Task 1: Raw Data Ingestion

Notebook 1: Ingest raw weather data (data_ingestion.py)

1. Read the CSV file

```
dbutils.fs.cp("file:/Workspace/Shared/weather_data.csv","dbfs:/FileStore/weather_data.csv")


from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, DateType, FloatType
from pyspark.sql.utils import AnalysisException


spark = SparkSession.builder.appName("WeatherDataIngestion").getOrCreate()


weather_schema = StructType([
    StructField("City", StringType(), True),
    StructField("Date", DateType(), True),
    StructField("Temperature", FloatType(), True),
    StructField("Humidity", FloatType(), True)
])


try:
    weather_df = spark.read.csv("dbfs:/FileStore/weather_data.csv", schema=weather_schema,
      header=True)
except AnalysisException:
    print("Error: File not found.")
```

2. Save the data to a delta table

```
>>  weather_df.write.format("delta").mode("overwrite").save("/delta/raw_weather_data")
```

Task 2: Data Cleaning

Notebook 2: clean raw weather data (data_cleaning.py)

1. Load Data from Delta table

```
>>  weather_raw_df = spark.read.format("delta").load("/delta/raw_weather_data")
```

2. **Remove any rows that contain missing or null values.**

```
>> cleaned_weather_df = weather_raw_df.dropna()
```

3. **Save the cleaned data to a new Delta table.**

```
>> cleaned_weather_df.write.format("delta").mode("overwrite").save("/delta/weather_cleaned")
```

Task 3: Data Transformation

Notebook 3: Data Transformation (data_transformation.py)

1. **Load the cleaned data from the Delta table**

```
>> weather_cleaned_df = spark.read.format("delta").load("/delta/weather_cleaned")
```

2. **Calculate the average temperature and humidity for each city.**

```
>> from pyspark.sql.functions import avg
>> transformed_df = weather_cleaned_df.groupBy("City").agg(
    avg("Temperature").alias("Avg_Temperature"),
    avg("Humidity").alias("Avg_Humidity")
    )
```

3. **Save the transformed data to a Delta table.**

```
>> transformed_df.write.format("delta").mode("overwrite").save("/delta/weather_transformed")
```

Task 4: Create a Pipeline to Execute Notebooks

Notebook 4: Master Notebook to execute other notebooks

```
import subprocess
notebooks = [
   "/ delta/raw_weather_data /data_ingestion.py",
   "/delta/weather_cleaned/data_cleaning.py",
   "/delta/weather_transformed/data_transformation.py"
]
```

```
for notebook in notebooks:
    try:
        subprocess.run(["databricks", "workspace", "import", notebook], check=True)
        print(f"Successfully executed {notebook}")
    except subprocess.CalledProcessError as e:
        print(f"Error occurred while executing {notebook}: {e}")
```

2. **Add Logging to track progress and errors**

```
import logging
logging.basicConfig(filename='/path/to/pipeline_log.log', level=logging.INFO)
try:
    logging.info(f'Successfully executed {notebook}')
except Exception as e:
    logging.error(f'Failed to execute {notebook}: {e}')
```

Bonus Task: Error Handling

1. **Add error handling to manage scenarios like missing files or corrupted data.**

```
import os
if not os.path.exists("dbfs:/FileStore/weather_data.csv"):
    raise FileNotFoundError("Weather data file not found")
```

2. **Log Errors for Analysis**

```
try:
    except Exception as e:
    logging.error(f"Error: {str(e)}")
    error_df = spark.createDataFrame([(str(e),)], ["Error"])
    error_df.write.format("delta").mode("append").save("/delta/error_log")
```