SQL Views – Complete Guide

1. What is a View in SQL?

✅ A **View** is a **virtual table** based on the result of an SQL query.
✅ It does **not store data physically**, only the SQL logic.
✅ Data shown in a view comes from base tables.

---

2. When & Where to Use a View?

🔹 **Use Cases:**

- **Data abstraction**: Hide complex joins or calculations.
- **Security**: Restrict column or row-level access.
- **Simplification**: Simplify repeated complex queries.
- **Read-only dashboard reports**.

🔹 **When to Use:**

- When multiple users need access to **specific filtered data**.
- When logic is complex but needs **reusability and clarity**.
- When needing a **layer of security** over sensitive columns.

---

3. How Views Work Internally?

✅ The **view query is stored** in the data dictionary.
✅ When queried, MySQL **executes the stored SELECT query dynamically**.
✅ So, **changes in base tables reflect in the view automatically**.

---

4. How to Create a View?

Syntax:

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

Example:

```
CREATE VIEW active_customers AS
SELECT id, name, email
FROM customers
WHERE status = 'Active';
```

---

5. How to Use/View the View?

```
SELECT * FROM active_customers;
```

✅ Acts like a normal table for read operations.

---

6. How to Drop a View?

Syntax:

```
DROP VIEW view_name;
```

Example:

```
DROP VIEW active_customers;
```

---

7. Where Are Views Stored in SQL?

✅ Views are stored in the `information_schema.VIEWS` table.

To list all views:

```
SHOW FULL TABLES IN your_database WHERE TABLE_TYPE LIKE 'VIEW';
```

To see view definition:

```
SHOW CREATE VIEW view_name;
```

Complete SQL View Query Example – Step-by-Step Guide

```
-- Step 1: Create the customers table

CREATE TABLE customers (
   id INT PRIMARY KEY,
   name VARCHAR(50),
   email VARCHAR(100),
   status VARCHAR(20)
);

-- Step 2: Insert sample data

INSERT INTO customers (id, name, email, status) VALUES
(1, 'Vijay', 'vijay@example.com', 'Active'),
(2, 'Harsha', 'harsha@example.com', 'Inactive'),
(3, 'Ravi', 'ravi@example.com', 'Active'),
(4, 'Srinivas', 'srinu@example.com', 'Inactive');

-- Step 3: Create a view for active customers only

CREATE VIEW active_customers AS
SELECT id, name, email
FROM customers
WHERE status = 'Active';
```

-- Step 4: Query the view

- **SELECT * FROM active_customers;**

-- Step 5: Show all views in the current database

- **SHOW FULL TABLES WHERE TABLE_TYPE = 'VIEW';**

-- Step 6: Show the definition of a specific view

- **SHOW CREATE VIEW active_customers;**

Advantages of Views:

1. **Data Security:**
   Restrict access to sensitive columns/rows by exposing only selected data.

2. **Simplification:**
   Hide complex joins or logic, making it easier for users to query.

3. **Reusability:**
   Write once, reuse multiple times in different queries or reports.

Disadvantages of Views:

1. **Performance Overhead:**
   Complex views (especially nested or with joins) can slow down queries.

2. **Limited DML Support:**
   Some views are not updatable (e.g., those with `GROUP BY`, `JOIN`, or `DISTINCT`).
