1. E-commerce Sales Data Cleaning and Aggregation

1) Loading the csv file

- This file consists of 1k unique rows with 7 different columns

Column Name	TransactionID	ProductID	Product Category	Quantity	Price	CustomerCity	TransactionDate
Data Type	String	String	String	Integer	Float	String	Date

- Loaded the file in colab directory and into dataframe with custom schema - df

2) Cleaning the data frame

- Quantity - There are negative which needs to be removed - using filter - stored in df_valid_data
```
df_valid_data = df.filter((col("Quantity") > 0) |
col("Quantity").isNull())
```
- Quantity - There are NULL values, we fill the missing values as 1(assuming a missing quantity implies a single item)
```
df_updated_quantity = df_valid_data.fillna(1, subset=["Quantity"])
```
- Price - Drop rows where price is null (since calculating revenue requires a price).
```
df_updated_quantity_price =
df_updated_quantity.dropna(subset=["Price"])
```

3) Aggregation

- Total_per_transaction - Calculating it by multiplying Quantity and Price
```
df_total_per_transaction =
df_updated_quantity_price.withColumn("total_per_transaction",
(col("Quantity")*col("Price")))
```
- Total_per_day - Calculating the revenue per day using window function
```
from pyspark.sql.window import Window
from pyspark.sql.functions import sum
window_spec = Window.partitionBy("TransactionDate")
df_total_per_day =
df_total_per_transaction.withColumn("total_per_day",

sum(col("total_per_transaction")).over(window_spec))
df_result =
df_total_per_day.select("TransactionDate","total_per_day").distinct(
).orderBy("TransactionDate")
```

4) Writing the df_total_per_day to output folder
```
output_folder = "daily_sales_output"
df_final_result.coalesce(1).write.csv(output_folder, mode="overwrite",
header=True)
print(f"Data successfully written to the folder: {output_folder}")
```