SQL Interview Preparation Guide (Brief Version)

----
1) Primary Key, Foreign Key, Unique Key, Composite Key

- Primary Key: Unique, not null. Example: EmpID in Employee table.
- Foreign Key: References another table. Example: DeptID in Employee table referencing Department.
- Unique Key: Unique, allows one NULL. Example: DeptName unique.
- Composite Key: Multiple columns as key. Example: StudentID+CourseID in StudentCourse.

----
2) Constraints

Rules to maintain data integrity: NOT NULL, PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK, DEFAULT.
Example: Salary CHECK(Salary>0), JoiningDate DEFAULT GETDATE().

----
3) Normalization

Organizing tables to reduce redundancy and dependency.
Example: Splitting StudentPhone into separate rows.

----
4) Normal Forms (1NF,2NF,3NF,BCNF,4NF,5NF)

- 1NF: Atomic values. Example: Separate phone numbers into rows.
- 2NF: No partial dependency. Example: Separate Course table.
- 3NF: No transitive dependency. Example: DeptName in separate table.
- BCNF: Stronger 3NF, every determinant is candidate key.
- 4NF: No multi-valued dependency. Example: Separate StudentHobby & StudentLanguage.
- 5NF: Remove join dependency.

----
5) Temp Table

Temporary table for session. Example: CREATE TABLE #TempEmployee(...).

----
6) View

Virtual table based on query. Example: CREATE VIEW ActiveEmployees AS SELECT * FROM Employee WHERE Status='Active';

----
7) WITH Keyword / CTE

Temporary result set for query readability. Example:
WITH DeptSalary AS (SELECT DeptID, AVG(Salary) AS AvgSal FROM Employee GROUP BY DeptID)
SELECT * FROM DeptSalary WHERE AvgSal>50000;

----
8) Types of JOIN

INNER, LEFT, RIGHT, FULL, CROSS. Example: SELECT * FROM A INNER JOIN B ON A.ID=B.ID;

----
9) Self Join

Joining table with itself. Example:
SELECT E1.EmpName, E2.EmpName AS Manager FROM Employee E1 JOIN Employee E2 ON E1.ManagerID=E2.EmpID;

----
10) GROUP BY

Groups rows for aggregates. Example: SELECT DeptID, AVG(Salary) FROM Employee GROUP BY DeptID;

----
11) Aggregate Functions

COUNT, SUM, AVG, MAX, MIN. Example: SELECT AVG(Salary) FROM Employee;

----
12) Query Execution Flow

FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY

----
13) Aggregate in WHERE?

Not allowed, use HAVING instead.

----
14) WHERE vs HAVING

WHERE: before grouping, HAVING: after grouping. Example: HAVING AVG(Salary)>50000;

----
15) HAVING before WHERE?

Not allowed, WHERE executes first.

----
16) Window Functions

Operate across set of rows. ROW_NUMBER(), RANK(), DENSE_RANK(), SUM() OVER(), AVG() OVER().
Example: SELECT EmpID, Salary, RANK() OVER(ORDER BY Salary DESC) AS Rank FROM Employee;

----
17) Stored Procedure

Reusable SQL statements. Example: CREATE PROCEDURE GetEmployees AS SELECT * FROM

Employee;

----
18) Functions

Return value from SQL code. Example: CREATE FUNCTION GetYear(@Date DATE) RETURNS INT AS RETURN YEAR(@Date);

----
19) Types of Functions

System (LEN, GETDATE), User Defined Functions (UDF).

----
20) User Defined Function

Custom logic function. Example: GetYear(@Date).

----
21) Insert/Update/Delete in Function?

Not allowed in UDF.

----
22) Trigger

Executes on table events. Example:
CREATE TRIGGER trgAfterInsert ON Employee AFTER INSERT AS PRINT 'Record Inserted';

----
23) Cursor

Fetch row-by-row. Example: DECLARE cur CURSOR FOR SELECT EmpName FROM Employee; OPEN cur; FETCH NEXT FROM cur;

----
24) SQL Injection & Prevention

Malicious SQL via input. Prevent using Parameterized Queries / Stored Procedures.
Example: SqlCommand cmd = new SqlCommand("SELECT * FROM Users WHERE Username=@user", con);

----
25) Transaction

Unit of work, ACID properties. Example:
BEGIN TRAN; UPDATE Account SET Balance=Balance-500 WHERE AccID=1;
UPDATE Account SET Balance=Balance+500 WHERE AccID=2; IF @@ERROR<>0 ROLLBACK ELSE COMMIT;

----
26) SQL Components

DDL: CREATE, ALTER, DROP; DML: INSERT, UPDATE, DELETE; DCL: GRANT, REVOKE; TCL: COMMIT, ROLLBACK; DQL: SELECT

----

27) Rollback

Undo uncommitted changes. Example: BEGIN TRAN; DELETE FROM Employee; ROLLBACK;

----
28) Indexing

Improves query speed. Types: Clustered, Non-Clustered, Unique, Filtered, Composite.
Example: CREATE CLUSTERED INDEX IX_EmployeeID ON Employee(EmpID);

----
29) Clustered vs Non-Clustered Index

Clustered: Physical order, 1 per table; Non-Clustered: Logical, many allowed.
Example: CREATE NONCLUSTERED INDEX IX_Name ON Employee(EmpName);

----
30) Subquery

Query inside another. Example: SELECT * FROM Employee WHERE Salary>(SELECT AVG(Salary) FROM Employee);

----
31) IN vs EXISTS vs ANY

IN: match list/subquery; EXISTS: returns true if subquery has rows; ANY: compares to any value in subquery.

----
32) DELETE vs TRUNCATE vs DROP

DELETE: remove rows, can rollback; TRUNCATE: remove all rows, faster; DROP: remove table.

----
33) CTE

Common Table Expression. Example:
WITH EmpCTE AS (SELECT * FROM Employee) SELECT * FROM EmpCTE;

----
34) Pivot

Rotate rows into columns. Example:
SELECT Year, [A],[B] FROM (SELECT Year, Product, Amount FROM Sales) src
PIVOT (SUM(Amount) FOR Product IN ([A],[B])) pvt;

----
35) UNION vs UNION ALL

UNION: removes duplicates; UNION ALL: keeps duplicates.

Queries:
1) Second Largest Salary:
SELECT MAX(Salary) FROM Employee WHERE Salary < (SELECT MAX(Salary) FROM Employee);

2) Employees > Department Avg:

```sql
SELECT EmpName, Salary, DeptID FROM Employee e
WHERE Salary>(SELECT AVG(Salary) FROM Employee WHERE DeptID=e.DeptID);
```

3) Nth Highest Salary:

```sql
SELECT Salary FROM (SELECT Salary,DENSE_RANK() OVER(ORDER BY Salary DESC) AS rnk
FROM Employee) t WHERE rnk=3;
```

4) Employees without Manager:

```sql
SELECT E1.EmpName FROM Employee E1 LEFT JOIN Employee E2 ON
E1.ManagerID=E2.EmpID WHERE E2.EmpID IS NULL;
```

5) Top 2 salaries per dept:

```sql
SELECT * FROM (SELECT EmpName, Salary, DeptID, ROW_NUMBER() OVER(PARTITION BY
DeptID ORDER BY Salary DESC) AS rn FROM Employee) t WHERE rn<=2;
```

6) Count of employees per dept:

```sql
SELECT DeptID, COUNT(*) FROM Employee GROUP BY DeptID;
```