

## Incedo Azure Data Engineer Interview Guide – Experienced 3+

### Technical Round 1

#### 1. Create Two Columns for Sum of Positive and Negative Numbers

Input:

Input:

X
1
2
-1
0
-2
1
5
-4

Output:

Positive_Sum	Negative_Sum
9	-7

Solution:

```
sql
SELECT
  SUM(CASE WHEN num > 0 THEN num ELSE 0 END) AS Positive_Sum,
  SUM(CASE WHEN num < 0 THEN num ELSE 0 END) AS Negative_Sum
FROM X;
```

Explanation:

- CASE is used to conditionally check each number.
- **Positive numbers** are summed when `num > 0`.
- **Negative numbers** are summed when `num < 0`.
- ELSE 0 ensures non-matching conditions do not affect the result.

## 2. Identify Consecutive Numbers in a Column

Input:

Input:	
ID	Num
1	1
2	1
3	1
4	2
Expected Output:	
Num	
1	

**Solution:**

```
SELECT num
FROM (
    SELECT num,
           ROW_NUMBER() OVER (PARTITION BY num ORDER BY id) AS rn,
           ROW_NUMBER() OVER (ORDER BY id) - ROW_NUMBER() OVER (PARTITION BY
num ORDER BY id) AS grp
    FROM table
) sub
GROUP BY num, grp
HAVING COUNT(*) >= 3;
```

**Explanation:**

- ROW\_NUMBER() is used to assign row numbers to each entry.
- A calculated grp column groups consecutive numbers.
- HAVING COUNT(\*) >= 3 filters sequences of at least 3.

### 3. Replace Words and Perform String Operations in Python

```
input_str = "Spark is fast and Spark is scalable"

# Replace "Spark" with "Snowflake"
replaced = input_str.replace("Spark", "Snowflake")

# Replace vowels with white space
vowels = "aeiouAEIOU"
no_vowels = ''.join([' ' if char in vowels else char for char in input_str])

# Count occurrences of each word
from collections import Counter
word_count = Counter(input_str.split())

# Check for pattern "sas"
import re
pattern_exists = bool(re.search(r'sas', input_str))

# Results
print(replaced) # Snowflake is fast and Snowflake is scalable
print(no_vowels) # Sp rk s f st nd Sp rk s sc l bl
print(word_count) # {'Spark': 2, 'is': 2, 'fast': 1, 'and': 1, 'scalable': 1}
print(pattern_exists) # False
```

#### Explanation:

- **String Replacement:** `.replace()` is used for direct word substitution.
- **Vowel Removal:** A list comprehension with `join()` creates a modified string.
- **Word Count:** `Counter` splits the string and counts word occurrences.
- **Pattern Check:** `re.search()` finds the pattern 'sas'.

#### 4. SQL Where Employee Earns More Than Manager

**Problem:** Identify employees whose salary is higher than their manager's.

**Table Structure:**

employee_id	name	manager_id	salary
1	Alice	NULL	1500
2	Bob	1	2000
3	Charlie	1	1800

**Query:**

sql

Copy code

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

**Explanation:**

- **Self Join:** The table employees is joined to itself (e and m) to compare employee-manager relationships.
- **Condition:** WHERE e.salary > m.salary filters for employees earning more than their manager.
- **Example Output:**

employee\_id name

2 Bob

## Technical Round 2

### 1. Find the Second-Highest Salary in a Table

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

### 2. Count the Number of Nulls in Each Column of a Table

```
SELECT
    SUM(CASE WHEN column1 IS NULL THEN 1 ELSE 0 END) AS column1_nulls,
    SUM(CASE WHEN column2 IS NULL THEN 1 ELSE 0 END) AS column2_nulls
FROM table_name;
```

### 3. SQL Query to Remove Duplicates from a Table

```
DELETE FROM table_name
WHERE id NOT IN (
    SELECT MIN(id)
    FROM table_name
    GROUP BY column1, column2, column3
);
```

### 4. Write a Query to Find Employees in the Same Department as 'John'

```
SELECT e.name
FROM employees e
JOIN employees john ON e.department_id = john.department_id
WHERE john.name = 'John' AND e.name != 'John';
```

### 5. Python Program to Reverse Words in a String

```
input_str = "Hello World from Python"
reversed_words = ' '.join(input_str.split()[::-1])
print(reversed_words) # Output: "Python from World Hello"
```

## 6. How to Optimize a Spark Job

- Use proper partitioning and bucketing.
- Avoid shuffles by optimizing join keys.
- Use broadcast joins for small datasets.
- Cache intermediate results when reused multiple times.
- Enable Dynamic Resource Allocation in Databricks.

## 7. Scenario-Based Question: Query Optimization for a Large Dataset

- Check for proper indexing on queried columns.
- Use EXPLAIN PLAN to identify bottlenecks.
- Partition the dataset by frequently queried columns.
- Use parallel processing for large transformations.

## 8. Find Employees Who Earn the Third-Highest Salary

```
SELECT employee_id, salary
FROM (
    SELECT employee_id, salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS
rank
    FROM employees
) WHERE rank = 3;
```

## 9. Difference Between Lazy Evaluation and Eager Execution in PySpark

- Lazy Evaluation: Transformations are not executed immediately; they are recorded in a DAG.
- Eager Execution: Actions like collect() or show() trigger execution.

## 10. Python Program to Check if a String is a Palindrome

```
def is_palindrome(s):
    s = s.lower().replace(" ", "")
    return s == s[::-1]

print(is_palindrome("A man a plan a canal Panama")) # Output: True
```

### 11. SQL Query to Find Departments with More Than 10 Employees

```
SELECT department_id, COUNT(*) AS employee_count  
FROM employees  
GROUP BY department_id  
HAVING COUNT(*) > 10;
```

### 12. Explain PySpark's Catalyst Optimizer

- Catalyst is PySpark's query optimization engine.
- Performs logical optimization (predicate pushdown, projection pruning).
- Converts logical plans to physical plans for efficient execution.

© Shubham Wadekar

## Technical Round 3

### 13. Python Program to Find Consecutive Numbers in a List

```
nums = [1, 1, 1, 2, 1, 2, 2, 3, 3, 3]
from collections import Counter

def find_consecutive(nums):
    result = []
    count = Counter(nums)
    for num, freq in count.items():
        if freq >= 3:
            result.append(num)
    return result

print(find_consecutive(nums)) # Output: [1, 3]
```

### 14. SQL Query to Replace Specific Patterns in a String Column

Replace "Spark" with "Snowflake":

```
SELECT REPLACE(column_name, 'Spark', 'Snowflake') AS updated_column
FROM table_name;
```

Replace All Vowels with White Spaces:

```
SELECT REGEXP_REPLACE(column_name, '[AEIOUaeiou]', ' ') AS no_vowels
FROM table_name;
```

Count Occurrences of Each Word:

```
SELECT word, COUNT(*) AS occurrences
FROM (SELECT EXplode(SPLIT(column_name, ' ')) AS word FROM table_name) AS words
GROUP BY word;
```

### 15. Difference Between Partition Count and Query Performance in Spark

- Partition count affects parallelism.
- Too few partitions lead to underutilization.
- Too many partitions create task scheduling overhead.
- Use repartition() to increase partitions and coalesce() to reduce partitions.



## 16. Incremental Load in ADF and Databricks

- **In ADF:** Use copy activity with a filter on the last\_modified\_date column.
- **In Databricks:**
  - incremental\_data = full\_data.filter(full\_data.updated\_at > last\_load\_timestamp)

```
incremental_data.write.format("delta").mode("append").save(target_path)
```

## 17. SQL Query to Find Top 3 Earners in Each Department

```
SELECT department_id, employee_id, salary
```

```
FROM (
```

```
    SELECT department_id, employee_id, salary, RANK() OVER (PARTITION BY  
    department_id ORDER BY salary DESC) AS rank
```

```
    FROM employees
```

```
) WHERE rank <= 3;
```

## 18. Explain the Purpose of SparkSession vs SparkContext

- SparkSession: Unified entry point for DataFrame and SQL APIs.
- SparkContext: Low-level entry point to interact with Spark's core APIs (RDDs).

## 19. How to Remove Duplicate Rows in PySpark

```
df = df.dropDuplicates(["column1", "column2"])
```

## 20. Caching Techniques in Databricks

```
df.cache()
```

```
df.show() # This triggers caching
```

## 21. Python Code to Generate Prime Numbers

```
def is_prime(num):  
    if num < 2:  
        return False  
    for i in range(2, int(num**0.5) + 1):  
        if num % i == 0:  
            return False  
    return True  
  
primes = [x for x in range(1, 101) if is_prime(x)]  
print(primes)
```

## 22. Broadcast Join in PySpark

```
from pyspark.sql.functions import broadcast  
result_df = large_df.join(broadcast(small_df), "common_key")
```

## 23. Explain Fact and Dimension Tables

- Fact Table: Contains quantitative data (e.g., sales, revenue).
- Dimension Table: Contains descriptive attributes (e.g., product, customer).

## 24. Difference Between Managed and External Tables in Databricks

- Managed Table: Databricks manages the data storage location.
- External Table: Data resides outside Databricks but is referenced in the metastore.

## 25. SQL Query to Fetch Employees Earning More Than Their Manager

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

## 26. Explain Z-Ordering in Databricks

- Z-ordering optimizes file layout by co-locating similar data.
- Improves query performance for frequently filtered columns.