Views

What is a Views?
A view is a saved SQL query treated like a virtual table. It presents a specific representation of data from
one or more tables without storing the data separately (unless materialized).

Step 1 — Create sample tables (Employees, Departments)

Run these statements in your SQL client to create tables and insert sample data.

```
CREATE TABLE Departments (DeptID INT PRIMARY KEY, DeptName
VARCHAR(50));

CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, Name
VARCHAR(50), DeptID INT, Salary INT);

INSERT INTO Departments (DeptID, DeptName) VALUES (10,
'Sales'), (20, 'HR'), (30, 'IT');

 INSERT INTO Employees (EmployeeID, Name,
DeptID, Salary) VALUES
(1, 'John', 10, 5000),
(2, 'Alice', 10, 7000),
 (3, 'Bob', 20, 4500),
 (4, 'Eve', 20, 6000),
(5, 'Carol', 30, 3000);
```

Step 2 — Create a simple view

Create a view that shows employee names with their department IDs.
`CREATE VIEW EmployeeNames AS SELECT EmployeeID, Name, DeptID FROM Employees;`
Use the view like a table: `SELECT * FROM EmployeeNames;`

Step 3 — Create a view with JOIN (combined data)

Create a view that shows employee name with department name using a JOIN.

```
CREATE VIEW EmpDept AS SELECT e.EmployeeID, e.Name, d.DeptName,
e.Salary FROM
Employees e JOIN Departments d ON e.DeptID = d.DeptID;
```

Then query: `SELECT * FROM EmpDept WHERE DeptName = 'Sales';`
Sample output (EmpDept WHERE DeptName='Sales'):

Step 4 — Updatable views & WITH CHECK OPTION (step-by-step)

Some views are updatable (you can INSERT, UPDATE, DELETE through them).
General rules for a view to be updatable:
- It typically references a single base table (no aggregates or GROUP BY).
- Does not use DISTINCT, GROUP BY, UNION, or complex expressions on columns you want to update.
- Has the necessary permissions and writable underlying columns.

Example: Create a filtered view and update through it.

```
CREATE VIEW SalesEmployees AS SELECT EmployeeID, Name, DeptID, Salary
FROM Employees
WHERE DeptID = 10; -- Update via view (if your DB supports updates
through views)
UPDATE SalesEmployees SET Salary = Salary + 500 WHERE EmployeeID =
```

`1;`WITH CHECK OPTION ensures any INSERT/UPDATE through the view does not produce rows outside
the view's WHERE clause. Example:

```
CREATE VIEW SalesEmployeesChk AS SELECT EmployeeID, Name, DeptID,
Salary FROM
Employees WHERE DeptID = 10 WITH CHECK OPTION;
```

Step 5 — Materialized / Indexed Views (concept)

A materialized view stores the result set on disk and must be refreshed when base data changes —
useful for fast reads with expensive queries. An indexed view (SQL Server) or materialized view

example:

```
CREATE MATERIALIZED VIEW DeptSalary AS SELECT DeptID,
AVG(Salary) AS
AvgSalary FROM Employees GROUP BY DeptID;
```

Step 6 — Use views for security and abstraction

Views help expose only required columns or rows to certain users. Steps to secure data using views: 1.
Create a view that selects only non-sensitive columns (or filters rows).
2. GRANT SELECT on the view to specific users, but not on the base table.
Example:

```
CREATE VIEW PublicEmployees AS SELECT EmployeeID, Name, DeptID
FROM
Employees;
GRANT SELECT ON PublicEmployees TO readonly_user;
```

Step 7 — Drop, Replace, and Alter views

To remove a view: `DROP VIEW view_name;`
To replace a view in some DBs you can use CREATE OR REPLACE VIEW: `CREATE OR REPLACE
VIEW view_name AS …`

Step 8 — Limitations & best practices

• Views can hide complexity but may hide performance costs — a view might expand into a heavy query
at runtime.
• Avoid using views for frequently updated OLTP paths if they are complex.
• Be careful with nested views (views built on views) — they can be harder to debug.

.Interview Questions

1.What is a Views?

A virtual table defined by a SQL query. It does not necessarily store data by itself.

2.Can we update data through a view?
Sometimes — if the view is updatable (simple, single-table mappings) and the DB allows it.

3.What is a materialized view?
A stored result set that can be refreshed; used for performance.

4.Difference between view and table?
A table stores data; a view is a saved query representing data from tables.

5.How to drop a view?
```
DROP VIEW view_name;
```

6.Why use Views?
To simplify queries, provide abstraction, and secure/limit data exposure.

7.Can we create indexed views?
Some DBs support indexed or materialized views to improve performance.

8.How to secure data using views?
Expose only needed columns/rows via a view and grant permissions on the view, not the base table.

9.What are Limitations of views?
Performance overhead, non-updatability for complex views, and nested-view complexity.

10.How does WITH CHECK OPTION work?
It prevents INSERT/UPDATE through the view that would produce rows not visible in the view (enforces
the view's WHERE clause).