[Students] AS [S]
    LEFT JOIN [dbo].[StudentDetails] AS [SD]
        ON [SD].[StudentId] = [S].[StudentId];
```

Results | Messages

StudentId	StudentName	StudentLastName	Email	Phone	City	DirectionId
1000	ერეკლე	ბურჯაძე	aaa@gmail.com	578464543	თელავი	1
1001	ნინო	ჭურდაძე	aaa@gmail.com	577324543	ბათუმი	1
1002	ანნა	გაგუა	aaa@gmail.com	578324543	ჭუთაისი	2
1003	გიგო	პაიციძე	aaa@gmail.com	57731243	თელავი	3
1004	ნუცა	ურიდია	aaa@gmail.com	NULL	NULL	1
1005	მაკა	ჯოჯუა	aaa@gmail.com	577324543	ბათუმი	2
1006	გიორგი	სალია	aaa@gmail.com	NULL	NULL	2
1007	დავითი	ზვიადაძე	aaa@gmail.com	571224543	თბილისი	3
1008	ნიკა	ცინცაძე	aaa@gmail.com	NULL	NULL	1
1009	ლელა	დანელია	aaa@gmail.com	599324543	თბილისი	1

როგორც ხედავთ მივიღეთ სტუდენტთა სრული სია, ხოლო ტელეფონისა და ქალაქის ველები იმ სტუდენტებისთვის რომლებსაც არ აქვთ დეტალური შევსებულია NULL მნიშვნელობით.

ზოგიერთ მონაცემთა ბაზაში მარჯვენა შეერთებას უწოდებენ LEFT OUTER JOIN.

დავუკვირდეთ მოთხოვნას, `StudentId` გადაბმის ველი გვხვდება ორივე ცხრილში, თუმცა ჩვენ მოთხოვნაში მას ვიყენებთ მარცხენა, სტუდენტების ცხრილიდან. ჩავამატოთ იგივე ველი დეტალურის ცხრილიდან და უფრო თვალსაჩინო იქნება რა მომენტში მოხდა ამ ორი ცხრილის გადაბმა და რა მომენტში ვერა:

```
SELECT [S].[StudentId],
       [SD].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [SD].[Phone],
       [SD].[City],
       [S].[DirectionId]
FROM [dbo].[Students] AS [S]
    LEFT JOIN [dbo].[StudentDetails] AS [SD]
        ON [SD].[StudentId] = [S].[StudentId];
```

შედეგად მივიღებთ:

```sql
SELECT [S].[StudentId],
       [SD].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [S].[Email],
       [SD].[Phone],
       [SD].[City],
       [S].[DirectionId]
FROM [dbo].[Students] AS [S]
    LEFT JOIN [dbo].[StudentDetails] AS [SD]
        ON [SD].[StudentId] = [S].[StudentId];
```

Results | Messages

StudentId	StudentId	StudentName	StudentLastName	Email	Phone	City	DirectionId
1000	1000	ერეკლე	ბურჯაძე	aaa@gmail.com	578464543	თელავი	1
1001	1001	ნინო	ჭურდაძე	aaa@gmail.com	577324543	ბათუმი	1
1002	1002	ანნა	გაგუა	aaa@gmail.com	578324543	ჭუთაისი	2
1003	1003	გიგო	ჰაიციძე	aaa@gmail.com	57731243	თელავი	3
1004	NULL	ნუცა	ურიდია	aaa@gmail.com	NULL	NULL	1
1005	1005	მაკა	ჯოჯუა	aaa@gmail.com	577324543	ბათუმი	2
1006	NULL	გიორგი	სალია	aaa@gmail.com	NULL	NULL	2
1007	1007	დავითი	ზვიადაძე	aaa@gmail.com	571224543	თბილისი	3
1008	NULL	ნიკა	ცინცაძე	aaa@gmail.com	NULL	NULL	1
1009	1009	ლელა	დანელია	aaa@gmail.com	599324543	თბილისი	1

RIGHT JOIN მოთხოვნა

RIGHT JOIN წინამორბედის მსგავსად არის შეერთების ტიპი, სადაც მნიშვნელოვანია რომელი ცხრილი წერია JOIN სიტყვის მარცხნივ და რომელი მარჯვნივ, ვინდაიდან იგი აბრუნებს ყველა ჩანაწერს მარჯვენა ცხრილიდან და შესატყვის ჩანაწერებს მარცხენადან. აღიშნულ შემთხვევაში უკვე მარჯვენა ცხრილი არის პრიორიტეტული, იგი მოთხოვნაში მოდის სრულად, ხოლო მარცხენა ცხრილიდან ივესბა შესაბამისი ველები აღნიშნული ცხრილისთვის. ხოლო შეუსაბამო ჩანაწერისთვის მარცხენა ცხრილიდან აბრუნებს NULL-ს.

დავწეროთ მოთხოვნა რომელიც დააბრუნებს ყველა სტუდენტს და საგნებს რომლებზეც რეგისტრირებული არიან ისინი:

```sql
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
         [SS].[StudentId],
       [SS].[SubjectId],
       [SS].[RegisterDate],
         [SJ].[SubjectId],
         [SJ].[SubjectName]
```

```
FROM [dbo].[Students] AS [S]
    LEFT JOIN [dbo].[StudentSubjects] AS [SS] ON [SS].[StudentId] =
[S].[StudentId]
     RIGHT JOIN [dbo].[Subjects] AS [SJ] ON [SJ].[SubjectId] =
[SS].[SubjectId]
```

დავიწყეთ სტუდენტების ცხრილიდან და გადავედით სტუდენტების და საგნების შემაერთებელ ცხრილზე მარცხენა შეერთებით, რათა აგვეღო ყველა სტუდენტის რეგისტრაციის მონაცემი, როგორც ხედავთ სრულადაა შევსებული ყველა სტუდენტზე საგნები, რაც ნიშნავს რომ ყველა სტუდენტი არის რეგისტრირებული ერთ საგანზე მაინც, შემდეგ კი შუალედური ცხრილიდან მარჯვენა შეერთებით გადავედით საგნების ცხრილზე, რათა პრიორიტეტი მიგვენიჭებინა მარჯვენა ცხრილითვის და წამოგვეღო ყველა საგანი იმისდა მიუხედავად იყო თუ არა მასზე ვინმე რეგისტრირებული. ამ შემთხვევაში აღმოჩნდა რომ ქსელებზე არავინ არაა რეგისტრირებული.

```
SELECT [S].[StudentId],
       [S].[StudentName],
       [S].[StudentLastName],
       [SS].[StudentId],
       [SS].[SubjectId],
       [SS].[RegisterDate],
       [SJ].[SubjectId],
       [SJ].[SubjectName]
FROM [dbo].[Students] AS [S]
    LEFT JOIN [dbo].[StudentSubjects] AS [SS] ON [SS].[StudentId] = [S].[StudentId]
    RIGHT JOIN [dbo].[Subjects] AS [SJ] ON [SJ].[SubjectId] = [SS].[SubjectId]
```

Results | Messages

StudentId	StudentName	StudentLastName	StudentId	SubjectId	RegisterDate	SubjectId	SubjectName
1000	ერეკლე	ბურჭაძე	1000	1	2023-04-10 11:08:09.497	1	ნეირონული ქსელები
1001	ნინო	ჭურდაძე	1001	1	2023-04-10 11:08:09.497	1	ნეირონული ქსელები
1007	დავითი	ზვიადაძე	1007	1	2023-04-10 11:08:09.497	1	ნეირონული ქსელები
1001	ნინო	ჭურდაძე	1001	2	2023-04-10 11:08:09.497	2	პროექტი
1002	ანნა	გაგუა	1002	2	2023-04-10 11:08:09.497	2	პროექტი
1003	გიგო	პაიკიძე	1003	2	2023-04-10 11:08:09.497	2	პროექტი
1007	დავითი	ზვიადაძე	1007	2	2023-04-10 11:08:09.497	2	პროექტი
1009	ლელა	დანელია	1009	2	2023-04-10 11:08:09.497	2	პროექტი
1000	ერეკლე	ბურჭაძე	1000	3	2023-04-10 11:08:09.497	3	ალგორითმები
1001	ნინო	ჭურდაძე	1001	3	2023-04-10 11:08:09.497	3	ალგორითმები
1005	მაკა	ჯოჯუა	1005	3	2023-04-10 11:08:09.497	3	ალგორითმები
NULL	NULL	NULL	NULL	NULL	NULL	4	ქსელები
1000	ერეკლე	ბურჭაძე	1000	5	2023-04-10 11:08:09.497	5	NLP
1003	გიგო	პაიკიძე	1003	5	2023-04-10 11:08:09.497	5	NLP
1007	დავითი	ზვიადაძე	1007	5	2023-04-10 11:08:09.497	5	NLP
1001	ნინო	ჭურდაძე	1001	6	2023-04-10 11:08:09.497	6	დაპროგრამება
1005	მაკა	ჯოჯუა	1005	6	2023-04-10 11:08:09.497	6	დაპროგრამება
1002	ანნა	გაგუა	1002	7	2023-04-10 11:08:09.497	7	მანქანური სწავლება
1003	გიგო	პაიკიძე	1003	7	2023-04-10 11:08:09.497	7	მანქანური სწავლება

FULL JOIN მოთხოვნა

FULL JOIN აბრუნებს ყველა ჩანაწერს ყველა მითითებული ცხრილიდან. ნებისმიერი შეუსაბამო ჩანაწერისთვის მარჯვენა ცხრილიდან ინება იგი თუ მარცხენა ცარიელი ველები ივსება NULL-ით.

ზოგიერთ მონაცემთა ბაზაში FULL JOIN ეწოდება FULL OUTER JOIN. მას შეუძლია დააბრუნოს ძალიან დიდი შედეგი, ვინაიდან იგი აბრუნებს ყველა სტრიქონს ყველა ცხრილიდან.

წავუშალოთ საგან კავშირი მიმართულებასთან (FOREIGN KEY) აღნიშნული საშუალებას მოგვცემს საგანი დავარეგისტრიროთ არარსებულ მიმართულებაზე. ამის შემდეგ დავამატოთ ერთი ახალი მიმართულება, რომელზეც არ გვექნება საგანი, და ორიც ახალი საგანი არარსებულ მიმართულებაზე.

```
INSERT INTO [dbo].[Directions]
([DirectionName])
VALUES
(N'ფიზიკა');

insert into Subjects
values
 (N'ლიტერატურა',6, 10)
,(N'ფერწერა',6,5)
```

დავწეროთ სრული შეერთების მოთხოვნა:

```
SELECT [S].[SubjectId],
       [S].[SubjectName],
       [S].[DirectionId],
       [D].[DirectionId],
       [D].[DirectionName]
        FROM [dbo].[Subjects] AS [S]
FULL JOIN [dbo].[Directions] AS [D]
ON [D].[DirectionId] = [S].[DirectionId]
```

შედეგად მივიღებთ ყველა ველს, ორივე ცხრილიდან, მარცხენა ცხრილი სრულად, თუ კი რომელიმე ჩანაწერს არ ჰქონდა შესაბამისობა მარჯვენა ცხრილიდან შესაბამისი ველები შეივსო NULL-ით. ანალოგიურად სრულად ვიღებთ მარჯვენა ცხრილსაც და თუ კი რომელიმე ჩანაწერს არ ჰქონდა შესაბამისობა მარცხენა ცხრილიდან შესაბამისი ველები შეივსო NULL-ით.

```sql
SELECT [S].[SubjectId],
       [S].[SubjectName],
       [S].[DirectionId],
       [D].[DirectionId],
       [D].[DirectionName]
       FROM [dbo].[Subjects] AS [S]
FULL JOIN [dbo].[Directions] AS [D]
ON [D].[DirectionId] = [S].[DirectionId]
```

Results | Messages

SubjectId	SubjectName	DirectionId	DirectionId	DirectionName
1	ნეირონული ქსელები	3	3	ინფორმაციული ტექნოლოგიები
2	პროექტი	1	1	კომპიუტერული მეცნიერება
3	ალგორითმები	2	2	ინფორმაციული სისტემები
4	ქსელები	3	3	ინფორმაციული ტექნოლოგიები
5	NLP	1	1	კომპიუტერული მეცნიერება
6	დაპროგრამება	2	2	ინფორმაციული სისტემები
7	მანქანური სწავლება	1	1	კომპიუტერული მეცნიერება
8	ლიტერატურა	6	NULL	NULL
9	ფერწერა	6	NULL	NULL
NULL	NULL	NULL	4	ფიზიკა

სურათზე ვხედავთ რომ საგნებს (ლიტერატურა და ფერწერა) რომლებსაც არ ქონდათ შესაბამისი მიმართულება და ფიზიკის მიმართულებას რომელსაც არ ჰქონდა საგნები შესაბამისი ველები შეევსო NULL-ით.

CROSS JOIN მოთხოვნა

ჯვარედინი შეერთების ტიპი, როდესაც ორ ცხრილს არ აქვს საერთო ველი და თითეულ სვეტს ერთი ცხრილიდან შეესაბამება ყველა სვეტი მეორედან, ვინაიდან არ არსებობს გადაბმის On პირობა:

```
SELECT [S].[SubjectId],
       [S].[SubjectName],
       [S].[DirectionId],
       [D].[DirectionId],
       [D].[DirectionName]
        FROM [dbo].[Subjects] AS [S]
CROSS JOIN [dbo].[Directions] AS [D]
```

თუკი გვაქვს 4 მიმართულება და 9 საგანი შედეგად ვიღებთ 4x9=36 სტრიქონს.

```
SELECT [S].[SubjectId],
       [S].[SubjectName],
       [S].[DirectionId],
       [D].[DirectionId],
       [D].[DirectionName]
       FROM [dbo].[Subjects] AS [S]
CROSS JOIN [dbo].[Directions] AS [D]
```

137 %

Results | Messages

	SubjectId	SubjectName	DirectionId	DirectionId	DirectionName
1	1	ნეირონული ქსელები	3	1	კომპიუტერული მეცნიერება
2	2	პროექტი	1	1	კომპიუტერული მეცნიერება
3	3	ალგორითმები	2	1	კომპიუტერული მეცნიერება
4	4	ქსელები	3	1	კომპიუტერული მეცნიერება
5	5	NLP	1	1	კომპიუტერული მეცნიერება
6	6	დაპროგრამება	2	1	კომპიუტერული მეცნიერება
7	7	მანქანური სწავლება	1	1	კომპიუტერული მეცნიერება
8	8	ლიტერატურა	6	1	კომპიუტერული მეცნიერება
9	9	ფერწერა	6	1	კომპიუტერული მეცნიერება
10	1	ნეირონული ქსელები	3	2	ინფორმაციული სისტემები
11	2	პროექტი	1	2	ინფორმაციული სისტემები
12	3	ალგორითმები	2	2	ინფორმაციული სისტემები
13	4	ქსელები	3	2	ინფორმაციული სისტემები
14	5	NLP	1	2	ინფორმაციული სისტემები
15	6	დაპროგრამება	2	2	ინფორმაციული სისტემები
16	7	მანქანური სწავლება	1	2	ინფორმაციული სისტემები
17	8	ლიტერატურა	6	2	ინფორმაციული სისტემები
18	9	ფერწერა	6	2	ინფორმაციული სისტემები
19	1	ნეირონული ქსელები	3	3	ინფორმაციული ტექნოლოგიები
20	2	პროექტი	1	3	ინფორმაციული ტექნოლოგიები
21	3	ალგორითმები	2	3	ინფორმაციული ტექნოლოგიები
22	4	ქსელები	3	3	ინფორმაციული ტექნოლოგიები
23	5	NLP	1	3	ინფორმაციული ტექნოლოგიები
24	6	დაპროგრამება	2	3	ინფორმაციული ტექნოლოგიები
25	7	მანქანური სწავლება	1	3	ინფორმაციული ტექნოლოგიები
26	8	ლიტერატურა	6	3	ინფორმაციული ტექნოლოგიები
27	9	ფერწერა	6	3	ინფორმაციული ტექნოლოგიები
28	1	ნეირონული ქსელები	3	4	ფიზიკა
29	2	პროექტი	1	4	ფიზიკა
30	3	ალგორითმები	2	4	ფიზიკა
31	4	ქსელები	3	4	ფიზიკა
32	5	NLP	1	4	ფიზიკა
33	6	დაპროგრამება	2	4	ფიზიკა
34	7	მანქანური სწავლება	1	4	ფიზიკა
35	8	ლიტერატურა	6	4	ფიზიკა
36	9	ფერწერა	6	4	ფიზიკა

აღნიშნული შედეგი მიიღებოდა პირდაპირ ცხრილების მძიმით გამოყოფისას From პირობაში Where -ის გარეშე.

```
SELECT [S].[SubjectId],
       [S].[SubjectName],
       [S].[DirectionId],
       [D].[DirectionId],
       [D].[DirectionName]
        FROM [dbo].[Subjects] AS [S],[dbo].[Directions] AS [D]
```

0

```sql
CREATE DATABASE [Faculty];
GO

USE [Faculty];

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]
(
```

```sql
    [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]
                )
);


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

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);

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);


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);


INSERT INTO [dbo].[Lecturers]
(
    [LecturerName],
    [LecturerLastName],
    [Phone],
    [Email]
)
VALUES
(   N'თამარ', N'იარაული', '577238432', 't.ijarauli@gmail.com' ),

(   N'გიორგი', N'სამხარაძე', '598723432', 'g.samkharadze@gmail.com' ),
(   N'ლევან', N'ცაბაძე', '585237433', 'l.tsabadze@gmail.com' ),
(   N'ია', N'კილასონია', '511986098', 'i.kilasonia@gmail.com' ),
(   N'თამთა', N'ბერიძე', '5772301293', 't.beridze@gmail.com' ),
(   N'დავით', N'ბურდული', '555128712', 'd.burduli@gmail.com' )
```

```sql
INSERT INTO [dbo].[SubjectLecturers]
(
    [SubjectId],
    [LecturerId]
)
VALUES
(   1, 1  ),
(   1, 3  ),
(   2, 2  ),
(   4, 6  ),
(   5, 3  ),
(   6, 4  ),
(   7, 4  ),
(   7, 5  )

INSERT INTO [dbo].[StudentSubjects]
(
    [StudentId],
    [SubjectId],
    [RegisterDate],
    [IsPassed]
)
VALUES
(   1000, 1,  GETDATE(), 0),
(   1000, 3,  GETDATE(), 0),
(   1000, 5,  GETDATE(), 0),
(   1001, 2,  GETDATE(), 0),
(   1001, 3,  GETDATE(), 0),
(   1001, 6,  GETDATE(), 0),
(   1001, 1,  GETDATE(), 0),
(   1002, 2,  GETDATE(), 0),
(   1002, 7,  GETDATE(), 0),
(   1003, 7,  GETDATE(), 0),
(   1003, 5,  GETDATE(), 0),
(   1003, 2,  GETDATE(), 0),
(   1005, 3,  GETDATE(), 0),
(   1005, 6,  GETDATE(), 0),
(   1007, 1,  GETDATE(), 0),
(   1007, 2,  GETDATE(), 0),
(   1007, 5,  GETDATE(), 0),
(   1009, 2,  GETDATE(), 0)
```