CLOUSER, FUNCTION & DDL

What is Clousers In Mysql ?

Definitions and examples of the **HAVING**, **GROUP BY**, and **ORDER BY** clauses in MySQL:

1. **HAVING Clause:**
   - **Definition:** The **HAVING** clause is used in conjunction with the **GROUP BY** clause to filter the results of a query based on aggregate conditions. It is applied after the grouping has been done.

Example:

sqlCopy code

SELECT department, AVG(salary) as avg_salary FROM employees GROUP BY department HAVING AVG(salary) > 50000;

In this example, the query retrieves the average salary for each department and filters the result to include only those departments where the average salary is greater than 50,000.

2. **GROUP BY Clause:**
   - **Definition:** The **GROUP BY** clause is used to group rows that have the same values in specified columns into summary rows, like "total sales per category" or "average salary per department."

Example:

sqlCopy code

```
SELECT department, AVG(salary) as avg_salary FROM employees
GROUP BY department;
```

This query groups employees by department and calculates the average salary for each department.

3. **ORDER BY Clause:**
    - **Definition:** The **ORDER BY** clause is used to sort the result set based on one or more columns. It can sort in ascending (**ASC**) or descending (**DESC**) order.

Example:

sqlCopy code

```
SELECT employee_id, employee_name, salary FROM employees
ORDER BY salary DESC;
```

This query retrieves employee details and orders the result set in descending order based on the salary, showing employees with higher salaries first.

These clauses are often used together to perform more complex queries. For instance, you might want to group data, filter the groups using **HAVING**, and then sort the result using **ORDER BY**. Here's an example combining all three:

sqlCopy code

```
SELECT department, AVG(salary) as avg_salary FROM employees
GROUP BY department HAVING AVG(salary) > 50000 ORDER BY
avg_salary DESC;
```

This query groups employees by department, calculates the average salary for each department, filters out those with an average salary less than or equal to 50,000, and finally, orders the result by average salary in descending order.

What is Functions in mysql ?

In the context of databases, a function typically refers to a built-in operation that performs a specific task or calculation on data. MySQL provides a variety of functions for tasks such as mathematical calculations, string manipulation, date and time operations, and more.

Here is a general definition for a function and examples for some commonly used functions in MySQL:

Function Definition:

A function in MySQL is a predefined operation that takes one or more arguments as input, performs a specific task or calculation, and returns a result. Functions can be used in SQL queries to manipulate data, perform calculations, or retrieve information from the database.

Examples of Common MySQL Functions:

1. **Mathematical Functions:**

    - **ABS(x):** Returns the absolute value of x.

sqlCopy code

```
SELECT ABS(-10); -- Result: 10
```

- **ROUND(x, d):** Rounds x to d decimal places.

sqlCopy code

```
SELECT ROUND(3.14159, 2); -- Result: 3.14
```

- **SQRT(x):** Returns the square root of x.

sqlCopy code

```
SELECT SQRT(25); -- Result: 5
```

2. **String Functions:**

    - **CONCAT(str1, str2, ...):** Concatenates two or more strings.

sqlCopy code

```
SELECT CONCAT('Hello', ' ', 'World'); -- Result: Hello World
```

- **LENGTH(str):** Returns the length of a string.

sqlCopy code

```
SELECT LENGTH('MySQL'); -- Result: 5
```

- **UPPER(str):** Converts a string to uppercase.

sqlCopy code

```
SELECT UPPER('mysql'); -- Result: MYSQL
```

3. **Date and Time Functions:**

    - **NOW():** Returns the current date and time.

sqlCopy code

```
SELECT NOW(); -- Result: Current date and time
```

- **DATE_FORMAT(date, format):** Formats a date as specified.

sqlCopy code

```
SELECT DATE_FORMAT(NOW(), '%Y-%m-%d'); -- Result: Current date in 'YYYY-MM-DD' format
```

- **DATEDIFF(date1, date2):** Returns the difference in days between two dates.

sqlCopy code

```
SELECT DATEDIFF('2022-01-15', '2022-01-01'); -- Result: 14
```

These are just a few examples of the many functions available in MySQL. Depending on your specific requirements, you may need to use different functions for your queries. Refer to the MySQL documentation for a comprehensive list of functions and their usage.

What is DDL Commands ?

DDL, or Data Definition Language, commands in MySQL are used to define and manage the structure of a database, including creating, altering, and deleting database objects such as tables, indexes, and views. DDL commands allow you to specify how the data is stored in the database. Here are some commonly used DDL commands in MySQL with definitions and examples:

1. **CREATE TABLE:**
    - **Definition:** Creates a new table with specified columns and data types.

Example:

sqlCopy code

```
CREATE TABLE employees ( employee_id INT PRIMARY KEY,
employee_name VARCHAR(50), salary DECIMAL(10, 2) );
```

This command creates a table named "employees" with columns **employee_id**, **employee_name**, and **salary**.

2. **ALTER TABLE:**
    - **Definition:** Modifies the structure of an existing table, such as adding or dropping columns.

Example:

sqlCopy code

```
ALTER TABLE employees ADD COLUMN hire_date DATE;
```

This command adds a new column named "hire_date" to the existing "employees" table.

3. **DROP TABLE:**
    - **Definition:** Deletes an existing table and its data.

Example:

sqlCopy code

```
DROP TABLE employees;
```

This command deletes the "employees" table along with all its data.

4. **CREATE INDEX:**
    - **Definition:** Creates an index on one or more columns of a table to improve query performance.

Example:

sqlCopy code

```
CREATE INDEX idx_salary ON employees (salary);
```

This command creates an index named "idx_salary" on the "salary" column of the "employees" table.

5. **DROP INDEX:**
    - **Definition:** Removes an existing index from a table.

Example:

sqlCopy code

```
DROP INDEX idx_salary ON employees;
```

This command removes the "idx_salary" index from the "employees" table.

6. **CREATE DATABASE:**
    - **Definition:** Creates a new database.

Example:

sqlCopy code

```
CREATE DATABASE company;
```

This command creates a new database named "company."

7. **DROP DATABASE:**
    - **Definition:** Deletes an existing database and all its tables.

Example:

sqlCopy code

```
DROP DATABASE company;
```

This command deletes the "company" database along with all its tables.

These DDL commands are powerful and can have a significant impact on the database structure. It's important to use them with caution, especially when working with production databases, as they can result in data loss or schema changes. Always make sure to have proper backups before executing DDL commands, and thoroughly understand the consequences of the changes you are making.