Task 1: Vehicle Maintenance Data Ingestion

CSV data representing Vehicle maintenance records

VehicleID	Date	ServiceType	ServiceCost	Mileage
V001	2024-04-01	Oil Change	50.00	15000
V002	2024-04-05	Tire Replacement	400.00	30000
V003	2024-04-10	Battery Replacement	120.00	25000
V004	2024-04-15	Brake Inspection	200.00	40000
V005	2024-04-20	Oil Change	50.00	18000

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

- - Ingest this CSV data into a Delta table in Databricks.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when
import logging

spark = SparkSession.builder \
    .appName("Vehicle Maintenance Data Ingestion") \
    .getOrCreate()

logging.basicConfig(level=logging.INFO, filename="/dbfs/logs/vehicle_data_ingestion.log")

file_path = "dbfs:/FileStore/vehicle_records.csv"

try:
    df = spark.read.option("header", True).csv(file_path)

    df_clean = df.withColumn("ServiceCost", col("ServiceCost").cast("double")) \
                .withColumn("Mileage", col("Mileage").cast("int")) \
                .filter((col("ServiceCost").isNotNull()) & (col("Mileage") >= 0))

    df_invalid = df.subtract(df_clean)
    if df_invalid.count() > 0:
        logging.error(f"Invalid data found: {df_invalid.show(truncate=False)}")
```

```
    df_clean.write.format("delta").mode("overwrite").save("/delta/vehicle_records")
```

- - Add error handling for cases where the file is missing or contains incorrect data

```
except FileNotFoundError as e:
    logging.error(f"File not found: {e}")
except Exception as e:
    logging.error(f"Error during data ingestion: {e}")
```

Task 2: Data Cleaning

```
    from pyspark.sql import functions as F

    df = spark.read.format("delta").load("/delta/vehicle_maintenance")
```

- - Ensure that the ServiceCost and Mileage columns contain valid positive values.

```
    df_clean = df.filter((F.col("ServiceCost") > 0) & (F.col("Mileage") > 0))
```

- - Remove any duplicate records based on VehicleID and Date .

```
    df_clean = df_clean.dropDuplicates(["VehicleID", "Date"])
```

- - Save the cleaned data to a new Delta table.

```
    df_clean.write.format("delta").mode("overwrite").save("/delta/cleaned_vehicle_maintenance")
    df_clean.show()
```

Task 3: Vehicle Maintenance Analysis

```
df_cleaned = spark.read.format("delta").load("/delta/cleaned_vehicle_maintenance")

- - Calculate the total maintenance cost for each vehicle.
df_total_cost = df_cleaned.groupBy("VehicleID") \
    .agg(F.sum("ServiceCost").alias("TotalMaintenanceCost"))
df_total_cost.show()

- - Identify vehicles that have exceeded a certain mileage threshold
mileage_threshold = 30000
df_high_mileage = df_cleaned.filter(F.col("Mileage") > mileage_threshold)
df_high_mileage.show()

- - Save the analysis results to a Delta table.
df_total_cost.write.format("delta").mode("overwrite").save("/delta/total_maintenance_cost")
df_high_mileage.write.format("delta").mode("overwrite").save("/delta/high_mileage_vehicles")
```

Task 4: Data Governance with Delta Lake

```
delta_table_path = "/delta/cleaned_vehicle_maintenance"

- - Use VACUUM to clean up old data from the Delta table.
spark.sql(f"VACUUM '{delta_table_path}' RETAIN 168 HOURS")

- - Use DESCRIBE HISTORY to check the history of updates to the maintenance records.
history_df = spark.sql(f"DESCRIBE HISTORY '{delta_table_path}'")
history_df.show(truncate=False)
```