Learning Objectives:

- Retrieving and Restricting data using the SQL SELECT statement
- Understanding wildcards, GROUP BY and ORDER BY

Helping Material:

Capabilities of SQL SELECT Statements

A SELECT statement retrieves information from the database. With SELECT statement you can use the following capabilities:

- **Projection**: Select the columns in a table that are returned by a query. Select as few or as many of the columns as required.
- **Selection**: Select the rows in a table that are returned by a query. Various criteria can be used to restrict the rows that are retrieved.
- **Joining**: Bring together data that is stored in different tables by specifying the link between them. SQL joins are covered in more detail in the next labs.

Structure of basic select statement

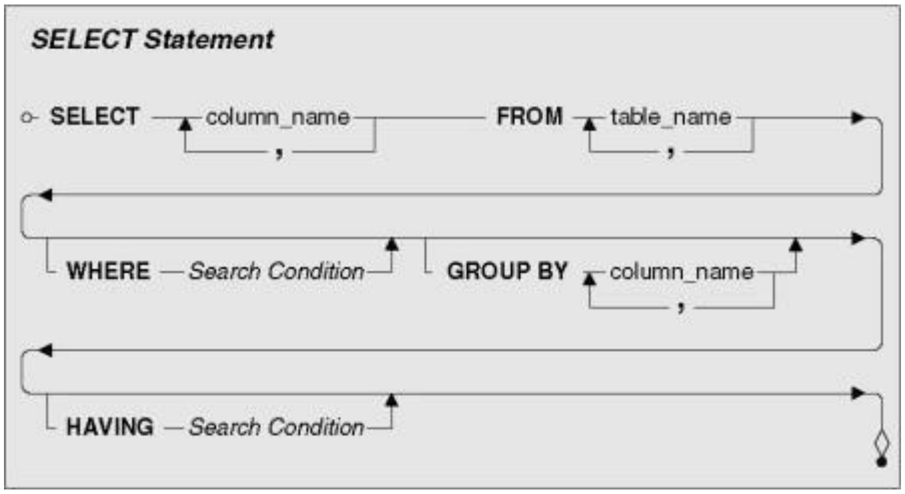

Figure 1: Select statement graphical flow

In its simplest form, a SELECT statement must include the follow in:

- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which identifies the table containing the columns that are listed in the SELECT statement

Note: Throughout this course, the words **keyword, *clause*: and *statement*** are used as follows:

- A keyword refers to an individual SQL element. For example, SELECT and FROM are keywords.

- A clause is a part of a SQL statement. For example, SELECT name, cnic and so on is a clause.
- A statement is a combination of two or more clauses. For example, SELECT FROM Student is a SQL statement.

Note:

- SQL statements are not case-sensitive.
- SQL statements can be entered on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- SQL statements can optionally be terminated by a semicolon (;). Semicolons are required when you execute multiple SQL statements.

```
Select * from TableName;                    -- Display all the contents of a table
```

```
SELECT * from Student
```

Select column:

To select a column from the table, specify the correct column name.
Syntax:

```
Select columnName from TableName;                 -- Display a specific column of a table
```

Select multiple columns:

For selecting multiple columns from a table write names of all columns separated by comma (,)
Syntax:

```
Select column1, column2 from TableName;            -- Display specific columns of a table
```

```
SELECT firstName, lastName from Student
```

Select top:

If we want to show the top few values from a table, we use TOP command and a number, that is the rows you want to show, with SELECT command.
Syntax:

```
Select TOP number columnName from TableName;       -- Display top rows of a table
```

```
SELECT TOP 5 age from Student -- Display top 5 rows of age column from Students table
```

Select distinct:

To know the unique values in a column we use DISTINCT, with SELECT command.

Syntax:

```
Select DISTINCT (columnName) from TableName;          -- Display all the unique values in a column
```

```
SELECT   DISTINCT(AGE)   from   Student
```

Select count:

If you want to know the number of entries in a row, use COUNT command.
Syntax:

```
Select COUNT (columnName) from TableName;          -- Display number of entries in a column
```

```
SELECT COUNT(firstName) from Student
```

Operator Precedence:

Operator	**Precedence**
Unary operators, bitwise NOT	1
Multiplication and division	2
Addition, subtraction, and concatenation	3
SQL conditions	4

Wildcard Characters:

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the ***LIKE*** operator. The ***LIKE*** operator is used in a ***WHERE*** clause to search for a specified pattern in a column.
The variations of wildcards and their usages are mentioned in the table below

Symbol	Description	Example
%	Represents zero or more characters	**bl%** finds bl, black, blue, and blob
_	Represents a single character	**h_t** finds hot, hat, and hit
[]	Represents any single character within the brackets	**h[oa]t** finds hot and hat, but not hit
^	Represents any character not in the brackets	**h[^oa]t** finds hit, but not hot and hat
-	Represents any single character within the specified range	**c[a-b]t** finds cat and cbt

Comparison and Logical Operators:

- Following comparison operators are commonly used in SQL server
  - =
  - < >
  - >
  - >=
  - <=
  - <
  - IS NULL
  - IS NOT NULL
  - LIKE
- Following Logical operators are used
  - NOT
  - OR
  - AND

ORDER BY Clause

ORDER BY clause is used for ascending order.
The ORDER BY keyword is used to sort the result-set in ascending or descending order.
Syntax:

```
Select column1, column2 from tableName ORDER BY column1, column2 ASC|DESC;
```

```
--Sort ascending/descending

SELECT LastName, Age from Student ORDER BY Student ASC
```

This query will select the last names and ages of all the students in the ascending order of their ages.

Group By:

Group by is used to show all the unique values in a column.

Syntax:

```
Select column1, column2 from tableName GROUP BY column1, column2,…
```

-- Display all values using fully qualified name

Now, you must be wondering if we already have DISTINCT statement, why are we using GROUP BY for the same purpose?
The answer is that the difference between GROUP BY and DISTINCT is that the later only tells you the unique values in a column, while the former not only tells you the unique values in the column but also tells the number of entries under each value.

For example: If we write this query,

```
SELECT Gender from Student GROUP BY Gender
```

It will return the same result as the DISCTINCT statement. But if we modify our query as

```
SELECT Gender, COUNT(Gender)  from Student GROUP BY Gender
```

It will show how many numbers of entries are there in male and female category

TOP N clause:

- TOP N clause is used to limit the result.
- Limit is also used for this purpose.

Lab Task:

- Create a new table Student which have the following schema Student(RegNo: String, FirstName: String, LastName: String, GPA: Float, Contact: Integer)
- Add at least 5 records of your own class in which one or two students have GPA undefined.
- Display all the data from the table Student
- Display specific columns form the table Student
- Display all the data of students where GPA > 3.5
- Display all the data of students where GPA <= 3.5
- Does the above 2 queries covers all the data?
- Display first and last name of all students as single column using concatenation operator “||”.
- Your task is to write SQL statements corresponding to each operator.
- Identify at least one SQL statement in which precedence can affect the result of query.
- Identify how the result of a mathematical expression on null value affect the result of a query.
- Use the distinct operator to eliminate the duplicates in your SQL statement.