National University of Computer and Emerging Sciences

Lab Manual

"Data Retrieval Select-from-where, Joins, Order by, Aggregate functions, Group by"

Database Systems Lab

Spring 2024

Department of Computer Science
FAST-NU, Lahore, Pakistan

1. Contents

2. Objective

- The purpose of this manual is to get stared with data retrieval queries, starting from Simple Select-From-Where, going towards Join operation, covering Order by clause and Aggregate functions, Group by.

3. Pre-requisites

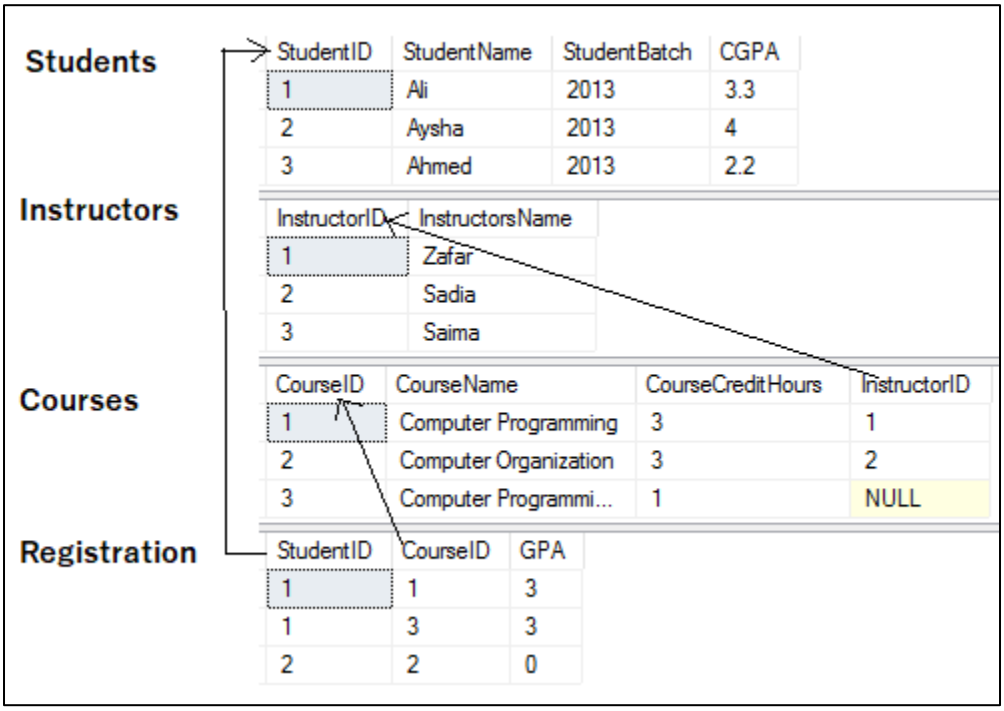
- Lab 3 manual, on how to get started with MS-SQL server
- How Select from Where clause work
- How Joining and all its type work
- How Order by clause works
- Aggregate functions, Group by

Task Distribution

Total Time	170 Minutes
Select from where	15 Minutes
Order by	15 Minutes
Joining	15 Minutes
Group by	15 Minutes
Exercise	90 Minutes
Evaluation	Last 20 Minutes

4. SELECT-FROM-WHERE

Select from where is equivalent to projection and selection in Relational Algebra, it will give output in form of a table. The most basic select statement includes Select and from clause, and it will retrieve all columns and rows from the table.
We will use the following schema and database for the examples. Script to create this schema is given in Lab4Manual.sql file



Most Basic Select:

SELECT *
FROM <tableName>

* after select means that all columns will be retrieved

Try this

```
select * from students
```

Results

Results Messages

	StudentID	StudentName	StudentBatch	CGPA
1	1	Ali	2013	3.3
2	2	Aysha	2013	4
3	3	Ahmed	2013	2.2

Retrieving certain Columns from Select

To retrieve only certain columns give a comma separated list of those columns after Select keyword

SELECT ColumnX, ColumnY, ColumnZ
FROM <tableName>

Try this

```
Select Course Name, CourseCreditHours
from courses
```

Results

Results Messages

	CourseName	CourseCreditHours
1	Computer Programming	3
2	Computer Organization	3
3	Computer Programming Lab	1

Retrieving certain Rows from Select- WHERE CLAUSE

Like Selection in RA, rows are filter in SQL using WHERE clause, rows that fulfill where clause conditions will be projected in result. Where clause can put condition on original columns of tables mentioned on from clause, or derived columns.

SELECT *
FROM <tableName>
where <conditions>

Try this

```
Select CourseName, CourseCreditHours
from courses
where CourseName like '%Programming%' and CourseCreditHours>= 1
```

Results

Results Messages

	CourseName	CourseCreditHours
1	Computer Programming	3
2	Computer Programming Lab	1

Renaming Resulting Column

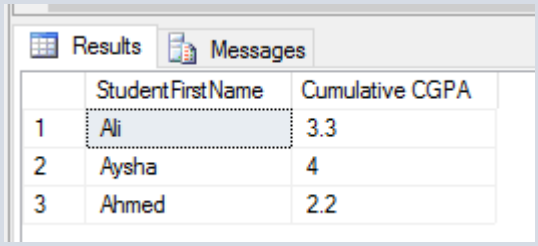
You can rename a column in result by using AS keyword also called Alias. The scope of this renaming is only to that select query, this is useful in joining where more than one table have same column names.

SELECT ColumnX as X , ColumnY as Y, ColumnZ
FROM <tableName> as Table1

Try this

```
select StudentName AS StudentFirstName
, CGPA AS [Cumulative CGPA]
from students AS StudentsTable
```

Results



Results | Messages

	StudentFirstName	Cumulative CGPA
1	Ali	3.3
2	Aysha	4
3	Ahmed	2.2

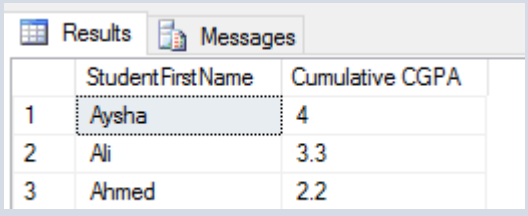
5. Order by Clause

Order by clause is used to arrange the rows in ascending or descending order of one or more columns
SELECT ColumnX as X , ColumnY as Y, ColumnZ
FROM <tableName> as Table1
ORDER BY ColumnX asc/desc, ColumnZ asc/desc

Try this

```
select StudentName AS StudentFirstName
, CGPA AS [Cumulative CGPA]
from students AS StudentsTable
order by  CGPA desc
```

Results



Results | Messages

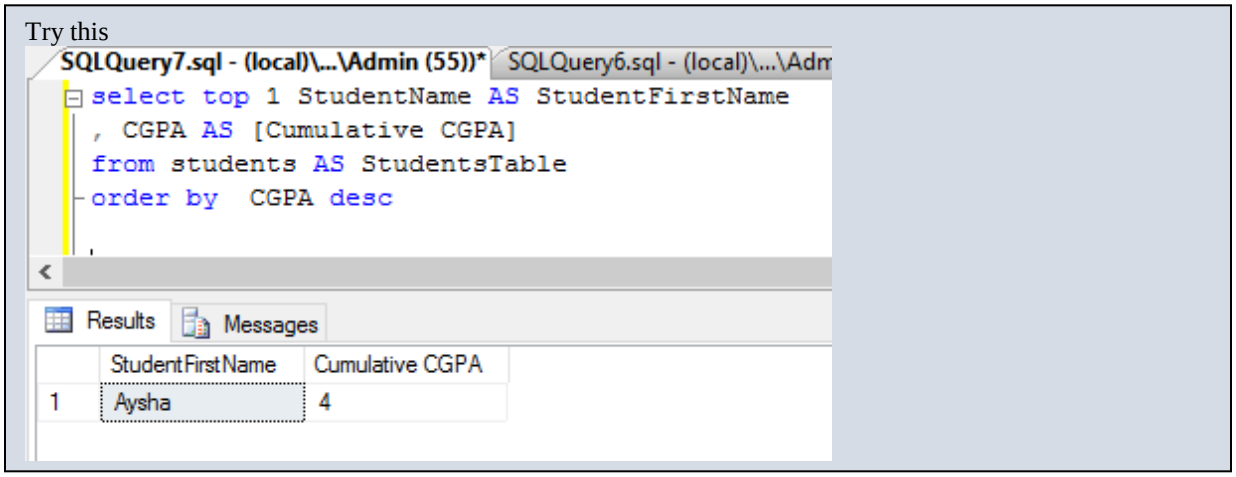
	StudentFirstName	Cumulative CGPA
1	Aysha	4
2	Ali	3.3
3	Ahmed	2.2

TOP Clause

Top n clause will give you first n rows from result instead of all the rows.

SELECT TOP <n> *
FROM <tableName>
where <conditions>
Order by <column Name> asc/desc

Try this



6. Join Operation

We will use the following tables in examples

Inner Join:

Returns only those rows that match in both tables.

SELECT *
FROM <table1> inner join <table2>
ON <Joining Condition>

```
select * from Instructors
inner join courses
on Courses.InstructorID=Instructors.InstructorID
```

Results | Messages

	InstructorID	InstructorsName	CourseID	CourseName	CourseCreditHours	InstructorID
1	1	Zafar	1	Computer Programming	3	1
2	2	Sadia	2	Computer Organization	3	2

Left/Right/Full Outer Join

Left Join: Returns all the rows of Left table with corresponding row or null row of right table
Right Join: Returns all the rows of Right table with corresponding row or null row of Left table
Full Join: Union of Left and Right Outer join
SELECT * FROM <table1> Left/Right/Full join <table2> ON <Joining Condition>

Try these

```
select * from Instructors
left join courses
on Courses.InstructorID=Instructors.InstructorID
```

Results Messages

	InstructorID	InstructorsName	CourseID	CourseName	CourseCreditHours	InstructorID
1	1	Zafar	1	Computer Programming	3	1
2	2	Sadia	2	Computer Organization	3	2
3	3	Saima	NULL	NULL	NULL	NULL

```
select * from Instructors
right join courses
on Courses.InstructorID=Instructors.InstructorID
```

Results Messages

InstructorID	InstructorsName	CourseID	CourseName	CourseCreditHours	InstructorID
1	Zafar	1	Computer Programming	3	1
2	Sadia	2	Computer Organization	3	2
NULL	NULL	3	Computer Programming Lab	1	NULL

```
select * from Instructors
full join courses
on Courses.InstructorID=Instructors.InstructorID
```

Results Messages

	InstructorID	InstructorsName	CourseID	CourseName	CourseCreditHours	InstructorID
1	1	Zafar	1	Computer Programming	3	1
2	2	Sadia	2	Computer Organization	3	2
3	3	Saima	NULL	NULL	NULL	NULL
4	NULL	NULL	3	Computer Programming Lab	1	NULL

Cross Join

It's a cross product of two tables, no ON condition is required here

SELECT * FROM <table1> cross Join <table2>

Try this

```
select * from Instructors
cross  join courses
```

Results Messages

	InstructorID	InstructorsName	CourseID	CourseName	CourseCreditHours	InstructorID
1	1	Zafar	1	Computer Programming	3	1
2	2	Sadia	1	Computer Programming	3	1
3	3	Saima	1	Computer Programming	3	1
4	1	Zafar	2	Computer Organization	3	2
5	2	Sadia	2	Computer Organization	3	2
6	3	Saima	2	Computer Organization	3	2
7	1	Zafar	3	Computer Programming Lab	1	NULL
8	2	Sadia	3	Computer Programming Lab	1	NULL
9	3	Saima	3	Computer Programming Lab	1	NULL

Joining More than two tables

SELECT *
FROM <table1>
Left/Right/Full/Inner join <table2> ON <Joining Condition>
Left/Right/Full/Inner join <table3> ON <Joining Condition>
Left/Right/Full/Inner join <table4> ON <Joining Condition>

Try this

```
select studentName,courseName,instructorsName
from students S
join Registration R on R.studentID=S.studentID
join Courses c on R.courseID=c.courseID
join Instructors i on i.InstructorID=c.InstructorID
```

Results Messages

	studentName	courseName	instructorsName
1	Ali	Computer Programming	Zafar
2	Aysha	Computer Organization	Sadia

7. 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
```

Results | Messages

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
```

Results | Messages

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

8. Set operations

Result of two (or more) select queries can be combined using Set operations such as UNION, INTESECT, EXCEPT.

Syntax

```
Select ColumnX, ColumnY
From Table1

Union/Intersect/Except

Select ColumnA, ColumnB
From Table2
```

NOTE: The output of first select query should have same number and type of column as of second select query.

Try this –Set operations

```
--List IDs of all the students that have not registed in any course
select StudentID From Students
except
select StudentID from Registration
```

Results Messages

StudentID
3
4
5

```
---list ID  of all the instructors that are taking some course
Select InstructorID from Instructors
Intersect
Select InstructorID from Courses
```

Results Messages

InstructorID
1
2

```
--List all the Names of instructors and Students
Select StudentName From Students
Union
Select InstructorsName From Instructors
```

Results Messages

StudentName
Ahmed
Ali
Aysha
Bilal
Sadia
Saima
Zafar

Try this- error to look out for in set operations

```
--Fails because Datatype of Corresponding Columns is not same
Select StudentName From Students
Union
Select StudentID From Students
```

Results Messages

```
Msg 245, Level 16, State 1, Line 1
Conversion failed when converting the varchar value 'Ali' to data type int.
```

```
--Fails becsue Number of Columns in 1st select query are not same as number of columns in 2nd select query
Select StudentName From Students
Union
Select StudentID, StudentName From Students
```

Messages

```
Msg 205, Level 16, State 1, Line 2
All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.
```