

DATA PROFILING REPORT : SAMPLE SUPERSTORE DATASET

Dataset: Sample_Superstore.csv

1. Dataset Structure

- The Superstore dataset contains 9,994 rows and 21 columns.

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** Task1.py, Task 1 Correlation, Task 1 Outlier.py, Task 1 duplicate.py, Task 1 missing values.py, Task 1 datatype.py.
- Code Cell:** SuperStore > Task1.py > ...
1 import pandas as pd
2 import numpy as np
3
4 df = pd.read_csv(r"D:\Task\Sample - Superstore.csv", encoding='latin1')
5 print("1) Dataset shape (rows, columns):", df.shape)
- Toolbar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS.
- Terminal Output:** PS D:\Task>
& C:\Users\KUMARAN\AppData\Local\Programs\Python\Python313\python.exe d:/Task/SuperStore/Task1.py
1) Dataset shape (rows, columns): (9994, 21)

2. Column Names & Data Types

- Row ID (int)
 - Order ID (object)
 - Order Date (object)
 - Ship Date (object)
 - Ship Mode (object)
 - Customer ID (object)
 - Customer Name (object)
 - Segment (object)
 - Country (object)
 - City (object)
 - State (object)
 - Postal Code (int)
 - Region (object)
 - Product ID (object)
 - Category (object)
 - Sub-Category (object)
 - Product Name (object)
 - Sales (float)
 - Quantity (int)
 - Discount (float)
 - Profit (float)

The screenshot shows a code editor interface with several tabs at the top: Task1.py, Task 1 Corelation, Task 1 Outlier.py, Task 1 duplicate.py, Task 1 missing values.py, and Task 1 datatype.py (which is the active tab). Below the tabs, the code for Task 1 datatype.py is displayed:

```
1 import pandas as pd
2 import numpy as np
3
4 df = pd.read_csv(r"D:\Task\Sample - Superstore.csv", encoding='latin1')
5 print("\n2) Columns and Data Types:")
6 print(df.dtypes)
7
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the command run and the output:

```
PS D:\Task>
& C:\Users\KUMARAN\AppData\Local\Programs\Python\Python313\python.exe "d:/Task/SuperStore/Task 1 datatype.py"
```

2) Columns and Data Types:

Column	Data Type
Row ID	int64
Order ID	object
Order Date	object
Ship Date	object
Ship Mode	object
Customer ID	object
Customer Name	object
Segment	object
Country	object
City	object
State	object
Postal Code	int64
Region	object
Product ID	object
Category	object
Sub-Category	object
Product Name	object
Sales	float64
Quantity	int64
Discount	float64
Profit	float64
dtype: object	

3. Missing Value Analysis

- The dataset contains 0% missing values across all columns.
- No missing values found in key analytical columns such as Sales, Profit, Quantity, Discount, Category, Segment, Region.

The screenshot shows a Jupyter Notebook interface with several tabs at the top: Task1.py, Task 1 Corelation, Task 1 Outlier.py, Task 1 duplicate.py, Task 1 missing values.py (which is active), and Task 1 datatype.py. Below the tabs is a code cell containing Python code to calculate missing value percentages:

```
SuperStore > Task 1 missing values.py > ...
1 import pandas as pd
2 import numpy as np
3
4 df = pd.read_csv(r"D:\Task\Sample - Superstore.csv", encoding='latin1')
5
6 print("\n3) Missing Value Percentage per Column:")
7 print((df.isna().mean() * 100).round(2))
```

Below the code cell are navigation buttons: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal output shows the command run and the resulting output:

```
PS D:\Task>
& C:/Users/KUMARAN/AppData/Local/Programs/Python/Python313/python.exe "d:/Task/SuperStore/Task 1 missing values.py"

3) Missing Value Percentage per Column:
Row ID      0.0
Order ID    0.0
Order Date  0.0
Ship Date   0.0
Ship Mode   0.0
Customer ID 0.0
Customer Name 0.0
Segment     0.0
Country    0.0
City       0.0
State      0.0
Postal Code 0.0
Region     0.0
Product ID 0.0
Category   0.0
Sub-Category 0.0
Product Name 0.0
Sales      0.0
Quantity   0.0
Discount   0.0
Profit     0.0
dtype: float64
```

4. Duplicate Records

- Duplicate rows detected: 0
- Final dataset contains only unique order-line entries.

The screenshot shows a Jupyter Notebook interface with several tabs at the top: Task1.py, Task 1 Corelation, Task 1 Outlier.py, Task 1 duplicate.py (which is active), Task 1 missing values.py, and Task 1 datatype.py. Below the tabs, there is a code cell containing Python code to read a CSV file and count duplicate rows:

```
SuperStore > Task 1 duplicate.py > ...
1 import pandas as pd
2 import numpy as np
3
4 df = pd.read_csv(r"D:\Task\Sample - Superstore.csv", encoding='latin1')
5
6 print("\n4) Duplicate Rows Count:", df.duplicated().sum())
```

Below the code cell, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the command used to run the script:

```
PS D:\Task>
& C:\Users\KUMARAN\AppData\Local\Programs\Python\Python313\python.exe "d:/Task/SuperStore/Task 1 missing values.py"
```

Under the terminal output, the results of the missing value analysis are displayed:

3) Missing Value Percentage per Column:

Column	Missing Value Percentage
Row ID	0.0
Order ID	0.0
Order Date	0.0
Ship Date	0.0
Ship Mode	0.0
Customer ID	0.0
Customer Name	0.0
Segment	0.0
Country	0.0
City	0.0
State	0.0
Postal Code	0.0
Region	0.0
Product ID	0.0
Category	0.0
Sub-Category	0.0
Product Name	0.0
Sales	0.0
Quantity	0.0
Discount	0.0
Profit	0.0

dtype: float64

5. Outlier Analysis

- Sales Outliers: 1,167
- Profit Outliers: 1,881

The screenshot shows a Jupyter Notebook interface with several tabs at the top: Task1.py, Task 1 Corelation (highlighted), Task 1 Outlier.py, Task 1 duplicate.py, Task 1 missing values.py, and Task 1 datatype.py. Below the tabs, the code for 'Task 1 Outlier.py' is displayed:

```
SuperStore > Task 1 Outlier.py > ...
1 import pandas as pd
2 import numpy as np
3
4 df = pd.read_csv(r"D:\Task\Sample - Superstore.csv", encoding='latin1')
5 import pandas as pd
6
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The terminal output shows the execution of the script and the results of outlier detection:

```
PS D:\Task>
& C:\Users\KUMARAN\AppData\Local\Programs\Python\Python313\python.exe "d:/Task/SuperStore/Task 1 Outlier.py"
Sales Outliers Count: 1167
Sales Lower Bound: -271.71000000000004
Sales Upper Bound: 498.93

Profit Outliers Count: 1881
Profit Lower Bound: -39.724125
Profit Upper Bound: 70.816875
```

6. Relationship Analysis (Correlation)

Discount vs Profit (-0.21 correlation)

- Small negative correlation confirms that increasing discount generally reduces profitability.

Sales vs Quantity (+0.20 correlation)

- Positive but weak relationship: increasing quantity does not always mean higher sales value.

The screenshot shows a Jupyter Notebook interface with several tabs at the top: Task1.py, Task 1 Corelation (highlighted), Task 1 Outlier.py, Task 1 duplicate.py, Task 1 missing values.py, and Task 1 datatype.py. Below the tabs, the code for 'Task 1 Corelation.py' is displayed:

```
SuperStore > Task 1 Corelation > ...
6 corr_discount_profit = df['Discount'].corr(df['Profit'])
7 corr_sales_quantity = df['Sales'].corr(df['Quantity'])
8
9 print("\n Correlation Checks:")
10 print(f" Discount vs Profit: {corr_discount_profit:.4f}")
11 print(f" Sales vs Quantity: {corr_sales_quantity:.4f}")
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The terminal output shows the execution of the script and the results of correlation checks:

```
PS D:\Task>
& C:\Users\KUMARAN\AppData\Local\Programs\Python\Python313\python.exe "d:/Task/SuperStore/Task 1 Corelation"

Correlation Checks:
Discount vs Profit: -0.2195
Sales vs Quantity: 0.2008
```

7. Summary of Insights

- Dataset is clean with no missing values or duplicates.
- Sales & Profit contain heavy outliers.
- Discounting reduces profitability.
- Quantity has weak influence on Sales.
- Profit volatility is high.