**Create Database and Tables**

-- Create Database

CREATE DATABASE company\_db;

USE company\_db;

-- Create Employees Table

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

salary DECIMAL(10, 2),

hire\_date DATE,

department\_id INT

);

-- Create Departments Table

CREATE TABLE departments (

department\_id INT AUTO\_INCREMENT PRIMARY KEY,

department\_name VARCHAR(50)

);

-- Create Sales Table

CREATE TABLE sales (

sale\_id INT AUTO\_INCREMENT PRIMARY KEY,

employee\_id INT,

sale\_amount DECIMAL(10, 2),

sale\_date DATE,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id)

);

**2. Insert Data into Tables**

-- Insert Data into Employees

INSERT INTO employees (name, salary, hire\_date, department\_id) VALUES

('Alice Johnson', 75000, '2021-01-15', 1),

('Bob Smith', 60000, '2022-03-22', 1),

('Charlie Brown', 55000, '2020-07-19', 2),

('Daisy Miller', 80000, '2019-09-01', 2),

('Eva Green', 45000, '2023-01-10', 3);

-- Insert Data into Departments

INSERT INTO departments (department\_name) VALUES

('HR'),

('Finance'),

('Engineering');

-- Insert Data into Sales

INSERT INTO sales (employee\_id, sale\_amount, sale\_date) VALUES

(1, 1500.00, '2024-07-01'),

(1, 2000.00, '2024-07-15'),

(2, 1200.00, '2024-06-15'),

(3, 1000.00, '2024-05-10'),

(4, 2500.00, '2024-04-20'),

(5, 800.00, '2024-07-05');

**3. Basic Queries**

1. **Select All Data from Employees**

SELECT \* FROM employees;

**Output:**

| **employee\_id** | **name** | **salary** | **hire\_date** | **department\_id** |
| --- | --- | --- | --- | --- |
| 1 | Alice Johnson | 75000 | 2021-01-15 | 1 |
| 2 | Bob Smith | 60000 | 2022-03-22 | 1 |
| 3 | Charlie Brown | 55000 | 2020-07-19 | 2 |
| 4 | Daisy Miller | 80000 | 2019-09-01 | 2 |
| 5 | Eva Green | 45000 | 2023-01-10 | 3 |

1. **Select All Data from Departments**

SELECT \* FROM departments;

**Output:**

| **department\_id** | **department\_name** |
| --- | --- |
| 1 | HR |
| 2 | Finance |
| 3 | Engineering |

**4. Using SQL Functions and Clauses**

1. **Using GROUP BY and Aggregate Functions**

-- Find the average salary and total number of employees in each department

SELECT department\_id,

AVG(salary) AS Average\_Salary,

COUNT(\*) AS Number\_Of\_Employees

FROM employees

GROUP BY department\_id;

**Output:**

| **department\_id** | **Average\_Salary** | **Number\_Of\_Employees** |
| --- | --- | --- |
| 1 | 67500.00 | 2 |
| 2 | 67500.00 | 2 |
| 3 | 45000.00 | 1 |

1. **Using ORDER BY**

sql

Copy code

-- List employees in descending order of their salary

SELECT name,

salary

FROM employees

ORDER BY salary DESC;

**Output:**

| **name** | **salary** |
| --- | --- |
| Daisy Miller | 80000 |
| Alice Johnson | 75000 |
| Bob Smith | 60000 |
| Charlie Brown | 55000 |
| Eva Green | 45000 |

1. **Using HAVING**

sql

Copy code

-- Find departments with an average salary greater than 60000

SELECT department\_id,

AVG(salary) AS Average\_Salary

FROM employees

GROUP BY department\_id

HAVING AVG(salary) > 60000;

**Output:**

| **department\_id** | **Average\_Salary** |
| --- | --- |
| 1 | 67500.00 |
| 2 | 67500.00 |

1. **Using LIMIT**

sql

Copy code

-- Get the top 2 highest-paid employees

SELECT name,

salary

FROM employees

ORDER BY salary DESC

LIMIT 2;

**Output:**

| **name** | **salary** |
| --- | --- |
| Daisy Miller | 80000 |
| Alice Johnson | 75000 |

1. **Using String Functions**

-- Extract the first 3 characters of employee names and convert them to uppercase

SELECT name,

UPPER(SUBSTRING(name, 1, 3)) AS Short\_Name

FROM employees;

**Output:**

| **name** | **Short\_Name** |
| --- | --- |
| Alice Johnson | ALI |
| Bob Smith | BOB |
| Charlie Brown | CHA |
| Daisy Miller | DAI |
| Eva Green | EVA |

1. **Using Numeric Functions**

-- Round the salary of each employee to the nearest thousand

SELECT name,

ROUND(salary, -3) AS Rounded\_Salary

FROM employees;

**Output:**

| **name** | **Rounded\_Salary** |
| --- | --- |
| Alice Johnson | 75000 |
| Bob Smith | 60000 |
| Charlie Brown | 60000 |
| Daisy Miller | 80000 |
| Eva Green | 50000 |

1. **Using Date/Time Functions**

-- Find the number of employees hired in the last 2 years

SELECT COUNT(\*) AS Number\_Of\_New\_Hires

FROM employees

WHERE hire\_date > DATE\_SUB(CURDATE(), INTERVAL 2 YEAR);

**Output:**

| **Number\_Of\_New\_Hires** |
| --- |
| 2 |

1. **Using Aggregate Functions**

-- Calculate the total sales amount and average sale amount per employee

SELECT employee\_id,

SUM(sale\_amount) AS Total\_Sales,

AVG(sale\_amount) AS Average\_Sale

FROM sales

GROUP BY employee\_id;

**Output:**

| **employee\_id** | **Total\_Sales** | **Average\_Sale** |
| --- | --- | --- |
| 1 | 3500.00 | 1750.00 |
| 2 | 1200.00 | 1200.00 |
| 3 | 1000.00 | 1000.00 |
| 4 | 2500.00 | 2500.00 |
| 5 | 800.00 | 800.00 |

1. **Using Subqueries**

-- Find employees whose salary is above the average salary of their department

SELECT name,

salary

FROM employees e

WHERE salary > (

SELECT AVG(salary)

FROM employees

WHERE department\_id = e.department\_id

);

**Output:**

| **name** | **salary** |
| --- | --- |
| Alice Johnson | 75000 |
| Daisy Miller | 80000 |

10. **Using HAVING with AVG and SUM**

-- Find departments where the total salary exceeds 100000 and the average salary is greater than 50000

SELECT department\_id,

SUM(salary) AS Total\_Salary,

AVG(salary) AS Average\_Salary

FROM employees

GROUP BY department\_id

HAVING Total\_Salary > 100000 AND Average\_Salary > 50000;

1. Find the names and salaries of employees hired in the year 2021.

SELECT name, salary FROM employees WHERE YEAR(hire\_date) = 2021;

1. List all employees whose name starts with 'A' and ends with 'e'. **Hint:** Use string pattern matching.

SELECT name, salary FROM employees WHERE name LIKE 'A%e';