
✅SQL Interview Questions (All Levels)

□ Basic Level

1. **What is SQL? What are the different types of SQL commands?**

Answer:
SQL (Structured Query Language) is a standard language used to **communicate with relational databases** for performing operations such as querying, inserting, updating, and deleting data.

SQL command categories:

- **DDL (Data Definition Language):** `CREATE`, `ALTER`, `DROP`, `TRUNCATE`
- **DML (Data Manipulation Language):** `INSERT`, `UPDATE`, `DELETE`
- **DQL (Data Query Language):** `SELECT`
- **DCL (Data Control Language):** `GRANT`, `REVOKE`
- **TCL (Transaction Control Language):** `COMMIT`, `ROLLBACK`, `SAVEPOINT`

2. **What is a primary key and foreign key?**

Answer:

- **Primary Key:**
  - Uniquely identifies each row in a table.
  - Cannot have NULLs.
  - Only one primary key per table (can be composite).
  - Example: `EmployeeID` in an `Employees` table.
- **Foreign Key:**
  - Creates a link between two tables.
  - Refers to the primary key in another table.
  - Ensures referential integrity.
  - Example: `DepartmentID` in `Employees` referencing `Departments`.

3. **What is the difference between WHERE and HAVING clauses?**

Answer:

Clause	Purpose	Used With
`WHERE`	Filters rows **before** aggregation	Any SQL query
`HAVING`	Filters results **after** aggregation (GROUP BY)	Aggregate queries

Example:

```
sql
CopyEdit
SELECT Department, COUNT(*)
FROM Employees
WHERE Active = 1
GROUP BY Department
HAVING COUNT(*) > 5;
```

4. **What are joins? Explain different types of joins (INNER, LEFT, RIGHT, FULL).**

Answer:
Joins are used to **retrieve data from multiple tables** based on related columns.

- **INNER JOIN**: Returns records that match in both tables.
- **LEFT JOIN (LEFT OUTER)**: All records from the left table + matched records from the right.
- **RIGHT JOIN (RIGHT OUTER)**: All records from the right table + matched from the left.
- **FULL JOIN (FULL OUTER)**: All records when there's a match in either table.

Example (INNER JOIN):

```
sql
CopyEdit
SELECT e.Name, d.DepartmentName
FROM Employees e
INNER JOIN Departments d ON e.DeptId = d.Id;
```

5. **What is normalization? What are its types?**

Answer:
Normalization is the process of **organizing data to reduce redundancy** and improve data integrity.

Common normal forms:

- **1NF (First Normal Form):** Remove repeating groups, use atomic values.
- **2NF:** Remove partial dependencies (applies to composite keys).
- **3NF:** Remove transitive dependencies.
- **BCNF:** A stricter version of 3NF.

In my experience, normalization helps maintain **data consistency** and simplifies **database maintenance**, but in some cases (e.g., reporting), denormalization is used for performance.

6. **What is a view? How is it different from a table?**

Answer:
A **view** is a **virtual table** based on a SELECT query. It does **not store data** physically, only the SQL logic.

Differences:

Feature	Table	View
Data storage	Stores data	No data, just a definition
Editable	Yes	Sometimes (if no joins, group by, etc.)
Performance	Faster (indexed)	Slightly slower (computed at runtime)

Use views to **simplify complex queries**, apply **security** (restricted columns), or create **reusable business logic**.

7. **What is the difference between DELETE, TRUNCATE, and DROP?**

Answer:

Command	Deletes Data	Removes Structure	Rollback Possible	Speed
DELETE	Yes	No	Yes (if in transaction)	Slower (row-by-row)
TRUNCATE	Yes (all rows)	No	No	Fast
DROP	Yes	Yes	No	Fastest

Example:

```
sql
CopyEdit
DELETE FROM Employees WHERE DeptId = 5; -- Conditional delete
TRUNCATE TABLE Employees; -- Deletes all data
DROP TABLE Employees; -- Deletes table structure and data
```

8. **What is indexing? Why is it important?**

Answer:
An index is a **performance optimization** structure that allows faster data retrieval.

- Works like a book index — speeds up searches on large tables.
- **Types**: Clustered, Non-clustered, Unique, Composite, Full-text
- SQL Server uses **B-Trees** internally.

Example:

```
sql
CopyEdit
CREATE NONCLUSTERED INDEX idx_name ON Employees(Name);
```

In my experience, indexing **significantly improves SELECT** performance but should be balanced as it can **slow down INSERT/UPDATE** operations.

9. **What is a stored procedure? How do you call it?**

Answer:
A **stored procedure** is a precompiled group of SQL statements saved in the database, often used for **modular and secure database logic**.

Creating:

```
sql
CopyEdit
CREATE PROCEDURE GetEmployeeById
    @EmpId INT
AS
BEGIN
    SELECT * FROM Employees WHERE Id = @EmpId;
END
```

Calling:

```
sql
CopyEdit
EXEC GetEmployeeById @EmpId = 1;
```

I use stored procedures for **reusable logic, better performance, and security** (e.g., role-based execution).

10. **What is a trigger in SQL?**

Answer:
A **trigger** is a **special stored procedure** that automatically runs **in response to INSERT, UPDATE, or DELETE events** on a table.

Example:

```
sql
CopyEdit
CREATE TRIGGER trg_LogDelete
ON Employees
AFTER DELETE
AS
BEGIN
    INSERT INTO EmployeeLog (EmpId, DeletedOn)
    SELECT Id, GETDATE() FROM deleted;
END
```

Triggers are useful for **auditing**, **enforcing business rules**, and **automatic logging**, but I use them cautiously due to their hidden execution and potential performance impact.

 Intermediate Level

1. **Write a SQL query to fetch the second-highest salary from an employee table.**

Answer:

```
sql
CopyEdit
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

✅Alternate using `ROW_NUMBER()`:

```
sql
CopyEdit
SELECT Salary
FROM (
  SELECT Salary, ROW_NUMBER() OVER (ORDER BY Salary DESC) AS rn
  FROM Employees
) AS ranked
WHERE rn = 2;
```

2. **How do you find duplicate rows in a table?**

Answer:

```
sql
CopyEdit
```

```
SELECT Name, COUNT(*)
FROM Employees
GROUP BY Name
HAVING COUNT(*) > 1;
```

If checking across multiple columns:

```
sql
CopyEdit
SELECT Name, DeptId, COUNT(*)
FROM Employees
GROUP BY Name, DeptId
HAVING COUNT(*) > 1;
```

3. **Explain ACID properties in SQL.**

Answer:

Property	Description
Atomicity	All steps in a transaction are completed or none at all.
Consistency	The database remains in a valid state before and after the transaction.
Isolation	Concurrent transactions are isolated from each other.
Durability	Once committed, changes remain even if the system crashes.

These ensure **data integrity** and **reliable transactions**.

4. **What is the difference between clustered and non-clustered index?**

Answer:

Feature	Clustered Index	Non-Clustered Index
Storage	Sorts and stores the actual data rows	Stores pointers to data rows
Count per table	Only one (because data can be sorted one way)	Can have multiple
Speed	Faster for range queries	Slower for large reads but flexible

Example:

```
sql
CopyEdit
CREATE CLUSTERED INDEX idx_emp_id ON Employees(Id);
CREATE NONCLUSTERED INDEX idx_emp_name ON Employees(Name);
```

5. **How do you optimize a slow-running SQL query?**

Answer:
I usually follow these steps:

- Use **indexes** on frequently searched columns.
- Avoid `SELECT *`; only select required columns.
- Analyze **execution plan** to find bottlenecks.

- Optimize **joins** (especially nested or cross joins).
- Replace **subqueries** with **joins** when applicable.
- Use **WHERE**, **LIMIT**, or **TOP** to reduce scanned rows.
- Check for **parameter sniffing** and use **option (recompile)** if needed.

6. **What is the difference between UNION and UNION ALL?**

Answer:

Feature	UNION	UNION ALL
Duplicates	Removes duplicates	Includes duplicates
Performance	Slower due to distinct sort	Faster
Use case	When you need distinct records	When duplicates are okay

```
sql
CopyEdit
SELECT Name FROM Table1
UNION
SELECT Name FROM Table2;

-- vs
SELECT Name FROM Table1
UNION ALL
SELECT Name FROM Table2;
```

7. **What are subqueries and correlated subqueries?**

Answer:

- **Subquery**: A query inside another query, executed once.

```
sql
CopyEdit
SELECT Name FROM Employees
WHERE DeptId = (SELECT Id FROM Departments WHERE Name = 'HR');
```

- **Correlated Subquery**: Uses data from the outer query and runs **per row**.

```
sql
CopyEdit
SELECT e.Name
FROM Employees e
WHERE Salary > (SELECT AVG(Salary) FROM Employees WHERE DeptId = e.DeptId);
```

8. **How do you implement transactions in SQL Server?**

Answer:

```
sql
CopyEdit
BEGIN TRANSACTION;

BEGIN TRY
    UPDATE Accounts SET Balance = Balance - 100 WHERE Id = 1;
```

```
    UPDATE Accounts SET Balance = Balance + 100 WHERE Id = 2;

    COMMIT;
END TRY
BEGIN CATCH
    ROLLBACK;
    -- Log error
END CATCH;
```

I use transactions for **data consistency** in multi-step operations.

9. **What is a CTE (Common Table Expression)?**

Answer:
A **CTE** is a temporary result set defined within a `WITH` clause, used to simplify complex queries like recursion or joins.

Example:

```
sql
CopyEdit
WITH TopEmployees AS (
  SELECT Name, Salary
  FROM Employees
  WHERE Salary > 50000
)
SELECT * FROM TopEmployees;
```

✅Used in recursive queries like hierarchical (tree/manager) structures.

10. **What is a temporary table? When do you use it?**

Answer:
A **temporary table** is a table created in **tempdb** for **storing intermediate results** during session execution.

Syntax:

```
sql
CopyEdit
CREATE TABLE #TempEmployees (Id INT, Name VARCHAR(50));
INSERT INTO #TempEmployees SELECT Id, Name FROM Employees;
```

✅Use cases:

- When processing large intermediate datasets.
- For breaking complex queries into parts.
- Used in **stored procedures** to store results temporarily.

 Advanced Level

1. **How do you handle deadlocks in SQL Server?**

Answer:
A **deadlock** occurs when two or more processes block each other by holding locks the other needs.

To handle deadlocks:

- Use **`TRY...CATCH`** to handle the deadlock error (error code 1205).
- Use `SET DEADLOCK_PRIORITY LOW` to allow lower priority processes to be chosen as the victim.
- Avoid long-running transactions and **keep transactions short**.
- Always access resources in the **same order** across procedures.
- Use **`WITH (NOLOCK)`** cautiously for reads when real-time consistency is not critical.

You can also detect them using **SQL Server Profiler**, **Extended Events**, or `sys.dm_tran_locks`.

2. **Explain isolation levels in SQL and their impact on performance.**

Answer:
Isolation levels define how one transaction **is isolated from the effects of others**:

Isolation Level	Read Uncommitted	Read Committed	Repeatable Read	Serializable	Snapshot
Dirty Read	✓	✗	✗	✗	✗
Non-repeatable Read	✓	✓	✗	✗	✗
Phantom Read	✓	✓	✓	✗	✗
Locks Taken	Low	Moderate	High	Highest	None
Performance	Fastest	Balanced	Slower	Slowest	High (uses TempDB)

I typically use **Read Committed (default)**, and **Snapshot** when high concurrency is required with minimal blocking.

3. **What are indexed views and when should you use them?**

Answer:
An **indexed view** is a materialized view where the **results are stored on disk** with a clustered index.

Use when:

- Query involves complex **joins/aggregations** and is frequently used.
- You need **faster reads** and can tolerate slower writes (as changes to base tables update the view index).
- Under strict conditions (e.g., deterministic functions, schema binding).

Example:

```
sql
CopyEdit
CREATE VIEW SalesSummary WITH SCHEMABINDING AS
SELECT ProductId, SUM(Amount) AS Total
FROM dbo.Sales
GROUP BY ProductId;

CREATE UNIQUE CLUSTERED INDEX idx_SalesSummary ON SalesSummary(ProductId);
```

4. **How do you implement pagination in SQL Server?**

Answer:
Using `OFFSET` and `FETCH NEXT` in SQL Server 2012+:

```
sql
CopyEdit
SELECT Name, Salary
FROM Employees
ORDER BY Id
OFFSET 10 ROWS FETCH NEXT 10 ROWS ONLY;
```

For earlier versions:

```
sql
CopyEdit
WITH CTE AS (
  SELECT ROW_NUMBER() OVER (ORDER BY Id) AS RowNum, Name
  FROM Employees
)
SELECT * FROM CTE WHERE RowNum BETWEEN 11 AND 20;
```

5. **What is a performance execution plan? How do you read it?**

Answer:
An **execution plan** shows how SQL Server **executes a query** — the steps and their cost.

View it using:

- `SET SHOWPLAN_XML ON` / SSMS "Display Estimated Execution Plan" (Ctrl+L)
- "Actual Execution Plan" after running a query (Ctrl+M)

Key elements to watch:

- **Table Scans** (slow) vs. **Index Seeks** (fast)
- **Join types** (Nested Loops, Merge Join, Hash Join)
- **Operator costs (%)**
- **Missing index suggestions**

I regularly tune queries using the execution plan and add **covering indexes** where needed.

6. **How can you monitor long-running queries in SQL Server?**

Answer:
You can monitor them using:

- **Activity Monitor** in SSMS
- `sys.dm_exec_requests` to view currently running queries
- `sys.dm_exec_query_stats` to find historically slow queries
- **Query Store** (SQL Server 2016+)
- **Extended Events** for custom tracking
- **Profiler** or **third-party tools** like Redgate or SolarWinds

Example:

```
sql
CopyEdit
SELECT * FROM sys.dm_exec_requests
WHERE status = 'running' AND total_elapsed_time > 10000;
```

7. **What are SQL Server functions (scalar, table-valued)?**

Answer:

- **Scalar Function**: Returns a single value.

```
sql
CopyEdit
CREATE FUNCTION fn_GetAge(@DOB DATE) RETURNS INT AS
BEGIN
    RETURN DATEDIFF(YEAR, @DOB, GETDATE());
END
```

- **Inline Table-Valued Function (iTVF)**: Returns a table using a single `SELECT`.

```
sql
CopyEdit
CREATE
```

8. **How do you implement full-text search in SQL Server?**

Answer:
Steps:

1. Enable **Full-Text Search** feature in SQL Server.
2. Create a **Full-Text Catalog** and **Full-Text Index**.
3. Use `CONTAINS` or `FREETEXT` for querying.

Example:

```
sql
CopyEdit
SELECT * FROM Articles
WHERE CONTAINS(Content, '("AI" AND "Machine Learning")');
```

It's powerful for **natural language searches** on large text columns like blog content or product descriptions.

9. **What is the difference between IN, EXISTS, and ANY?**

Operator	Checks	Best Used When
`IN`	List of static or subquery values	Subquery is small or fixed
`EXISTS`	Subquery returns rows	Subquery is large and indexed
`ANY`	Compares with any returned value	With =, >, < comparisons

Examples:

```
sql
```

```
CopyEdit
-- IN
SELECT * FROM Employees WHERE DeptId IN (SELECT Id FROM Departments WHERE Name = 'HR');

-- EXISTS
SELECT * FROM Employees e WHERE EXISTS (SELECT 1 FROM Departments d WHERE d.Id = e.DeptId AND
d.Name = 'HR');

-- ANY
SELECT * FROM Employees WHERE Salary > ANY (SELECT Salary FROM Employees WHERE DeptId = 3);
```

10. **Have you used Window Functions (ROW_NUMBER(), RANK(), LEAD(), LAG())? Explain with examples.**

Answer:
Yes, I regularly use window functions for **analytics and pagination**.

- `ROW_NUMBER()`: Unique row number per partition/order
- `RANK()`: Gives same rank for ties (gaps in sequence)
- `DENSE_RANK()`: Same rank for ties (no gaps)
- `LEAD()/LAG()`: Access next/previous row's value

Example:

```
sql
CopyEdit
-- Get top 3 salaries per department
SELECT Name, DepartmentId, Salary,
       RANK() OVER (PARTITION BY DepartmentId ORDER BY Salary DESC) AS Rank
FROM Employees;

-- Compare current and previous salary
SELECT Name, Salary,
       LAG(Salary) OVER (ORDER BY HireDate) AS PreviousSalary
FROM Employees;
```

Window functions are great for **running totals, comparisons, pagination, and trends**.