Table of Contents

1. SQL Syntax Overview

SQL (Structured Query Language) is used to manage and manipulate databases.

---

2. Database and Table Management

1. Create a Database

```
CREATE DATABASE my_database;
```

---

2. Delete a Database

```
DROP DATABASE my_database;
```

---

3. Use a Database

```
USE my_database;
```

---

4. Create a Table

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

---

5. Delete a Table

```
DROP TABLE employees;
```

---

6. Modify a Table

Add a Column:

```
ALTER TABLE employees ADD age INT;
```

Delete a Column:

```
ALTER TABLE employees DROP COLUMN age;
```

Rename a Column:

```
ALTER TABLE employees RENAME COLUMN name TO full_name;
```

---

3. Inserting Data

1. Insert Single Row

```
INSERT INTO employees (id, name, position, salary)
VALUES (1, 'Alice', 'Manager', 75000);
```

---

2. Insert Multiple Rows

```
INSERT INTO employees (id, name, position, salary)
VALUES
(2, 'Bob', 'Developer', 60000),
(3, 'Carol', 'Analyst', 55000);
```

---

4. Querying Data (SELECT)

1. Select All Columns

```
SELECT * FROM employees;
```

---

2. Select Specific Columns

```
SELECT name, salary FROM employees;
```

---

3. Rename Columns (Alias)

```
SELECT name AS employee_name, salary AS earnings FROM employees;
```

---

4. Distinct Values

```
SELECT DISTINCT position FROM employees;
```

---

5. Filtering Data (WHERE Clause)

1. Basic WHERE Clause

```
SELECT * FROM employees WHERE salary > 60000;
```

---

2. Multiple Conditions (AND/OR)

```
SELECT * FROM employees
WHERE salary > 50000 AND position = 'Developer';
```

---

3. Range Filter (BETWEEN)

```
SELECT * FROM employees
WHERE salary BETWEEN 50000 AND 70000;
```

---

4. Set Membership (IN)

```
SELECT * FROM employees
WHERE position IN ('Manager', 'Analyst');
```

---

5. Pattern Matching (LIKE)

```
SELECT * FROM employees
```

```
WHERE name LIKE 'A%';  -- Names starting with A
```

- % – Matches any number of characters
- _ – Matches a single character

---

6. Sorting Data (ORDER BY)

1. Sort by Single Column

```
SELECT * FROM employees
ORDER BY salary DESC;
```

---

2. Sort by Multiple Columns

```
SELECT * FROM employees
ORDER BY position ASC, salary DESC;
```

---

7. Limiting and Paginating Results

1. Limit the Number of Results

```
SELECT * FROM employees LIMIT 5;
```

---

2. Paginate Results (OFFSET)

```
SELECT * FROM employees LIMIT 5 OFFSET 10;
```

---

8. Aggregation and Grouping

1. Aggregate Functions

```
SELECT COUNT(*) FROM employees;          -- Count
SELECT AVG(salary) FROM employees;       -- Average
SELECT MAX(salary) FROM employees;       -- Maximum
SELECT MIN(salary) FROM employees;       -- Minimum
SELECT SUM(salary) FROM employees;       -- Sum
```

---

2. Grouping Data (GROUP BY)

```
SELECT position, AVG(salary)
FROM employees
GROUP BY position;
```

---

3. Filtering Groups (HAVING)

```
SELECT position, COUNT(*)
FROM employees
GROUP BY position
HAVING COUNT(*) > 1;
```

---

9. Updating Data

1. Update a Single Row

```
UPDATE employees
SET salary = 80000
WHERE id = 1;
```

---

2. Update Multiple Rows

```
UPDATE employees
SET position = 'Senior Developer'
WHERE position = 'Developer';
```

---

10. Deleting Data

1. Delete Specific Rows

```
DELETE FROM employees
WHERE id = 3;
```

---

2. Delete All Rows

```
DELETE FROM employees;
```

---

11. Table Joins

1. Inner Join

```
SELECT employees.name, departments.name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.id;
```

---

2. Left Join

```
SELECT employees.name, departments.name
FROM employees
LEFT JOIN departments
```

```
ON employees.department_id = departments.id;
```

---

3. Right Join

```
SELECT employees.name, departments.name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.id;
```

---

4. Full Outer Join

```
SELECT employees.name, departments.name
FROM employees
FULL OUTER JOIN departments
ON employees.department_id = departments.id;
```

---

12. Subqueries

1. Subquery in WHERE Clause

```
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

---

2. Subquery in SELECT

```
SELECT name,
       (SELECT AVG(salary) FROM employees) AS average_salary
FROM employees;
```

---

13. Indexing

1. Create an Index

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

---

2. Drop an Index

```
DROP INDEX idx_salary;
```

---

14. Backup and Restore

1. Backup a Database

```
mysqldump -u username -p my_database > backup.sql
```

---

2. Restore a Database

```
mysql -u username -p my_database < backup.sql
```

---

15. Common SQL Functions

String Functions

```
SELECT UPPER(name) FROM employees;  -- Uppercase
SELECT LOWER(name) FROM employees;  -- Lowercase
SELECT LENGTH(name) FROM employees; -- String Length
```

Date Functions

```
SELECT NOW();              -- Current date and time
SELECT YEAR(hire_date);    -- Extract Year
SELECT DATEDIFF(NOW(), hire_date) FROM employees; -- Date difference
```

Tips for SQL

- Always backup your database before performing large operations.
- Use LIMIT to avoid selecting large datasets unintentionally.
- Test queries in a sandbox environment before applying to production.
- Use JOINs to efficiently combine data from multiple tables.