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 select multiple 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;
```