National University of Computer and Emerging Sciences

Lab Manual 4

"Nested Queries"

Database Systems Lab

Spring 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

1. Contents

2. Objective

- The purpose of this manual is to get started with nested queries. This lab will cover all the topics we have covered before. Starting from simple Select-From-Where, Joins, Order by, Aggregate functions & Group by, all of these will be used in combination with the nested queries.

3. Pre-requisites

- Lab manual 2 & 3 which includes:
    - Select-From-Where clause
    - Joins and all its types

Task Distribution

Total Time	170 Minutes
Nested Queries	30 Minutes
Exercise	120 Minutes
Evaluation	Last 20 Minutes

4. Nested Queries

For this in-lab manual, use the **InLab5TryThisSchema.sql** script to create database and practice the queries given below.

4.1.1. A subquery (inner query) is a SQL select query nested inside a another select query (outer query)

A subquery may occur in:
- SELECT clause of outer query
- FROM clause of outer query
- WHERE clause of outer query (most commonly used)

4.1.2. A subquery can be nested inside:

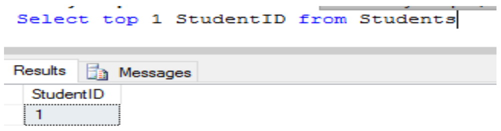
- SELECT statement
- INSERT statement
- UPDATE statement
- DELETE statement
- Another subquery.

4.1.3. There are two types of subqueries

- Correlated subqueries: where we use some attribute of outer query in inner query, result of inner query will then change according to the attribute of outer query.
- Non-correlated subqueries: where no attribute of outer query is used in inner query, in this case inner query always return same value.

4.1.4. Scalar Vs Non-scalar

A select query can return a scalar value or a table. Scalar value means one column and one row
Example: result of the following query is scalar



```
Select top 1 StudentID from Students
```

```
StudentID
1
```

A select query can also return non-scalar value, with more than one column and/or more than one row
Example:

```
Select StudentID from Students
```

Will give non-scalar result.

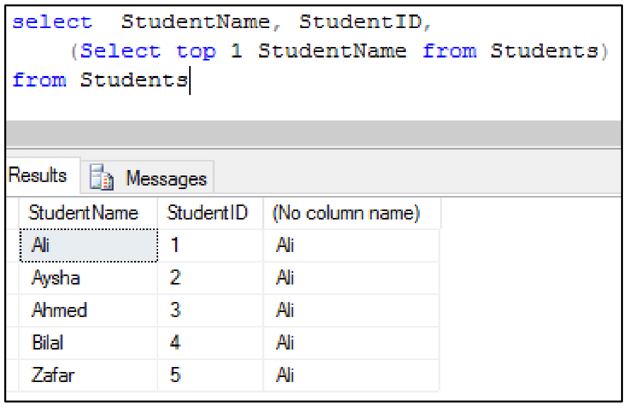
If you are writing a sub query in Select Clause, the inner query should be Scalar
If you are writiing a subquey in From Clause, inner query can be scalar or non-Scalar
If you are writing a subquery in Where Clause, inner query can be scalar or non-Scalar depending on conditon.

Non-Correlated Query:

4.1.5. Non-Correlated Subqueries in SELECT clause

```
SELECT <List of columns of T>
       (select ColumnName from <TableName>)
FROM <tablename> AS T
WHERE <condition>
**inner query should be scalar
```

TRY IT: Non-correlated nested query in Select is not very useful



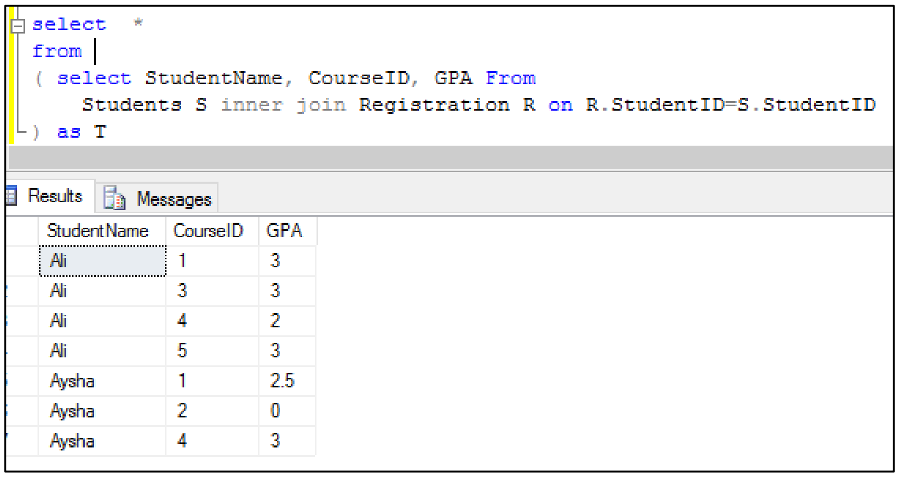
```
select  StudentName, StudentID,
    (Select top 1 StudentName from Students)
from Students
```

StudentName	StudentID	(No column name)
Ali	1	Ali
Aysha	2	Ali
Ahmed	3	Ali
Bilal	4	Ali
Zafar	5	Ali

4.1.6. Non-Correlated Subqueries in From Clause

```
SELECT <List of columns of T ( result of inner query)>
FROM (select ColumnName from <TableName>) as T WHERE <condition>
**inner query can be scalar of non-scalar
***always give alias to inner query in from clause
```

TRY THIS



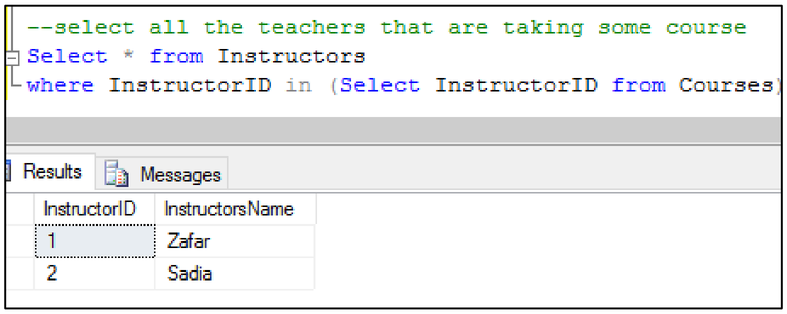
```
select  *
from
( select StudentName, CourseID, GPA From
    Students S inner join Registration R on R.StudentID=S.StudentID
) as T
```

StudentName	CourseID	GPA
Ali	1	3
Ali	3	3
Ali	4	2
Ali	5	3
Aysha	1	2.5
Aysha	2	0
Aysha	4	3

4.1.7. Non-Correlated Subqueries in Where Clause

```
SELECT <List of columns of T >
FROM TableName as T
WHERE <condition> (select ColumnName from <TableName>)
```

TRY THIS



```
--select all the teachers that are taking some course
Select * from Instructors
where InstructorID in (Select InstructorID from Courses)
```

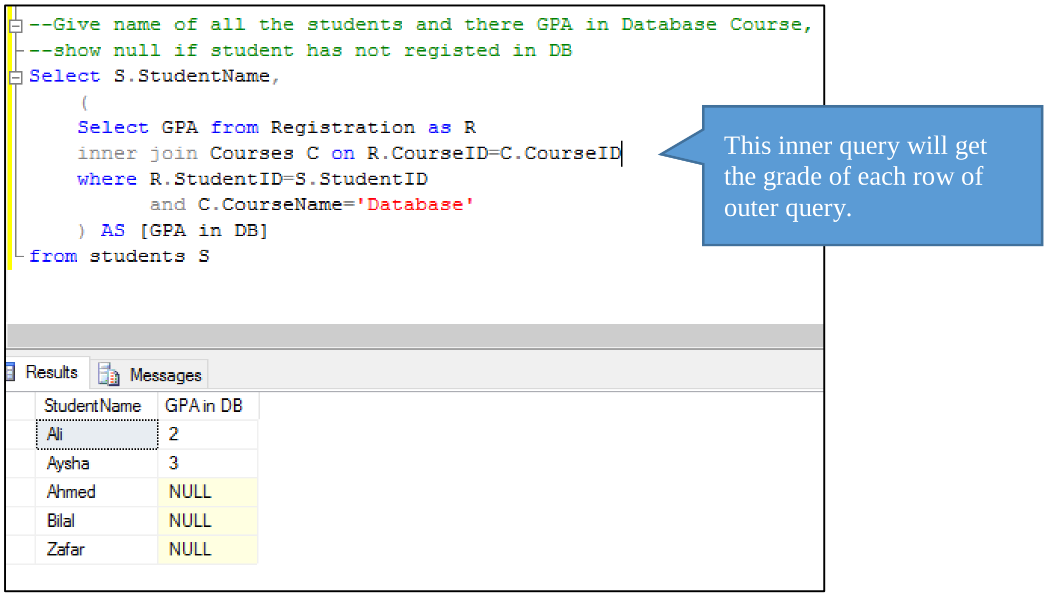
InstructorID	InstructorsName
1	Zafar
2	Sadia

Correlated queries

When inner query is correlated with outer query, then the inner query is executed for each row of outer query.

4.1.8. Correlated Subquery in Select Clause

TRY THIS



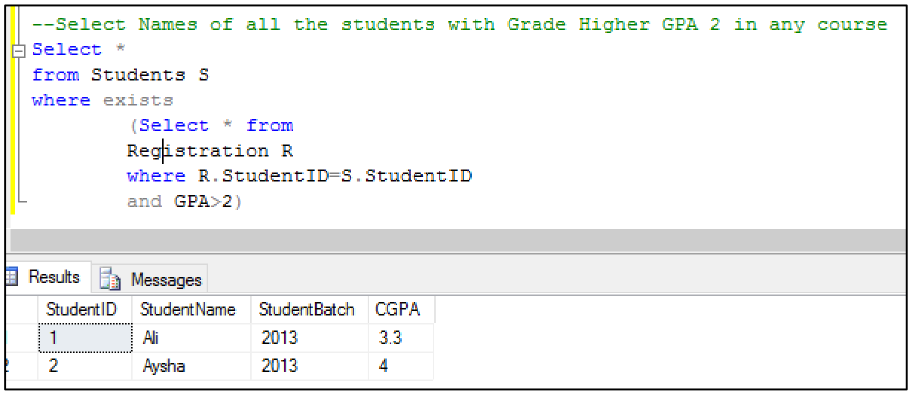
```
--Give name of all the students and there GPA in Database Course,
--show null if student has not registed in DB
Select S.StudentName,
    (
    Select GPA from Registration as R
    inner join Courses C on R.CourseID=C.CourseID
    where R.StudentID=S.StudentID
          and C.CourseName='Database'
    ) AS [GPA in DB]
from students S
```

This inner query will get the grade of each row of outer query.

StudentName	GPA in DB
Ali	2
Aysha	3
Ahmed	NULL
Bilal	NULL
Zafar	NULL

4.1.9. Correlated Subquery in Where Clause

TRY THIS



```
--Select Names of all the students with Grade Higher GPA 2 in any course
Select *
from Students S
where exists
        (Select * from
        Registration R
        where R.StudentID=S.StudentID
        and GPA>2)
```

Results | Messages

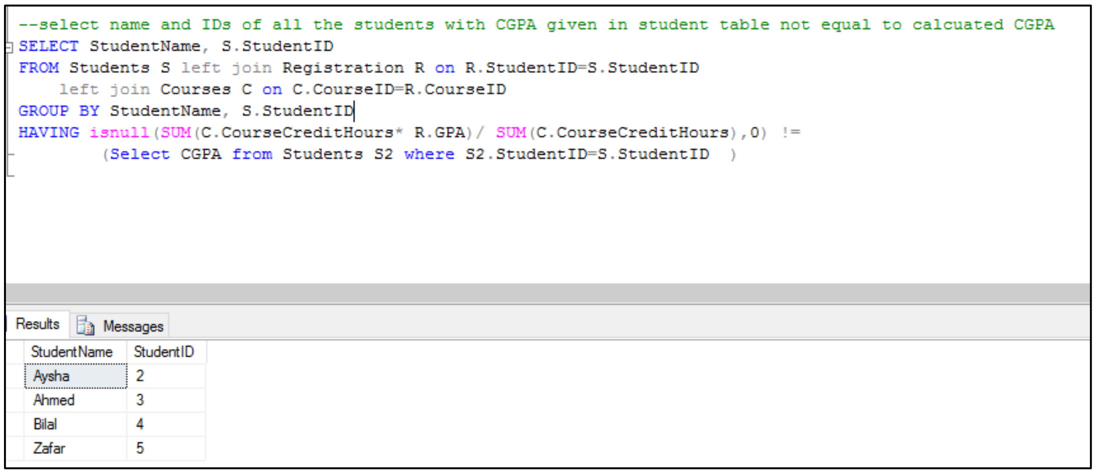
StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	3.3
2	Aysha	2013	4

** WHAT DOES THE EXIST CLAUSE DO?

4.1.10. Correlated Subquery in Having Clause

You can also use subquery in having clause (correlated on non-correlated)

TRY THIS



```
--select name and IDs of all the students with CGPA given in student table not equal to calcuated CGPA
SELECT StudentName, S.StudentID
FROM Students S left join Registration R on R.StudentID=S.StudentID
    left join Courses C on C.CourseID=R.CourseID
GROUP BY StudentName, S.StudentID
HAVING isnull(SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours),0) !=
        (Select CGPA from Students S2 where S2.StudentID=S.StudentID  )
```

Results | Messages

StudentName	StudentID
Aysha	2
Ahmed	3
Bilal	4
Zafar	5

Modify the query given above to, Show name, IDs, Calcuated CGPA and CGPA given in Student table of all the students with CGPA given in student table lesser to calcuated CGPA

5. Aggregation-Grouping

Aggregation allows you to apply calculation on values of column, and it will return a scalar value. Adding the GROUP BY Clause allows you to aggregate on groups of data, a scalar value will be returned for each group of data.
Some examples of Aggregate functions are given below.

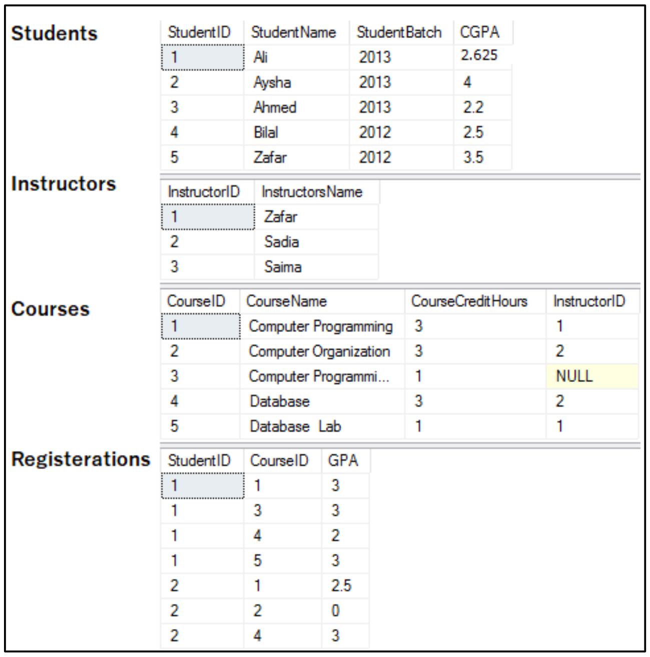
Aggregation Function Key work	How it works	No of Column Function can work on
AVG()	Returns the average of the values in a group. Null values are ignored.	Single column
COUNT()	Returns the number of items in a group. This function always returns an int data type value	Single Column or List of Columns or *
MAX()	Returns the maximum value in the expression.	Single column
MIN()	Returns the minimum value in the expression.	Single column
SUM()	Returns the sum of all the values in the expression. SUM can be used on numeric columns only and it ignores all the NULL values.	Single column

Figure 1 Aggregation Functions

Following is the syntax of Aggregation without grouping.

```
Select <AggregationFunction>(COLUMNs/Column) AS <AliasName>
From <TableName>
```

Use the script (Lab4TryManual.sql Figure 1) to create database to try the following queries.



Students

StudentID	StudentName	StudentBatch	CGPA
1	Ali	2013	2.625
2	Aysha	2013	4
3	Ahmed	2013	2.2
4	Bilal	2012	2.5
5	Zafar	2012	3.5

Instructors

InstructorID	InstructorsName
1	Zafar
2	Sadia
3	Saima

Courses

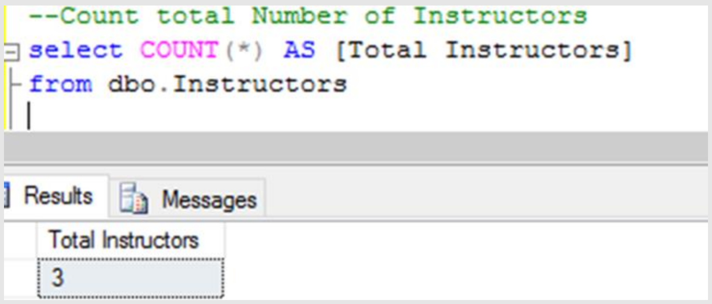
CourseID	CourseName	CourseCreditHours	InstructorID
1	Computer Programming	3	1
2	Computer Organization	3	2
3	Computer Programmi...	1	NULL
4	Database	3	2
5	Database Lab	1	1

Registerations

StudentID	CourseID	GPA
1	1	3
1	3	3
1	4	2
1	5	3
2	1	2.5
2	2	0
2	4	3

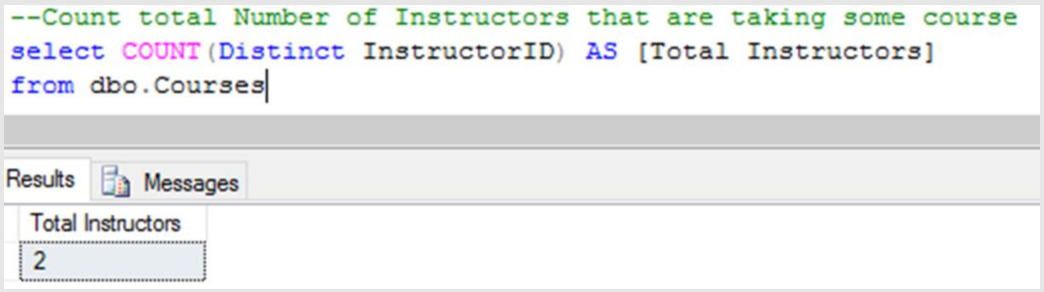
Figure 2 University Database

TRY THIS (Aggregation with Grouping)



```
--Count total Number of Instructors
select COUNT(*) AS [Total Instructors]
from dbo.Instructors
```

```
Results | Messages
Total Instructors
3
```

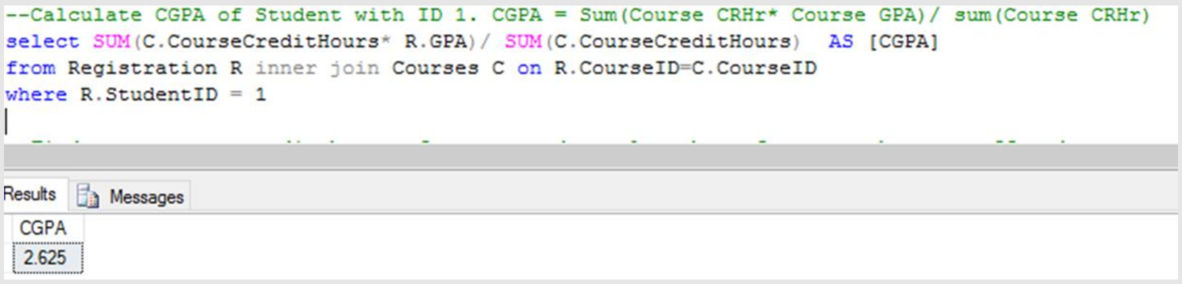


```
--Count total Number of Instructors that are taking some course
select COUNT(Distinct InstructorID) AS [Total Instructors]
from dbo.Courses
```

```
Results | Messages
Total Instructors
2
```

**NOTE THE DISTINCT KEY WORD. WHAT DOES IT DO?

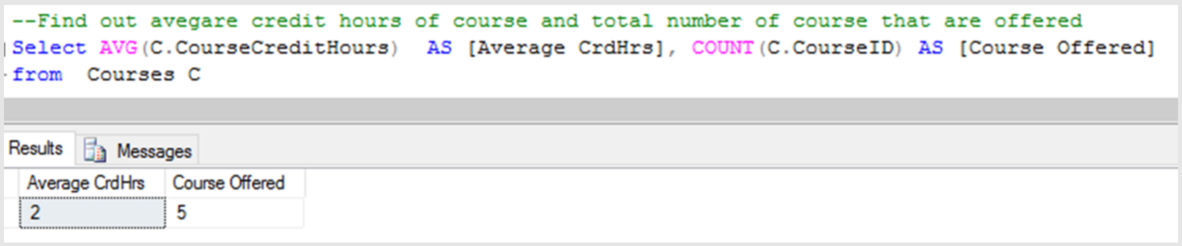
YOU CAN USE AGGREGATION AND JOING TOGETHER



```
--Calculate CGPA of Student with ID 1. CGPA = Sum(Course CRHr* Course GPA)/ sum(Course CRHr)
select SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours)  AS [CGPA]
from Registration R inner join Courses C on R.CourseID=C.CourseID
where R.StudentID = 1
```

```
Results | Messages
CGPA
2.625
```

USE MORE THAN ONE AGGREGATION FUNCTION IN SAME SELECT



```
--Find out avegare credit hours of course and total number of course that are offered
Select AVG(C.CourseCreditHours)  AS [Average CrdHrs], COUNT(C.CourseID) AS [Course Offered]
from  Courses C
```

```
Results | Messages
Average CrdHrs  Course Offered
2               5
```

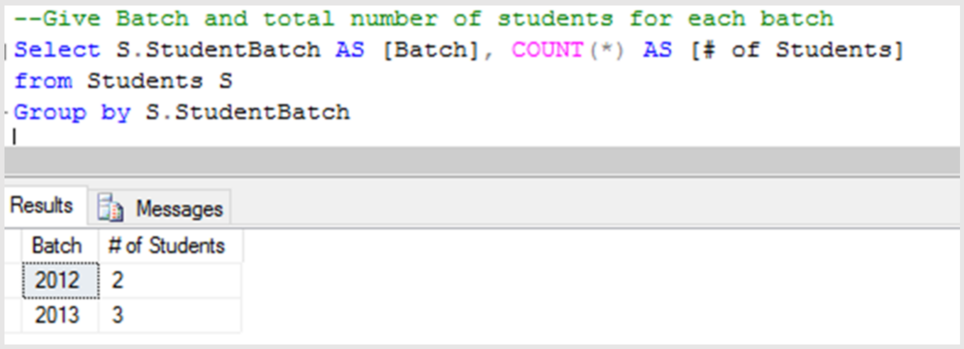
Grouping:

Syntax:

```
Select T.ColumnX, T.ColumnY Aggreation Function(Column/Columns) AS [Alias]
from TableName T
Group by T.ColumnX, T.ColumnY --comma seperated list of all the column of which
                              --groping is to be done
```

NOTE: ONLY THE COLUMNS THAT ARE USED IN GROUPING CAN BE USED IN SELECT CLAUSE

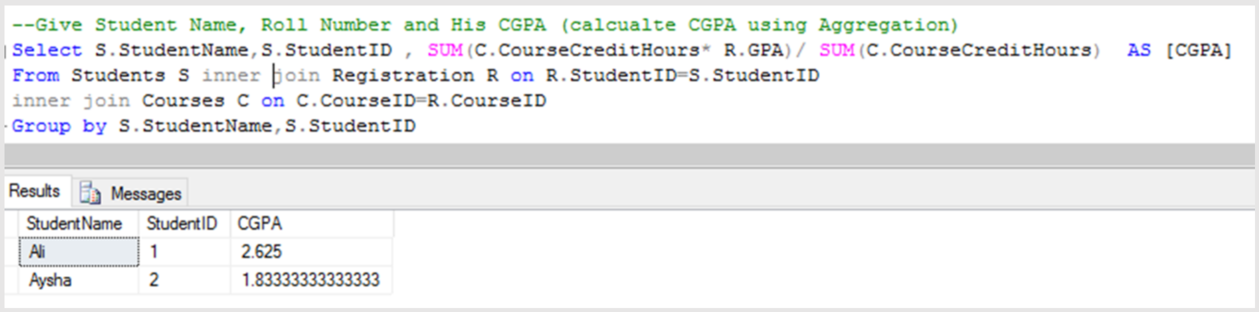
TRY THIS (Aggregate with grouping)



```
--Give Batch and total number of students for each batch
Select S.StudentBatch AS [Batch], COUNT(*) AS [# of Students]
from Students S
Group by S.StudentBatch
```

Results | Messages

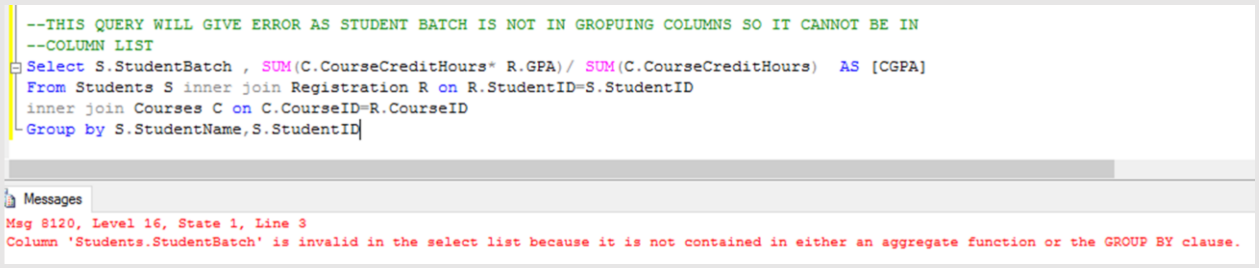
Batch	# of Students
2012	2
2013	3



```
--Give Student Name, Roll Number and His CGPA (calcualte CGPA using Aggregation)
Select S.StudentName,S.StudentID , SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours)  AS [CGPA]
From Students S inner join Registration R on R.StudentID=S.StudentID
inner join Courses C on C.CourseID=R.CourseID
Group by S.StudentName,S.StudentID
```

Results | Messages

StudentName	StudentID	CGPA
Ali	1	2.625
Aysha	2	1.83333333333333



```
--THIS QUERY WILL GIVE ERROR AS STUDENT BATCH IS NOT IN GROPUING COLUMNS SO IT CANNOT BE IN
--COLUMN LIST
Select S.StudentBatch , SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours)  AS [CGPA]
From Students S inner join Registration R on R.StudentID=S.StudentID
inner join Courses C on C.CourseID=R.CourseID
Group by S.StudentName,S.StudentID
```

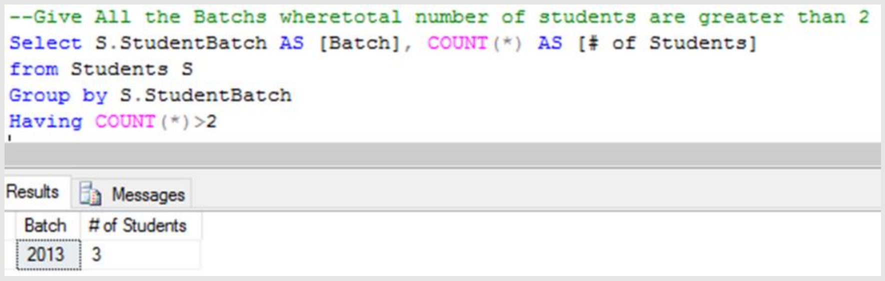
Messages

```
Msg 8120, Level 16, State 1, Line 3
Column 'Students.StudentBatch' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
```

Having Clause

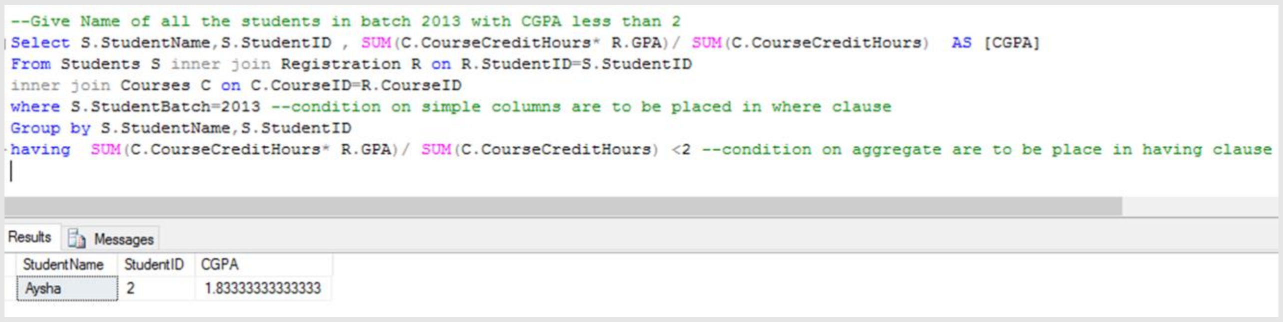
Having Clause allows us to filter the data based on the result of aggregation function, it’s the same as where clause except that we cannot use aggregate functions in where clause and we cannot use simple columns having clause.

Try this (aggregate group having)



```
--Give All the Batchs wheretotal number of students are greater than 2
Select S.StudentBatch AS [Batch], COUNT(*) AS [# of Students]
from Students S
Group by S.StudentBatch
Having COUNT(*)>2
```

```
Batch  # of Students
2013   3
```



```
--Give Name of all the students in batch 2013 with CGPA less than 2
Select S.StudentName,S.StudentID , SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours)  AS [CGPA]
From Students S inner join Registration R on R.StudentID=S.StudentID
inner join Courses C on C.CourseID=R.CourseID
where S.StudentBatch=2013 --condition on simple columns are to be placed in where clause
Group by S.StudentName,S.StudentID
having  SUM(C.CourseCreditHours* R.GPA)/ SUM(C.CourseCreditHours) <2 --condition on aggregate are to be place in having clause
```

```
StudentName  StudentID  CGPA
Aysha        2          1.83333333333333
```

NOTE: THE ORDER OF EACH CLAUSE IS TO BE MAINTAINED AS FOLLOW

1. SELECT (COMPULSORY)
2. FROM (COMPULSORY)
3. WHERE
4. GROUP
5. HAVING