Task 1: Data Ingestion - Reading Data from Various Formats

- - Ingest data from different formats (CSV, JSON, Parquet, Delta table)

- - Ingest CSV data (Student Information)

```
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

student_schema = StructType([
    StructField("StudentID", StringType(), True),
    StructField("Name", StringType(), True),
    StructField("Class", StringType(), True),
    StructField("Score", IntegerType(), True)
])

data = [("S001", "Anil Kumar", "10", 85),
        ("S002", "Neha Sharma", "12", 92),
        ("S003", "Rajesh Gupta", "11", 78)
]
columns = ["StudentID", "Name", "Class", "Score"]

student_df = spark.createDataFrame(data, schema=student_schema)
student_df.show()
```

- - Ingest JSON data (City Information)

```
city_json_data = '''
[
  {"CityID": "C001", "CityName": "Mumbai", "Population": 20411000},
  {"CityID": "C002", "CityName": "Delhi", "Population": 16787941},
  {"CityID": "C003", "CityName": "Bangalore", "Population": 8443675}
]
'''
city_rdd = spark.sparkContext.parallelize([city_json_data])
```

```
city_df = spark.read.json(city_rdd)
city_df.show()
```

- - **Ingest Parquet data**

Sample data for hospital_data.parquet

HospitalID	HospitalName	City	Beds	Specialty
H001	Apollo Hospital	Hyderabad	500	Cardiology
H002	Vasan Eye Hospital	Vizag	300	Ophthalmology
H003	Manipal Hospital	Bangalore	450	Orthopedics
H004	Kokilaben Hospital	Mumbai	400	Oncology
H005	Amrita Hospital	Delhi	350	Gastroenterology

```
dbutils.fs.cp("file:/Workspace/hospital_data.parquet","dbfs:/FileStore/hospital_data.parquet")

from pyspark.sql.types import StructType, StructField, StringType, IntegerType
hospital_schema = StructType([
    StructField("HospitalID", StringType(), True),
    StructField("HospitalName", StringType(), True),
    StructField("City", StringType(), True),
    StructField("Beds", IntegerType(), True),
    StructField("Specialty", StringType(), True)
])

data =spark.read.parquet("dbfs:/FileStore/hospital_data.parquet").option("header","true")

parquet_file_path = "dbfs:/FileStore/hospital_data.parquet"
hospital_df = spark.read.parquet(parquet_file_path)
hospital_df.show()
```

- - **Ingest Delta Table**

```
from delta.tables import DeltaTable
delta_table_path = "/delta/hospital_records"

try:
    if DeltaTable.isDeltaTable(spark, delta_table_path):
        hospital_records_df = spark.read.format("delta").load(delta_table_path)
        hospital_records_df.show()
    else:
        raise Exception("Delta table does not exist at the specified path.")
except Exception as e:
    print(f"Error: {str(e)}")
```

Task 2: Writing Data to Various Formats

- - **Write Student Data to CSV**

```
csv_output_path = "dbfs:/FileStore/students.csv"
student_df.write.mode("overwrite").option("header", "true").csv(csv_output_path)
print(f"Student data written to: {csv_output_path}")
```

- - **Write City Data to JSON**

```
json_output_path = "dbfs:/FileStore/cities.json"
city_df.write.mode("overwrite").json(json_output_path)
print(f"City data written to: {json_output_path}")
```

- - **Write Hospital Data to Parquet**

```
parquet_output_path = "dbfs:/FileStore/hospitals_output.parquet"
hospital_df.write.mode("overwrite").parquet(parquet_output_path)
print(f"Hospital data written to: {parquet_output_path}")
```

- - Write Hospital Data to a Delta Table

```
delta_table_path = "/delta/hospitals_delta"
hospital_df.write.format("delta").mode("overwrite").save(delta_table_path)
print(f"Hospital data written to Delta table at: {delta_table_path}")
```

Task 3: Running One Notebook from Another

1. **Create two notebooks**

   - - Notebook A: Ingest, Clean, and Save Data as Delta Table

```
csv_input_path = "dbfs:/FileStore/students.csv"
student_df = spark.read.option("header", "true").csv(csv_input_path)

cleaned_student_df = student_df.dropDuplicates().na.drop()

delta_table_path = " /delta/students_delta"
cleaned_student_df.write.format("delta").mode("overwrite").save(delta_table_path)

print(f"Cleaned student data saved to Delta table at: {delta_table_path}")

dbutils.notebook.run("file:/Workspace/Users/Notebook B", 60)
```

   - - Notebook B: Analyze Delta Table and Write Results to Another Delta Table

```
delta_table_path = "/delta/students_delta"
students_delta_df = spark.read.format("delta").load(delta_table_path)
average_score_df = students_delta_df.groupBy().avg("Score")\
                .withColumnRenamed("avg(Score)", "Average_Score")

result_delta_table_path = "/delta/students_analysis_delta"
average_score_df.write.format("delta").mode("overwrite").save(result_delta_table_path)

print(f"Analysis results saved to Delta table at: {result_delta_table_path}")
```

Task 4: Databricks Ingestion

1. Read data from the following sources

- - CSV file from Azure Data Lake.

```
csv_file_path = "dbfs:/FileStore/students.csv"
csv_df = spark.read.option("header", "true").csv(csv_file_path)
```

- - JSON file stored on Databricks FileStore.

```
json_file_path = "dbfs:/FileStore/cities.json"
json_df = spark.read.json(json_file_path)
```

- - Parquet file from an external data source

```
parquet_file_path = "s3:/FileStore/hospitals.parquet"
parquet_df = spark.read.parquet(parquet_file_path)
```

- - Delta table stored in a Databricks-managed database.

```
delta_table_path = "/delta/hospitals_delta"
delta_df = spark.read.format("delta").load(delta_table_path)
```

2. Write the cleaned data

```
filtered_csv_df = csv_df.filter(csv_df["Score"] > 80)
filtered_csv_df.write.mode("overwrite").option("header", "true")\
                        .csv("dbfs:/FileStore/filtered_students.csv")

json_df.write.mode("overwrite").json("dbfs:/FileStore/cleaned_cities.json")

parquet_df.write.mode("overwrite").parquet("dbfs:/FileStore/cleaned_hospitals.parquet")

delta_df.write.format("delta").mode("overwrite").save("/delta/cleaned_hospitals_delta")

print("Data written to CSV, JSON, Parquet, and Delta formats.")
```

Additional Tasks

Optimization Task

```
delta_table_path = "/delta/hospitals_delta"
spark.sql(f"OPTIMIZE delta.`{delta_table_path}`")
print(f"Delta table at {delta_table_path} optimized.")
```

Z-ordering Task

```
spark.sql(f"OPTIMIZE delta.`{delta_table_path}` ZORDER BY (CityName)")
print(f"Z-ordering applied on 'CityName' column for faster querying.")
```

Vacuum Task

```
spark.sql(f"VACUUM delta.`{delta_table_path}` RETAIN 168 HOURS")
print(f"Vacuum operation completed for Delta table at {delta_table_path}.")
```