Task 1: Raw Data Ingestion

Notebook 1: Load Weather Data into Delta Table

Sample CSV Data

City	Date	Temperature	Humidity
New York	2024-01-01	30.5	60
Los Angeles	2024-01-01	25.0	65
Chicago	2024-01-01	-5.0	75
Houston	2024-01-01	20.0	80
Phoenix	2024-01-01	15.0	50

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

Load the CSV data into a Delta table

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, DateType, FloatType
import os
import logging
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)
])
file_path = "dbfs:/FileStore/weather_data.csv"
logging.basicConfig(filename='/dbfs/mnt/logs/ingestion_log.log', level=logging.INFO)

if not os.path.exists(file_path):
    logging.error(f"File not found: {file_path}")
else:
    weather_df = spark.read.csv(file_path, schema=weather_schema, header=True)
    weather_df.write.format("delta").mode("overwrite").save("/delta/weather_raw")
    logging.info(f"Weather data successfully ingested and saved to Delta at /delta/weather_raw")
```

Task 2: Data Cleaning

Notebook 2: Clean the Ingested weather data

1. Handle null or incorrect values in the temperature and humidity columns.

```
from pyspark.sql.functions import col
weather_raw_df = spark.read.format("delta").load("/delta/weather_raw")
cleaned_weather_df = weather_raw_df.na.drop()
cleaned_weather_df = cleaned_weather_df.filter((col("Temperature") >= -50) & (col("Humidity")
    >= 0))
```

2. After cleaning, save the updated data to a new Delta table.

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

Task 3: Data Transformation

Notebook 3: Calculate average temperature and humidity.

1. Calculating the average temperature and humidity for each city.

```
from pyspark.sql.functions import avg
cleaned_weather_df = spark.read.format("delta").load("/delta/weather_cleaned")
transformed_df = cleaned_weather_df.groupBy("City").agg(
  avg("Temperature").alias("Avg_Temperature"),
  avg("Humidity").alias("Avg_Humidity")
)
```

2. Save the transformed data into a new Delta table.

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

Task 4: Build and Run a Pipeline

Notebook 4: Databricks Pipeline Execution

1. **Pipeline that executes the following notebooks in sequence**

```
import subprocess
import logging
logging.basicConfig(filename='/dbfs/mnt/logs/pipeline_log.log', level=logging.INFO)
notebooks = [
    "/delta/weather_raw",
    "/delta/weather_cleaned",
    "/delta/weather_transformed"
]

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