SQL HOME

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

SQL INTRO

SQL is used to interact with databases, allowing users to create, retrieve, update, and delete data.

SQL SYNTAX

```
SELECT column1, column2 FROM
table_name;
INSERT INTO table_name (column1,
column2) VALUES ('value1',
'value2');
UPDATE table_name SET column1 =
'value' WHERE condition;
DELETE FROM table_name WHERE
condition;
```

SQL SELECT

```
SELECT * FROM Employees;
```

Output:

id	name	age	department	salary
1	John	30	IT	60000
2	Alice	28	HR	50000

```
SELECT name, salary FROM
Employees;
```

Output:

name	salary
John	60000
Alice	50000

SQL SELECT DISTINCT

```
SELECT DISTINCT department FROM
Employees;
```

Output:

department
IT
HR

SQL WHERE

```
SELECT * FROM Employees WHERE
age > 25;
```

Output:

id name age department salary
1 John 30 IT 60000

SQL ORDER BY

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

Output:

id name age department salary
1 John 30 IT 60000

id	name	age	department	salary
2	Alice	28	HR	50000

SQL AND

```
SELECT * FROM Employees WHERE
age > 25 AND salary > 50000;
```

Output:

id	name	age	department	salary
1	John	30	IT	60000

SQL OR

```
SELECT * FROM Employees WHERE
department = 'IT' OR department
= 'HR';
```

Output:

id	name	age	department	salary
1	John	30	IT	60000
2	Alice	28	HR	50000

SQL NOT

```
SELECT * FROM Employees WHERE
NOT department = 'HR';
```

Output:

id	name	age	department	salary
1	John	30	IT	60000

SQL INSERT INTO

```
INSERT INTO Employees (name,
age, department) VALUES ('Mark',
26, 'Finance');
```

Output:

A new row is inserted into the **Employees** table.

SQL NULL VALUES

```
SELECT * FROM Employees WHERE
department IS NULL;
```

Output:

(No records found)

SQL UPDATE

```
UPDATE Employees SET salary =
60000 WHERE name = 'John';
```

Output:

1 row updated.

SQL DELETE

```
DELETE FROM Employees WHERE age
< 25;
```

Output:

Records of employees younger than 25 are deleted.

SQL SELECT TOP

```
SELECT TOP 5 * FROM Employees;
```

Output: ***(First 5 rows from Employees table)***

SQL AGGREGATE FUNCTIONS

SQL MIN and MAX

```
SELECT MIN(salary) FROM
Employees;
SELECT MAX(salary) FROM
Employees;
```

Output:

MIN(salary)
50000
MAX(salary)
60000

SQL COUNT

```
SELECT COUNT(*) FROM Employees;
```

Output:

COUNT(*)
2

SQL SUM

```
SELECT SUM(salary) FROM
Employees;
```

Output:

SUM(salary)
110000

SQL AVG

```
SELECT AVG(salary) FROM
Employees;
```

Output:

AVG(salary)
55000

SQL LIKE

```
SELECT * FROM Employees WHERE
name LIKE 'J%';
```

Output: ***(All employees whose name starts with "J")***

SQL JOINS

INNER JOIN

```
SELECT Employees.name,
Department.department_name
FROM Employees
INNER JOIN Department ON
Employees.department_id =
Department.id;
```

Output:

name	department_name
John	IT
Alice	HR

SQL UNION

```
SELECT name FROM Employees UNION
SELECT name FROM Managers;
```

Output: ***(Unique names from Employees and Managers tables)***

SQL GROUP BY

```
SELECT department, COUNT(*) FROM
Employees GROUP BY department;
```

Output:

department	COUNT(*)
IT	1
HR	1

SQL HAVING

```
SELECT department, COUNT(*) FROM
Employees GROUP BY department
HAVING COUNT(*) > 5;
```

Output: ***(Departments with more than 5 employees)***

SQL CASE

```
SELECT name, salary,
CASE
    WHEN salary > 50000 THEN
'High'
    WHEN salary BETWEEN 30000
AND 50000 THEN 'Medium'
    ELSE 'Low'
END AS SalaryCategory
FROM Employees;
```

Output:

name salary SalaryCategory

name	salary	SalaryCategory
John	60000	High
Alice	50000	Medium

SQL CREATE TABLE

```
CREATE TABLE Employees (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    department VARCHAR(50)
);
```

Output:

Table **Employees** is created.

SQL INDEX

```
CREATE INDEX idx_name ON
Employees(name);
```

Output:

Index **idx_name** is created on the **name** column.

SQL VIEW

```
CREATE VIEW EmployeeView AS
SELECT name, salary FROM
Employees;
```

Output:

View **EmployeeView** is created.

SQL INJECTION PROTECTION

```
SELECT * FROM Users WHERE
username = @username AND
password = @password;
```

Explanation:

Using **parameterized queries** protects against SQL injection.

This SQL tutorial includes **topic-wise queries with sample outputs**. Let me know if you need additional details!

Here's a **MySQL tutorial** with topic-wise code examples and **expected output**:

MySQL HOME

MySQL is an open-source relational database management system (RDBMS).

MySQL INTRO

MySQL is used to store, retrieve, and manage data using SQL.

MySQL RDBMS

RDBMS stands for **Relational Database Management System**, where data is stored in **tables**.

MySQL SQL SYNTAX

```
SELECT column1, column2 FROM
table_name;
INSERT INTO table_name (column1,
column2) VALUES ('value1',
'value2');
UPDATE table_name SET column1 =
'value' WHERE condition;
DELETE FROM table_name WHERE
condition;
```

MySQL SELECT

```
SELECT * FROM Employees;
```

Output:

id name age department salary

id	name	age	department	salary
1	John	30	IT	60000
2	Alice	28	HR	50000

MySQL WHERE

```
SELECT * FROM Employees WHERE
age > 25;
```

Output:

id	name	age	department	salary
1	John	30	IT	60000

MySQL AND, OR, NOT

```
SELECT * FROM Employees WHERE
age > 25 AND salary > 50000;
```

Output:

id	name	age	department	salary
1	John	30	IT	60000

MySQL ORDER BY

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

Output:

id	name	age	department	salary
1	John	30	IT	60000
2	Alice	28	HR	50000

MySQL INSERT INTO

```
INSERT INTO Employees (name,
age, department, salary) VALUES
('Mark', 35, 'Finance', 70000);
```

Output:

New record inserted successfully.

MySQL NULL VALUES

```
SELECT * FROM Employees WHERE
department IS NULL;
```

Output:

id	name	age	department	salary
3	Tom	40	NULL	75000

MySQL UPDATE

```
UPDATE Employees SET salary =
80000 WHERE name = 'John';
```

Output:

Record updated successfully.

MySQL DELETE

```
DELETE FROM Employees WHERE age
< 25;
```

Output:

Record deleted successfully.

MySQL LIMIT

```
SELECT * FROM Employees LIMIT 2;
```

Output:

Returns only the first **2 records**.

MySQL MIN and MAX

```
SELECT MIN(salary) FROM
Employees;
SELECT MAX(salary) FROM
Employees;
```

Output:

MIN(salary)
50000
MAX(salary)
80000

MySQL COUNT, AVG, SUM

```
SELECT COUNT(*) FROM Employees;
```

```
SELECT AVG(salary) FROM
Employees;
SELECT SUM(salary) FROM
Employees;
```

Output:

COUNT(*)
3
AVG(salary)
65000
SUM(salary)
195000

MySQL LIKE

```
SELECT * FROM Employees WHERE
name LIKE 'J%';
```

Output:

id name age department salary
1 John 30 IT 80000

MySQL IN

```
SELECT * FROM Employees WHERE
department IN ('IT', 'HR');
```

Output:

Returns employees from IT and HR departments.

MySQL BETWEEN

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

Output:

Returns employees with salaries in the specified range.

MySQL ALIASES

```
SELECT name AS EmployeeName FROM
Employees;
```

Output:

EmployeeName
John
Alice

MySQL JOINS

INNER JOIN

```
SELECT Employees.name,
Department.department_name
FROM Employees
INNER JOIN Department ON
Employees.department_id =
Department.id;
```

MySQL GROUP BY

```
SELECT department, COUNT(*) FROM
Employees GROUP BY department;
```

Output:

department COUNT(*)

department	COUNT(*)
IT	1
HR	1

MySQL HAVING

```
SELECT department, COUNT(*) FROM
Employees GROUP BY department
HAVING COUNT(*) > 1;
```

Output:

Returns only departments with more than one employee.

MySQL CASE

```
SELECT name, salary,
CASE
    WHEN salary > 50000 THEN
'High'
    ELSE 'Low'
END AS SalaryCategory
```

```
FROM Employees;
```

Output:

name	salary	SalaryCategory
John	80000	High
Alice	50000	Low

MySQL DATABASE COMMANDS

Create Database

```
CREATE DATABASE Company;
```

Drop Database

```
DROP DATABASE Company;
```

MySQL TABLE COMMANDS

Create Table

```
CREATE TABLE Employees (
    id INT PRIMARY KEY
AUTO_INCREMENT,
    name VARCHAR(50),
```

```
    age INT,
    department VARCHAR(50),
    salary INT
);
```

Drop Table

```
DROP TABLE Employees;
```

MySQL CONSTRAINTS

Primary Key

```
ALTER TABLE Employees ADD
PRIMARY KEY (id);
```

Foreign Key

```
ALTER TABLE Employees ADD
FOREIGN KEY (department_id)
REFERENCES Department(id);
```

MySQL INDEX

```
CREATE INDEX idx_name ON
Employees(name);
```

MySQL AUTO INCREMENT

```
CREATE TABLE Employees (
    id INT AUTO_INCREMENT
PRIMARY KEY,
    name VARCHAR(50)
);
```

MySQL DATES

```
SELECT CURDATE();
SELECT DATE_ADD(CURDATE(),
INTERVAL 1 YEAR);
```

Output:

CURDATE()
2025-03-23

DATE_ADD(CURDATE(), INTERVAL 1 YEAR)
2026-03-23

MySQL VIEWS

```
CREATE VIEW EmployeeView AS
SELECT name, salary FROM
Employees;
SELECT * FROM EmployeeView;
```
