**SQL Views** and **Indexes** along with **practical use cases** — especially useful for interviews or real-time projects.

**✅ 1. SQL Views**

**🔷 What is a View?**

A **View** is a **virtual table** based on the result of a SQL query. It doesn’t store data physically but fetches from the original table(s) when queried.

**🔸 Syntax**

CREATE VIEW view\_name AS

SELECT column1, column2 FROM table\_name WHERE condition;

**🔸 Use Cases**

| **Use Case** | **Description** |
| --- | --- |
| 1. **Security / Access Restriction** | Only expose specific columns to users. |

CREATE VIEW vw\_EmployeeBasicInfo AS

SELECT EmpID, Name FROM Employees;

| 2. **Simplify Complex Queries** | Reuse long joins/subqueries using a view.

CREATE VIEW vw\_OrderDetails AS

SELECT o.OrderID, c.CustomerName, p.ProductName

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON o.ProductID = p.ProductID;

| 3. **Data Aggregation / Reporting** | Show summarized data like totals, counts.

CREATE VIEW vw\_DeptSalarySummary AS

SELECT Department, AVG(Salary) AS AvgSalary

FROM Employees GROUP BY Department;

| 4. **Join Multiple Tables** | Combine related data from multiple tables.  
| 5. **Mask Sensitive Columns** | Hide confidential data like salaries or passwords.

**✅ 2. SQL Indexes**

**🔷 What is an Index?**

An **index** is a performance-tuning method that allows faster retrieval of records from a table. It's like a book index – helping you locate data faster.

**🔸 Syntax**

CREATE INDEX idx\_Employee\_Name ON Employees(Name);

**🔸 Use Cases**

| **Use Case** | **Description** |
| --- | --- |
| 1. **Speed up SELECT queries** | Particularly on large datasets with WHERE or JOIN conditions. |

SELECT \* FROM Employees WHERE Name = 'John';

| 2. **Improve JOIN performance** | Index on join keys improves efficiency.

CREATE INDEX idx\_Order\_CustomerID ON Orders(CustomerID);

| 3. **Optimize Sorting (ORDER BY)** | Index helps speed up queries with sorting.  
| 4. **Speed up GROUP BY / Aggregate Functions** | Especially when grouping on indexed columns.  
| 5. **Partial Indexing** | Index only rows that meet a condition (supported in PostgreSQL, etc.)

**🔸 Types of Indexes**

| **Type** | **Description** |
| --- | --- |
| **Clustered Index** | Reorders the table physically. (Only one per table) |

In SQL Server, the **primary key** is a clustered index by default.  
| **Non-Clustered Index** | Does not affect physical order. Can have **many**.  
| **Unique Index** | Prevents duplicate values.  
| **Composite Index** | Created on multiple columns.

CREATE INDEX idx\_name\_dob ON Employees(Name, DOB);

| **Full-text Index** | Used for fast text searches.  
| **Bitmap Index** | Efficient for low-cardinality columns (Oracle).

**🔸 Important Notes**

* Indexes **slow down INSERT/UPDATE/DELETE** (because the index also needs to be updated).
* Choose indexes wisely for frequently **searched or filtered** columns.
* Avoid over-indexing — it can increase maintenance time and storage.