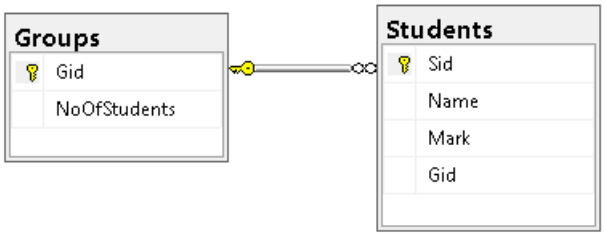
Examples




```
USE Lab2E
go

CREATE TABLE Groups(
Gid int primary key,
NoOfStudents int)

CREATE TABLE Students(
Sid int primary key,
Name varchar(50),
Mark float,
Gid int foreign key references Groups(Gid))

insert into Groups values (921, 28), (922, 27), (923, 28)
insert into Students values (1, 'Paul', 10, 921), (2, 'Cristi', 9, 921),
(3, 'Tania', 8, 923), (4, 'Daniel', 9, 922)

select * from Groups
select * from Students
```

Results | Messages

	Gid	NoOfStudents
1	921	28
2	922	27
3	923	28

	Sid	Name	Mark	Gid
1	1	Paul	10	921
2	2	Cristi	9	921
3	3	Tania	8	923
4	4	Dan...	9	922

-- a. 2 queries with the union operation; use UNION [ALL] and OR;

```
-- a. the students that have the name starting with T and has at least 2 letters OR have the
mark greater or equal than 10.
SELECT *
FROM Students
WHERE Name like 'T_%' OR Mark>=10
-- equivalent
SELECT *
FROM Students
WHERE Name like 'T_%'
UNION -- OR - the unique values
SELECT *
FROM Students
WHERE Mark>=10
--
SELECT s1.Sid
FROM Students s1
WHERE Name like 'T_%'
UNION
SELECT s2.Sid
FROM Students s2
WHERE Mark>=10
```

Results | Messages

	Sid	Name	Mark	Gid
1	1	Paul	10	921
2	3	Tania	8	923

	Sid	Name	Mark	Gid
1	1	Paul	10	921
2	3	Tania	8	923

```
SELECT s1.Sid, s1.Name
FROM Students s1
```

```sql
WHERE Name like 'T_%'
UNION
SELECT s2.Sid
FROM Students s2
WHERE Mark>=10
```

```
Msg 205, Level 16, State 1, Line 41
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.
```

```sql
SELECT s1.Name
FROM Students s1
WHERE Name like 'T_%'
ORDER BY s1.Name
UNION
SELECT s2.Name
FROM Students s2
WHERE Mark>=10
```

```
Msg 156, Level 15, State 1, Line 55
Incorrect syntax near the keyword 'UNION'.
```

```sql
SELECT s1.Name
FROM Students s1
WHERE Name like 'T_%'
UNION
SELECT s2.Name
FROM Students s2
WHERE Mark>=10
ORDER BY s1.Name
```

Results | Messages

	Name
1	Paul
2	Tania

```sql
SELECT s1.Name
FROM Students s1
WHERE Name like 'T_%'
UNION
SELECT s2.Name
FROM Students s2
WHERE Mark>=10
ORDER BY s1.Name DESC
```

Results | Messages

	Name
1	Tania
2	Paul

```sql
SELECT s1.Name
FROM Students s1
WHERE Name like 'T_%'
UNION ALL -- all values, including the duplicates
SELECT s2.Name
FROM Students s2
WHERE Mark>=10
ORDER BY s1.Name
```

Results | Messages

	Name
1	Paul
2	Tania

-- b. 2 queries with the intersection operation; use INTERSECT and IN;

```sql
-- b. the students that have the name starting with T and has at least 2 letters AND have the mark greater or equal than 10.
SELECT s1.Name
FROM Students s1
WHERE Name like 'T_%'
INTERSECT -- AND
SELECT s2.Name
FROM Students s2
WHERE Mark>=10
ORDER BY s1.Name

select * from Students
```

Results | Messages

	Name

	Sid	Name	Mark	Gid
1	1	Paul	10	921
2	2	Cristi	9	921
3	3	Tania	8	923
4	4	Daniel	9	922

-- c. 2 queries with the difference operation; use EXCEPT and NOT IN;

```
-- c. the students that have the name starting with T and has at
least 2 letters BUT NOT have the mark greater or equal than 10.
SELECT s1.Name
FROM Students s1
WHERE Name like 'T_%'
EXCEPT -- IN first NOT IN second
SELECT s2.Name
FROM Students s2
WHERE Mark>=10
ORDER BY s1.Name
-- Tania
```

Results | Messages

	Name
1	Tania

d. 4 queries with INNER JOIN, LEFT JOIN, RIGHT JOIN and FULL JOIN; one query will join at least 3 tables, while another one will join at least two *many-to-many* relationships;

```
-- d. the students from each group
select * from Groups
select * from Students
```

Results | Messages

	Gid	NoOfStudents
1	921	28
2	922	27
3	923	28
4	924	27

	Sid	Name	Mark	Gid
1	1	Paul	10	921
2	2	Cristi	9	921
3	3	Tania	8	923
4	4	Daniel	9	922

```
select *
from Groups, Students
-- all combinations
```

Results | Messages

	Gid	NoOfStudents	Sid	Name	Mark	Gid
1	921	28	1	Paul	10	921
2	922	27	1	Paul	10	921
3	923	28	1	Paul	10	921
4	924	27	1	Paul	10	921
5	921	28	2	Cristi	9	921
6	922	27	2	Cristi	9	921
7	923	28	2	Cristi	9	921
8	924	27	2	Cristi	9	921
9	921	28	3	Tania	8	923
10	922	27	3	Tania	8	923
11	923	28	3	Tania	8	923
12	924	27	3	Tania	8	923
13	921	28	4	Dan...	9	922
14	922	27	4	Dan...	9	922
15	923	28	4	Dan...	9	922
16	924	27	4	Dan...	9	922

```
select *
from Groups, Students
WHERE Groups.Gid=Students.Gid
-- equivalent
select *
from Groups g, Students s
WHERE g.Gid=s.Gid
-- equivalent
```

Results | Messages

	Gid	NoOfStudents	Sid	Name	Mark	Gid
1	921	28	1	Paul	10	921
2	921	28	2	Cristi	9	921
3	923	28	3	Tania	8	923
4	922	27	4	Daniel	9	922

```sql
select *
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
```

```sql
-- LEFT OUTER JOIN
insert into Groups values(924, 27) -- (1 row(s)
affected)
select *
from Groups g LEFT OUTER JOIN Students s ON
g.Gid=s.Gid
-- RIGHT OUTER JOIN
select *
from Groups g RIGHT OUTER JOIN Students s ON
g.Gid=s.Gid
-- FULL OUTER JOIN
select *
from Groups g FULL OUTER JOIN Students s ON
g.Gid=s.Gid
```

Results | Messages

	Gid	NoOfStudents	Sid	Name	Mark	Gid
1	921	28	1	Paul	10	921
2	921	28	2	Cristi	9	921
3	922	27	4	Daniel	9	922
4	923	28	3	Tania	8	923
5	924	27	NULL	NULL	NULL	NULL

	Gid	NoOfStudents	Sid	Name	Mark	Gid
1	921	28	1	Paul	10	921
2	921	28	2	Cristi	9	921
3	923	28	3	Tania	8	923
4	922	27	4	Dan...	9	922

	Gid	NoOfStudents	Sid	Name	Mark	Gid
1	921	28	1	Paul	10	921
2	921	28	2	Cristi	9	921
3	922	27	4	Daniel	9	922
4	923	28	3	Tania	8	923
5	924	27	NULL	NULL	NULL	NULL

```sql
-- more tables
select *
from Groups g INNER JOIN Students s ON g.Gid=s.Gid
INNER JOIN Exams e ON e.
INNER JOIN
```

```
Msg 156, Level 15, State 1, Line 146
Incorrect syntax near the keyword
'INNER'.
```

e. 2 queries using the IN operator to introduce a subquery in the WHERE clause; in at least one query, the subquery should include a subquery in its own WHERE clause;
f. 2 queries using the EXISTS operator to introduce a subquery in the WHERE clause;
g. 2 queries with a subquery in the FROM clause;
-- the Groups with the students that have the Mark>5 (requirements e, f, g)

```sql
SELECT s.Gid, s.Mark
FROM Groups g INNER JOIN Students s ON g.Gid=s.Gid
WHERE Mark>5
```

```sql
SELECT s.Gid, s.Mark
FROM Students s
WHERE Mark>5 and s.Gid IN (SELECT g.Gid FROM Groups g)
```

```sql
SELECT s.Gid, s.Mark
FROM Students s
WHERE Mark>5 and EXISTS (SELECT * FROM Groups g
                         WHERE g.Gid=s.Gid)
```

```sql
SELECT A.Gid, A.Mark
FROM (SELECT g.Gid, s.Name, s.Mark
        FROM Groups g INNER JOIN Students s ON g.Gid=s.Gid
        WHERE Mark>5) A
```

Results | Messages

	Gid	Mark
1	921	10
2	921	9
3	923	8
4	922	9

h. 4 queries with the GROUP BY clause, 3 of which also contain the HAVING clause; 2 of the latter will also have a subquery in the HAVING clause; use the aggregation operators: COUNT, SUM, AVG, MIN, MAX;

```sql
-- h. the average of the marks for each groups
select g.gid
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
```

Results | Messages

	gid
1	921
2	922
3	923

```sql
select g.gid, s.Mark
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
```

```
Msg 8120, Level 16, State 1, Line 174
Column 'Students.Mark' is invalid in the select
list because it is not contained in either an
aggregate function or the GROUP BY clause.
```

```sql
select g.gid, s.Mark
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid, s.Mark
```

Results | Messages

	gid	Mark
1	921	9
2	921	10
3	922	9
4	923	8

```sql
select g.gid, AVG(s.Mark)
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
```

Results | Messages

	gid	(No column name)
1	921	9.5
2	922	9
3	923	8

```sql
select g.gid, AVG(s.Mark) AS average
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
```

Results | Messages

	gid	average
1	921	9.5
2	922	9
3	923	8

```sql
select AVG(s.Mark) AS average
from Students s
```

Results | Messages

	average
1	9

```sql
-- for each group the average if it is grater or equal than 9
```

```sql
select g.gid, AVG(s.Mark)
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
HAVING AVG(s.Mark)>=9
```

Results | Messages

	gid	(No column name)
1	921	9.5
2	922	9

```sql
select g.gid, AVG(s.Mark) as average
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
HAVING average>=9
```

```
Msg 207, Level 16, State 1, Line 205
Invalid column name 'average'.
```

```
-- for each group less than 923 the average if
it is grater or equal than 9
select g.gid, AVG(s.Mark)
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
GROUP BY g.gid
HAVING g.Gid<=923
-- OR
select g.gid, AVG(s.Mark)
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
WHERE g.Gid<=923
GROUP BY g.gid
```

Results | Messages

	gid	(No column name)
1	921	9.5
2	922	9
3	923	8

	gid	(No column name)
1	921	9.5
2	922	9
3	923	8

```
select g.gid, AVG(s.Mark)
from Groups g INNER JOIN Students s ON
g.Gid=s.Gid
WHERE AVG(s.Mark)>=9
GROUP BY g.gid
```

```
Msg 147, Level 15, State 1, Line 222
An aggregate may not appear in the WHERE clause
unless it is in a subquery contained in a
HAVING clause or a select list, and the column
being aggregated is an outer reference.
```

i. 4 queries using ANY and ALL to introduce a subquery in the WHERE clause; 2 of them should be rewritten with aggregation operators, while the other 2 should also be expressed with [NOT] IN.

<u>ANY – ALL</u>

ALL – all records check the condition

ANY – at least one record check the condition

```
-- > all  - equivalent with MAX
SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark>ALL(SELECT s1.Mark
                 FROM Students s1
                 WHERE s.Sid=s1.Sid)


SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark>(SELECT MAX(s1.Mark)
              FROM Students s1
              WHERE s.Sid=s1.Sid)
```

Results | Messages

Name	Mark

Name	Mark

```
-- any <  - equivalent with MIN
SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark<ANY(SELECT s1.Mark
                 FROM Students s1
                 WHERE s.Sid=s1.Sid)


SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark<(SELECT MIN(s1.Mark)
              FROM Students s1
              WHERE s.Sid=s1.Sid)
```

Results | Messages

Name	Mark

Name	Mark

```
-- <> all  - equivalent with NOT IN
SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark<>ALL(SELECT s1.Mark
```

```
-- any = equivalent with IN
SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark=ANY(SELECT s1.Mark
```

```
                                FROM Students s1
                                WHERE s.Sid=s1.Sid)


SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark NOT IN (SELECT s1.Mark
                                FROM Students s1
                                WHERE s.Sid=s1.Sid)
```

Results | Messages

Name	Mark

Name	Mark

```
                                FROM Students s1
                                WHERE s.Sid=s1.Sid)


SELECT s.Name, s.Mark
FROM Students s
WHERE s.Mark IN (SELECT s1.Mark
                                FROM Students s1
                                WHERE s.Sid=s1.Sid)
```

Results | Messages

	Name	Mark
1	Paul	10
2	Cristi	9
3	Tania	8
4	Daniel	9

	Name	Mark
1	Paul	10
2	Cristi	9
3	Tania	8
4	Daniel	9

You need to use:

- arithmetic expressions in the SELECT clause in at least 3 queries;

```
-- arithmetic expresions
select s.Name, (s.Mark+5 )/6
from Students s
```

Results | Messages

	Name	(No column name)
1	Paul	2.5
2	Cristi	2.33333333333333
3	Tania	2.16666666666667
4	Daniel	2.33333333333333

- conditions with AND, OR, NOT and parantheses in the WHERE clause in at least 3 queries;
- DISTINCT in at least 3 queries, ORDER BY in at least 2 queries and TOP in at least 2 queries.

```
--distinct
select DISTINCT s.Name, s.Mark
from Students s

select s.Mark
from Students s

select DISTINCT s.Mark
from Students s
```

Results | Messages

	Name	Mark
1	Cristi	9
2	Daniel	9
3	Paul	10
4	Tania	8

	Mark
1	10
2	9
3	8
4	9

	Mark
1	8
2	9
3	10

```
-- top
-- the first 2 records
select TOP 2 s.Name, s.Mark
from Students s
```

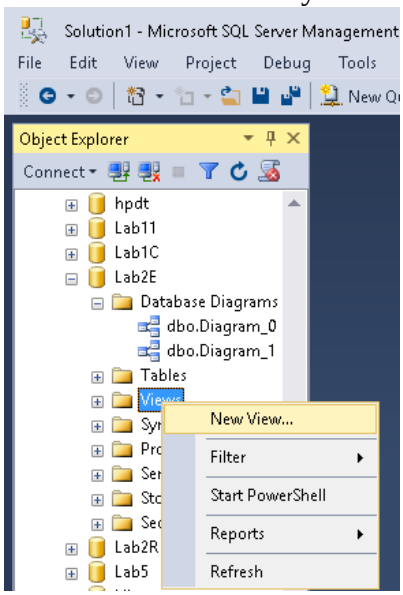
Results | Messages

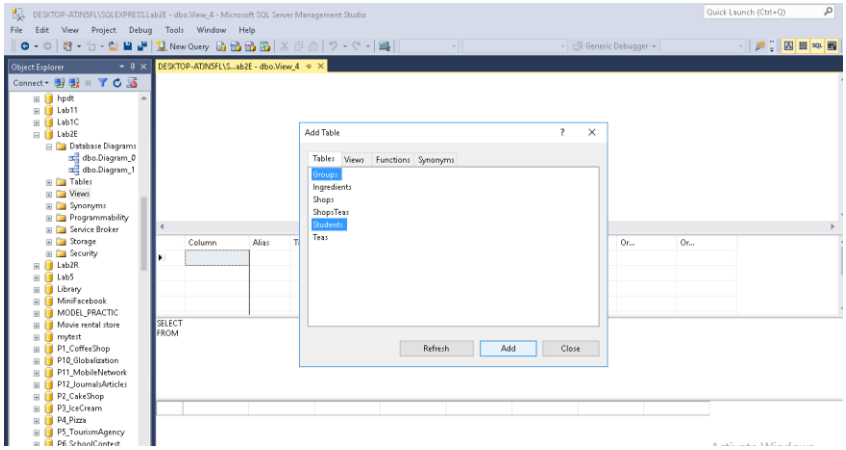
	Name	Mark
1	Paul	10
2	Cristi	9

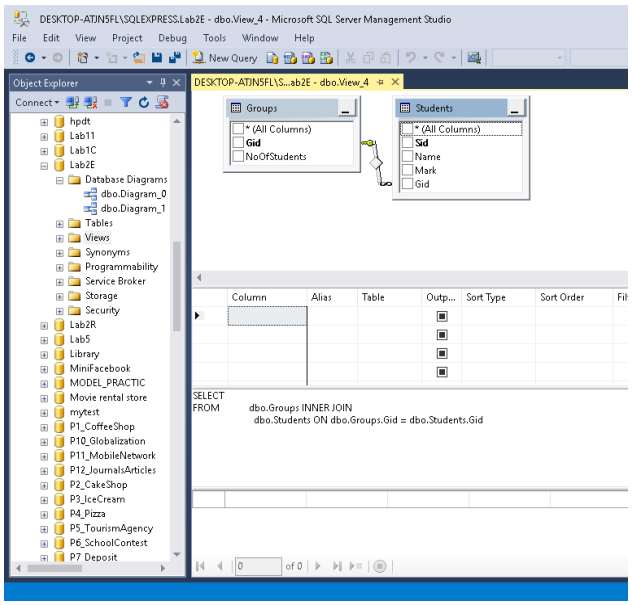
Obs.

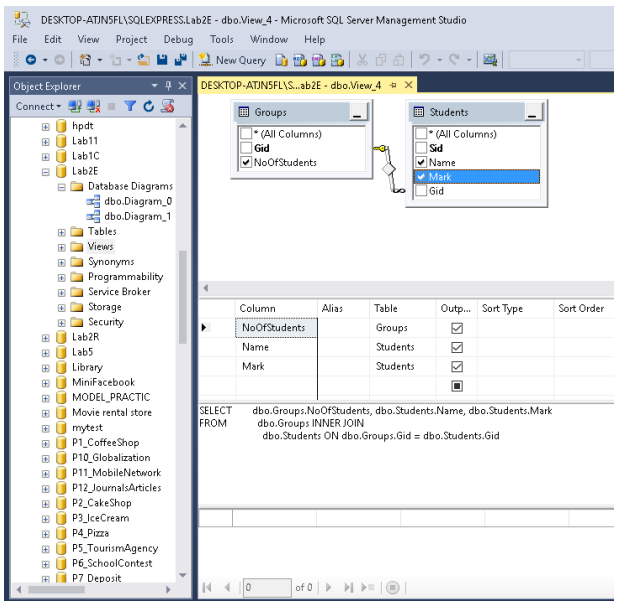
You can use views in at most 3 queries.

- Create a view: right click on the database on the View tab;
- choose the table you want to add; -> Add
- select the column you want to see in the result

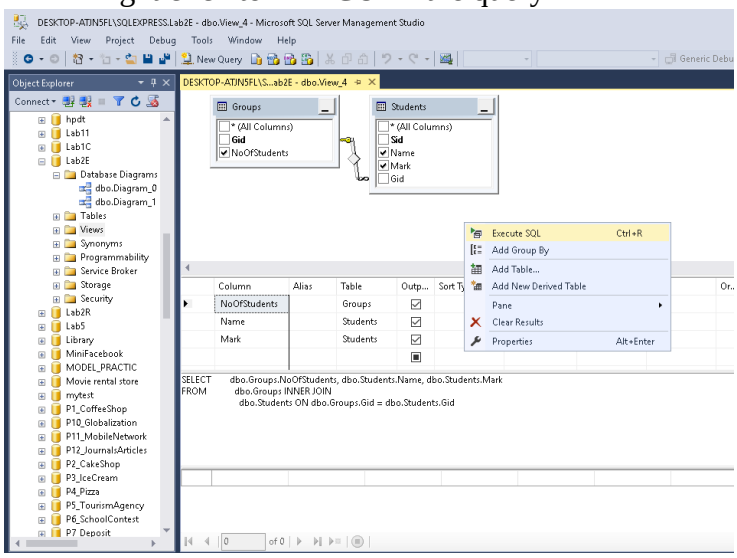


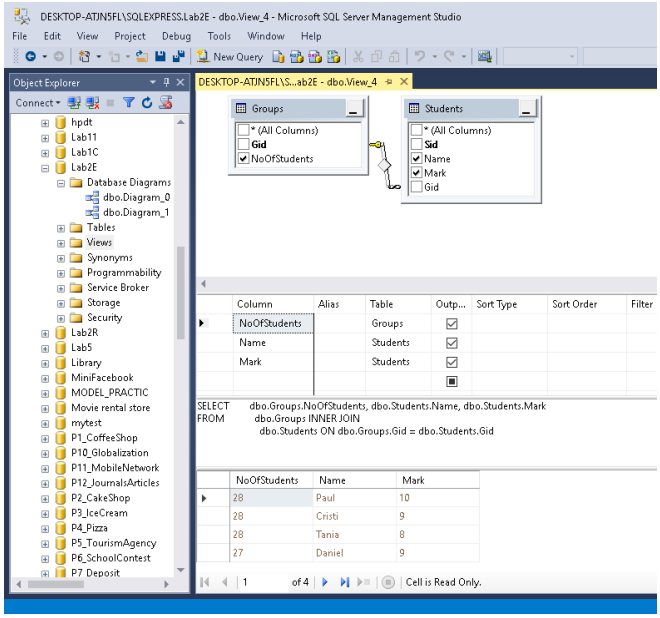


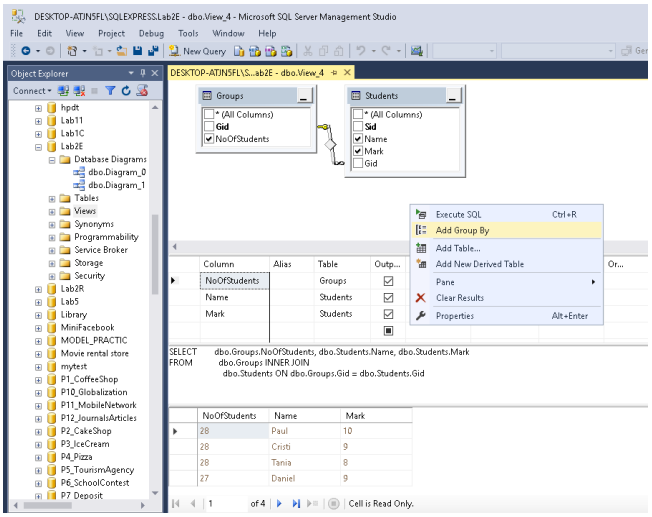




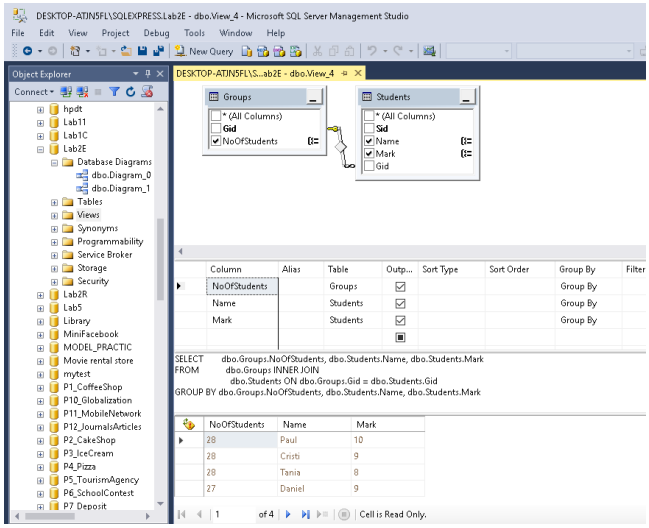
- right click to EXECUTE the query







- add group by- by right click and select



You can change the relational structure created for the first lab.
The queries must be relevant to the problem domain and provide data of interest to a potential user.