SQL Select

The SQL SELECT Statement

The most commonly used SQL command is **SELECT statement**. It is used to query the database and retrieve selected data that follow the conditions we want.

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SELECT Syntax

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

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, then :

```
SELECT * FROM table_name;
```

Example:

REG_NO	NAME	BRANCH	EMAIL
20184165	Saurav Chaudhary	CSE	saurav@saurav.com
20184149	Lokesh Raj Singhi	CSE	lokesh@lokesh.com
20184063	Rajan Kr Jaiswal	CSE	rajan@rajan.com
20185157	Rajan Jayswal	it	rajan@rajan.com

To Select Desired Column

```
select Reg_No,Name from Students;
```

To Select All Column

```
select * from students;
```

The SQL SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values. There might be sometime duplicate values in database so to list the different values.

SELECT DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

To Selected Distinct values in Email Column

```
SELECT DISTINCT email
FROM students;
```

To get count of distinct values in a column

```
SELECT COUNT(DISTINCT ColumnName) FROM table_name;
```

```
SELECT count(DISTINCT email)
FROM students;
```

The SQL SELECT IN Statement

SQL IN is an operator used in a SQL query to help reduce the need to use multiple SQL "OR" conditions.

```
SELECT *
FROM table_name
WHERE students_name IN ( value1 , value2 ......);
```

```
SELECT email FROM students
where Reg_No in (20184165,20184149);
```

The SQL SELECT TOP Clause

The SELECT TOP clause is used to specify the number of records to return.

MySQL

```
SELECT TOP 3 * FROM Students;
```

Oracle

```
SELECT * FROM Students
where ROWNUM<=3;
```

Optional clauses in SELECT statement

WHERE CLAUSE

The WHERE clause is used to filter records on the basis of certain condition.

WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;

/*OR*/

SELECT *
FROM table_name
WHERE condition;
```

To Select Student whose name is Rajan

```
SELECT * FROM Students
WHERE Name='Rajan';
```

> **NOTES:** SQL requires single quotes around text values & No quotes for numeric field.

The SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Operators in The WHERE Clause

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

BETWEEN

```
SELECT * FROM Products
WHERE Price BETWEEN 50 AND 60;
```

LIKE

Returns those tuples whose City starts with s.

```
SELECT * FROM Customers
WHERE City LIKE 's%';
```

IN

Refer to SELECT IN .

IS NULL & IS NOT NULL

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

ORDER BY

This is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the **DESC** keyword.

Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Example

Ascending by default

```
SELECT *
FROM students
ORDER BY Reg_No;
```

Descending

```
SELECT *
FROM students
ORDER BY Reg_No DESC;
```

ORDER BY Several Columns

```
SELECT *
FROM students
ORDER BY Name ASC, Reg_No DESC ;
```

The SQL GROUP BY Clause

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of students with same blood group".

```
Select Blood_Group,count(Blood_Group)
From Students
Group BY Blood_Group;
```

The SQL HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.