**JADE Global Snowflake Interview Questions:**

**Q:** Okay, so you said you work with Snowflake, right? Let's say there is one scenario. The scenario is while working with a production support issue, we simulate the issue in the backup of a table. But somehow, we need to drop those backup tables. The requirement is we have to identify the backup table which is created prior to 90 days. Once we identify that, we have to create one log table, log those entries, whatever the backup table we found, and once we log it, we have to drop it. So could you please tell me your approach? **A:** I can't hear the question clearly.

**Q:** What I'm saying is, we have a backup of a table. In a single database, there are multiple schemas, like x, y, z. Our requirement is to identify the backup tables from all these schemas which were created 90 days prior. **A:** The backup table name ends with BKB. For this, we have information\_schema.tables where the table\_name is LIKE '...%BKB' and the creation date is greater than 90 days.

**A Production Support Scenario in Snowflake**

While working on a production support issue, we need to create a backup of a table to simulate and troubleshoot a problem. Once the issue is resolved, it's essential to clean up these backup tables to maintain a tidy and efficient environment.

The specific requirements for this cleanup process are as follows:

1. **Identify:** We need to find all backup tables that were created more than 90 days ago.
2. **Log:** Before dropping any tables, we must create a log table and record all the identified backup tables in it for auditing purposes.
3. **Drop:** After logging the entries, we need to drop the identified tables from the database.

**Question:** What is the best approach to automate this process in Snowflake? Please outline the steps required to identify, log, and drop these old backup tables efficiently and safely

**Question 2:**

**Q:** Do you know the list and dictionary from the Python? What about dictionary?

Question 3:

Write a SQL query to:

1. Calculate the **average salary per department**.
2. **Filter** only those departments where the average salary is **greater than 50,000**.
3. From those departments, **identify employees whose salary is greater than their department's average salary**.

CREATE OR REPLACE TABLE emp\_with (

emp\_id INT,

emp\_name STRING,

dept\_no INT,

salary NUMBER(10,2)

);

INSERT INTO emp\_with (emp\_id, emp\_name, dept\_no, salary) VALUES

(101, 'Alice', 10, 60000),

(102, 'Bob', 10, 52000),

(103, 'Charlie', 10, 48000),

(104, 'David', 20, 70000),

(105, 'Eve', 20, 72000),

(106, 'Frank', 20, 68000),

(107, 'Grace', 30, 45000),

(108, 'Heidi', 30, 47000),

(109, 'Ivan', 30, 49000);

-- Step 1: CTE to calculate department-wise average salary

WITH dept\_avg AS (

SELECT

dept\_no,

AVG(salary) AS avg\_salary

FROM emp\_with

GROUP BY dept\_no

),

-- Step 2: Filter departments with avg\_salary > 50000

high\_avg\_dept AS (

SELECT \*

FROM dept\_avg

WHERE avg\_salary > 50000

),

-- Step 3: Use analytic function to compare employee salary with department average

emp\_with\_avg AS (

SELECT

emp\_id,

emp\_name,

dept\_no,

salary,

AVG(salary) OVER (PARTITION BY dept\_no) AS dept\_avg\_salary

FROM emp\_with

)

-- Final output: Employees with salary > department average and department avg > 50000

SELECT

e.emp\_id,

e.emp\_name,

e.dept\_no,

e.salary,

e.dept\_avg\_salary

FROM emp\_with\_avg e

JOIN high\_avg\_dept d ON e.dept\_no = d.dept\_no

WHERE e.salary > e.dept\_avg\_salary;

EMP\_ID EMP\_NAME DEPT\_NO SALARY DEPT\_AVG\_SALARY

101 Alice 10 60000.00 53333.33333

105 Eve 20 72000.00 70000.00000