MYSQL

Q) how to create databases in MYSQL?
Ans)To create a new database, you can use the `CREATE DATABASE` statement.

Syntax:- CREATE DATABASE database_name;

Example:- **CREATE DATABASE imad;**

Q)how to drop databases in MYSQL?
Ans)To drop an existing database, you can use the `DROP DATABASE` statement.

Syntax:- DROP DATABASE database_name;

Example:-**DROP DATABASE imad**;

Q)Can we provide heading to database?
Ans)In MySQL, there is no specific command to give a heading to a database. The database name itself serves as an identifier or label for the database. However, you can add comments to provide additional information or a description about the database.

To add a comment to a database, you can use the `COMMENT` clause when creating the database. Here's an example:

CREATE DATABASE mydatabase COMMENT 'This is a sample database.';

CREATE TABLE

Q)how **create tables** in MYSQL?

Ans)In MySQL, you can provide column headings or column names to tables by defining them when creating the table. Each column in the table is defined with a name and a data type.

Here's an Example of creating a table with column headings:

```
CREATE TABLE table_name (
    column1_name datatype,
    column2_name datatype,
    column3_name datatype,
    ...
);
```

For instance, let's say we want to create a table called "employees" with three columns: "id", "name", and "salary". We can define it as follows:

```
CREATE TABLE employees (
id INT,
name VARCHAR(50),
salary DECIMAL(10,2));
```

In this example, the "employees" table has three columns:

"id" (integer), "name" (varchar), and "salary" (decimal).

Once the table is created, you can use these column names to insert, update, or query data from the table. For example:

```
INSERT INTO employees (id, name, salary) VALUES (1, 'John Doe', 5000.00);
```

Q)how to check the table employees?

Ans)To check the structure and contents of the table "employees" in MySQL, you can use the `DESCRIBE` statement or the `SHOW COLUMNS` statement. Both statements provide information about the columns in a table.

1>Using DESCRIBE:

The `DESCRIBE` statement provides a concise description of the table structure, including the column names, data types, and any additional attributes.

Syntax:- **DESCRIBE employees;**

mysql> DESCRIBE employees;

```
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id     | int          | YES  |     | NULL    |       |
| name   | varchar(50)  | YES  |     | NULL    |       |
| salary | decimal(10,2) | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

2>Using SHOW COLUMNS:

3>The `SHOW COLUMNS` statement provides a more detailed view of the table structure, including information like column names, data types, column order, default values, and more.

Syntax:- **SHOW COLUMNS FROM employees;**

R) how to check this [INSERT INTO employees (id, name, salary) VALUES (1, 'John Doe', 5000.00);] in table?

Ans)The SQL query you provided is an example of an `INSERT` statement that inserts a new row into the "employees" table with values for the "id", "name", and "salary" columns.

SELECT * FROM employees;

mysql> SELECT * FROM employees;

```
+------+------------+-------------+
| id   | name       | salary      |
+------+------------+-------------+
|  570 | imaduddin | 500000.00  |
|  571 | akbar      | 75000.00  |
+------+------------+-------------+
2 rows in set (0.00 sec)
```

Q)What is **asterisk (*)** ?
Ans) The **asterisk (*)** in the `SELECT` statement represents **all columns**.
If you only want to see specific columns, you can replace the asterisk with the column names you want to retrieve. For example:

SELECT id, name, salary FROM employees;

Q) how to remove rows in employees table?
Ans)To remove rows from the "employees" table in MySQL, you can use the `DELETE` statement. The `DELETE` statement allows you to specify a condition that determines which rows to delete from the table.

Here's an example of how to use the `DELETE` statement to remove rows from the "employees" table:

DELETE FROM employees WHERE condition;

Example:-

DELETE FROM employees WHERE id = 1;

Output:-

```
mysql> select * from employees;
+------+-----------+-----------+
| id     | name      | salary     |
+------+-----------+-----------+
|   1    | John Doe   |  5000.00 |
|   1    | John Doe   |  5000.00 |
|   1    | John Doe   |  5000.00 |
|   1    | John Doe   |  5000.00 |
| 570  | imaduddin | 50000.00 |
| 571  | akbar    | 750000.00  |
+------+-----------+-----------+
```

```
6 rows in set (0.00 sec)

mysql> DELETE FROM employees WHERE id = 1;

Query OK, 4 rows affected (0.00 sec)

mysql> select * from employees;

+------+-----------+-----------+

| id   | name      | salary    |

+------+-----------+-----------+

|  570 | imaduddin | 500000.00 |

|  571 | akbar     | 750000.00 |
```

Q)How to **add values in ‘name’ column**?
Ans)To add values to the "name" column in the "details" table, you can use the `UPDATE` statement in MySQL. Here's an **example**:

```
UPDATE details
SET name = 'John Doe'
WHERE id = 1;
```

TABLE OPERATIONS

Q)Whats are the operations that can be performed in employees table?
Ans)**1>Inserting Rows:**
2>Updating Rows:
3>Altering rows
3>Deleting Rows:
4>Querying Rows:
5>Filtering Rows:
6>Sorting Rows:
7>Aggregating Data:
8>Joining Tables:

Q)How to Insert Rows in table?
Ans)To **insert rows** into the "employees" table in MySQL, you can use the `INSERT INTO` statement. Here's the general syntax:

Example:-
INSERT INTO employees (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

INSERT INTO employees (id, name, salary) VALUES (1, 'John Doe', 5000.00);

Q)How to Insert **MULTIPLE** Rows in table?
Ans)**INSERT INTO employees (id, name, salary) VALUES**
(1, 'John Doe', 5000.00),
(2, 'Jane Smith', 6000.00),
(3, 'Michael Johnson', 5500.00);

Q)Updating Rows in employees table?
Ans)Ans)To update rows in the "employees" table in MySQL, you can use the `UPDATE` statement.
syntax:
UPDATE employees SET column1 = value1, column2 = value2, ... WHERE condition;

Example:-

```
UPDATE employees SET   name='mohammed',
                       salary=15000  where id=570;

mysql>  UPDATE employees SET name='mohammed',salary=15000
where id=570;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees;
+------+----------------+-----------+
| id   | name           | salary    |
+------+----------------+-----------+
|  570 | mohammed       |  15000.00 |
|  571 | akbar          | 750000.00 |
|    1 | John Doe       |   5000.00 |
|    2 | Jane Smith     |   6000.00 |
|    3 | Michael Johnson |   5500.00 |
+------+----------------+-----------+
5 rows in set (0.00 sec)
```

Q)How to update multiple rows simultaneously using a comma-separated list of conditions in the WHERE clause. Each row needs its own separate UPDATE statement.
Ans)1> We can use the `CASE` statement in the combination with the `UPDATE` statement.
2>In this example, the `CASE` statement is used to check the value of the "name" column for each row.
3>Based on the name, the corresponding salary value is assigned.
4>The `UPDATE` statement then updates the "salary" column for all the matching rows simultaneously.
5>Make sure to adjust the column names, values, and conditions based on your specific table structure and data requirements.
6>Executing this statement will update the salary for all the employees whose names match the specified conditions in a single query.
Here's an **example**:

```
mysql> update employees
    -> set salary=case
    -> when name='mohammed' then 789652
    -> when name='akbar' then 60000
    -> when name='John Doe' then 7000
    -> when name='Jane Smith' then 120000
    -> else salary end;
```

```
mysql> select * from employees;
+------+----------------+-----------+
| id   | name           | salary    |
+------+----------------+-----------+
|  570 | mohammed       | 789652.00 |
|  571 | akbar          |  60000.00 |
|  572 | John Doe       |   7000.00 |
|  573 | Jane Smith     | 120000.00 |
|  574 | Michael Johnson |   5500.00 |
+------+----------------+-----------+
5 rows in set (0.00 sec)
```

Q)**Altering rows**?
Ans)To add a new column to an existing table in MySQL, you can use the `ALTER TABLE` statement. Here's an example:

```
 ALTER TABLE employee
ADD COLUMN age INT;
```

You can also specify additional properties for the new column, such as whether it allows NULL values, a default value, or any constraints. Here's an example that includes some additional properties:

```
ALTER TABLE employee
ADD COLUMN age INT NOT NULL DEFAULT 0;
```

Q)**Deleting Rows**?
Ans)To delete rows from a table in MySQL, you can use the `DELETE FROM` statement. Here's an example:

```
DELETE FROM details
WHERE id = 2;
```

You can also use additional conditions in the `WHERE` clause to specify more specific criteria for deleting rows. For example:

```
DELETE FROM details
WHERE id = 2 AND section = 'CSE';
```

In this updated example, the `DELETE FROM` statement deletes rows from the "details" table where the "id" column is equal to 2 and the "section" column is equal to 'CSE'.

If you want to delete all rows from the table, you can omit the `WHERE` clause:

DELETE FROM details;

This query will delete all rows from the "details" table, effectively emptying the table.

Please exercise caution when using the `DELETE FROM` statement as it permanently removes data from the table.

Q)**Querying rows?**
Ans)We can use the `SELECT` statement. The `SELECT` statement **allows you to retrieve data from one or more tables based on specified conditions.**
Here's an example:

SELECT * FROM details;

the `SELECT` statement **retrieves all columns (`*`) from the "details" table.** The result will include **all rows and columns** from the table.

You can also **specify specific columns** to retrieve by listing them instead of using `*`. For example:

SELECT id, rollnumber, section FROM details;

This query will retrieve only the "id", "rollnumber", and "section" columns from the "details" table.

To filter the rows based on specific conditions, you can use the `WHERE` clause. For example:

SELECT * FROM details WHERE section = 'CSE';

This query will retrieve all columns and rows from the "details" table where the "section" column is equal to 'CSE'.

You can combine multiple conditions using logical operators such as `AND` and `OR` in the `WHERE` clause to further refine the query.

Make sure to adjust the table name, column names, and conditions according to your specific database schema and requirements.

Q)Filtering rows?

Ans) To filter rows from a table in MySQL, you can use the `WHERE` clause in the `SELECT` statement.

The `WHERE` clause allows you to specify conditions to filter the rows based on certain criteria.

Here's an example:

SELECT * FROM details WHERE section = 'CSE' AND age > 25;

You can use various comparison operators in the `WHERE` clause to define the conditions, such as:

- `=`: Equal to
- `<>` or `!=`: Not equal to
- `<`: Less than
- `>`: Greater than
- `<=`: Less than or equal to
- `>=`: Greater than or equal to

You can also use logical operators such as `AND`, `OR`, and `NOT` to combine multiple conditions. For example:

SELECT * FROM details WHERE section = 'CSE' OR section = 'IT';
Additionally, you can use pattern matching with the `LIKE` operator and wildcard characters (`%` and _) to perform more flexible filtering.
For example:

SELECT * FROM details WHERE name LIKE 'J%';
This query retrieves all columns and rows from the "details" table where the "name" starts with the letter 'J'.
These are just a few examples of how you can filter rows using the `WHERE` clause in MySQL.
You can combine different conditions and operators to create more complex filters based on your specific requirements.

Q)**Sorting Rows?**

Ans)To sort rows in MySQL, you can use the `ORDER BY` clause in the `SELECT` statement. The `ORDER BY` clause allows you to specify the column(s) by which you want to sort the result set. Here's an example:

SELECT * FROM details ORDER BY age;

the `SELECT` statement retrieves all columns (`*`) from the "details" table and sorts the result set in ascending order based on the "age" column. The result will be displayed from the lowest age to the highest age.

You can also specify multiple columns to sort by, separating them with commas. For example:

SELECT * FROM details ORDER BY section, rollnumber;

This query retrieves all columns from the "details" table and sorts the result set first by the "section" column in ascending order, and then by the "rollnumber" column in ascending order within each section.

By default, the `ORDER BY` clause sorts the result set in ascending order. If you want to sort in descending order, you can use the `DESC` keyword. For example:

SELECT * FROM details ORDER BY age DESC;

This query retrieves all columns from the "details" table and sorts the result set in descending order based on the "age" column.

You can also mix ascending and descending sorting for different columns. For example:

SELECT * FROM details ORDER BY section ASC, age DESC;

This query retrieves all columns from the "details" table and sorts the result set first by the "section" column in ascending order and then by the "age" column in descending order within each section.

Please note that the `ORDER BY` clause should be the last part of your `SELECT` statement, after any filtering conditions specified in the `WHERE` clause.

Q)**Aggregating Data?**

Ans) To aggregate data in MySQL, you can use aggregate functions in combination with the `SELECT` statement. Aggregate functions perform **calculations on a set of rows and return a single value as the result.** Here are some commonly used aggregate functions in MySQL:

1. `COUNT`: Counts the number of rows that match a specified condition.
2. `SUM`: Calculates the sum of a column's values.
3. `AVG`: Calculates the average of a column's values.
4. `MIN`: Retrieves the minimum value from a column.
5. `MAX`: Retrieves the maximum value from a column.

Here's an example that demonstrates the usage of aggregate functions:

2> SUM:- To use the `SUM` aggregate function in MySQL, you can follow the syntax:

SELECT SUM(column_name) FROM table_name;

Here's an example that demonstrates how to use the `SUM` function to calculate the total salary from a table named `employees`:

SELECT SUM(salary) FROM employees;

3>COUNT:-To use the `COUNT` aggregate function in MySQL, you can follow this syntax:

SELECT COUNT(column_name) FROM table_name;

Here's an example of using the `COUNT` function to count the number of rows in a hypothetical "employees" table:

SELECT COUNT(*) FROM employees;

This query will return the total number of rows in the "employees" table.

Additionally, you can use `COUNT` in combination with `DISTINCT` to count the number of distinct values in a column:

SELECT COUNT(DISTINCT column_name) FROM table_name;

For example, to count the number of unique job titles in the "employees" table:

SELECT COUNT(DISTINCT job_title) FROM employees;

3> Average :- To calculate the **average (mean) of a column** in MySQL, you can use the `AVG` aggregate function.

The syntax is as follows:-

SELECT AVG(column_name) FROM table_name;

4> **Min:-** To find the **minimum value in a column** in MySQL, you can use the given `MIN` aggregate function.

The syntax is as follows:

SELECT MIN(column_name) FROM table_name;

R) Rename Table?
S) Ans)you can use the `RENAME TABLE` statement to **rename one or more tables** in a database. This statement allows you to **change the name of an existing table without altering its structure or data.** Here's the syntax for the `RENAME TABLE` statement:

```
RENAME TABLE current_table_name TO new_table_name;
```

You can also specify multiple table renames in a single `RENAME TABLE` statement by separating each rename with a comma.
Here's an example:

```
RENAME TABLE employees TO staff, departments TO divisions;
```