Name: Maclang, Waken Cean C. **CSDS 312:** Machine Problem 2
Course: BSCS 3A **Date:** October 23, 2025

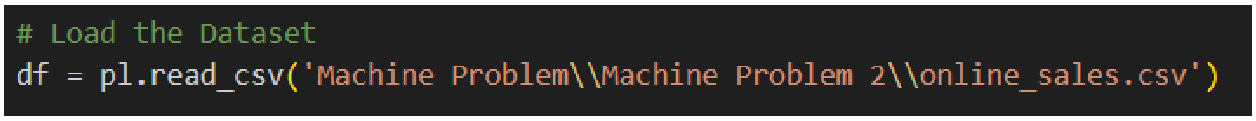
Submission

Document and report the steps and show the results following the tasks in order with reference to the expected output. In reporting the cleaned dataset, paste a screenshot of the table taken from the sheet's or MS Excels's interface. Upload the report in a PDF using the link provided.

Note: You can find the code used for this Machine Problem in this link (MachineProblem2.py)

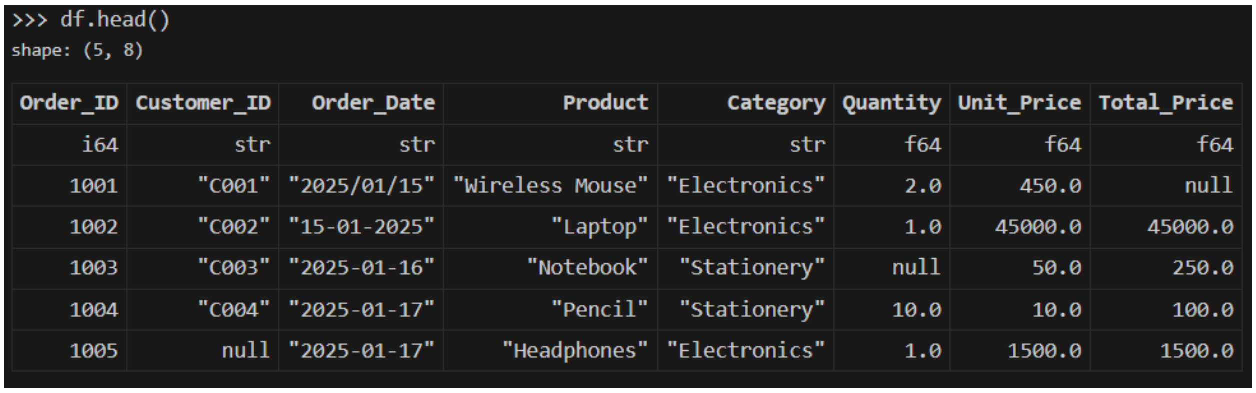
Part 1: Collecting and Exploring Data

1. Load the dataset using polars.



```
# Load the Dataset
df = pl.read_csv('Machine Problem\\Machine Problem 2\\online_sales.csv')
```

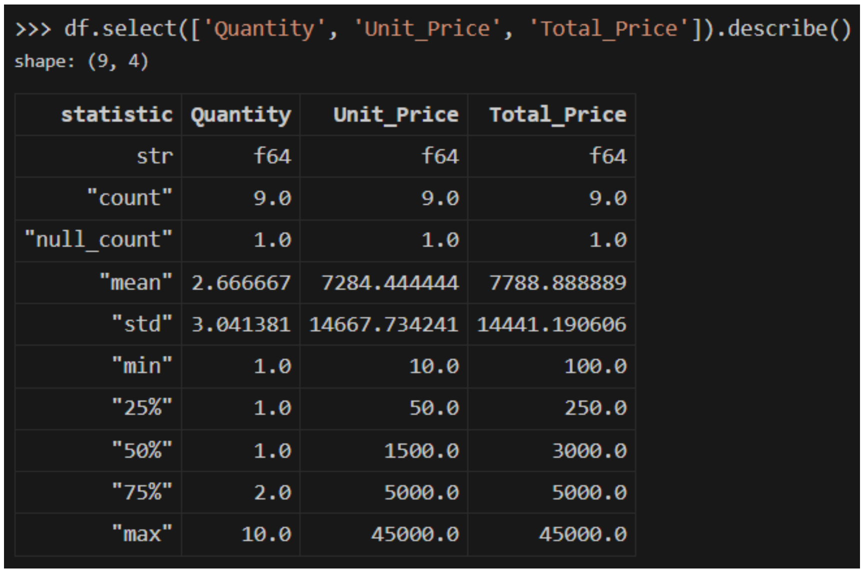
2. Display the first few rows and basic info about the dataset.



```
>>> df.head()
shape: (5, 8)
```

Order_ID	Customer_ID	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
i64	str	str	str	str	f64	f64	f64
1001	"C001"	"2025/01/15"	"Wireless Mouse"	"Electronics"	2.0	450.0	null
1002	"C002"	"15-01-2025"	"Laptop"	"Electronics"	1.0	45000.0	45000.0
1003	"C003"	"2025-01-16"	"Notebook"	"Stationery"	null	50.0	250.0
1004	"C004"	"2025-01-17"	"Pencil"	"Stationery"	10.0	10.0	100.0
1005	null	"2025-01-17"	"Headphones"	"Electronics"	1.0	1500.0	1500.0

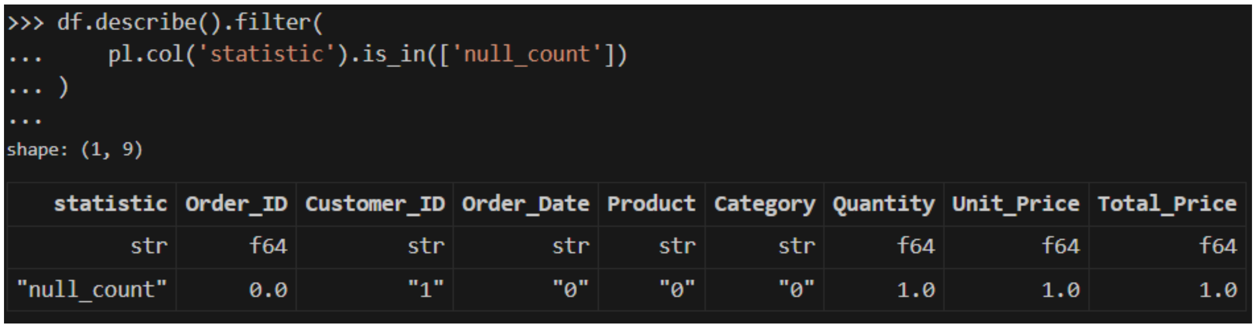
3. Compute summary statistics for numeric columns (Quantity, Unit_Price, Total_Price).



```
>>> df.select(['Quantity', 'Unit_Price', 'Total_Price']).describe()
shape: (9, 4)
```

statistic	Quantity	Unit_Price	Total_Price
str	f64	f64	f64
"count"	9.0	9.0	9.0
"null_count"	1.0	1.0	1.0
"mean"	2.666667	7284.444444	7788.888889
"std"	3.041381	14667.734241	14441.190606
"min"	1.0	10.0	100.0
"25%"	1.0	50.0	250.0
"50%"	1.0	1500.0	3000.0
"75%"	2.0	5000.0	5000.0
"max"	10.0	45000.0	45000.0

4. Find the number of missing values per column.

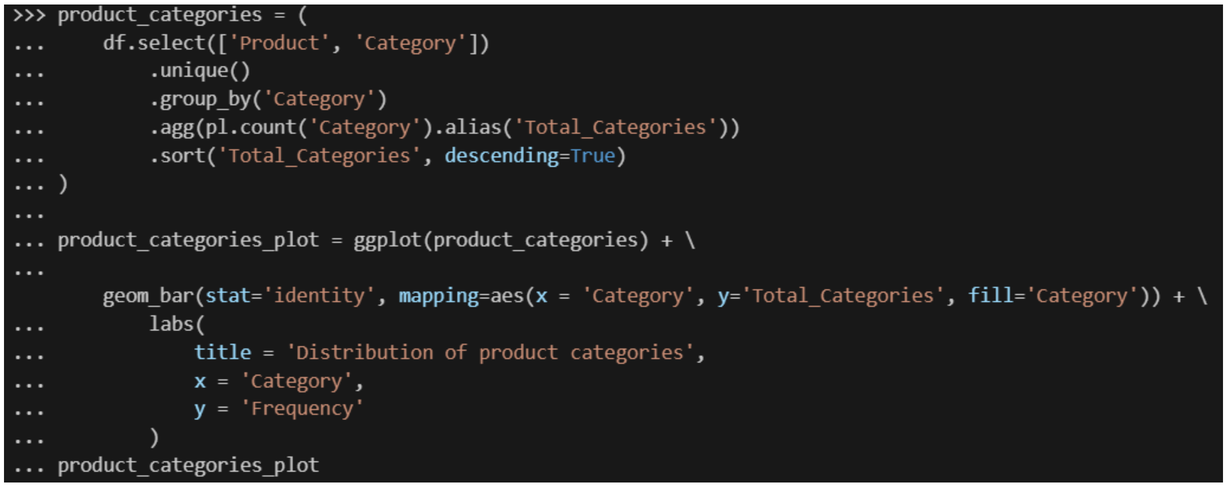


```
>>> df.describe().filter(
...     pl.col('statistic').is_in(['null_count'])
... )
...
shape: (1, 9)
```

statistic	Order_ID	Customer_ID	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
str	f64	str	str	str	str	f64	f64	f64
"null_count"	0.0	"1"	"0"	"0"	"0"	1.0	1.0	1.0

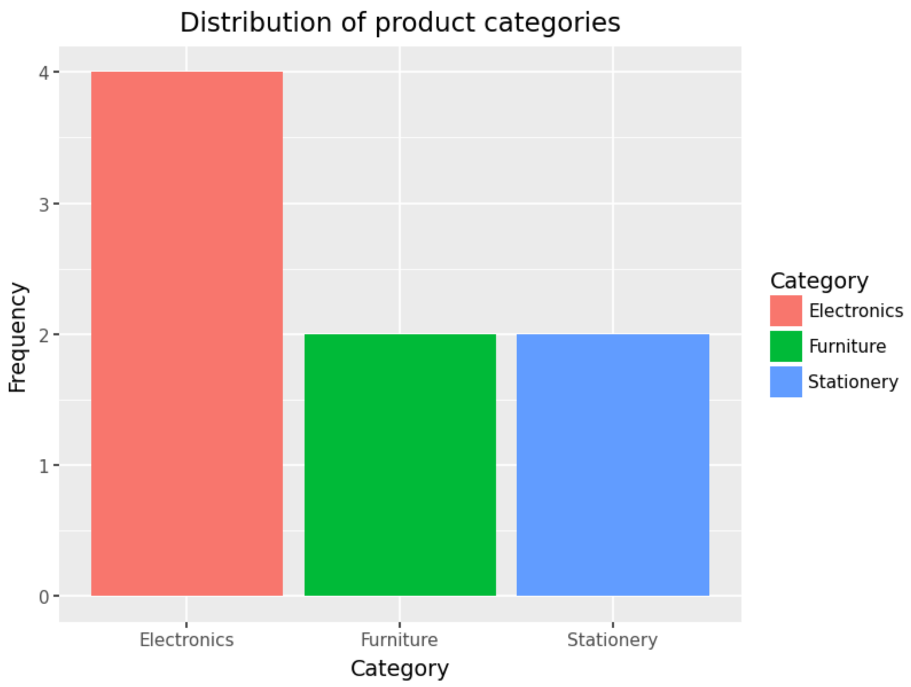
We can see that there are null values in the Customer_ID, Quantity, Unit_Price, and Total_Price table, thus, we will need to drop null values or use imputation to clean our dataset.

5. Create visualizations using plotnine:
   - Distribution of product categories



```
>>> product_categories = (
...     df.select(['Product', 'Category'])
...         .unique()
...         .group_by('Category')
...         .agg(pl.count('Category').alias('Total_Categories'))
...         .sort('Total_Categories', descending=True)
... )
...
... product_categories_plot = ggplot(product_categories) + \
...
    geom_bar(stat='identity', mapping=aes(x = 'Category', y='Total_Categories', fill='Category')) + \
...         labs(
...             title = 'Distribution of product categories',
...             x = 'Category',
...             y = 'Frequency'
...         )
... product_categories_plot
```

Getting the aggregate number of products per category, then plotting them in a bar plot

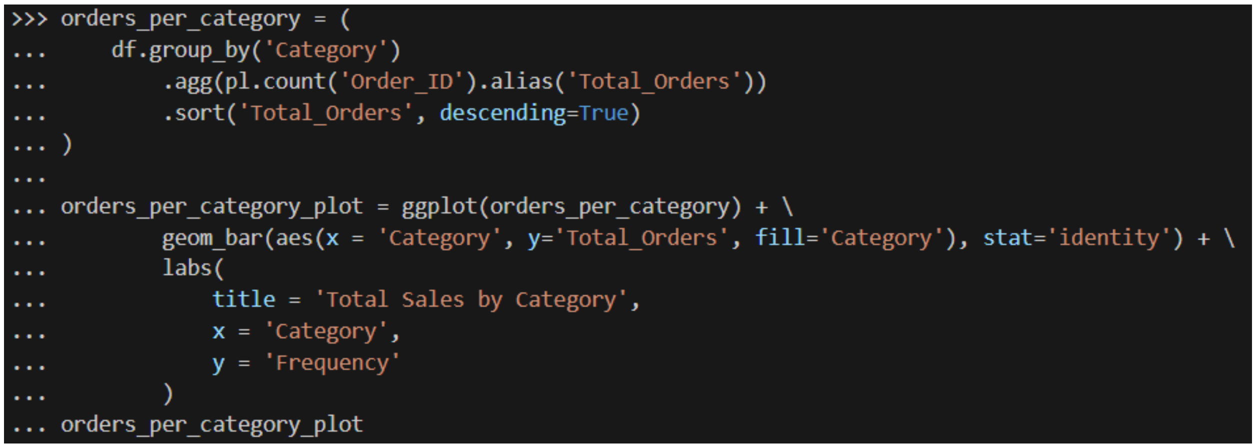


Distribution of product categories

We can see from the bar chart that out of all of the orders in our dataset there are twice as much Electronic products than Furniture and Stationary products. Though this is possibly biased as this is generated from the list of orders rather than a literal product list or inventory.

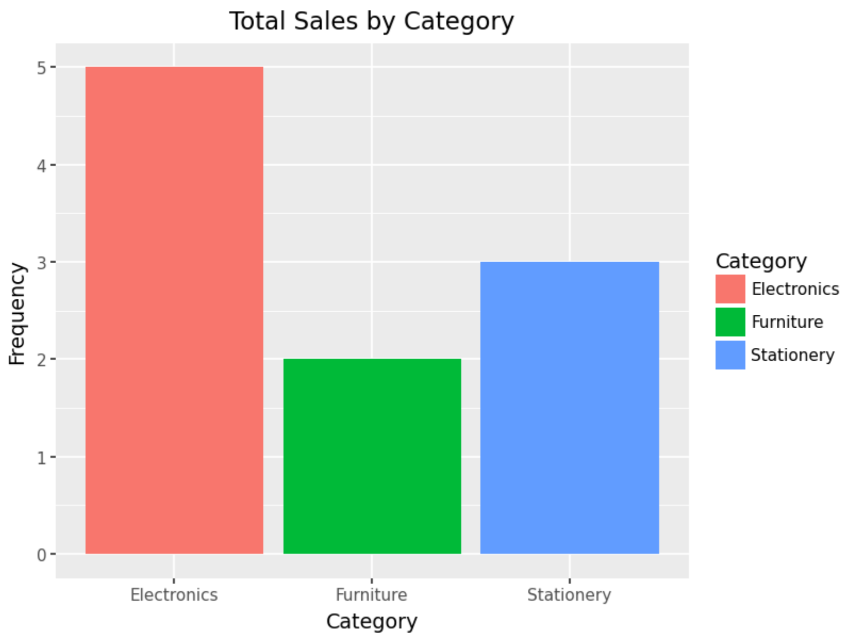
Aside from that, this can be an insight to how electronics may be a more salable product category than Furniture and Stationary products.

- Total sales by category



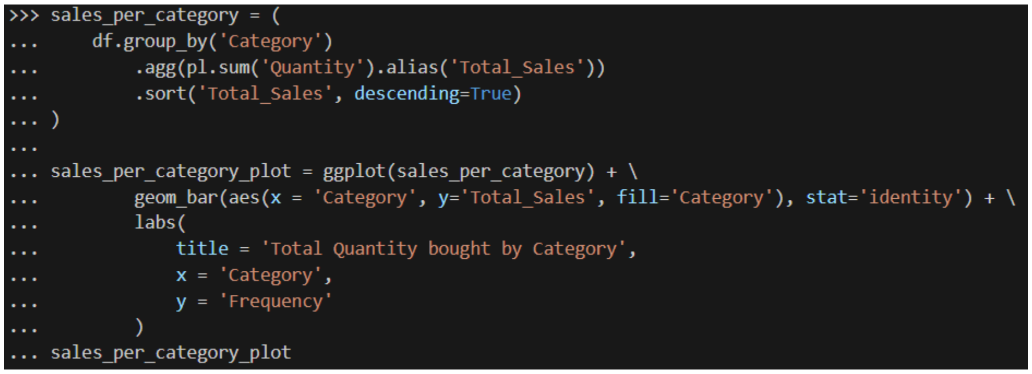
```
>>> orders_per_category = (
...     df.group_by('Category')
...         .agg(pl.count('Order_ID').alias('Total_Orders'))
...         .sort('Total_Orders', descending=True)
... )
...
... orders_per_category_plot = ggplot(orders_per_category) + \
...         geom_bar(aes(x = 'Category', y='Total_Orders', fill='Category'), stat='identity') + \
...         labs(
...             title = 'Total Sales by Category',
...             x = 'Category',
...             y = 'Frequency'
...         )
... orders_per_category_plot
```

Getting the number of orders per category then plotting it in a bar plot



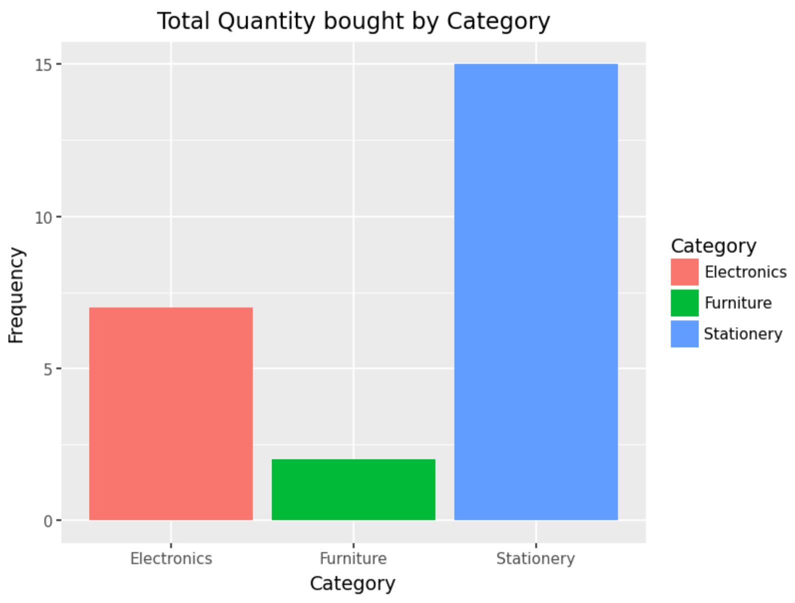
Total number of sales by category (Electronics has the most amount)

If we compare this to our previous plot, we can derive that Electronics and Stationary products are salable, each having at least one product bought twice (i.e., this is because there are 5 and 3 items bought for Electronics and Stationary categories even though there are 4 and 2 products under them), whereas both sales under the Furniture category are different products. However, this cannot be the true number of sales as this is counted based on the number of orders, and doesn't account for the number of items bought per order.



```
>>> sales_per_category = (
...     df.group_by('Category')
...         .agg(pl.sum('Quantity').alias('Total_Sales'))
...         .sort('Total_Sales', descending=True)
... )
...
... sales_per_category_plot = ggplot(sales_per_category) + \
...         geom_bar(aes(x = 'Category', y='Total_Sales', fill='Category'), stat='identity') + \
...         labs(
...             title = 'Total Quantity bought by Category',
...             x = 'Category',
...             y = 'Frequency'
...         )
... sales_per_category_plot
```

Code to aggregate the quantity of items bought per category, and plot them.

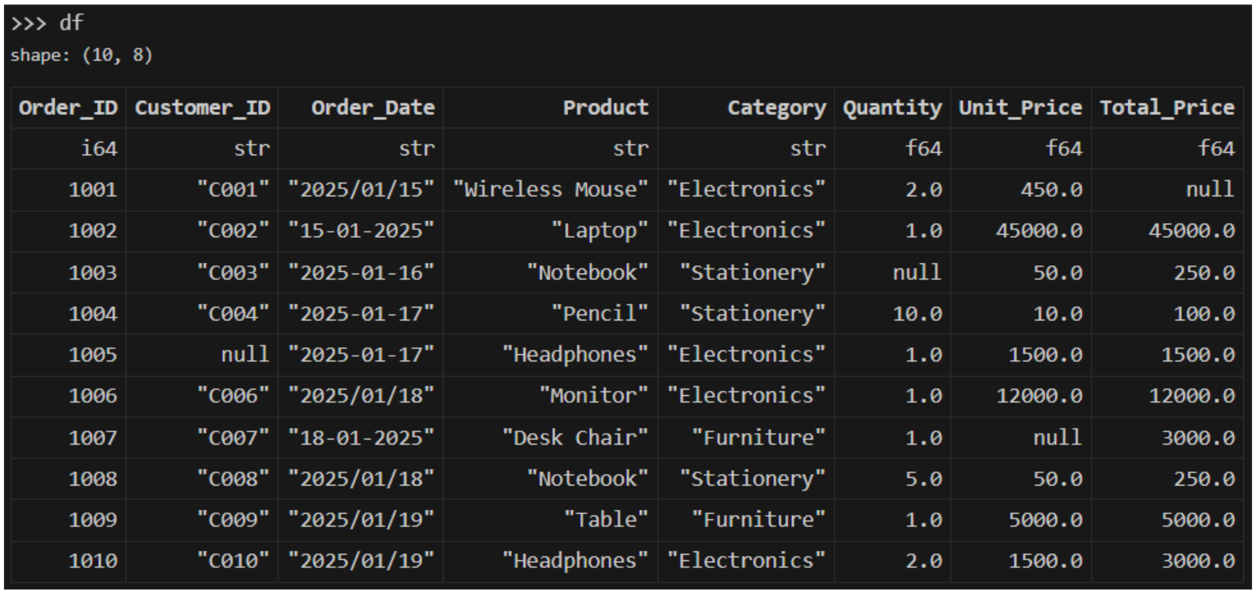


Bar plot of total quantity of items bought

Now, we finally get an better interpretation to which products are more salable, though it is understandable that furniture will have the least amount of sales as compared to stationary products (e.g., Notebooks and Pencils) and electronic products (e.g., Laptop, Monitor, Wireless Mouse, and Headphones).

Through this, we can focus our advertising more on Furniture and Electronic products to increase sales, especially due to the high price of Furniture and Electronic products (i.e., Desk Chair costing 5,000 and Laptop costing 45,000). Similarly, we now know that Stationary products should be checked regularly as they may be out of stock more often than other product categories.

Part 2: Cleaning and Preparing Data

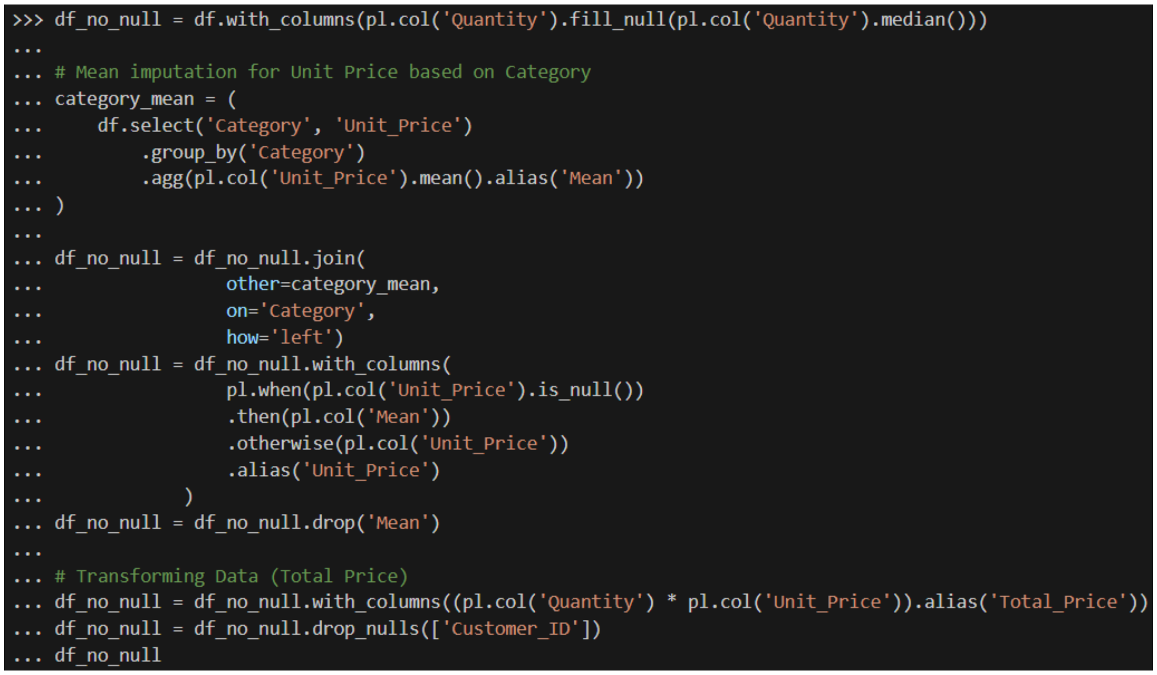


```
>>> df
shape: (10, 8)
```

Order_ID	Customer_ID	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
i64	str	str	str	str	f64	f64	f64
1001	"C001"	"2025/01/15"	"Wireless Mouse"	"Electronics"	2.0	450.0	null
1002	"C002"	"15-01-2025"	"Laptop"	"Electronics"	1.0	45000.0	45000.0
1003	"C003"	"2025-01-16"	"Notebook"	"Stationery"	null	50.0	250.0
1004	"C004"	"2025-01-17"	"Pencil"	"Stationery"	10.0	10.0	100.0
1005	null	"2025-01-17"	"Headphones"	"Electronics"	1.0	1500.0	1500.0
1006	"C006"	"2025/01/18"	"Monitor"	"Electronics"	1.0	12000.0	12000.0
1007	"C007"	"18-01-2025"	"Desk Chair"	"Furniture"	1.0	null	3000.0
1008	"C008"	"2025/01/18"	"Notebook"	"Stationery"	5.0	50.0	250.0
1009	"C009"	"2025/01/19"	"Table"	"Furniture"	1.0	5000.0	5000.0
1010	"C010"	"2025/01/19"	"Headphones"	"Electronics"	2.0	1500.0	3000.0

DataFrame before removing null values

1. Handle missing values:
    - Fill missing Quantity with the median quantity.
    - Fill missing Unit_Price with mean price per category.
    - Compute missing Total_Price as Quantity x Unit_Price.
    - Drop rows with missing Customer_ID.

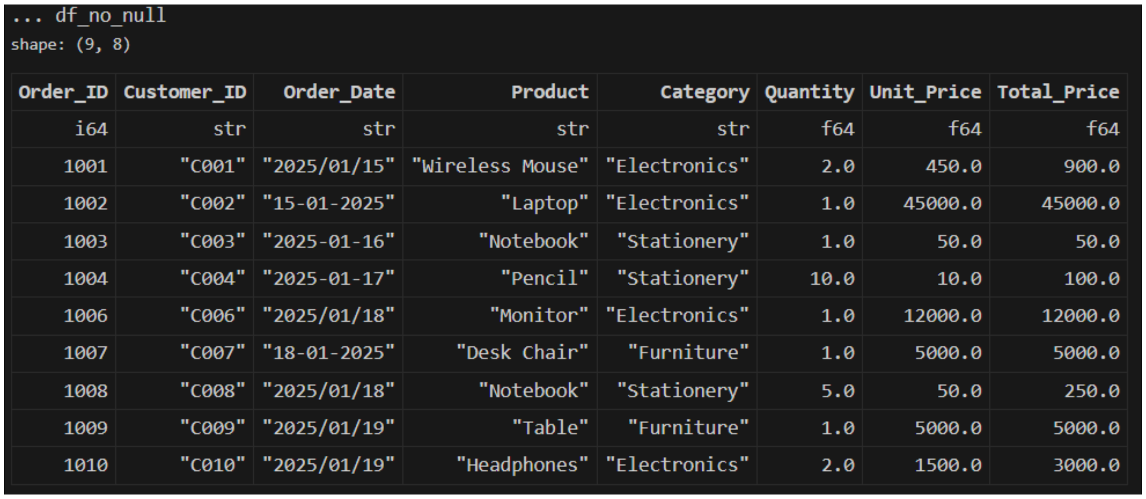


```
>>> df_no_null = df.with_columns(pl.col('Quantity').fill_null(pl.col('Quantity').median()))
...
... # Mean imputation for Unit Price based on Category
... category_mean = (
...     df.select('Category', 'Unit_Price')
...         .group_by('Category')
...         .agg(pl.col('Unit_Price').mean().alias('Mean'))
... )
...
... df_no_null = df_no_null.join(
...                 other=category_mean,
...                 on='Category',
...                 how='left')
... df_no_null = df_no_null.with_columns(
...                 pl.when(pl.col('Unit_Price').is_null())
...                 .then(pl.col('Mean'))
...                 .otherwise(pl.col('Unit_Price'))
...                 .alias('Unit_Price')
...             )
... df_no_null = df_no_null.drop('Mean')
...
... # Transforming Data (Total Price)
... df_no_null = df_no_null.with_columns((pl.col('Quantity') * pl.col('Unit_Price')).alias('Total_Price'))
... df_no_null = df_no_null.drop_nulls(['Customer_ID'])
... df_no_null
```

Code to handle missing values and missing columns

The block of code is made to remove null values through the following steps:

1. Use median imputation for our Quantity column
2. Aggregate the DataFrame grouped by Category and get the mean Unit_Price, have the column be named “Mean”
3. Join the aggregated DataFrame to our original DataFrame, using a one-to-many left join on the “Categories” column
4. Use an if-else statement to use mean impute on null values in Unit_Price based on the its category.
5. Drop the “Mean” column
6. Replace the null value for Total_Price using feature engineering
7. Drop rows with null values on the “Customer_ID” column

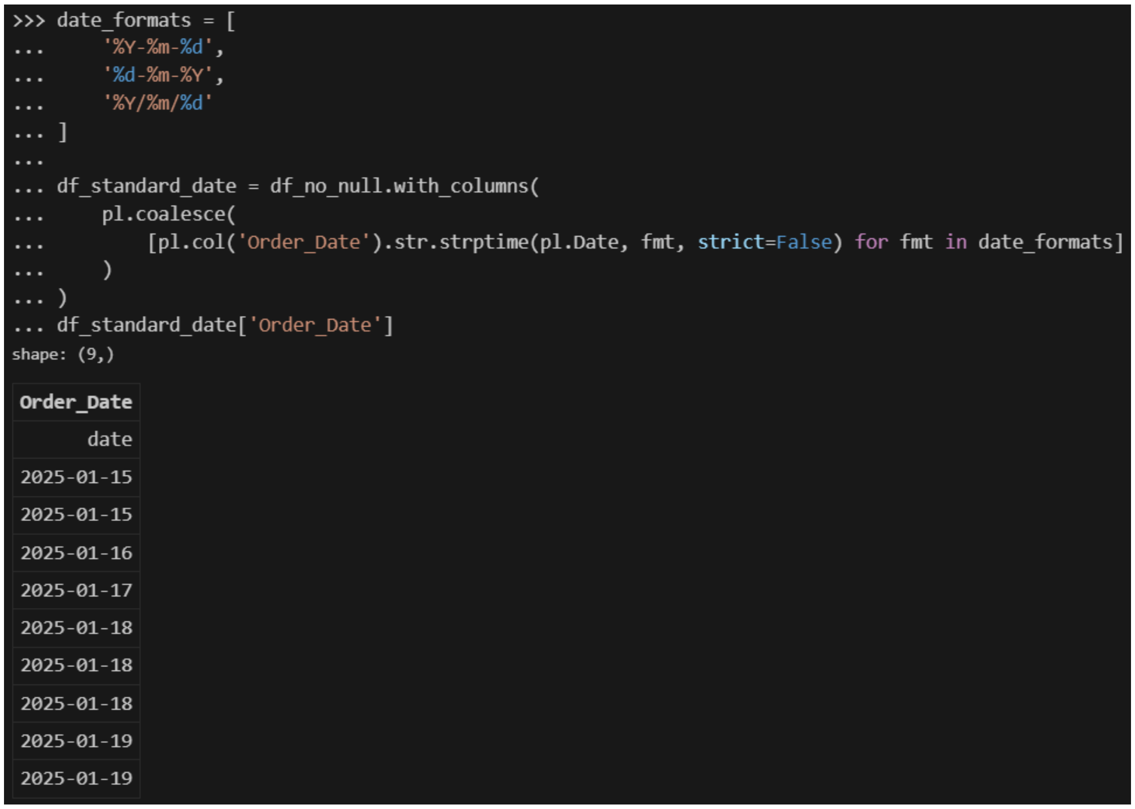


```
... df_no_null
shape: (9, 8)
```

Order_ID	Customer_ID	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
i64	str	str	str	str	f64	f64	f64
1001	"C001"	"2025/01/15"	"Wireless Mouse"	"Electronics"	2.0	450.0	900.0
1002	"C002"	"15-01-2025"	"Laptop"	"Electronics"	1.0	45000.0	45000.0
1003	"C003"	"2025-01-16"	"Notebook"	"Stationery"	1.0	50.0	50.0
1004	"C004"	"2025-01-17"	"Pencil"	"Stationery"	10.0	10.0	100.0
1006	"C006"	"2025/01/18"	"Monitor"	"Electronics"	1.0	12000.0	12000.0
1007	"C007"	"18-01-2025"	"Desk Chair"	"Furniture"	1.0	5000.0	5000.0
1008	"C008"	"2025/01/18"	"Notebook"	"Stationery"	5.0	50.0	250.0
1009	"C009"	"2025/01/19"	"Table"	"Furniture"	1.0	5000.0	5000.0
1010	"C010"	"2025/01/19"	"Headphones"	"Electronics"	2.0	1500.0	3000.0

Resulting DataFrame after running the code

2. Standardize the Order_date format to YYYY-MM-DD.



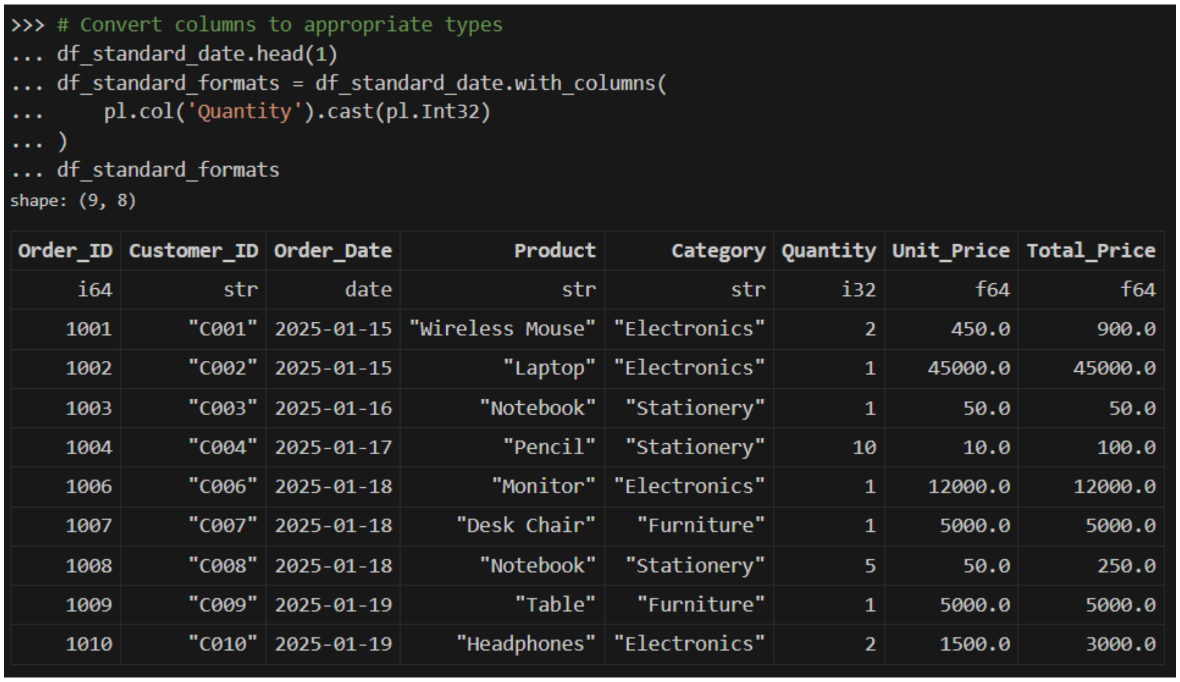
```
>>> date_formats = [
...     '%Y-%m-%d',
...     '%d-%m-%Y',
...     '%Y/%m/%d'
... ]
...
... df_standard_date = df_no_null.with_columns(
...     pl.coalesce(
...         [pl.col('Order_Date').str.strptime(pl.Date, fmt, strict=False) for fmt in date_formats]
...     )
... )
... df_standard_date['Order_Date']
shape: (9,)
```

Order_Date
date
2025-01-15
2025-01-15
2025-01-16
2025-01-17
2025-01-18
2025-01-18
2025-01-18
2025-01-19
2025-01-19

Using coalesce() to standardize dates with multiple formats

I used a list comprehension to apply different formats to our Order_Date column, and wrap it all with the pl.coalesce() method to combine the different non-null results. This results in a code that can standardize the Order_Date column regardless of the different date formats in our original csv.

3. Convert columns to proper data types.



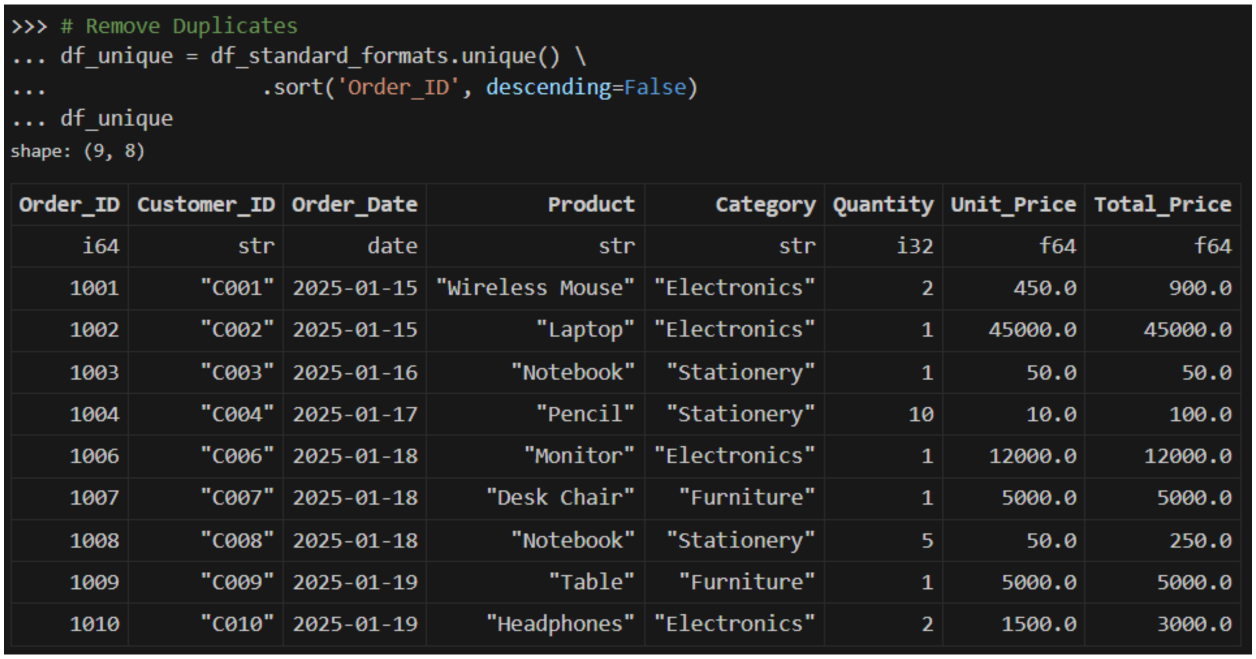
```
>>> # Convert columns to appropriate types
... df_standard_date.head(1)
... df_standard_formats = df_standard_date.with_columns(
...     pl.col('Quantity').cast(pl.Int32)
... )
... df_standard_formats
shape: (9, 8)
```

Order_ID	Customer_ID	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
i64	str	date	str	str	i32	f64	f64
1001	"C001"	2025-01-15	"Wireless Mouse"	"Electronics"	2	450.0	900.0
1002	"C002"	2025-01-15	"Laptop"	"Electronics"	1	45000.0	45000.0
1003	"C003"	2025-01-16	"Notebook"	"Stationery"	1	50.0	50.0
1004	"C004"	2025-01-17	"Pencil"	"Stationery"	10	10.0	100.0
1006	"C006"	2025-01-18	"Monitor"	"Electronics"	1	12000.0	12000.0
1007	"C007"	2025-01-18	"Desk Chair"	"Furniture"	1	5000.0	5000.0
1008	"C008"	2025-01-18	"Notebook"	"Stationery"	5	50.0	250.0
1009	"C009"	2025-01-19	"Table"	"Furniture"	1	5000.0	5000.0
1010	"C010"	2025-01-19	"Headphones"	"Electronics"	2	1500.0	3000.0

Converting quantity to an int.

We will then convert the Quantity column to an integer, as there are no floats for quantity.

4. Remove duplicates if any.

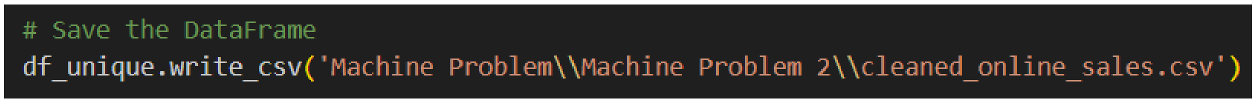


```
>>> # Remove Duplicates
... df_unique = df_standard_formats.unique() \
...                 .sort('Order_ID', descending=False)
... df_unique
shape: (9, 8)
Order_ID  Customer_ID  Order_Date   Product           Category       Quantity  Unit_Price  Total_Price
i64       str          date         str               str            i32       f64         f64
1001      "C001"       2025-01-15   "Wireless Mouse"  "Electronics"  2         450.0       900.0
1002      "C002"       2025-01-15   "Laptop"          "Electronics"  1         45000.0     45000.0
1003      "C003"       2025-01-16   "Notebook"        "Stationery"   1         50.0        50.0
1004      "C004"       2025-01-17   "Pencil"          "Stationery"   10        10.0        100.0
1006      "C006"       2025-01-18   "Monitor"         "Electronics"  1         12000.0     12000.0
1007      "C007"       2025-01-18   "Desk Chair"      "Furniture"    1         5000.0      5000.0
1008      "C008"       2025-01-18   "Notebook"        "Stationery"   5         50.0        250.0
1009      "C009"       2025-01-19   "Table"           "Furniture"    1         5000.0      5000.0
1010      "C010"       2025-01-19   "Headphones"      "Electronics"  2         1500.0      3000.0
```

DataFrame after handling null values, standardizing dates, formatting data types, and removing duplicates.

We then remove duplicate entries in our DataFrame.

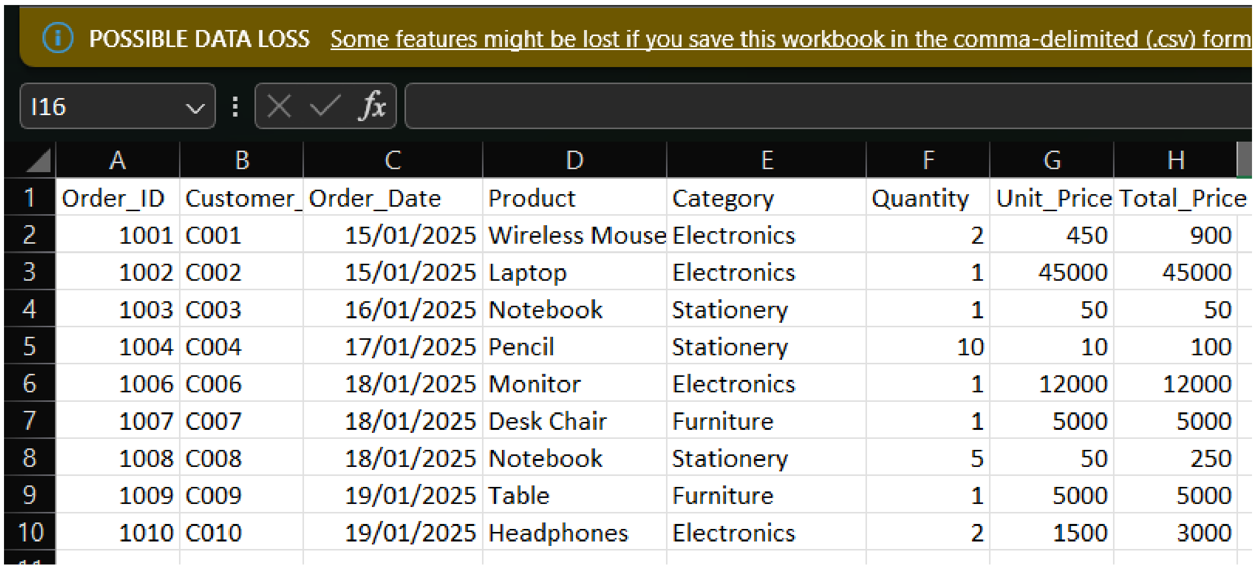
5. Save the cleaned dataset as cleaned_online_sales.csv.



```
# Save the DataFrame
df_unique.write_csv('Machine Problem\\Machine Problem 2\\cleaned_online_sales.csv')
```

Saving the Polars DataFrame into a csv.

Name: Maclang, Waken Cean C. **CSDS 312:** Machine Problem 2
Course: BSCS 3A **Date:** October 23, 2025



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) form

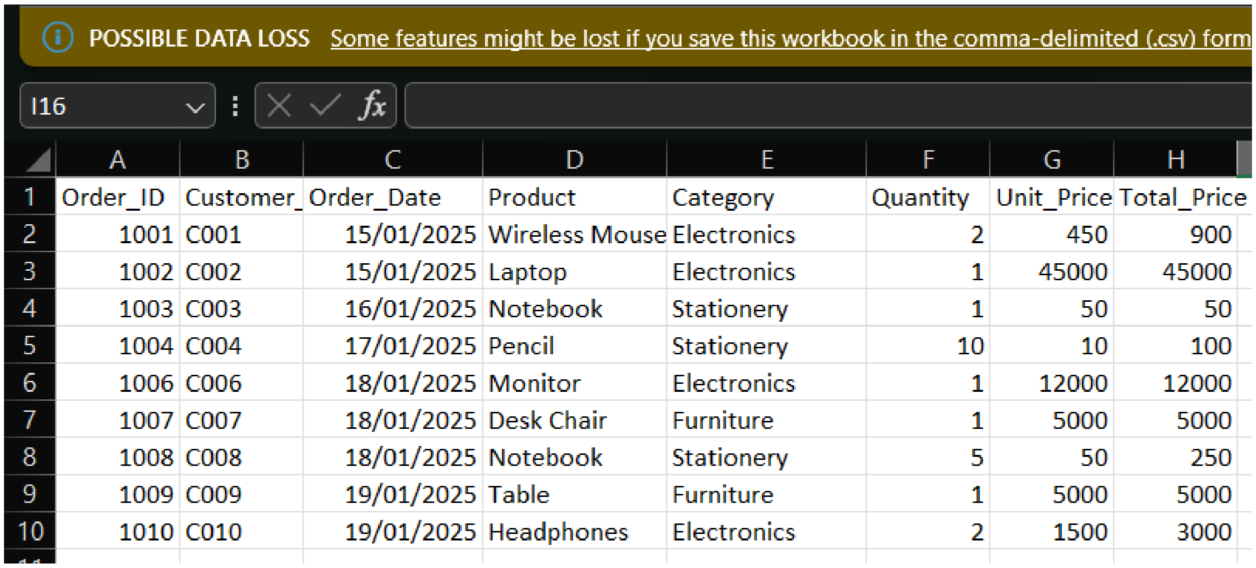
I16

	A	B	C	D	E	F	G	H
1	Order_ID	Customer_	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
2	1001	C001	15/01/2025	Wireless Mouse	Electronics	2	450	900
3	1002	C002	15/01/2025	Laptop	Electronics	1	45000	45000
4	1003	C003	16/01/2025	Notebook	Stationery	1	50	50
5	1004	C004	17/01/2025	Pencil	Stationery	10	10	100
6	1006	C006	18/01/2025	Monitor	Electronics	1	12000	12000
7	1007	C007	18/01/2025	Desk Chair	Furniture	1	5000	5000
8	1008	C008	18/01/2025	Notebook	Stationery	5	50	250
9	1009	C009	19/01/2025	Table	Furniture	1	5000	5000
10	1010	C010	19/01/2025	Headphones	Electronics	2	1500	3000

Viewing the saved csv in Microsoft Excel

Expected Output

- Bar chart of total sales per category.
- Distribution plot of product categories.
- Explanation for each cleaning step.
- Interpretation of the charts.
- Cleaned dataset file cleaned_online_sales.csv.



POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) form

I16

	A	B	C	D	E	F	G	H
1	Order_ID	Customer_	Order_Date	Product	Category	Quantity	Unit_Price	Total_Price
2	1001	C001	15/01/2025	Wireless Mouse	Electronics	2	450	900
3	1002	C002	15/01/2025	Laptop	Electronics	1	45000	45000
4	1003	C003	16/01/2025	Notebook	Stationery	1	50	50
5	1004	C004	17/01/2025	Pencil	Stationery	10	10	100
6	1006	C006	18/01/2025	Monitor	Electronics	1	12000	12000
7	1007	C007	18/01/2025	Desk Chair	Furniture	1	5000	5000
8	1008	C008	18/01/2025	Notebook	Stationery	5	50	250
9	1009	C009	19/01/2025	Table	Furniture	1	5000	5000
10	1010	C010	19/01/2025	Headphones	Electronics	2	1500	3000

Note: You can find the dataset file in the github link (cleaned_online_sales.csv)