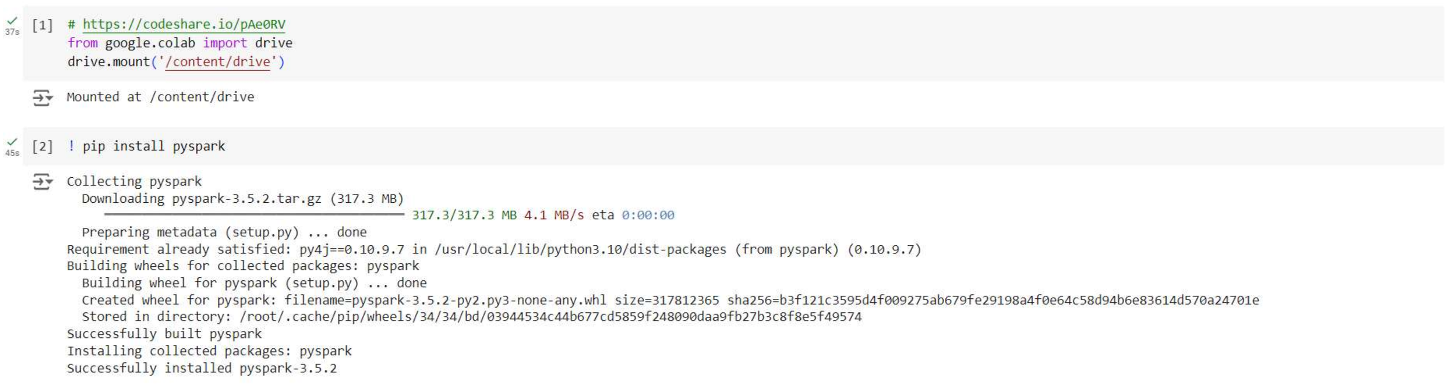
PySpark_WorkingWithCSV_HandsOn_3Sep

SaiPrabath Chowdary S

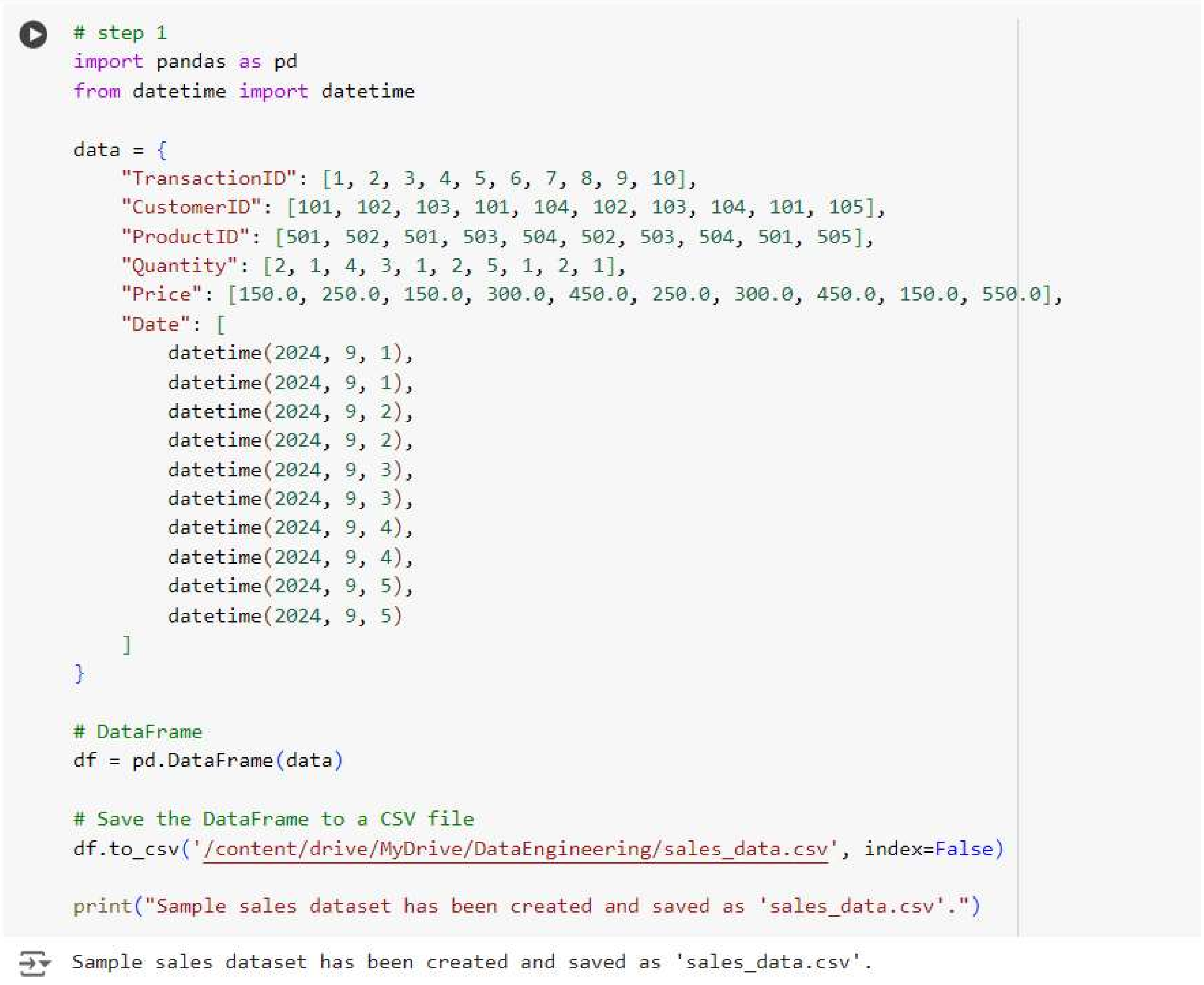


```
[1] # https://codeshare.io/pAe0RV
    from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[2] ! pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.5.2.tar.gz (317.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.3/317.3 MB 4.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.2-py2.py3-none-any.whl size=317812365 sha256=b3f121c3595d4f009275ab679fe29198a4f0e64c58d94b6e83614d570a24701e
  Stored in directory: /root/.cache/pip/wheels/34/34/bd/03944534c44b677cd5859f248090daa9fb27b3c8f8e5f49574
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.2
```



```python
# step 1
import pandas as pd
from datetime import datetime

data = {
    "TransactionID": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    "CustomerID": [101, 102, 103, 101, 104, 102, 103, 104, 101, 105],
    "ProductID": [501, 502, 501, 503, 504, 502, 503, 504, 501, 505],
    "Quantity": [2, 1, 4, 3, 1, 2, 5, 1, 2, 1],
    "Price": [150.0, 250.0, 150.0, 300.0, 450.0, 250.0, 300.0, 450.0, 150.0, 550.0],
    "Date": [
        datetime(2024, 9, 1),
        datetime(2024, 9, 1),
        datetime(2024, 9, 2),
        datetime(2024, 9, 2),
        datetime(2024, 9, 3),
        datetime(2024, 9, 3),
        datetime(2024, 9, 4),
        datetime(2024, 9, 4),
        datetime(2024, 9, 5),
        datetime(2024, 9, 5)
    ]
}

# DataFrame
df = pd.DataFrame(data)

# Save the DataFrame to a CSV file
df.to_csv('/content/drive/MyDrive/DataEngineering/sales_data.csv', index=False)

print("Sample sales dataset has been created and saved as 'sales_data.csv'.")
```

```
Sample sales dataset has been created and saved as 'sales_data.csv'.
```

```
# step 2

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Sales Dataset Analysis").getOrCreate()

sales_df = spark.read.csv("/content/drive/MyDrive/DataEngineering/sales_data.csv", header=True, inferSchema=True)

sales_df.show(4) #verify
```

```
+-------------+----------+---------+--------+-----+----------+
|TransactionID|CustomerID|ProductID|Quantity|Price|      Date|
+-------------+----------+---------+--------+-----+----------+
|            1|       101|      501|       2|150.0|2024-09-01|
|            2|       102|      502|       1|250.0|2024-09-01|
|            3|       103|      501|       4|150.0|2024-09-02|
|            4|       101|      503|       3|300.0|2024-09-02|
+-------------+----------+---------+--------+-----+----------+
only showing top 4 rows
```

[6]
```
# step 3

# printing schema
sales_df.printSchema()

# first 5 rows
sales_df.show(5)

# summary statistics for numeric columns
sales_df.describe(["Quantity", "Price"]).show()
```

```
root
 |-- TransactionID: integer (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- ProductID: integer (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- Price: double (nullable = true)
 |-- Date: date (nullable = true)

+-------------+----------+---------+--------+-----+----------+
|TransactionID|CustomerID|ProductID|Quantity|Price|      Date|
+-------------+----------+---------+--------+-----+----------+
|            1|       101|      501|       2|150.0|2024-09-01|
|            2|       102|      502|       1|250.0|2024-09-01|
|            3|       103|      501|       4|150.0|2024-09-02|
|            4|       101|      503|       3|300.0|2024-09-02|
|            5|       104|      504|       1|450.0|2024-09-03|
+-------------+----------+---------+--------+-----+----------+
only showing top 5 rows

+-------+-----------------+-----------------+
|summary|         Quantity|            Price|
+-------+-----------------+-----------------+
|  count|               10|               10|
|   mean|              2.2|            300.0|
| stddev|1.398411797560202|141.4213562373095|
|    min|                1|            150.0|
|    max|                5|            550.0|
+-------+-----------------+-----------------+
[19] # step 4

     # 1.Calculate the Total Sales Value for Each Transaction
     sales_df = sales_df.withColumn("TotalSales", sales_df["Quantity"] * sales_df["Price"])
     sales_df.show()
```

```
+-------------+----------+---------+--------+-----+----------+----------+
|TransactionID|CustomerID|ProductID|Quantity|Price|      Date|TotalSales|
+-------------+----------+---------+--------+-----+----------+----------+
|            1|       101|      501|       2|150.0|2024-09-01|     300.0|
|            2|       102|      502|       1|250.0|2024-09-01|     250.0|
|            3|       103|      501|       4|150.0|2024-09-02|     600.0|
|            4|       101|      503|       3|300.0|2024-09-02|     900.0|
|            5|       104|      504|       1|450.0|2024-09-03|     450.0|
|            6|       102|      502|       2|250.0|2024-09-03|     500.0|
|            7|       103|      503|       5|300.0|2024-09-04|    1500.0|
|            8|       104|      504|       1|450.0|2024-09-04|     450.0|
|            9|       101|      501|       2|150.0|2024-09-05|     300.0|
|           10|       105|      505|       1|550.0|2024-09-05|     550.0|
+-------------+----------+---------+--------+-----+----------+----------+
```

```
[20] # 2.Group By ProductID and Calculate Total Sales Per Product
     total_sales_by_product = sales_df.groupBy("ProductID").sum("TotalSales").withColumnRenamed("sum(TotalSales)","TotalSales")\
                              .orderBy("sum(TotalSales)", ascending=False)
     total_sales_by_product.show()
```

```
+---------+----------+
|ProductID|TotalSales|
+---------+----------+
|      503|    2400.0|
|      501|    1200.0|
|      504|     900.0|
|      502|     750.0|
|      505|     550.0|
+---------+----------+
```

```
[21] # 3.Identify the Top-Selling Product
     top_selling_product = total_sales_by_product.limit(1)
     top_selling_product.show()
```

```
+---------+----------+
|ProductID|TotalSales|
+---------+----------+
|      503|    2400.0|
+---------+----------+
```

```
# 4.Calculate the Total Sales by Date
total_sales_by_date = sales_df.groupBy("Date").sum('TotalSales').orderBy("Date")
total_sales_by_date.show()
```

```
+----------+---------------+
|      Date|sum(TotalSales)|
+----------+---------------+
|2024-09-01|          550.0|
|2024-09-02|         1500.0|
|2024-09-03|          950.0|
|2024-09-04|         1950.0|
|2024-09-05|          850.0|
+----------+---------------+
```

```python
# 5.Filter High-Value Transactions
high_value_transactions = sales_df.filter(sales_df["TotalSales"] > 500)
high_value_transactions.show()
```

```
+-------------+----------+---------+--------+-----+----------+----------+
|TransactionID|CustomerID|ProductID|Quantity|Price|      Date|TotalSales|
+-------------+----------+---------+--------+-----+----------+----------+
|            3|       103|      501|       4|150.0|2024-09-02|     600.0|
|            4|       101|      503|       3|300.0|2024-09-02|     900.0|
|            7|       103|      503|       5|300.0|2024-09-04|    1500.0|
|           10|       105|      505|       1|550.0|2024-09-05|     550.0|
+-------------+----------+---------+--------+-----+----------+----------+
```

[24]
```python
# 6.Identify Repeat Customers
repeat_customers = sales_df.groupBy("CustomerID").count().filter("count > 1").orderBy("count", ascending=False)
repeat_customers.show()
```

```
+----------+-----+
|CustomerID|count|
+----------+-----+
|       101|    3|
|       103|    2|
|       102|    2|
|       104|    2|
+----------+-----+
```

```python
# 7.Calculate the Average Sale Price Per Product
average_sale_price = sales_df.groupBy("ProductID").avg('Price').withColumnRenamed("avg(Price)","AveragePrice").orderBy("AveragePrice", ascending=False)
average_sale_price.show()
```

```
+---------+------------+
|ProductID|AveragePrice|
+---------+------------+
|      505|       550.0|
|      504|       450.0|
|      503|       300.0|
|      502|       250.0|
|      501|       150.0|
+---------+------------+
```