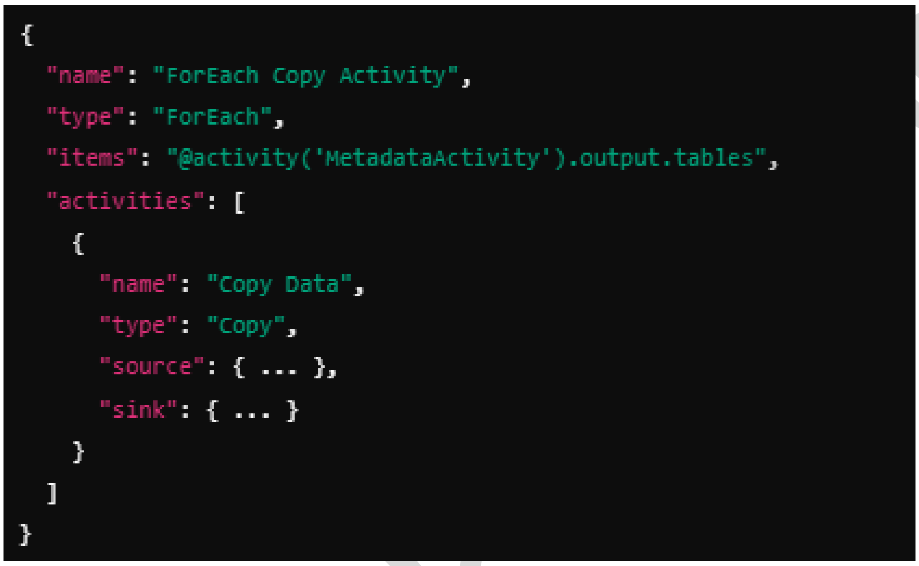
FedEx Dataworks Data Engineer Interview Guide – Experienced 3+

Technical round 1 and 2 combined

1. How to Copy All 1000 Tables from Source to Target in ADF

- Use Metadata Activity: Use the Metadata activity to fetch the table details (table names and columns) from the source.
- Use ForEach Activity: After gathering the list of tables, use the ForEach activity to iterate through each table name.
- Use Copy Activity: Within the ForEach activity, use the Copy Activity to copy the data from the source to the target.

Example:



```
{
  "name": "ForEach Copy Activity",
  "type": "ForEach",
  "items": "@activity('MetadataActivity').output.tables",
  "activities": [
    {
      "name": "Copy Data",
      "type": "Copy",
      "source": { ... },
      "sink": { ... }
    }
  ]
}
```

2. Trigger Types in ADF – Discuss Scheduled, Tumbling Window, and Event-Based Triggers

- **Scheduled Trigger**:

  Definition: Triggers the pipeline on a set schedule (e.g., daily, hourly).

  Use Case: Use when you want a pipeline to run at regular intervals.

- **Tumbling Window Trigger**:

  Definition: Creates a window of fixed duration (e.g., every hour, day). Each trigger is independent.

  Use Case: Use when you need to perform time-based operations on a continuous schedule.

- **Event-Based Trigger**:

  Definition: Triggers a pipeline based on an event (e.g., file upload in Blob Storage).

Use Case: Use when you want to trigger a pipeline after a specific event occurs, like a new file landing in a storage location.

3. Linked Service and Dataset – Explain Their Roles in ADF

- **Linked Service**:

  Definition: A Linked Service defines the connection information for a specific data store (e.g., Azure SQL Database, Blob Storage). It allows ADF to connect to source or destination.

- **Dataset**:

  Definition: A Dataset represents the data structure (e.g., a table, file, or folder) that is being used in the pipeline. It is often associated with a linked service to point to the data source.

4. Data Warehouse vs Data Lake vs Data Lakehouse – Compare Features and Use Cases

- **Data Warehouse**:

  Definition: A structured repository designed to support reporting and analytics. It typically uses schemas like star or snowflake.

  Use Case: Best for structured data with historical reporting needs.

- **Data Lake**:

  Definition: A large, unstructured data repository that can store raw data in any format (e.g., text, images, logs).

  Use Case: Ideal for big data, machine learning, and raw, unstructured data storage.

- **Data Lakehouse**:

  Definition: Combines the features of both data lakes and data warehouses. It supports structured, semi-structured, and unstructured data, with features like ACID transactions.

  Use Case: Ideal for organizations that need both analytical queries and unstructured data storage.

5. How to Run One Notebook in Another Notebook – Example in Databricks

In Databricks, you can run one notebook from another using the dbutils.notebook.run() method:

```
result = dbutils.notebook.run("/path/to/other_notebook", 60)
```

- 60 is the timeout in seconds.
- This will execute the notebook and return the result.

6. Workflow in Databricks – Steps to Create and Schedule a Job

- **Steps to Create a Job**:
    1. Navigate to the Jobs section in Databricks.
    2. Click on Create Job and provide a name for the job.
    3. Add a task that points to the notebook or JAR you want to execute.
    4. Set the cluster configuration (e.g., job cluster or shared cluster).
- **Steps to Schedule**:
    1. Under the job settings, configure the Schedule.
    2. Set the frequency (e.g., daily, weekly).
    3. Specify time and recurrence.

7. How Many Jobs/Stages/Tasks Are Created – Explain Spark's Execution Plan

- **Jobs**: Each action (like collect(), save(), count()) creates a job.
- **Stages**: Spark divides jobs into stages based on transformations that require shuffling.
- **Tasks**: A stage is divided into tasks, each corresponding to a partition of the data.

To see the execution plan in Databricks, you can use:

```
df.explain(True)
```

8. Repartition vs Coalesce – Differences and Scenarios for Use

- **Repartition**:

    Definition: Increases or decreases the number of partitions by reshuffling the data.

    Use Case: Use when increasing partitions to optimize parallel processing (e.g., repartition(10)).
- **Coalesce**:

    Definition: Reduces the number of partitions without full shuffle.

    Use Case: Use when reducing partitions (e.g., before saving output) to avoid small file problems (e.g., coalesce(1)).

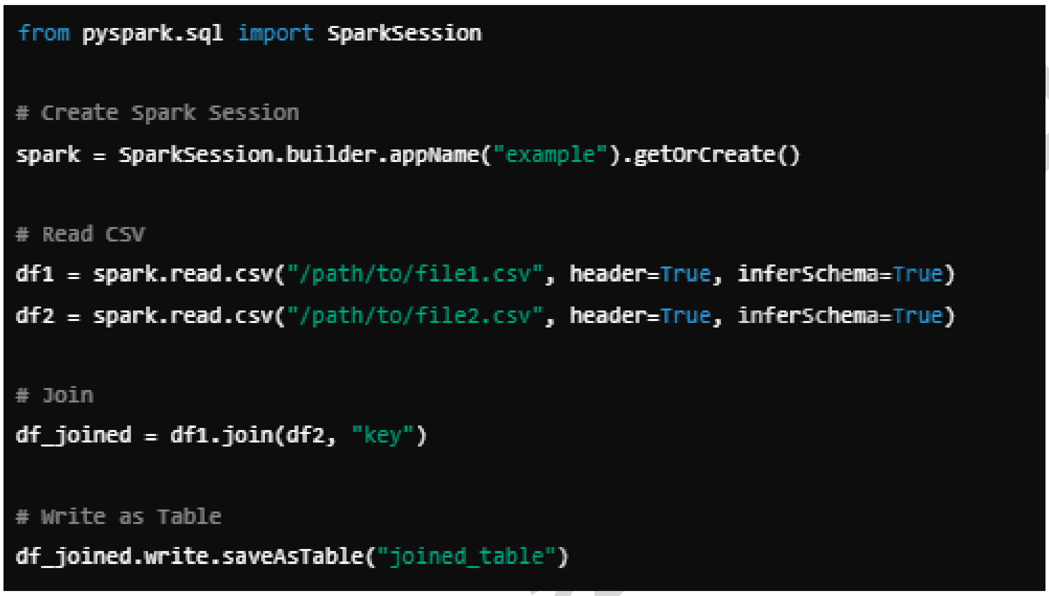
9. Broadcast Join – When and How to Use It

- **When to Use**: Broadcast joins are used when one dataset is significantly smaller than the other. Broadcasting the smaller dataset avoids shuffling the larger dataset.
- **How to Use**:

from pyspark.sql.functions import broadcast

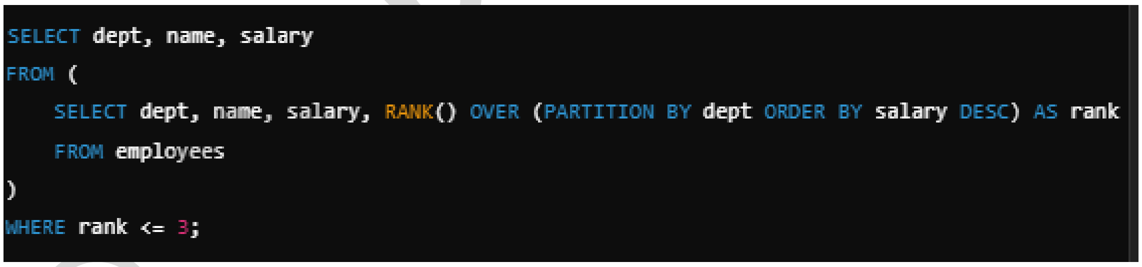
df_large.join(broadcast(df_small), "key")

10. Create Spark Session, Read CSV, Join, and Write as Table – Provide Example Code



```
from pyspark.sql import SparkSession

# Create Spark Session
spark = SparkSession.builder.appName("example").getOrCreate()

# Read CSV
df1 = spark.read.csv("/path/to/file1.csv", header=True, inferSchema=True)
df2 = spark.read.csv("/path/to/file2.csv", header=True, inferSchema=True)

# Join
df_joined = df1.join(df2, "key")

# Write as Table
df_joined.write.saveAsTable("joined_table")
```

11. SQL Query to Find Top 3 Earners in Each Department



```
SELECT dept, name, salary
FROM (
    SELECT dept, name, salary, RANK() OVER (PARTITION BY dept ORDER BY salary DESC) AS rank
    FROM employees
)
WHERE rank <= 3;
```

12. Rank vs Dense Rank – Differences and Use Cases

- **Rank**:

  Definition: RANK() assigns a rank to each row, skipping ranks in case of ties.

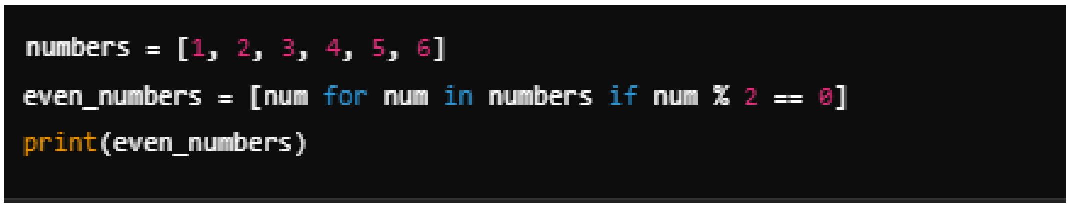
  Use Case: Use when you want gaps in ranking for tied rows.

- **Dense Rank**:

  Definition: DENSE_RANK() assigns a rank without skipping any values in case of ties.

  Use Case: Use when you want continuous ranking, even for ties.

13. Print Even Numbers from a List – Python Example



```python
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = [num for num in numbers if num % 2 == 0]
print(even_numbers)
```

14. Wide vs Narrow Transformations – Explain with Examples

- **Wide Transformations**:

  Definition: These transformations involve shuffling data (e.g., groupByKey(), reduceByKey()).

  Example: groupByKey() performs a wide transformation as it requires data shuffling.

- **Narrow Transformations**:

  Definition: These transformations don't involve shuffling (e.g., map(), filter()).

  Example: map() applies a function to each element in the dataset.

15. Adaptive Query Execution (AQE) in Databricks – Explain its Benefits

- **Definition**: AQE in Databricks optimizes query execution plans dynamically at runtime based on the data distribution and runtime statistics.
- **Benefits**:

  Dynamic Partition Pruning: Avoid scanning unnecessary partitions.

  Join Optimization: Dynamically adjust join strategies based on runtime data.

  Cost-based Optimizations: AQE adapts the plan to minimize the cost of query execution.

16. Interactive Cluster vs Job Cluster – Differences and Use Cases

- **Interactive Cluster**:

  Definition: A cluster that is used for interactive analysis and notebook execution.

  Use Case: Ideal for development, testing, and debugging.

- **Job Cluster**:

  Definition: A cluster created specifically for running jobs and terminated once the job is finished.

  Use Case: Use when running scheduled jobs and batch processes.

17. Autoscaling in Databricks – How it Works and Its Benefits

- **How it Works**: Databricks automatically scales the cluster based on the workload. If there are more tasks to process, it will add more nodes; if the tasks reduce, it will scale down.
- **Benefits**:

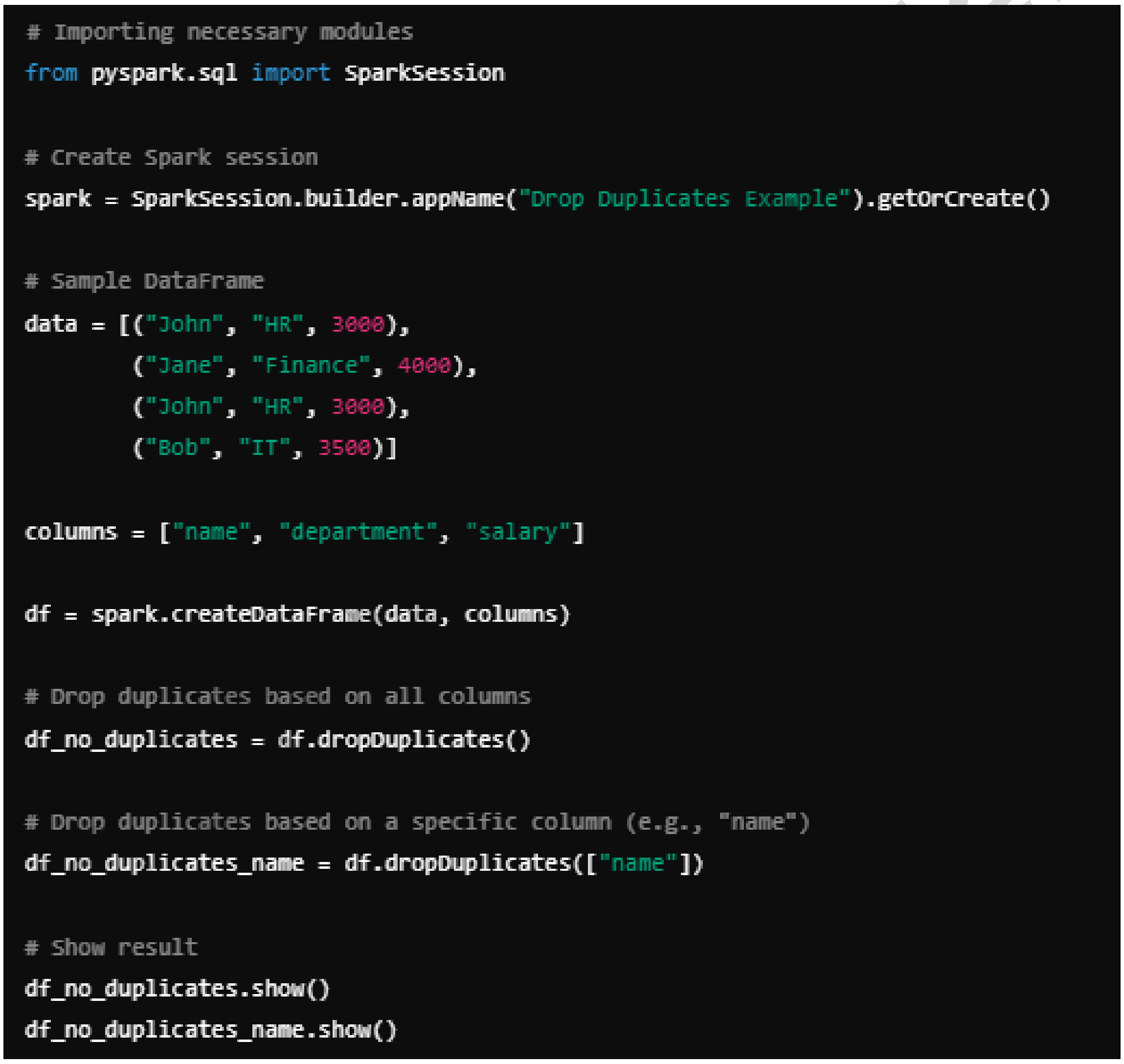
  Cost-efficient: Reduces costs by scaling up or down based on workload.

  Performance: Ensures the cluster is always appropriately sized for the job.

18. Drop Duplicates in PySpark – Example Code

In PySpark, you can remove duplicates from a DataFrame using the dropDuplicates() method. You can specify columns to remove duplicates based on specific columns or remove all duplicate rows from the DataFrame.

Example Code:



```
# Importing necessary modules
from pyspark.sql import SparkSession

# Create Spark session
spark = SparkSession.builder.appName("Drop Duplicates Example").getOrCreate()

# Sample DataFrame
data = [("John", "HR", 3000),
        ("Jane", "Finance", 4000),
        ("John", "HR", 3000),
        ("Bob", "IT", 3500)]

columns = ["name", "department", "salary"]

df = spark.createDataFrame(data, columns)

# Drop duplicates based on all columns
df_no_duplicates = df.dropDuplicates()

# Drop duplicates based on a specific column (e.g., "name")
df_no_duplicates_name = df.dropDuplicates(["name"])

# Show result
df_no_duplicates.show()
df_no_duplicates_name.show()
```

19. How to Give Permission to a Notebook to Other Users – Steps in Databricks

To give permission to a notebook to other users in Databricks, you need to set the appropriate permissions for the notebook. Here's how you can do it:

Steps:

1. **Navigate to the Notebook**:

   Go to the Workspace section in Databricks.

   Locate the notebook for which you want to grant access.

2. **Open Permissions Settings**:

   Right-click on the notebook and select Permissions from the context menu. Alternatively, you can open the notebook, and then click on the "Share" button at the top right corner.

3. **Add Users or Groups**:

   In the permissions panel, you will see the option to add Users or Groups.

   Type the name of the user or group you want to give access to.

4. **Set Permissions**:

   You can assign different levels of access:

   - **Can Edit**: Allows the user to modify the notebook.
   - **Can View**: Allows the user to only view the notebook but not make any changes.
   - **Can Run**: Allows the user to run the notebook.

5. **Save Permissions**:

   After selecting the appropriate permissions, click Save to apply the changes.

This ensures that the user or group will have the required permissions to access and work with the notebook.