Task 7: Creating Views

Objective:

Learn to create and use views in SQL for abstraction, reuse, and security.

Tools:

DB Browser for SQLite

MySQL Workbench

1. What is a View in SQL?

A view is a virtual table based on the result-set of a SQL SELECT query. It doesn't store data itself; it pulls data from one or more tables.

2. Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

3. Example Setup: Tables:

```
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    Name TEXT,
    Department TEXT,
    Salary INT
);

CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName TEXT,
```

Task 7: Creating Views

```
    Location TEXT
);
```

Sample Data:

```
INSERT INTO Employees VALUES (101, 'Alice', 'HR', 50000);
INSERT INTO Employees VALUES (102, 'Bob', 'IT', 75000);
INSERT INTO Employees VALUES (103, 'Charlie', 'Finance', 60000);
```

4. Creating a View: Example 1: High Salary Employees

```
CREATE VIEW HighSalary AS
SELECT Name, Salary
FROM Employees
WHERE Salary > 60000;
```

Using the View:

```
SELECT * FROM HighSalary;
```

Example 2: Department-wise Employee Info

```
CREATE VIEW DeptInfo AS
SELECT E.Name, E.Department, D.Location
FROM Employees E
JOIN Departments D ON E.Department = D.DeptName;
```

5. Updating Data through Views

If the view is simple (based on 1 table, no GROUP BY/aggregates), you can update it:

```
UPDATE HighSalary SET Salary = 80000 WHERE Name = 'Bob';
```

6. Dropping a View

Task 7: Creating Views

```
DROP VIEW IF EXISTS HighSalary;
```

7. Use Cases

- Hiding sensitive columns (Security)
- Simplifying complex joins
- Reusability in applications
- Logical data independence

Outcome:

You now understand:

- How to create views
- How to use views for queries
- How views support abstraction and simplify SQL logic