**Frieght Tiger Data Engineer Interview Guide – Experienced 3+**

**Round 1: Technical Interview – SQL Focused**

The SQL round evaluates a candidate's ability to handle real-world database queries, perform data analysis, and write efficient SQL code. Key areas tested include:

 Query structuring

 Aggregations

 Joins

 Subqueries

 Window functions

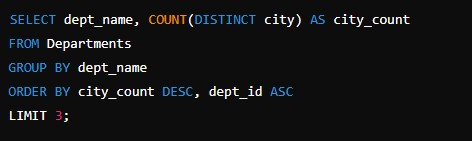
Candidates are expected to demonstrate proficiency in interpreting requirements, querying data, and handling complex logic using SQL.

**Questions and Solutions**

**1. Number of Cities Per Department**

**Question**: How many cities does each department operate in? List the top 3 departments in terms of the most number of cities. In case of a tie, order by dept\_id.

**Query**:



**Explanation**:

 COUNT(DISTINCT city) counts the unique cities each department operates in.

 GROUP BY dept\_name groups the results by department.

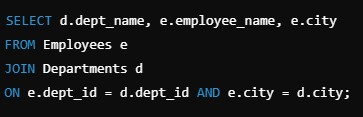
 ORDER BY city\_count DESC, dept\_id ASC ensures the results are sorted by the number of cities in descending order and by department ID in ascending order for ties.

 LIMIT 3 restricts the output to the top 3 departments.

**2. Employee-Department-City Combinations**

**Question**: List every combination of dept\_name, employee\_name, and city such that the employee belongs to the department and the same city in which the department is located.

**Query**:



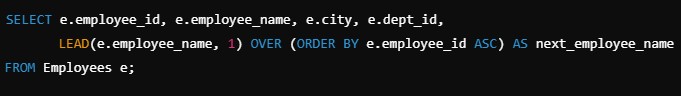
JOIN ensures that employees are matched with departments based on dept\_id and located in the same city.

 The SELECT clause retrieves the department name, employee name, and city.

**3. Next Employee by ID**

**Question**: Add a column to the Employees table that shows the name of the employee with the next higher employee\_id.

**Query**:



**Explanation**:

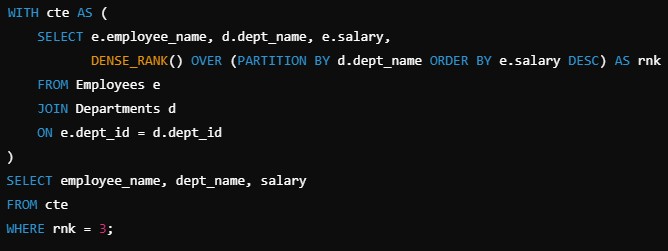
The LEAD() function retrieves the next employee’s name in ascending order of employee\_id.

OVER (ORDER BY e.employee\_id ASC) defines the ordering logic for the window function.

**4. Third-Highest Salary Per Department**

**Question**: Find the third-highest salary for each department.

**Query**:



**Explanation**:

 The WITH clause creates a Common Table Expression (CTE) to rank employees' salaries within each department using DENSE\_RANK().

 PARTITION BY d.dept\_name groups the ranking within each department.

 ORDER BY e.salary DESC ensures the ranking is in descending order of salary.

 WHERE rnk = 3 filters for the third-highest salary.

**Round 2: Technical Interview – Pyspark Focused**

**Question 1: PySpark Script for Filtering and Writing a CSV File**

**Task:**

Write a PySpark script to read a CSV file, filter rows where the age column is less than 18, and write the result to a new CSV file.

**Solution:**



**Explanation:**

 option("header", "True") ensures that the CSV header is read.

 filter(col("age") > 18) applies a filter condition on the age column.

 write.csv("output\_path", header=True) saves the filtered DataFrame as a new CSV file.

**Question 2: Identifying Top 3 Employees per Department**

**Task:**

Write a PySpark job to find the top 3 employees of each department, where:

 Age < 30

 Salary > department average salary

**Solution:**



**Explanation:**

 groupBy("dept\_id").agg(mean("sal").alias("avg\_salary")): Calculates department-wise average salary.

 Filters ensure that only employees under 30 and earning above the department average are considered.

 Window.partitionBy("dept\_id").orderBy(col("sal").desc()): Ranks employees based on salary within each department.

 Top 3 employees are selected using filter(col("rank") <= 3).

**Question 3: Spark Job Execution**

**Task:**

How many jobs, stages, and tasks will be created for the given code?

**Code:**



**Execution Analysis:**

1. Jobs:

 1st Job: Reading the data (spark.read()).

 2nd Job: Repartitioning the data to 2 partitions.

 3rd Job: Applying filter, projection, and aggregation. Total Jobs: 3

2. Stages:

 Stage 1: Filter operation.

 Stage 2: GroupBy aggregation. Total Stages: 2 (for the 3rd job).

3. Tasks:

Tasks are based on the number of partitions. After repartitioning to 2 partitions, tasks in each stage = 2.

Total Tasks: Depends on data size but will scale with 2 partitions.

**Explanation:**

 Repartitioning changes the data layout and impacts task parallelism.

 Aggregations like groupBy create shuffle stages.