**DQL - **Data Query Language using MySql****

1. ****select****

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

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

**Syntax :**

**SELECT column1, column2, ...  
FROM table\_name;**

To select all columns from a table “\*” is used.

Syntax:

**SELECT \* FROM tablename;**

To select selected columns column names are used.

Syntax:

**SELECT column\_name,column\_name,column\_name... etc**

**FROM table\_name**

To select only unique value from a column DISTINCT key is used

Syntax:

**SELECT DISTINCT column\_name FROM table\_name;**

**WHERE CLAUSE IN SELECT**

“where” clause is used to select or filter rows in MySql database using a conditions.

Syntax :

**SELECT column\_name,column\_name**

**FROM table\_name**

**WHERE condition;**

Can also use in update, delete queries.

**CONDITIONAL OPERATORS IN MySql**

|  |  |  |
| --- | --- | --- |
| **AND** | **OR** | **NOT** |
| 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. |
| SELECT column1, column2, ... FROM table\_name WHERE condition1 AND condition2 AND condition3 ...; | SELECT column1, column2, ... FROM table\_name WHERE condition1 OR condition2 OR condition3 ...; | SELECT column1, column2, ... FROM table\_name WHERE NOT condition; |

**LIKE OPERATOR**

There are two wildcards often used in conjunction with the LIKE operator:

The percent sign (%) represents zero, one, or multiple characters

The underscore sign (\_) represents one, single character

Syntax :

**SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;**

Example **:**

**SELECT \* FROM Customers  
WHERE CustomerName LIKE '%a';**

**SELECT \* FROM Customers  
WHERE CustomerName LIKE '\_r%';**

**IN OPERATOR**

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

Syntax :

**SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (SELECT STATEMENT);**

**Example:**

**SELECT \* FROM Customers  
WHERE Country NOT IN ('Germany', 'France', 'UK');**

**SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');**

**BETWEEN OPERATOR**

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

Syntax:

**SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;**

Example :

**SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;**

**SELECT \* FROM Products  
WHERE Price NOT BETWEEN 10 AND 20;**

**ALIASES**

Aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

Syntax :

**SELECT column\_name AS alias\_name  
FROM table\_name;**

Example :

**SELECT column\_name(s)  
FROM table\_name AS alias\_name;**

**ORDER BY**

“order by” is used to sort the result set in ascending order or descending order.

Syntax:

//ASCENDING ORDER

**SELECT column\_name**

**FROM table\_name**

**ORDER BY column\_name ASC;**

//DESCENDING ORDER

**SELECT column\_name**

**FROM table\_name**

**ORDER BY column\_name DESC;**

**GROUP BY**

“group by”statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax:

**SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)  
ORDER BY column\_name(s);**

Example

**SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;**

**NULL AND NOT NULL**

To check the any rows in a column is null or not NULL & NOT NULL keys are used.

Syntax:

**SELECT column\_name**

**FROM table\_name**

**WHERE column\_name IS NULL ;**

**SELECT column\_name**

**FROM table\_name**

**WHERE column\_name IS NOT NULL;**

**LIMIT CLAUSE**

Limit is used to specify the number column that should be returned.

Syntax :

**SELECT column\_name**

**FROM table\_name**

**WHERE condition**

**LIMIT number;**

Can also set “offset” to specify from where the limit should start.

**SELECT column\_name**

**FROM table\_name**

**WHERE condition**

**LIMIT number OFFSET number;**

**Aggregation function**

**MIN():**

**SELECT MIN(column\_name)**

**FROM table\_name**

**WHERE condition;**

**MAX():**

**SELECT MAX(column\_name)**

**FROM table\_name**

**WHERE condition;**

**COUNT():**

**SELECT COUNT(column\_name)**

**FROM table\_name**

**WHERE condition;**

**AVG():**

**SELECT AVG(column\_name)**

**FROM table\_name**

**WHERE condition;**

**SUM():**

**SELECT SUM(column\_name)**

**FROM table\_name**

**WHERE condition;**

**HAVING CLAUSE**

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

Syntax :

**SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)  
HAVING condition  
ORDER BY column\_name(s);**

Example :

**SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5;**

**SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5  
ORDER BY COUNT(CustomerID) DESC;**

**EXAMPLE :**

**Step 1 :**

**USE lfschool;**

**Step 2:**

**CREATE TABLE std\_fee(id INT PRIMARY KEY,std\_id INT NOT NULL,amount VARCHAR(200) NOT NULL,a\_status VARCHAR(200) NOT NULL, a\_date DATE NOT NULL,CONSTRAINT FK\_stdid FOREIGN KEY (id) REFERENCES std\_details(ID));**

**DROP TABLE std\_fee;**

**Step 3:**

**INSERT INTO std\_fee(id,std\_id,amount,a\_status,a\_date) VALUES**

**(1,1,"10000","paid","2024-10-01"),**

**(2,2,"12000","pending","2024-12-30"),**

**(3,3,"5000","paid","2024-12-30"),**

**(4,4,"5000","pending","2024-12-30"),**

**(5,5,"15000","paid","2024-12-30");**

**Step 4:**

**SELECT \* FROM std\_fee;**

**SELECT \* FROM std\_details;**

**SELECT \* FROM sub\_details;**

**SELECT id,std\_name,std\_age FROM std\_details;**

**Step 5:**

**SELECT DISTINCT std\_age FROM std\_details;**

**Step 6:**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_gen = "M" ;**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_gen = "F" ;**

**Step 7:**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_gen = "M" AND std\_age = 23 ;**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_gen = "M" OR std\_age = 23 ;**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE NOT std\_gen = "M" AND std\_age = 23 ;**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE NOT std\_gen = "M" OR std\_age = 23 ;**

**Step 8:**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_name LIKE "a%" ;**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_name LIKE "%u" ;**

**SELECT id,std\_name,std\_age**

**FROM std\_details**

**WHERE std\_name LIKE "\_n%" ;**

**Step 9:**

**SELECT \***

**FROM std\_details**

**WHERE std\_sub NOT IN (1,2,3);**

**SELECT \***

**FROM std\_details**

**WHERE std\_sub IN (1,2,3);**

**Step 10:**

**SELECT \***

**FROM std\_details**

**WHERE std\_dob BETWEEN "2001-01-01" AND "2003-01-01";**

**SELECT \***

**FROM std\_details**

**WHERE std\_age BETWEEN "20" AND "24";**

**Step 11:**

**SELECT id AS Roll\_no,std\_name AS Name,std\_age AS Age**

**FROM std\_details;**

**Step 12:**

**SELECT \* FROM std\_fee;**

**SELECT \***

**FROM std\_fee**

**ORDER BY amount DESC;**

**SELECT \***

**FROM std\_fee**

**ORDER BY amount ASC;**

**Step 13 :**

**SELECT DISTINCT(a\_status),COUNT(a\_status)**

**FROM std\_fee**

**GROUP BY a\_status;**

**Step 14:**

**SELECT \***

**FROM std\_fee**

**WHERE a\_status = "paid"**

**LIMIT 3 ;**

**SELECT \***

**FROM std\_fee**

**WHERE a\_status = "paid"**

**LIMIT 3 OFFSET 2;**

**Step 15 :**

**SELECT MIN(amount)**

**FROM std\_fee;**

**SELECT MAX(amount)**

**FROM std\_fee;**

**SELECT AVG(amount)**

**FROM std\_fee;**

**SELECT COUNT(id)**

**FROM std\_fee;**

**SELECT SUM(amount)**

**FROM std\_fee;**