Exercise 1:

Used Google Colab

```python
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
import os

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

schema = "VehicleID STRING, Date DATE, ServiceType STRING, ServiceCost FLOAT,
Mileage INT"

csv_file_path = "/dbfs/FileStore/vehicle_maintenance.csv"

try:
    if not os.path.exists(csv_file_path):
        raise FileNotFoundError("File not found at path")

    vehicle_df = spark.read.csv(csv_file_path, schema=schema, header=True)

    if vehicle_df.filter(F.col("VehicleID").isNull() |
F.col("Date").isNull()).count() > 0:
        print("Data contains missing VehicleID or Date values")
    else:
        vehicle_df.write.format("delta").mode("overwrite").save("/delta/vehicle_mai
ntenance")

        print("Data ingestion completed")

except FileNotFoundError as e:
    print(e)

except Exception as e:
    print(f"Error during ingestion: {e}")
```

```python
# Task 2
vehicle_df = spark.read.format("delta").load("/delta/vehicle_maintenance")

cleaned_df = vehicle_df.filter((F.col("ServiceCost") > 0) & (F.col("Mileage") > 0))

cleaned_df = cleaned_df.dropDuplicates(["VehicleID"])

cleaned_df.write.format("delta").mode("overwrite").save("/delta/cleaned_vehicle_mai
ntenance")

print("data cleaned")
```

```
# Task 3
cleaned_vehicle_df =
spark.read.format("delta").load("/delta/cleaned_vehicle_maintenance")

total_cost_df = cleaned_vehicle_df.groupBy("VehicleID") \
                                  .agg(F.sum("ServiceCost").alias("TotalMaintenance
Cost"))

high_mileage_df = cleaned_vehicle_df.filter(F.col("Mileage") > 30000) \
                                    .select("VehicleID", "Mileage") \
                                    .distinct()

total_cost_df.write.format("delta").mode("overwrite").save("/delta/total_cost")

high_mileage_df.write.format("delta").mode("overwrite").save("/delta/high_mileage")

print("Vehicle maintenance analysis completed")
```

```
# Task 4

spark.sql("VACUUM '/delta/vehicle_maintenance_analysis' RETAIN 168 HOURS")

spark.sql("DESCRIBE HISTORY '/delta/vehicle_maintenance_analysis'")
```

Exercise 2:

```
# Task 1
from pyspark.sql import SparkSession
import os
import pyspark.sql.functions as F

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

schema = "UserID STRING, MovieID STRING, Rating INT, Timestamp STRING"

csv_file_path = "/dbfs/FileStore/movie_ratings.csv"

try:
    if not os.path.exists(csv_file_path):
        raise FileNotFoundError("File not found")

    ratings_df = spark.read.csv(csv_file_path, schema=schema, header=True)

    inconsistent_data = ratings_df.filter(F.col("UserID").isNull() |
                                          F.col("MovieID").isNull() |
                                          F.col("Rating").isNull() |
                                          (F.col("Rating") < 1) |
                                          (F.col("Rating") > 5))

    if inconsistent_data.count() > 0:
        print("missing data found")
    else:
        ratings_df.write.format("delta").mode("overwrite").save("/delta/movie_ratin
gs")

        print("Data ingestion successful")

except FileNotFoundError as e:
    print(e)

except Exception as e:
    print(f"Error during ingestion: {e}")
```

```
# Task 2
ratings_df = spark.read.format("delta").load("/delta/movie_ratings")

cleaned_df = ratings_df.filter((F.col("Rating") >= 1) & (F.col("Rating") <= 5))

cleaned_df = cleaned_df.dropDuplicates(["UserID", "MovieID"])

cleaned_df.write.format("delta").mode("overwrite").save("/delta/cleaned_movie_ratin
gs")

print("Movie ratings data cleaned")
```

```
# Task 3
ratings_df = spark.read.format("delta").load("/delta/cleaned_movie_ratings")

avg_ratings_df =
ratings_df.groupBy("MovieID").agg(F.avg("Rating").alias("AverageRating"))

highest_rated_movie =
avg_ratings_df.orderBy(F.col("AverageRating").desc()).limit(1)
lowest_rated_movie = avg_ratings_df.orderBy(F.col("AverageRating").asc()).limit(1)

avg_ratings_df.write.format("delta").mode("overwrite").save("/delta/movie_ratings_a
nalysis")
highest_rated_movie.write.format("delta").mode("overwrite").save("/delta/highest_ra
ted")
lowest_rated_movie.write.format("delta").mode("overwrite").save("/delta/lowest_rate
d")

print("Movie ratings analysis saved")
```

```
# Task 4

ratings_df = spark.read.format("delta").load("/delta/cleaned_movie_ratings")
updated_ratings_df = ratings_df.withColumn("Rating", F.when(F.col("MovieID") ==
"M001", 5).otherwise(F.col("Rating")))

updated_ratings_df.write.format("delta").mode("overwrite").save("/delta/cleaned_mov
ie_ratings")
```

```
original_ratings_df = spark.read.format("delta").option("versionAsOf",
0).load("/delta/cleaned_movie_ratings")

spark.sql("DESCRIBE HISTORY '/delta/cleaned_movie_ratings'")
```

```
# Task 5

spark.sql("OPTIMIZE '/delta/cleaned_movie_ratings' ZORDER BY (MovieID)")

spark.sql("VACUUM '/delta/cleaned_movie_ratings' RETAIN 168 HOURS")
```

Exercise 3:

```
# Exercise 3
# Task 1
from pyspark.sql import SparkSession

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

csv_data = [("S001", "Anil Kumar", 10, 85),
            ("S002", "Neha Sharma", 12, 92),
            ("S003", "Rajesh Gupta", 11, 78)]

csv_df = spark.createDataFrame(csv_data, ["StudentID", "Name", "Class", "Score"])

json_data = [
    {"CityID": "C001", "CityName": "Mumbai", "Population": 20411000},
    {"CityID": "C002", "CityName": "Delhi", "Population": 16787941},
    {"CityID": "C003", "CityName": "Bangalore", "Population": 8443675}
]

json_df = spark.read.json(json_data)

parquet_df = spark.read.parquet("/path/to/hospital_data.parquet")

try:
    delta_df = spark.read.format("delta").load("/delta/hospital_records")
except Exception as e:
    print("Error loading Delta table")
```

```
# Task 2
csv_df.write.csv("/dbfs/FileStore/student_data.csv", header=True)

json_df.write.json("/dbfs/FileStore/city_data.json")

parquet_df.write.parquet("/dbfs/FileStore/hospital_data.parquet")

parquet_df.write.format("delta").mode("overwrite").save("/delta/hospital_data")
```

```python
# Task 3
# Notebook one

student_df = spark.read.csv("/dbfs/FileStore/student_data.csv", header=True)

cleaned_student_df = student_df.dropDuplicates().na.fill({"Score": 0})

cleaned_student_df.write.format("delta").mode("overwrite").save("/delta/cleaned_stu
dent_data")

dbutils.notebook.run("/Workspace/user/Notebook_B", 60)
```

```python
# Notebook b
cleaned_student_df = spark.read.format("delta").load("/delta/cleaned_student_data")

average_score_df =
cleaned_student_df.groupBy("Class").agg(F.avg("Score").alias("AverageScore"))

average_score_df.write.format("delta").mode("overwrite").save("/delta/average_stude
nt_scores")
```

```python
# Task 4
azure_df = spark.read.csv("abfss://account/data/data.csv", header=True)

databricks_json_df = spark.read.json("/FileStore/path/to/data.json")

s3_parquet_df = spark.read.parquet("s3://<bucket>/data.parquet")

delta_table_df = spark.read.format("delta").load("/delta/databricks_table")

cleaned_df = azure_df.filter(F.col("Score") > 50)

total_score_df =
cleaned_df.groupBy("Class").agg(F.sum("Score").alias("TotalScore"))


cleaned_df.write.csv("/dbfs/FileStore/cleaned_data.csv", header=True)
cleaned_df.write.json("/dbfs/FileStore/cleaned_data.json")
cleaned_df.write.parquet("/dbfs/FileStore/cleaned_data.parquet")
cleaned_df.write.format("delta").mode("overwrite").save("/delta/cleaned_data")

total_score_df.write.format("delta").mode("overwrite").save("/delta/total_student_s
cores")
```

```
# Additional Tasks
spark.sql("OPTIMIZE '/delta/cleaned_data'")

spark.sql("OPTIMIZE '/delta/cleaned_data' ZORDER BY (CityName)")

spark.sql("VACUUM '/delta/cleaned_data' RETAIN 168 HOURS")
```