SPECIAL FORCE LAB
DBT Interview Questions

Section 1: Direct Interview Questions

These are direct questions without scenario:

---

1. What is a Database?
2. DBMS vs RDBMS with examples.
3. What are the different types of SQL commands (DDL, DML, etc.)?
4. What is the difference between `DELETE`, `TRUNCATE`, and `DROP`?
5. What are the different data types?
6. Are NULL values in a database the same as that of blank space or zero?
7. What is the default sorting order in SQL if `ORDER BY` is used?
8. What are constraints? Name a few.
9. Explain the difference between `INNER JOIN` and `LEFT JOIN`.
10. What happens if there is no matching record in a `RIGHT JOIN`?
11. Write a query to get employees and their department names from two tables.
12. What's the difference between `WHERE` and `HAVING`?
13. How do you count the number of students in each department?
14. Can we use `GROUP BY` with multiple columns? Give an example.
15. What is a correlated subquery? How is it different from a normal subquery?
16. Write a query to get customers who ordered more than the average order amount.
17. Explain different types of keys in a database.
18. Explain different types of Normalization forms in a DBMS?
19. What is the difference between a `PRIMARY KEY` and a `UNIQUE` key?
20. Explain 1NF, 2NF, and 3NF in your own words.
21. What is a composite key?
22. What is the difference between a stored procedure and a function in MySQL?
23. Can a stored procedure return a value?
24. How do you pass input and output parameters in stored procedures?
25. What is a view? Why do we use it?
26. Can we insert or update data using a view?
27. What are triggers? Name the types.
28. When would you use a `BEFORE INSERT` trigger?
29. What are the ACID properties in databases?
30. What is the purpose of `ROLLBACK` in SQL?
31. What is an index? How does it improve query performance?
32. What are the different types of index?
33. What is cursor?
34. What is meant by an entity-relationship (E-R) model?
35. Explain different types of relationships amongst tables in a DBMS.

SPECIAL FORCE LAB
DBT Interview Questions

Section 2: MySQL Interview Simulation (Including Follow-Up Questions)

INTERVIEWER: Can you explain the different types of SQL commands?
CANDIDATE: [***Expected***:

- DDL: CREATE, ALTER, DROP
- DML: INSERT, UPDATE, DELETE
- DCL: GRANT, REVOKE
- TCL: COMMIT, ROLLBACK, SAVEPOINT]

INTERVIEWER (Follow-up):
What's the difference between `DELETE` and `TRUNCATE`?

- *Checks understanding of **DML vs DDL**, rollback, and logging.*

INTERVIEWER (Follow-up):
Which commands are auto-committed?

- ***Expected**: DDL commands like `CREATE, DROP` are auto-committed.*

INTERVIEWER: What are the ACID properties of a transaction?
CANDIDATE:
[***Expected***:

- A: Atomicity – all or nothing
- C: Consistency – valid state transition
- I: Isolation – concurrent transactions don't interfere
- D: Durability – once committed, remains committed]

INTERVIEWER (Follow-up):
Give a real-world example of using `ROLLBACK`.

- *E.g., online ticket booking or banking.*

SPECIAL FORCE LAB
DBT Interview Questions

INTERVIEWER : Rapid-fire round – tell me the differences:

- `PRIMARY KEY` vs `UNIQUE`
- `UNION` vs `UNION ALL`
- `WHERE` vs `HAVING`
- `INNER JOIN` vs `LEFT JOIN`
- `CHAR` vs `VARCHAR`
- `CANDIDATE KEY` vs `ALTERNATE KEY`
- `SUPER KEY`

- *Tests conceptual clarity.*
-

INTERVIEWER (Follow-up):

What is a `composite key`? Can a table have more than one `PRIMARY KEY`?

- ***Expected**: One composite PK (multiple columns), but only one PRIMARY KEY per table.*

INTERVIEWER:

Create a `students` table with columns: id (auto-increment), name, email (unique), enrollment_date (default today).

- *Tests use of DDL, constraints, default values.*

INTERVIEWER (Follow-up):

How would you modify this table to add a `phone_number` field later?

- ***Expect**:* `ALTER TABLE students ADD phone_number VARCHAR(15);`

INTERVIEWER (Follow-up):

How to delete the column?

***Expect**:* `ALTER TABLE DROP COLUMN phone_number;`

INTERVIEWER:

You have:

- `products(product_id, name, price)`
- `orders(order_id, product_id, quantity)`

Write a query to find the top 3 products with the highest sales (price * quantity).

- *Expecting:* `JOIN, SUM(price*qty), ORDER BY, LIMIT 3`
-

INTERVIEWER (Follow-up):

Can you do the same using a view?

`CREATE VIEW top_selling_products AS ...`

SPECIAL FORCE LAB
DBT Interview Questions

INTERVIEWER:
What's the difference between a procedure and a function?

- *Procedure: can't return values directly, called with CALL.*
  Function: returns value, used inside SQL.

INTERVIEWER (Follow-up):
Write a stored procedure to insert a student and return their ID.

- *Using* `OUT` *parameter and* `LAST_INSERT_ID()`.

INTERVIEWER
What is a trigger?
CANDIDATE: [Expected: Trigger is a block of code that runs automatically before/after insert/update/delete on a table.]

INTERVIEWER (Follow-up):
Can you write a trigger that sets `status = 'active'` whenever a new user is inserted?

- *Using* `BEFORE INSERT` *trigger.*

INTERVIEWER:
Find departments where the number of students is more than the average across all departments.

- *Use subquery in* `HAVING` *clause.*

INTERVIEWER:
What is a view? Why do we use it?
CANDIDATE: [Expected: To simplify queries, hide complexity or restrict access.]

INTERVIEWER (Follow-up):
Can you update a view?

- *Yes, if view is updatable and doesn't involve group by, joins, or aggregate.*

INTERVIEWER (Follow-up):
How would you give a user read-only access to a view?

- *Use:* `GRANT SELECT ON view_name TO 'user';`

INTERVIEWER:
How would you optimize a slow query?
CANDIDATE:
[***Expected***: Use EXPLAIN, add indexes, avoid SELECT *, optimize joins.]

INTERVIEWER (Follow-up):
If we have `WHERE email = 'abc@x.com',` which index helps?

- *Single index on* `email`

SPECIAL FORCE LAB
DBT Interview Questions

INTERVIEWER (Follow-up):
What is the difference between clustered and non-clustered index?

INTERVIEWER: Can you explain what normalization is and why it's important?
CANDIDATE : [***Expected***: Reducing data redundancy, ensuring integrity, etc.]
INTERVIEWER (Follow-up):
Can you walk me through how you'd normalize a `Student` table that contains `student_name, course_name, instructor_name,` and `department`?
CANDIDATE: [***Expected***: Split into Student, Course, Instructor, Department; use foreign keys.]

INTERVIEWER:
Let's say one course is taught by two instructors — how would you model that?

- *Checks understanding of **many-to-many relationships** using junction tables.*

INTERVIEWER: Here's a table:
`orders(order_id, customer_id, product_id, quantity, order_date)`
Write a query to find the total quantity ordered by each customer.
CANDIDATE: [***Expected***: `GROUP BY customer_id` with `SUM(quantity)`]

INTERVIEWER (Follow-up):
Now, show only customers who have ordered **more than 100 items in total**.

- *Tests `HAVING` clause usage.*

INTERVIEWER: Let's say we have:

- `students(student_id, name)`
- `enrollments(student_id, course_id)`
- `courses(course_id, course_name)`

Write a query to get names of students along with the courses they're enrolled in.
CANDIDATE: [***Expected***: JOIN across 3 tables.]

INTERVIEWER (Follow-up):
Now, return all students, even if they are not enrolled in any course.

- *Expecting **LEFT JOIN** usage.*

INTERVIEWER:
Write a query to get the course(s) with the **highest number of enrollments**.
CANDIDATE: [Subquery or `ORDER BY COUNT()` with `LIMIT 1`]

SPECIAL FORCE LAB
DBT Interview Questions

INTERVIEWER (Follow-up):
Can you do it without using `LIMIT`?

- *Expecting use of a subquery in* `WHERE` *clause with* `MAX`*.*

INTERVIEWER:
Have you written a stored procedure before? What are some use cases?
CANDIDATE: [***Expected***: Encapsulating logic like auto-grading, bulk insert, etc.]

INTERVIEWER (Follow-up):
Write a stored procedure to insert a new student record and return their new ID.

- *Should use* `OUT` *parameter and* `LAST_INSERT_ID()`*.*

INTERVIEWER:
If a new student is inserted, we want to automatically insert a default course enrollment. How would you do this?

- *Expecting a* `BEFORE INSERT` or `AFTER INSERT` *trigger.*

INTERVIEWER:
In a banking system, how do you ensure money is debited from one account and credited to another without inconsistency?
CANDIDATE: [***Expected***: Use of `START TRANSACTION, COMMIT, ROLLBACK`]

INTERVIEWER:
What's the difference between a **clustered** and a **non-clustered index**? Does MySQL support both?

- *Advanced topic: MySQL's InnoDB uses clustered index by default.*

INTERVIEWER:
You're building a dashboard that shows:

- Total number of users
- Number of active users
- Average number of orders per user

Can you write one SQL query or a view that returns all three?

- *Tests* ***aggregate usage****, possibly subqueries or* ***CTEs****.*

SPECIAL FORCE LAB
DBT Interview Questions

Section 3: A few important Real-Time Examples: (Extremely important to mention real-time example in Interviews, Consider this as a cheat-sheet)

1. ACID Properties

Property	Real-Time Example
Atomicity	Online payment: You pay for an order, and the amount is deducted. But if the order fails, the amount is refunded. The transaction either **completes fully or rolls back**.
Consistency	Bank transfer: The total balance across two accounts before and after a transfer remains consistent. If ₹1,000 is transferred from A to B, A's balance drops and B's increases—**data integrity is maintained**.
Isolation	Two users booking the last movie ticket: Even if they click "Book" at the same time, only one booking is processed because transactions are **isolated**.
Durability	After you place an Amazon order, even if the system crashes, your order confirmation remains—because the **data is committed and saved permanently**.

---

2. DDL vs DML

Concept	Real-Time Example
DDL (Data Definition Language)	Like setting up a shop layout — building tables, shelves (tables/columns). Ex: `CREATE TABLE, ALTER TABLE.`
DML (Data Manipulation Language)	Adding or changing products in the shop. Ex: `INSERT INTO, UPDATE, DELETE.`
DCL (Control access)	Giving your manager read-only or full access to your inventory system. Ex: `GRANT, REVOKE.`
TCL (Transaction Control)	Think of "Save" and "Undo" in MS Word. `COMMIT, ROLLBACK, SAVEPOINT` control whether changes are permanent or not.

SPECIAL FORCE LAB
DBT Interview Questions

3. Primary Key vs Unique Key

- **Primary Key:** Like a student's **Roll Number** — always unique and required.
- **Unique Key:** Like a student's **email** — must be unique, but can be left blank in some cases.

---

4. Indexing

- **Real-Life:** An index in a book helps you quickly jump to the topic page instead of reading everything. Similarly, a **database index** speeds up search by pointing directly to the row.

---

5. Stored Procedure

- Think of it like a **pre-recorded voice message** you use in a call center. It runs the same process every time it's triggered. Reusable and consistent.

---

6. Trigger

- Like an **auto-reply email**. When you receive an email (event), your system responds automatically. A database trigger reacts automatically when a specific event (like insert or delete) happens.

---

7. Transactions

- **Real-Time:** ATM transaction – You withdraw money → Account is debited → ATM dispenses cash. If anything fails in between, the whole transaction is rolled back.

SPECIAL FORCE LAB
DBT Interview Questions

8. Subqueries

- Like asking your friend: “Who scored more than the class average?”
  You need to **first calculate the average**, then compare — that’s a **subquery** inside a query.

---

9. JOINs

JOIN Type	Real-Life Analogy
INNER JOIN	Two circles overlapping: Students who are in both “Coding Club” and “Robotics Club”.
LEFT JOIN	All from “Coding Club”, and only those from “Robotics Club” who match.
RIGHT JOIN	All from “Robotics Club”, and only those from “Coding Club” who match.
FULL OUTER JOIN	Everyone from both clubs, matched or not.

---

10. View

- Like creating a **read-only report** in Excel. You don’t need to see all 10 database tables—just the important summary in one sheet.

11. Normalization

Real-Time Example:
Imagine a school where the student’s name, course, and teacher are repeated in every row. It’s like writing the teacher’s name on every student’s notebook.
Normalization organizes this data into multiple related tables to **remove redundancy** and **ensure consistency**.

SPECIAL FORCE LAB
DBT Interview Questions

12. Foreign Key

Real-Life:
A foreign key is like a **passport number linked in both airport and immigration databases**. The immigration system references the passport details but doesn't own them—it just ensures **referential integrity**.

---

13. Auto Increment

Analogy:
Imagine giving **queue tokens** in a bank. Every new customer automatically gets the next number. No one manually assigns it—just like MySQL auto-generates IDs using `AUTO_INCREMENT`.

---

14. Data Types

Real-Life:
When filling out a form:

- **Name field:** Accepts only text (VARCHAR).
- **Age field:** Accepts only numbers (INT).
- **DOB field:** Accepts only dates (DATE).

This is like enforcing **data types** in columns.

---

15. Constraints (NOT NULL, CHECK, DEFAULT, etc.)

Real-Life:

- **NOT NULL** – Like mandatory fields in an online form.
- **CHECK(age >= 18)** – Like allowing only adults to register.
- **DEFAULT 'India'** – If country isn't selected, default to India.

16. Group By & Aggregates (SUM, AVG, COUNT, etc.)

Real-Life:
Generating **monthly sales reports** where you want to:

- **SUM** total sales per branch
- **COUNT** number of customers per city
- **AVG** salary per department

`GROUP BY` helps with such summarizations.

---

17. EXISTS vs IN

Real-Life:

- `IN:` Like checking if a student's name is on a printed list.
- `EXISTS:` Like confirming if the list itself **has at least one matching entry** before proceeding.

---

18. Pagination (LIMIT, OFFSET)

Analogy:
Scrolling through Instagram or Amazon search results—**only 10 or 20 results per page**.
MySQL uses `LIMIT` and `OFFSET` to paginate results, just like those platforms.

SPECIAL FORCE LAB
DBT Interview Questions

19. UNION vs UNION ALL

Analogy:
You invite guests from two parties:

- **UNION** → Add everyone, but **remove duplicates**.
- **UNION ALL** → Add everyone, **including duplicates** (maybe they were invited twice!).