BASIC TO ADVANCE ALL QUERIES !

MYSQL

Database Related Queries

- CREATE
- DROP
- USE
- SHOW

Table Related Queries

- CREATE
- DESC
- INSERT
- SELECT & VIEW
- DELETE
- UPDATE
- DROP
- DISTINCT

ALTER

- ADD
- DROP
- RENAME
- CHANGE
- MODIFY

Constraints

- NOT NULL
- DEFAULT
- PRIMARY KEY
- FOREIGN KEY
- AUTO_INCREMENT
- CHECK
- UNIQUE

Aggregate functions

- SUM()
- AVG()
- MAX()
- MIN()
- COUNT()

MYSQL

CLAUSE

WITH WHERE

- LIKE
- IN
- BETWEEN
- ANY
- EXISTS
- AND/OR/NOT

OTHER

- ORDER BY
- LIMIT
- GROUP BY
- HAVING

JOINS

- INNER JOIN
- OUTER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

STRING FUNCTIONS

- CONCAT()
- CONCAT_WS()
- SUBSTR()
- REPLACE()
- REVERSE()
- UPPER()/LOWER()
- CHAR_LENGTH()
- LEFT()
- RIGHT()
- TRIM()

DATE AND TIME FUNCTIONS

NOW()
CURDATE()
CURTIME()
DATE()
TIME()
YEAR()
MONTH()
DAY() /DAYOFMONTH()
HOUR()
MINUTE()
SECOND()

Database related Queries

- 1. CREATE

The CREATE statement is used to create a new database or a new table.

Syntax : CREATE DATABASE database_name;

- 2. DROP

The DROP statement is used to delete (remove) a database .

Syntax : DROP DATABASE database_name;

- 3.USE

The USE statement is used to select or switch to a specific database, making it the current working database for all subsequent queries.

Syntax : USE database_name;

- 4.SHOW

The SHOW statement is used to display information about databases.

Syntax : SHOW DATABASES;

Table related Queries

- 1. CREATE

The CREATE statement is used to create a new table in database.

Syntax : CREATE TABLE table_name (
  column1 datatype,
  column2 datatype,
  ...
);

- 2. DESC

The DESC (short for DESCRIBE) statement is used to show the structure of a table, including the columns and their data types.

Syntax : DESC table_name;

- 3.INSERT

The INSERT statement is used to add new rows to a table.

Syntax : INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);

- **4. SELECT**

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

Syntax : SELECT column1, column2, ...
FROM table_name;

- **5. DELETE**

The DELETE statement is used to remove rows from a table.

Syntax : DELETE FROM table_name
WHERE condition;

- **6.UPDATE**

The UPDATE statement is used to modify existing data in a table.

Syntax :UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

- **7. DROP**

The DROP statement is used to permanently remove a database or a table.

Syntax : DROP TABLE table_name;

- **8. DISTINCT**

The DISTINCT keyword is used with a SELECT query to return only unique values, removing duplicates.

Syntax : SELECT DISTINCT column1
FROM table_name;

Table related Queries with ALTER

- 1. ADD

The ADD statement is used to add a new column or constraint to an existing table.

Syntax :ALTER TABLE table_name
ADD column_name datatype;

- 2. DROP

The DROP statement is used to delete an existing column, constraint, or index from a table.

Syntax : ALTER TABLE table_name
DROP COLUMN column_name;

- 3. RENAME

The RENAME statement is used to rename a table or a column in a table.

Syntax :ALTER TABLE table_name
RENAME COLUMN old_column_name TO new_column_name;

- **4. CHANGE**

The CHANGE statement is used to rename a column and change its data type simultaneously.

Syntax :ALTER TABLE table_name
RENAME COLUMN old_column_name TO new_column_name;

- **5. MODIFY**

The MODIFY statement is used to change the datatype of an existing column without changing its name.

Syntax :ALTER TABLE table_name
MODIFY column_name new_datatype;

CONSTRAINTS

1.NOT NULL

The NOT NULL constraint ensures that a column cannot have a NULL value. It forces a column to always contain a value.

Syntax :CREATE TABLE table_name (
column_name datatype NOT NULL);

2.DEFAULT

The DEFAULT constraint provides a default value for a column when no value is specified during insertion.

Syntax :CREATE TABLE table_name (
column_name datatype DEFAULT default_value);

3.CHECK

The CHECK constraint ensures that all values in a column satisfy a specific condition.

Syntax :CREATE TABLE table_name (
column_name datatype CHECK (condition));

4.PRIMARY KEY

The PRIMARY KEY constraint uniquely identifies each record in a table. It must contain unique values and cannot contain NULL values.

```
Syntax :CREATE TABLE table_name (
        column_name datatype,
        PRIMARY KEY (column_name)
        );
```

5.FOREIGN KEY

The FOREIGN KEY constraint is used to link two tables. It ensures referential integrity by establishing a relationship between columns in different tables.

```
Syntax :CREATE TABLE table_name (
        column_name datatype,
 FOREIGN KEY (column_name) REFERENCES other_table(column_name)
);
```

6.AUTO_INCREMENT

The AUTO_INCREMENT constraint automatically generates a unique number for a column when a new record is inserted. This is often used with the PRIMARY KEY.

Syntax :
```
CREATE TABLE table_name (
    column_name INT AUTO_INCREMENT,
    PRIMARY KEY (column_name)
);
```

7.UNIQUE

The UNIQUE constraint ensures that all values in a column or group of columns are distinct (i.e., no duplicate values).

Syntax :
```
CREATE TABLE table_name (
    column_name datatype,
    UNIQUE (column_name)
);
```

Aggregate functions

1.SUM()

The SUM() function calculates the total sum of a numeric column.

Syntax :SELECT SUM(column_name) FROM table_name;

2.AVG()

The AVG() function returns the average value of a numeric column.

Syntax :SELECT AVG(column_name) FROM table_name;

3.MAX()

The MAX() function returns the highest value from a column.

Syntax :SELECT MAX(column_name) FROM table_name;

4.MIN()

The MIN() function returns the lowest value from a column.

Syntax :SELECT MIN(column_name) FROM table_name;

5.COUNT()

The COUNT() function returns the number of rows that match a specified condition or counts the total number of rows in a column.

Syntax :SELECT COUNT(column_name) FROM table_name;

CLAUSE

1.LIKE

The LIKE operator is used to search for a specified pattern in a column.

Syntax :SELECT * FROM table_name WHERE column_name LIKE pattern;

2.IN

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

Syntax :
SELECT * FROM table_name WHERE column_name IN (value1, value2, ...);

3.BETWEEN

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

Syntax :SELECT * FROM table_name WHERE column_name BETWEEN value1 AND value2;

4.ANY

The ANY operator is used to compare a value with any value in a list or subquery.

Syntax :SELECT * FROM table_name WHERE column_name operator ANY (subquery);

5.EXISTS

The EXISTS operator is used to test for the existence of any record in a subquery. It returns TRUE if the subquery returns one or more records.

Syntax :
SELECT * FROM table_name WHERE EXISTS (subquery);

6.AND/OR/NOT

These are logical operators used to filter records based on more than one condition:

- AND: Returns records that satisfy all conditions.
- OR: Returns records that satisfy any of the conditions.
- NOT: Excludes records that meet the condition.

Syntax :SELECT * FROM table_name WHERE condition1 AND/OR/NOT condition2;

OTHER CLAUSE

1.ORDER BY

The ORDER BY clause is used to sort the result set of a query by one or more columns in ascending or descending order.

Syntax :SELECT * FROM table_name
ORDER BY column_name [ASC | DESC];

2.LIMIT

The LIMIT clause is used to specify the number of records to return from the result set.

Syntax :SELECT * FROM table_name
LIMIT number_of_rows;

3.GROUP BY

The GROUP BY clause is used to group rows that have the same values into summary rows, often used with aggregate functions like COUNT, SUM, AVG, etc.

Syntax :SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name;

4.HAVING

The HAVING clause is used to filter groups after they are created by the GROUP BY clause. It's similar to the WHERE clause but works on aggregated data.

Syntax: SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name
HAVING condition;

JOINS

1.INNER JOIN

Returns records that have matching values in both tables.

Syntax :
```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

2.OUTER JOIN

- **LEFT OUTER JOIN (or LEFT JOIN)**
- **RIGHT OUTER JOIN (or RIGHT JOIN)**
- **FULL OUTER JOIN**

A) LEFT OUTER JOIN (or LEFT JOIN)

Returns all records from the left table, and the matched records from the right table. If no match, NULLs are returned from the right table.

Syntax :
```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.common_column = table2.common_column;
```

B) RIGHT OUTER JOIN (or RIGHT JOIN)

Returns all records from the right table, and the matched records from the left table. If no match, NULLs are returned from the left table.

Syntax :
```
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.common_column = table2.common_column;
```

C) FULL OUTER JOIN

Returns all records when there is a match in either left or right table. If there is no match, NULLs are returned from the non-matching table.

Syntax :
```
SELECT columns
FROM table1
FULL OUTER JOIN table2
ON table1.common_column = table2.common_column;
```

STRING FUNCTIONS

1.CONCAT()

Concatenates (joins) two or more strings.

Syntax :SELECT CONCAT('Hello', ' ', 'World');

2.CONCAT_WS()

Concatenates strings with a separator (WS stands for "With Separator")

Syntax:SELECT CONCAT_WS('-', '2024', '09', '09');

3.SUBSTR() / SUBSTRING()

Extracts a substring from a string starting from a specified position and length

Syntax:SELECT SUBSTRING('Hello World', 1, 5);

4.REPLACE()

Replaces occurrences of a substring within a string with another substring.

Syntax :SELECT REPLACE('Hello World', 'World', 'SQL');

5.REVERSE()

SELECT REPLACE('Hello World', 'World', 'SQL');

Syntax:SELECT REVERSE('SQL');

6.UPPER() / LOWER()

Converts a string to uppercase or lowercase.

Syntax:SELECT UPPER('hello');

SELECT LOWER('HELLO');

7.CHAR_LENGTH()

Returns the number of characters in a string.

Syntax :SELECT CHAR_LENGTH('Hello');

8.LEFT()

Returns a specified number of characters from the left side of a string.

Syntax:SELECT LEFT('Hello', 3);

9.RIGHT()

Returns a specified number of characters from the right side of a string.

Syntax:SELECT RIGHT('Hello', 3);

10.TRIM()

Removes leading and trailing spaces from a string.

Syntax:SELECT TRIM(' Hello ');

DATE AND TIME FUNCTIONS

1.NOW()

Returns the current date and time.

Syntax :SELECT NOW();

2.CURDATE()

Returns the current date.

Syntax:SELECT CURDATE();

3.CURTIME()

Returns the current time.

Syntax:SELECT CURTIME();

4.DATE()

Extracts the date part from a DATETIME value.

Syntax:SELECT DATE();

5.TIME()

Extracts the time part from a DATETIME value.

Syntax :SELECT TIME();

6.YEAR()

Extracts the year from a date.

Syntax:SELECT YEAR();

7.MONTH()

Extracts the month from a date.

Syntax:SELECT MONTH();

8.DAY() / DAYOFMONTH()

Extracts the day of the month from a date.

Syntax:SELECT DAY();

9.HOUR()

Extracts the hour from a time.

Syntax :SELECT HOUR()

10.MINUTE()

Extracts the minute from a time.

Syntax:SELECT MINUTE();

11.SECOND()

Extracts the seconds from a time.

Syntax:SELECT SECOND();