**Exercise 1:**

**Used Google Colab**

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\_maintenance")

        print("Data ingestion completed")

except FileNotFoundError as e:

    print(e)

except Exception as e:

    print(f"Error during ingestion: {e}")

# 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\_maintenance")

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("TotalMaintenanceCost"))

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\_ratings")

        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\_ratings")

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\_analysis")

highest\_rated\_movie.write.format("delta").mode("overwrite").save("/delta/highest\_rated")

lowest\_rated\_movie.write.format("delta").mode("overwrite").save("/delta/lowest\_rated")

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\_movie\_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")

# 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\_student\_data")

dbutils.notebook.run("/Workspace/user/Notebook\_B", 60)

# 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\_student\_scores")

# 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\_scores")

# 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")