```python
# -*- coding: utf-8 -*-
import os
import shutil
import pyspark
from pyspark.sql.window import Window
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import traceback
from datetime import datetime

#*********************************************************************//
# Note: Please refer to problem statements from challenge.htm file    //
# found under instructions folder for detailed requirements. We have  //
# defined placeholder functions where you need to add your code that  //
# may solve one or more of the the problem statements. Within the     //
# functions we have added DEFAULT code which when executed without    //
# any modification would create empty dataframe. This approach will   //
# enable you to test your code intermediate without worrying about    //
# syntax failure outside of your code. To SOLVE the problem, you are  //
# expected to REPLACE the code which creates dummy dataframe with     //
# your ACTUAL code.                                                   //
#*********************************************************************//
# Note: We have also added code to print sample data to console in    //
# each of the functions to help you visualize your intermediate data  //
#*********************************************************************//


def read_data(spark):
    '''
    spark_session : spark
    customSchema : we have given the custom schema
    '''
    print("--------------------")
    print("Starting read_data")
    print("--------------------")

    # TODO: REPLACE WITH YOUR ACTUAL BUCKET NAME
    # Your bucket name should start with "car-data" followed by random digits
    # Example: "car-data123456789"
    bucket_name = "YOUR_BUCKET_NAME_HERE"  # Replace with actual bucket name like "car-data123456789"

    s3_input_path = "s3://" + bucket_name + "/inputfile/car_data.csv"

    # Read CSV file from S3 with header=true
    df = spark.read.format("csv") \
        .option("header", "true") \
        .option("inferSchema", "true") \
        .load(s3_input_path)

    print("Data read successfully. Sample data:")
    df.show(5)
    print(f"Total rows: {df.count()}")
    print(f"Schema: {df.printSchema()}")

    return df

def clean_data(input_df):
    '''
    for input file: input_df is output of read_data function
    '''
    print("--------------------")
    print("Starting clean_data")
    print("--------------------")

    # Start with input dataframe
    df = input_df

    print(f"Original data count: {df.count()}")

    # 1. Drop rows with null values in any column
    df = df.dropna()
    print(f"After dropping null values: {df.count()}")

    # 2. Drop duplicate rows
```

```
    df = df.dropDuplicates()
    print(f"After dropping duplicates: {df.count()}")

    print("Clean data sample:")
    df.show(5)

    return df

def s3_load_data(data, file_name):
    '''
    data  : the output data of clean_data function
    file_name : the name of the output to be stored inside the s3
    '''
    # TODO: REPLACE WITH YOUR ACTUAL BUCKET NAME
    # Use the same bucket name as in read_data function
    bucket_name = "YOUR_BUCKET_NAME_HERE"  # Replace with actual bucket name like "car-data123456789"

    output_path = "s3://" + bucket_name + "/output/" + file_name

    if data.count() != 0:
        print("Loading the data", output_path)

        # Write data to S3 as single partition CSV with header
        data.coalesce(1) \
            .write \
            .mode("overwrite") \
            .option("header", "true") \
            .csv(output_path)

        print(f"Data successfully saved to {output_path}")
    else:
        print("Empty dataframe, hence cannot save the data", output_path)


def result_1(input_df):
    '''
    for input file: input_df is output of clean_data function
    '''
    print("-------------------------")
    print("Starting result_1")
    print("-------------------------")

    # Group by car_name and calculate average selling price and count
    df = input_df.groupBy("car_name") \
        .agg(
            avg("selling_price").alias("average_selling_price"),
            count("*").alias("car_count")
        ) \
        .filter(col("car_count") > 2) \
        .select("car_name", "average_selling_price", "car_count")

    print("Result 1 sample data:")
    df.show(10)
    print(f"Total records with car_count > 2: {df.count()}")

    return df


def result_2(input_df):
    '''
    for input file: input_df is output of clean_data function
    '''
    print("-------------------------")
    print("Starting result_2")
    print("-------------------------")

    # Create price_per_km column, filter and round
    df = input_df.withColumn("price_per_km",
                             round(col("selling_price") / col("km_driven"), 2)) \
        .filter(col("price_per_km") < 10) \
        .select("car_name", "year", "selling_price", "km_driven",
                "fuel", "seller_type", "transmission", "owner", "price_per_km")

    print("Result 2 sample data:")
    df.show(10)
```

```
    print(f"Total records with price_per_km < 10: {df.count()}")

    return df


def rds_mysql_load_data(data, table_name):
    if data.count() != 0:
        print(f"Loading the data into RDS table : {table_name}...")

        # TODO: UPDATE THESE RDS CONNECTION DETAILS
        # Replace with your actual RDS endpoint
        jdbcUrl = "jdbc:mysql://YOUR_RDS_ENDPOINT:3306/dev"  # Replace YOUR_RDS_ENDPOINT with actual RDS endpoint
        username = "admin"      # This should match what you set during RDS creation
        password = "Awsuser1"   # This should match what you set during RDS creation (case-sensitive)

        data.write.mode('overwrite') \
            .format('jdbc') \
            .option('url', jdbcUrl) \
            .option('dbtable', table_name) \
            .option('user', username) \
            .option('password', password) \
            .option("driver", "com.mysql.cj.jdbc.Driver") \
            .save()

        print(f"Data successfully loaded into RDS table: {table_name}")
    else:
        print(f"Empty dataframe, hence cannot load the data into RDS table : {table_name}...")


def main():
    """ Main driver program to control the flow of execution.
        Please DO NOT change anything here.
    """

    spark = (SparkSession.builder.appName("Car Data analysis").getOrCreate())
    spark.sparkContext.setLogLevel("ERROR")
    clean_data_path = "cleaned_data"

    try:
        task_1 = read_data(spark)
    except Exception as e:
        print("Getting error in the read_data function", e)
        traceback.print_exc()
    try:
        task_2 = clean_data(task_1)
    except Exception as e:
        print("Getting error in the clean_data function", e)
        traceback.print_exc()
    try:
        task_3 = result_1(task_2)
    except Exception as e:
        print("Getting error in the result_1 function", e)
        traceback.print_exc()
    try:
        task_4 = result_2(task_2)
    except Exception as e:
        print("Getting error in the result_2 function", e)
        traceback.print_exc()
    try:
        s3_load_data(task_2, clean_data_path)
    except Exception as e:
        print("Getting error while loading clean_data", e)
        traceback.print_exc()

    try:
        rds_mysql_load_data(task_3, "average_selling_price")
    except Exception as e:
        print("Getting error while loading rds data", e)
        traceback.print_exc()

    try:
        rds_mysql_load_data(task_4, "price_per_km")
    except Exception as e:
        print("Getting error while loading rds data", e)
        traceback.print_exc()
```

```
    spark.stop()


if __name__ == "__main__":
    main()
```