

MySQL CREATE, ALTER, DROP Table/Database Examples

1. CREATE DATABASE Statement

Create a new database.

```
CREATE DATABASE company_db;
```

2. ALTER DATABASE Statement

Modify database settings (e.g., changing the character set).

```
ALTER DATABASE company_db
```

```
CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

3. CREATE TABLE Statement

Create a new table in a database.

```
CREATE TABLE employees (
```

```
    employee_id INT AUTO_INCREMENT PRIMARY
```

```
    KEY, first_name VARCHAR(50),
```

```
    last_name VARCHAR(50),
```

```
    department
```

```
    VARCHAR(50), salary
```

```
    DECIMAL(10, 2)
```

```
);
```

4. ALTER TABLE Statement

Add a new column to an existing table.

ALTER TABLE employees

ADD hire_date DATE;

4. ALTER TABLE Statement (Modify Column)

Modify the data type of an existing column.

ALTER TABLE employees

MODIFY salary DECIMAL(12, 2);

5. ALTER TABLE Statement (Rename Column)

Rename a column in a table.

ALTER TABLE employees

CHANGE hire_date date_of_hire DATE;

6. DROP TABLE Statement

Remove a table from the database (with caution!).

DROP TABLE employees;

7. DROP DATABASE Statement

Remove an entire database (use with caution!).

DROP DATABASE company_db;

MySQL CREATE INDEX and DROP INDEX Commands

CREATE INDEX

The CREATE INDEX statement is used to create indexes on tables. Indexes help speed up queries.

Syntax:

```
CREATE INDEX index_name  
ON table_name (column1, column2, ...);
```

- *index_name*: The name of the index.
- *table_name*: The name of the table where the index will be created.
- (*column1*, *column2*, ...): The columns on which the index is created.

Example:

```
CREATE INDEX idx_customer_name  
ON customers (last_name,  
first_name);
```

CREATE UNIQUE INDEX

A unique index ensures that the indexed columns do not have duplicate values.

Syntax:

```
CREATE UNIQUE INDEX index_name  
ON table_name (column1, column2,  
...);
```

Example:

```
CREATE UNIQUE INDEX idx_email_unique  
ON users (email);
```

DROP INDEX

The DROP INDEX statement is used to delete an index from a table.

Syntax:

```
DROP INDEX index_name ON table_name;
```

- index_name: The name of the index to be dropped.
- table_name: The table associated with the index.

Example:

```
DROP INDEX idx_customer_name ON customers;
```

MySQL SELECT Statement Examples

1. Basic SELECT Statement

Retrieve all columns from a table.

```
SELECT * FROM employees;
```

2. Selecting Specific Columns

Retrieve specific columns from a table.

```
SELECT first_name, last_name, salary FROM employees;
```

3. Using WHERE Clause

Select rows that match a specific condition.

```
SELECT first_name, last_name FROM employees WHERE department = 'IT';
```

4. Using AND / OR Conditions

Use multiple conditions in a query.

```
SELECT first_name, last_name, salary FROM employees WHERE department = 'IT' AND salary > 50000;
```

5. Ordering the Result (ORDER BY)

Sort the result set by a specific column.

```
SELECT first_name, last_name, salary FROM employees WHERE department = 'IT' ORDER BY salary DESC;
```

6. Limiting Results (LIMIT)

Limit the number of rows returned by a query.

SELECT first_name, last_name FROM employees LIMIT 5;

7. Using LIKE for Pattern Matching

Search for a pattern in a string column.

SELECT first_name, last_name FROM employees WHERE first_name LIKE 'J%';

8. Aggregate Functions (COUNT, SUM, AVG)

- COUNT: Count the rows

***SELECT COUNT(*) FROM
employees;***

SUM: Sum values in a numeric column.

SELECT SUM(salary) FROM employees WHERE department = 'Sales';

AVG: Calculate the average of a numeric column.

SELECT AVG(salary) FROM employees;

9. Using GROUP BY for Aggregating Data

Group rows that have the same values in specified columns.

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

9. Using HAVING to Filter Aggregate Results

Filter results after aggregation.

```
SELECT department, AVG(salary) AS avg_salary FROM employees GROUP BY department  
HAVING avg_salary > 60000;
```

10. Join Queries (INNER JOIN)

Combine rows from two tables based on a related column.

```
SELECT employees.first_name, employees.last_name, departments.department_name  
FROM employees INNER JOIN departments ON employees.department_id =  
departments.department_id;
```

11. Left Join (LEFT JOIN)

Retrieve all rows from the left table, and matching rows from the right table.

```
SELECT employees.first_name, departments.department_name FROM employees LEFT JOIN  
departments ON employees.department_id = departments.department_id;
```

12. Using Aliases (AS)

Rename columns or tables for readability.

```
SELECT e.first_name, e.last_name, d.department_name FROM employees AS e JOIN  
departments AS d ON e.department_id = d.department_id;
```

MySQL UPDATE, DELETE, INSERT INTO Examples

1. INSERT INTO Statement

Insert a new row into a table.

```
INSERT INTO employees (first_name, last_name, department,  
salary) VALUES ('John', 'Doe', 'IT', 55000);
```

2. INSERT Multiple Rows

Insert multiple rows into a table.

```
INSERT INTO employees (first_name, last_name, department,  
salary) VALUES  
( 'Alice', 'Smith', 'HR', 60000),  
( 'Bob', 'Johnson', 'Sales', 45000);
```

3. UPDATE Statement

Update existing rows in a table.

```
UPDATE employees  
SET salary = 60000  
WHERE first_name = 'John' AND last_name = 'Doe';
```

4. UPDATE with Multiple Columns

Update multiple columns at once.

```
UPDATE employees  
SET salary = 65000, department =  
'Management' WHERE employee_id = 3;
```


5. DELETE Statement

Delete rows from a table.

DELETE FROM employees

WHERE first_name = 'Bob' AND last_name = 'Johnson';

6. DELETE All Rows

Delete all rows from a table (with caution!). DELETE FROM employees;

7. DELETE with LIMIT

Delete a specific number of rows.

DELETE FROM employees

ORDER BY employee_id

LIMIT 2;