**Select Statement Introduction**

**select**statement is used to fetch records from database tables.  
If **student**is a table, we use the following SQL statement to fetch all rows (records) and all columns from the table student.

select \* from student;

**Selecting a specific column**

recharge is a database table that contains information related to pre-paid mobile phone recharges

completed. The following columns are present in recharge table.

id s- unique identifier

phonenumber - mobile number for which the recharge was done

amount - recharge amount

rechargedon - the date and time of recharge

To select just the column containing the phone numbers (that is the column **phonenumber**), we use the following SQL.

select phonenumber from recharge;

**Select - Multiple Columns**

**SELECT**statement is used to retrieve data from tables. The result returned is called result set.

A table called **player**is created with the following DDL command:

create table player(id INT, name VARCHAR(100), score INT);

To selectmultiple columns **name** and **score**from the **player**table, we use the following select query.

select name, score FROM player;

**Select - count function**

student is a table to store id, name, registration number, year of study of students in a college.  
Zing Hua is the principal and he wants to know how many students are in the college. So he uses the following SQL.

select count(\*) from student;

**count(\*)** fetches the number of rows in the result set returned by the SQL.

**Select - where clause**

Very often it may not be required to fetch all the rows from a table but only specific rows meeting a certain condition or multiple conditions.

where clause is used to fetch the rows meeting the desired condition(s).

**movie**is a table to store name, language and  year of release of movies in a given country.  
The columns are name, language and releaseyear (language is like 'spanish' and releaseyear is a number like 2013).

Suppose we wish to fetch all the rows in the movie table where the language is chinese, we use the following SQL.

select \* from movie where language = 'chinese';

**Note:** For string values we must enclose the filtering value in where clause within **single quotes.**

**Select - where clause continued..**

**WHERE**clause is used to retrieve certain data from the table that complies to a particular condition.

A table called **employee**is created with the following DDL command:

create table employee(employeeid INT, name VARCHAR(100), salary INT);

To select the employee name(s) whose salary is **10000**from the table **employee**, we use the following SQL query.

select name from employee where salary = 10000 order by name;

**Note**: The result is ordered by the column name in ascending order

**Select - where clause multiple conditions**

In a select statement, where clause can have multiple conditions.

"**movie**" is a table to store name, language and  year of release of movies in a given country.  
The columns are name, language and releaseyear (language is like 'spanish' and releaseyear is a number like 2013).

Suppose we wish to fetch all the rows in the movie table where the language is chinese and the release year is 2012, we use the following SQL.

select \* from movie where language = 'chinese' and releaseyear=2012;

**Select - where clause OR condition**

In a select statement, where clause can have either or condition.

"**movie**" is a table to store name, language and  year of release of movies in a given country.  
The columns are name, language and releaseyear (language is like 'spanish' and releaseyear is a number like 2013).

Suppose we want to fetch all the rows in the movie table where the language is chinese or the release year is 2013, we use the following SQL.

select \* from movie where language = 'chinese' or releaseyear=2013;

**select - where clause NOT equal**

In a select statement, we can have a**not equal** to condition in where clause.

**movie** is a table to store name, language and  year of release of movies in a given country.  
The columns are name, language and releaseyear (language is like 'spanish' and releaseyear is a number like 2013).

Suppose we want to fetch all the rows in the movie table where the language is NOT chinese, we use the following SQL.

select \* from movie where language != 'chinese';

**select - where clause Greater than**

In a select statement, we can have a greater than / greater than or equal to condition in where clause.

"**movie**" is a table to store name, language and  year of release of movies in a given country.  
The columns are name, language and releaseyear (language is like 'spanish' and releaseyear is a number like 2013).

Suppose we want to fetch all the rows in the movie table where the release year is greater than or equal to 2012, we use the following SQL.

select \* from movie where releaseyear >= 2012;

Suppose we want to fetch all the rows in the movie table where the release year is greater than 2012, we use the following SQL. (We just omit the = symbol in the previous SQL).

select \* from movie where releaseyear > 2012;

**select - where clause Less than**

In a select statement, we can have a less than / less than or equal to condition in where clause.

"movie" is a table to store name, language and  year of release of movies in a given country.  
The columns are name, language and releaseyear (language is like 'spanish' and releaseyear is a number like 2013).

Suppose we want to fetch all the rows in the movie table where the release year is less than or equal to 2012, we use the following SQL.

select \* from movie where releaseyear <= 2012;

Suppose we want to fetch all the rows in the movie table where the release year is less than 2012, we use the following SQL. (We just omit the = symbol in the previous SQL).

select \* from movie where releaseyear < 2012;

**Select - order by**

**order by**is used to sort the records retrieved in either ascending or descending order. order by can be applied to one or more columns.

A table called **employee**is created with the following DDL command:

create table employee(employeeid INT, name VARCHAR(100), salary INT, city VARCHAR(100));

To select all the records in the table **employee**and to sort the records in ascending order by **name** and **salary**, we use the following query.

select \* from employee order by name, salary asc;

**Select - order by with or**

The OR operator selects a record**if any one of the conditions is TRUE.**  
   
A table called **student**is created with the following DDL command:

create table student(name VARCHAR(100), subject1 INT, subject2 INT, subject3 INT);

To select the names of the students who scored **more**than **50**in **subject1**or **less**than **80**in **subject2**, we use the following select query.

select name from student WHERE subject1 > 50 OR subject2 < 80 ORDER BY name;

**Select - limit**

**limit**clause is used to specify the maximum number of records to be retrieved.

A table called **player**is created with the following DDL command:

create table player(id INT, name VARCHAR(100), score INT);

To select the **players with top five** **scores** from the **player**table, we use the following select query.

select name, score FROM player order by score desc limit 5;

**Note**:  
1. The result set is ordered by the column **score**in descending order.  
2. The top five records are selected from the result using the **limit**clause.

**Select - order by desc**

A table called **player**is created with the following DDL command:

create table player(id integer, name varchar(20), score integer);

To select first five**names** and **scores** from the **player** table where the result set is ordered by**name** and **score** (both) in descending order, we use the following select query.

select name, score from player order by name desc, score desc limit 5;

**Select - IN operator**

IN operator is used to select records where the column value is present in the given set of data. It is a **shorthand**notation for using**multiple OR operators.**

A table called **player**is created with the following DDL command:

create table player(id INT, name VARCHAR(100), score INT);

To select the player **names**whose scores is any one of**187, 27, 43** from the **player**table and to list them in the descending order of their score, we use the following SQL query.

select name FROM player where score in (187, 27, 43) order by score desc;

**Select - BETWEEN operator**

**BETWEEN**operator is used to select values within a range. The between operator is**inclusive** of beginning and end of the range.

A table called **player**is created with the following DDL command:

create table player(id INT, name VARCHAR(100), score INT);

To select the player records who scored between **100 and 200** from the **player**table sorting them by their score in descending order, we use the following select query.

select name FROM player where score between 100 and 200 order by score desc;

**Select - NOT operator**

The NOT operator selects a record if a given condition is**false (not true).**  
   
A table called **student**is created with the following DDL command:

create table student(name VARCHAR(100), subject1 INT, subject2 INT, result VARCHAR(4));

To select all records where the value of the column result is **not FAIL**, we use the following select query.

select name, subject1, subject2 from student WHERE NOT result='FAIL' ORDER BY name;

**Note:**  
The order of the columns in the result set is **name, subject1, subject2.**  
The result set is ordered by the column **name**in ascending order.

**Select - NOT NULL**

In a table, if a column is **optional** and no value is specified for this optional column then a NULL value is assigned to this column value.

A table called **student**is created with the following DDL command:

create table student (id INT, name VARCHAR(100), department VARCHAR(100), college VARCHAR(100), city VARCHAR(100));

To select all records from the table **student**where the value for the column college is **NOT** **empty (NOT NULL)**, we use following select query.

select \* from student where college IS NOT NULL;

**Select - Distinct**

The **DISTINCT**keyword is used to retrieve only distinct (unique) values. If a table contains duplicate values for a given column, then DISTINCT will discard the duplicate values and return only the unique values.

A table called **supermarket**iscreated with the following DDL command:

create table supermarket(itemid INT, categoryid INT, name VARCHAR(100), price INT, city VARCHAR(100));

To fetch the count of uniquevalues in the column **city**, we use the following query.

select count(distinct city) from supermarket;

**Select - Arithmetic Operators**

Arithmetic operators can be used to perform arithmetic operations on column values in the returned result set.  
   
A table called **student**is created with the following DDL command:

create table student(id int,name varchar(100), physics int, maths int, chemistry int);

To select all the **student names**with their **average marks** ordered by their name we use the following select query.

select name, (physics + chemistry + maths)/3 as average from student order by name;