


MEESHO

Data Analyst Interview

SQL Questions

**Bhuvnesh Kumar** · 2nd
Business Intelligence Analyst@Concentrix | I help you elevate yo...
2d · 🌐




[+ Follow](#) [...](#)

Meesho Data Analyst interview experience (0-3) -

SQL Questions:

1. How would you optimize a slow-running query with multiple joins?
2. What is a recursive CTE, and can you provide an example of when to use it?
3. Explain the difference between clustered and non-clustered indexes and when to use each.
4. Write a query to find the second highest salary in each department.
5. How would you detect and resolve deadlocks in SQL?
6. Explain window functions and provide examples of ROW_NUMBER, RANK, and DENSE_RANK.
7. Describe the ACID properties in database transactions and their significance.
8. Write a query to calculate a running total with partitions based on specific conditions.

[#powerBi](#) [#Sql](#) [#dataanalyst](#)



Dhrubojyoti das and 201 others









5 comments · 9 reposts

Thank you for sharing these questions, Bhuvnesh!

Let's explore the answers




Q1: How would you optimize a slow-running query with multiple joins?

-  Add indexes on **JOIN**, **WHERE**, and **ORDER BY** columns
-  Use **EXPLAIN** to find slow parts of the query
-  Filter rows early using **WHERE** or CTEs
-  Avoid **SELECT *** (only select what you need)
-  Use **INNER JOIN** if you don't need unmatched records
-  Don't wrap join columns in functions like **UPPER()** as it breaks indexing
-  Keep join tables as small as possible
-  Rethink join order - start with smaller or filtered tables

Recursive CTEs



Q2: What is a recursive CTE, and can you provide an example of when to use it?

-  A recursive CTE is a query that calls itself.
- It has 2 parts: a base query (anchor), and a recursive step
- Great for working with hierarchies like:
 - Employee → Manager trees
 - Categories and subcategories
 - Folder structures

Example Use Case:

- ☛ Find all employees under a specific manager, directly or indirectly.

```
WITH RECURSIVE employee_hierarchy AS (
  SELECT employee_id, name, manager_id
  FROM employees
  WHERE manager_id = 101
```

```
UNION ALL
```

```
  SELECT e.employee_id, e.name, e.manager_id
  FROM employees e
  JOIN employee_hierarchy eh ON e.manager_id = eh.employee_id
)
SELECT * FROM employee_hierarchy;
```




Indexes



Q3: Explain the difference between clustered and non-clustered indexes and when to use each.

Indexes make queries faster, like a book index helps you find pages quickly.




Clustered Index

-  Physically sorts the table by one key
-  One per table (because rows are stored in that order)
-  Best for range queries and sorting

Example:

Use on order_date if you frequently query orders by date

Non-Clustered Index

-  Stores data separately with sorted key values and row pointers
-  Multiple per table allowed
-  Best for searching specific values

Example:

Use on customer_id if you often look up customer orders

When to use them?

 Use **clustered indexes** for sorting-heavy queries.

 Use **non-clustered indexes** for quick lookups and filtering.

Salary Query



Q4: Write a query to find the second highest salary in each department.

Approach: Use `DENSE_RANK()` to rank salaries in each department, then filter where the rank is 2.

```
SELECT department_id, employee_name, salary
FROM
(
  SELECT department_id, employee_name, salary,
    DENSE_RANK() OVER (
      PARTITION BY department_id ORDER BY salary DESC
    ) AS rank
  FROM employees
) ranked
WHERE rank = 2;
```

meesho




Deadlocks









Q5: How would you detect and resolve deadlocks in SQL?

Deadlocks happen when two transactions wait on each other. Prevent them with good locking practices and consistent access patterns.

Detect Deadlocks:

-  Enable logging: `log_lock_waits = on, deadlock_timeout = '1s'`
-  Check logs for ERROR: deadlock detected
-  Use views like `pg_locks` and `pg_stat_activity` to inspect blocking queries

Resolve Deadlocks:

-  Keep transactions short and focused
-  Access tables and rows in a consistent order
-  Use `SELECT ... FOR UPDATE` only when needed
-  Avoid waiting for user input inside open transactions
-  Add retry logic in application to handle deadlock errors
-  Monitor with `pg_locks` and logs to detect blocking patterns





Q6: Explain window functions and provide examples of ROW_NUMBER(), RANK(), and DENSE_RANK().

What are window functions?

Window functions let you perform calculations across rows without grouping them.

✨ They're great for:




-  Ranking
- + Running totals
-  Row-by-row comparisons

When to use them?

Use window functions when you need:

-  Ordered, row-level insight
-  Without collapsing the dataset (like GROUP BY would)

Key functions:

- ROW_NUMBER()
 Always gives unique ranks → 1, 2, 3...
- RANK()
 Skips numbers if there are ties → 1, 1, 3...
- DENSE_RANK()
 No gaps in ranks → 1, 1, 2...

Example



Example of Window Functions

Ranking employees by salary in each department

```
SELECT employee_name, department_id, salary,  
       ROW_NUMBER() OVER (  
         PARTITION BY department_id ORDER BY salary DESC) AS row_num,  
       RANK() OVER (  
         PARTITION BY department_id ORDER BY salary DESC) AS rank_num,  
       DENSE_RANK() OVER (  
         PARTITION BY department_id ORDER BY salary DESC) AS dense_rank  
FROM employees;
```

Sample Output

employee_name	department_id	salary	row_num	rank_num	dense_rank
Alice	10	9000	1	1	1
Bob	10	9000	2	1	1
Carol	10	8000	3	3	2
Dan	20	9500	1	1	1
Eva	20	8800	2	2	2

ACID



Q7: Describe the ACID properties in database transactions and their significance.

ACID stands for **A**tomicity, **C**onsistency, **I**solation, **D**urability. These properties ensure reliable and safe transactions in a database.

💡 **Why it is significant:**

ACID protects your data from corruption, conflicts, and crashes, making your database trustworthy.

The 4 Properties:

1. **A**tomicity

☞ All operations succeed or none do. No halfway changes.

2. **C**onsistency

☞ The database moves from one valid state to another. Rules stay intact.

3. **I**solation

☞ Transactions run independently. One doesn't affect another mid-process.

4. **D**urability

☞ Once committed, changes are permanent—even if the system crashes.

Running Total



Q8: Write a query to calculate a running total with partitions based on specific conditions

To calculate running totals within groups, we can use `SUM()` `OVER(PARTITION BY ... ORDER BY ...)`

Running total of sales per customer (only for completed orders)

```
SELECT customer_id, order_id, order_date, amount,  
       SUM(amount) OVER (  
         PARTITION BY customer_id ORDER BY order_date  
       ) AS running_total  
FROM orders  
WHERE status = 'Completed';
```

Sample Output:

customer_id	order_id	order_date	amount	running_total
101	O1	2024-01-01	100	100
101	O2	2024-01-10	150	250
102	O3	2024-01-05	200	200
102	O4	2024-01-15	100	300

Learnings



🚩 My Learnings

DATA ANALYST @ MEESHO

What is this interview focused on?

SQL depth, problem-solving, and clear thinking.

- ✓ Expect real-world data questions
- ✓ Know your window functions, CTEs, and indexing
- ✓ Be ready to write clean, optimized SQL on the spot

💬 The questions are a great mix of technical depth and business context, ideal for anyone who enjoys turning raw data into real impact.

What to expect when interviewing for this role?

- ✓ Break down problems step-by-step, without jumping to code
- ✓ Prepare for follow-up questions on logic and optimization
- ✓ Business scenarios framed like mini case studies

💬 Whether you're applying to Meesho or any data-driven startup: Know the tech, think like a stakeholder, and speak like a problem-solver.