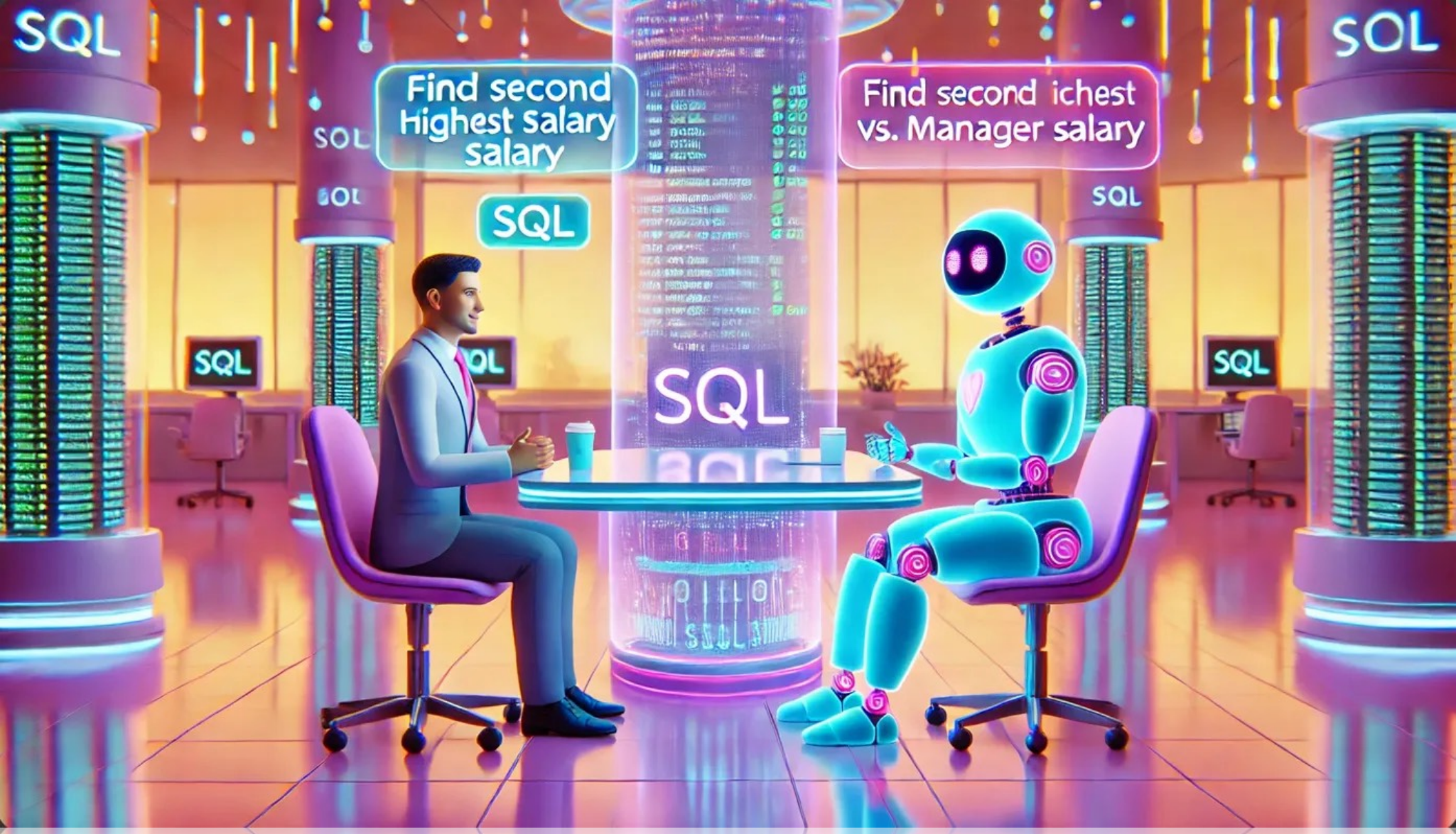
Ahmed Abdulwahid

Top 40 SQL Interview Questions & Answers 🏆

#DataGenius



SQL (Structured Query Language) is the backbone of databases, making it a crucial skill for data professionals. Whether you're a data scientist, data analyst, or backend developer, mastering SQL is non-negotiable. 💡

This epic deep dive 🔥 will help you ace SQL interviews by covering the Top 40 SQL Questions with detailed explanations, examples, and pro tips! 🚀📊

🟢 Beginner-Level SQL Interview Questions

1. What is SQL?

SQL (Structured Query Language) is used to store, manipulate, and retrieve data from relational databases. It's the standard language for interacting with databases like MySQL, PostgreSQL, SQL Server, and Oracle.

2. What are the types of SQL commands?

SQL commands are categorized into:

✅ *DDL (Data Definition Language) – `CREATE`, `ALTER`, `DROP`, `TRUNCATE`*

✅ *DML (Data Manipulation Language) – `INSERT`, `UPDATE`, `DELETE`*

✅ *DQL (Data Query Language) – `SELECT`*

✅ *TCL (Transaction Control Language) – `COMMIT`, `ROLLBACK`, `SAVEPOINT`*

✅ *DCL (Data Control Language) – `GRANT`, `REVOKE`*

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

🔍 DELETE

- *removes specific rows based on a WHERE condition (or all rows if no WHERE is provided).*
- *can be rolled back (if inside a transaction).*
- *is slower because it logs each row deletion.*
- *keeps the table structure and indexes intact.*
- *Type: DML (Data Manipulation Language) – modifies data but doesn't affect the table structure.*
- *Example:*

```sql
DELETE FROM employees WHERE department = 'HR';
```

⚡ TRUNCATE

- *removes all rows from a table.*
- *cannot be rolled back (it's a DDL command).*
- *is faster than DELETE because it doesn't log row deletions.*
- *keeps the table structure and indexes.*
- *resets identity columns (if present).*
- *Type: DDL (Data Definition Language) – modifies table structure but keeps the table itself.*
- *Example:*

```
TRUNCATE TABLE employees;
```

💣 DROP

- *completely removes the table, including its structure, indexes, and constraints.*

- *cannot be rolled back (it's a permanent action).*
- *is the fastest because it removes everything in one go.*
- *makes the table no longer exist after execution.*
- *Type: DDL (Data Definition Language) – removes the entire table from the database.*
- *Example:*

```
DROP TABLE employees;
```

📊 Key Differences Between DROP, TRUNCATE, and DELETE

🔑 Feature	🗑️ DROP	🚄 TRUNCATE	✏️ DELETE
Removes Data	✅ Yes	✅ Yes	✅ Yes
Removes Table Structure	✅ Yes	❌ No	❌ No
Can Be Rolled Back	❌ No *(unless in transaction)*	❌ No *(mostly)*	✅ Yes *(with transaction)*
Supports WHERE Clause	❌ No	❌ No	✅ Yes
Speed	⚡ **Fastest**	🚀 **Faster**	🦕 **Slower**
Affects Indexes/Constraints	✅ Yes *(drops them)*	❌ No *(keeps them)*	❌ No *(keeps them)*
Typical Use Case	**Permanently** delete a table	**Quickly** clear all data	**Selectively** delete specific rows

4. What is a Primary Key? 🔑

A Primary Key uniquely identifies each record in a table. It cannot have NULL values and must be unique.

```sql
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    name VARCHAR(100),
    salary DECIMAL(10,2)
);
```

5. What is a Foreign Key? 🌍

A Foreign Key maintains referential integrity between two tables. It links a column in one table to the Primary Key of another.

6. What is the difference between `WHERE` and `HAVING`?

```sql
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    emp_id INT,
    FOREIGN KEY (emp_id) REFERENCES employees(emp_id)
);
```

6. What is the difference between WHERE and HAVING?

Clause	Used With	Filters on
WHERE	`SELECT`, `UPDATE`, `DELETE`	**Rows before aggregation**
HAVING	`GROUP BY`	**Aggregated results**

Example:

```sql
SELECT department, AVG(salary)
FROM employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

7. Find the second highest salary.

```sql
SELECT MAX(salary) AS SecondHighestSalary
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

8. What is a Self Join?

A Self Join is when a table joins itself. It's useful for comparing rows within the same table.

9. Find employees who earn more than their managers.

```sql
SELECT e.name, e.salary
FROM employees e
JOIN employees m ON e.manager_id = m.emp_id
WHERE e.salary > m.salary;
```

10. Retrieve the top 3 highest salaries.

```sql
SELECT DISTINCT salary FROM employees ORDER BY salary DESC LIMIT 3;
```

11. What are Window Functions? 🪟

Window functions perform calculations across a set of table rows related to the current row.

Example — Running total of salaries per department:

```sql
SELECT name, department,
       salary,
       SUM(salary) OVER (PARTITION BY department ORDER BY salary)
                                                        AS running_total
FROM employees;
```

12. Find duplicate records in a table.

```sql
SELECT name, COUNT(*) FROM employees GROUP BY name HAVING COUNT(*) > 1;
```

13. Find the department with the highest total salary.

```sql
SELECT department, SUM(salary) AS TotalSalary
FROM employees GROUP BY department ORDER BY TotalSalary DESC LIMIT 1;
```

14. Find employees who joined in the last 6 months.

```sql
SELECT * FROM employees
WHERE hire_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH);
```

15. Write a query to display employee details along with their department names.

```sql
SELECT e.name, e.salary, d.department_name
FROM employees e
JOIN departments d ON e.department_id = d.department_id;
```

16. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN?

Join Type	Description
INNER JOIN	Returns matching rows in both tables
LEFT JOIN	Returns all rows from the left table + matching rows from the right
RIGHT JOIN	Returns all rows from the right table + matching rows from the left
FULL JOIN	Returns all rows from both tables

17. Write a SQL query to find the Nth highest salary.

```sql
SELECT DISTINCT salary FROM employees
ORDER BY salary DESC LIMIT 1 OFFSET N-1;
```

18. What is normalization? Why is it important?

Normalization is the process of structuring a database to reduce redundancy and improve data integrity.

Normal Forms (NF):

🔹 *1NF: Ensure atomicity (no multiple values in a single column).*

🔹 *2NF: Remove partial dependencies (every non-key column depends on the whole primary key).*

🔹 *3NF: Remove transitive dependencies (non-key columns shouldn't depend on other non-key columns).*

It improves consistency, efficiency, and scalability in databases. 🚀

19. What is denormalization? When should you use it?

Denormalization is the process of combining tables to improve read performance by reducing joins. It sacrifices redundancy for speed.

When to Use It?

✅ *Read-heavy applications (e.g., reporting dashboards).*

✅ *Minimizing complex joins for faster queries.*

✅ *Data warehouses, where speed is prioritized over storage efficiency.*

🚀 *Example: Storing customer details inside an Orders table instead of a separate Customers table to avoid frequent joins.*

20. What is the difference between `UNION` and `UNION ALL`?

✅ *`UNION` removes duplicates, while `UNION ALL` keeps all records (including duplicates).*

21. Find employees who have the same salary.

```sql
SELECT e1.name, e1.salary FROM employees e1
JOIN employees e2 ON e1.salary = e2.salary AND e1.emp_id != e2.emp_id;
```

🔵 Intermediate-Level SQL Interview Questions

22. What is a Common Table Expression (CTE)?

A CTE (Common Table Expression) is a temporary result set that simplifies complex queries and improves readability.

```sql
WITH DepartmentSalary AS (
    SELECT department_id, AVG(salary) AS avg_salary
    FROM employees
    GROUP BY department_id
)
SELECT e.name, e.salary, d.avg_salary
FROM employees e
JOIN DepartmentSalary d ON e.department_id = d.department_id
WHERE e.salary > d.avg_salary;
```

23. What is the difference between RANK(), DENSE_RANK(), and ROW_NUMBER()?

- ✅ *RANK(): Skips ranks when duplicates exist.*
- ✅ *DENSE_RANK(): Does not skip ranks.*
- ✅ *ROW_NUMBER(): Assigns a unique row number.*

```sql
SELECT name, salary,
       RANK() OVER (ORDER BY salary DESC) AS rank,
       DENSE_RANK() OVER (ORDER BY salary DESC) AS dense_rank,
       ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_number
FROM employees;
```

24. Find the third highest salary using DENSE_RANK().

```sql
SELECT name, salary FROM (
    SELECT name, salary,
           DENSE_RANK() OVER (ORDER BY salary DESC) AS rnk
    FROM employees
) t WHERE rnk = 3;
```

25. What is a Recursive CTE? Explain with an example.

Recursive CTEs allow queries to reference themselves, often used for hierarchical data (e.g., organizational structures).

```sql
WITH RECURSIVE EmployeeHierarchy AS (
    SELECT emp_id, name, manager_id
    FROM employees
    WHERE manager_id IS NULL
    UNION ALL
    SELECT e.emp_id, e.name, e.manager_id
    FROM employees e
    JOIN EmployeeHierarchy eh ON e.manager_id = eh.emp_id
)
SELECT * FROM EmployeeHierarchy;
```

26. How do you find employees with the highest salary in each department?

```sql
SELECT e.name, e.salary, e.department_id
FROM employees e
WHERE salary =
  (SELECT MAX(salary) FROM employees WHERE department_id = e.department_id);
```

27. How do you swap two column values in SQL?

```sql
UPDATE employees
SET salary = department_id,
    department_id = salary;
```

28. Find the running total of salaries ordered by date.

```sql
SELECT name, salary,
       SUM(salary) OVER (ORDER BY hire_date) AS running_total
FROM employees;
```

29. What is the difference between EXISTS and IN?

- ✅ *EXISTS: Returns TRUE if a subquery returns at least one row (more efficient for large data).*

- ✅ *IN: Checks if a value exists in a list (less efficient for larger datasets).*

```sql
SELECT name FROM employees e
WHERE EXISTS (
    SELECT 1 FROM managers m WHERE m.manager_id = e.emp_id
);
```

30. Find employees who do not have any direct reports.

```sql
SELECT name FROM employees e
WHERE NOT EXISTS (
    SELECT 1 FROM employees m WHERE m.manager_id = e.emp_id
);
```

31. What is database indexing? Why is it important?

Indexes speed up data retrieval by creating an ordered structure. Use them on frequently queried columns.

```sql
CREATE INDEX IX_salary ON employees(salary);
```

32. Explain the difference between Clustered and Non-Clustered Index.

Index Type	Description
Clustered	Sorts and stores data **physically** (Only one per table)
Non-Clustered	Creates a separate structure (Can have multiple)

33. How do you optimize a slow SQL query? ⚡

- ✅ *Use Indexing on large datasets.*
- ✅ *Avoid `SELECT *`, specify needed columns.*
- ✅ *Use `EXISTS` instead of `IN` for subqueries.*
- ✅ *Optimize Joins by filtering before joining.*
- ✅ *Use `LIMIT` or `TOP` to restrict query size.*

34. What is a Stored Procedure? How is it different from a Function?

- ✅ *Stored Procedures: Can perform multiple SQL operations and return multiple results.*
- ✅ *Functions: Must return a single value.*

```sql
CREATE PROCEDURE GetHighSalaryEmployees()
BEGIN
    SELECT * FROM employees WHERE salary > 100000;
END;
```

35. What are Triggers in SQL?

Triggers automatically execute when an event occurs in a table.

```sql
CREATE TRIGGER after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
    INSERT INTO audit_log (action, emp_id, timestamp)
    VALUES ('INSERT', NEW.emp_id, NOW());
END;
```

36. Find the cumulative sum of salaries per department.

```sql
SELECT name, department, salary,
       SUM(salary) OVER (PARTITION BY department ORDER BY salary)
                                                    AS cumulative_salary
FROM employees;
```

🔴 Advanced SQL Interview Questions

37. What is the difference between OLAP and OLTP?

🔷 **OLTP (Fast Transactions)**

- ✅ *Handles real-time transactions (banking, e-commerce).*
- ✅ *Focuses on speed & accuracy (INSERT, UPDATE, DELETE).*
- ✅ *Uses normalized data (avoids redundancy).*
- ✅ *Small, frequent operations.*

🔷 **OLAP (Deep Analysis)**

- 📊 *Used for reporting & analytics (BI, dashboards).*
- 📊 *Works with historical, aggregated data.*
- 📊 *Uses denormalized data (fast queries).*
- 📊 *Complex SELECT queries for insights.*

🔥 **OLTP = Fast & Small 🏎️ | OLAP = Big & Smart 📊**

38. How do you work with Time-Series Data in SQL?

Working with Time-Series Data in SQL ⏳

🔷 *1. Filtering by Date/Time*

```sql
SELECT * FROM sales WHERE order_date >= '2024-01-01';
```

🔷 *2. Aggregating Over Time (Daily, Monthly, Yearly)*

```sql
SELECT DATE_TRUNC('month', order_date) AS month, SUM(revenue)
FROM sales GROUP BY month;
```

🔷 *3. Using Window Functions (Running Total, Moving Average)*

```sql
SELECT order_date, revenue,
SUM(revenue) OVER (ORDER BY order_date
                   ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS moving_avg
FROM sales;
```

39. How do you solve Churn problem in SQL 📉

Churn Analysis in SQL 📉

🔷 *1. Identify Active & Churned Customers*

💡 *Helps track when each customer last made a purchase.*

```sql
SELECT customer_id, MAX(last_purchase_date) AS last_order
FROM orders
GROUP BY customer_id;
```

🔷 *2. Define Churn (e.g., No Orders in 90 Days)*

💡 *Finds customers who haven't purchased in 90+ days.*

```sql
SELECT customer_id
FROM (SELECT customer_id, MAX(last_purchase_date) AS last_order
      FROM orders GROUP BY customer_id) AS last_orders
WHERE last_order < NOW() - INTERVAL '90 days';
```

🔷 3. Churn Rate Calculation

💡 % of customers who left vs. total customers.

```sql
SELECT COUNT(DISTINCT customer_id) * 100.0 /
       (SELECT COUNT(DISTINCT customer_id) FROM customers) AS churn_rate
FROM orders
WHERE last_purchase_date < NOW() - INTERVAL '90 days';
```

🔷 4. Cohort Analysis (Tracking Churn Over Time)

💡 Tracks how different signup groups retain or churn over time.

```sql
SELECT DATE_TRUNC('month', signup_date) AS cohort,
       COUNT(DISTINCT customer_id) AS total_customers,
       COUNT(DISTINCT CASE WHEN last_purchase_date <
                               NOW() - INTERVAL '90 days'
                               THEN customer_id END) AS churned_customers
FROM customers
GROUP BY cohort;
```

40. What is the Execution Plan?

An Execution Plan shows how a SQL query is executed by the database, detailing the steps, indexes, and operations used.

Why is it Important?

✅ *Helps optimize queries by identifying slow operations.*

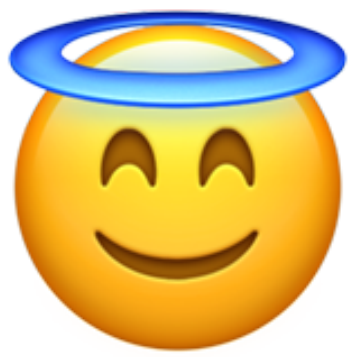
✅ *Shows if the query uses indexes or performs full table scans.*

✅ *Helps avoid unnecessary joins or sorting operations.*

🎯 Final Thoughts

Mastering SQL is a game-changer in landing data science & analytics jobs. 🚀 Practice daily, optimize queries, and explore real-world problems to excel! 🎉

Repost it



Thank you