

## Assignment 2: Working with JSON Data (product\_data.json)

### Tasks:

#### 1. Load the JSON data:

- Load the product\_data.json file into a DataFrame.
- Display the first 10 rows and inspect the schema.

▶

✓ 1 minute ago (1s)

6

```
1 from pyspark.sql import functions as F
2 # Copy the file from the local workspace to DBFS
3 dbutils.fs.cp("file:/Workspace/Shared/product_data.json", "dbfs:/FileStore/product_data.json")
4 # Load the JSON file
5 df = spark.read.format("json").load("dbfs:/FileStore/product_data.json",multiline=True)
6
7 # Display the first 10 rows
8 display(df.limit(10))
9 |
10 # Inspect the schema
11 df.printSchema()
```

Output

Terminal

Debug Console

	<sup>A</sup> <sub>C</sub> Category	<sup>1</sup> <sub>3</sub> Price	<sup>1</sup> <sub>3</sub> ProductID	<sup>A</sup> <sub>C</sub> ProductName	<sup>1</sup> <sub>3</sub> Stock
1	Electronics	1200	101	Laptop	35
2	Electronics	800	102	Smartphone	80
3	Furniture	150	103	Desk Chair	60
4	Electronics	300	104	Monitor	45
5	Furniture	350	105	Desk	25

↓

5 rows | 0.93 seconds runtime

root

```
|-- Category: string (nullable = true)
|-- Price: long (nullable = true)
|-- ProductID: long (nullable = true)
|-- ProductName: string (nullable = true)
|-- Stock: long (nullable = true)
```

## 2. Data Cleaning:

- Remove rows where Stock is less than 30.
- Filter the products that belong to the "Electronics" category.

```
▶ ✓ Just now (<1s) 6

1 # Remove rows where Stock is less than 30
2 df_cleaned = df.filter(col('Stock') >= 30)
3 df_cleaned.show()
4
5 # Filter products that belong to the "Electronics" category
6 filtered_products = df_cleaned.filter(df_cleaned['Category'] == 'Electronics')
7 filtered_products.show()
```

Output Terminal Debug Console

```
▶ df_cleaned: pyspark.sql.dataframe.DataFrame = [Category: string, Price: long ... 3 more fields]
▶ filtered_products: pyspark.sql.dataframe.DataFrame = [Category: string, Price: long ... 3 more fields]
```

```
+-----+-----+-----+-----+-----+
|  Category|Price|ProductID|ProductName|Stock|
+-----+-----+-----+-----+-----+
|Electronics| 1200|    101|    Laptop|   35|
|Electronics|   800|    102|Smartphone|   80|
| Furniture|   150|    103|Desk Chair|   60|
|Electronics|   300|    104|   Monitor|   45|
+-----+-----+-----+-----+-----+
```

```
+-----+-----+-----+-----+-----+
|  Category|Price|ProductID|ProductName|Stock|
+-----+-----+-----+-----+-----+
|Electronics| 1200|    101|    Laptop|   35|
|Electronics|   800|    102|Smartphone|   80|
|Electronics|   300|    104|   Monitor|   45|
+-----+-----+-----+-----+-----+
```

### 3. Data Aggregation:

- Calculate the total stock for products in the "Furniture" category.
- Find the average price of all products in the dataset.

```
Just now (<1s) 6

1 # Calculate the total stock for products in the "Furniture" category
2 total_furniture_stock = df_cleaned.filter(df_cleaned['Category'] == 'Furniture').agg(F.sum('Stock').alias('total_stock'))
3 total_furniture_stock.show()
4
5 # Find the average price of all products in the dataset
6 average_price = df_cleaned.agg(F.avg('Price').alias('average_price'))
7 average_price.show()
8
```

Output Terminal Debug Console

```
▶ total_furniture_stock: pyspark.sql.dataframe.DataFrame = [total_stock: long]
▶ average_price: pyspark.sql.dataframe.DataFrame = [average_price: double]

+-----+
|total_stock|
+-----+
|          60|
+-----+

+-----+
|average_price|
+-----+
|          612.5|
+-----+
```

### 4. Write the Data to JSON:

- Save the cleaned and aggregated data into a new JSON file.

```
1 minute ago (<1s) 6

1 # Save the cleaned and aggregated data to a new JSON file
2 df_cleaned.write.mode("overwrite").json("dbfs:/FileStore/cleaned_product_data.json")
```