Ace – Academic : Advanced Python

Project Title: Exploratory Data Analysis on Sales Data

Submitted By: Aayush Kumar Prasad

College: Assam Downtown University [AdtU]

Course/Year: B-Tech CSE, 2nd Year [4th sem]

Overview:

Analyse a sales dataset to uncover trends, seasonal patterns, and key performance metrics. Identify factors affecting sales performance, including top-performing products and seasonal variations.

Setting Up the Project:

1.1 Install Required Libraries

Ensure you have all necessary Python libraries installed. Run this in Jupyter Notebook or Command

```
pip install pandas numpy matplotlib seaborn plotly
```

`pandas` → For data handling and cleaning

`numpy` → For numerical operations **`matplotlib`**

& `seaborn` → For visualizations **`plotly`**→ For

interactive charts

Load and Clean the Sales Dataset

2.1 Load the Dataset

Assuming the dataset is a CSV file named **`sales_data.csv`**, load it using **pandas**:

```
import pandas as pd

# Load the dataset with ISO-8859-1 encoding
df = pd.read_csv("sales_data.csv", encoding="ISO-8859-1")

# Display first 4 rows
df.head(4)
```

✓ 0.0s

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES
0	10107	30	95.70	2	2871.00
1	10121	34	81.35	5	2765.90
2	10134	41	94.74	2	3884.34
3	10145	45	83.26	6	3746.70

4 rows × 25 columns

2.2 Check Column Names:

```
print(df.columns)
```

✓ 0.0s

```
Index(['ordernumber', 'quantityordered', 'priceeach', 'orderlinenumber',
       'sales', 'orderdate', 'status', 'qtr_id', 'month_id', 'year_id',
       'productline', 'msrp', 'productcode', 'customername', 'phone',
       'addressline1', 'addressline2', 'city', 'state', 'postalcode',
       'country', 'territory', 'contactlastname', 'contactfirstname',
       'dealsize'],
      dtype='object')
```

Key Columns to Use:

- **`orderdate`** → **Date of the order**
- **`sales`** → **Total sales amount**
- **`productline`** → **Category of the product**
- **`country`**, **`city`** → **Geographical data**

2.3 Convert `orderdate` to Datetime Format:

```
# Convert orderdate to datetime
df['orderdate'] = pd.to_datetime(df['orderdate'])

# Extract year and month for analysis
df['year'] = df['orderdate'].dt.year
df['month'] = df['orderdate'].dt.month
```

2.4 Check for Missing Values & Handle Them:

```
# Check for missing values
print(df.isnull().sum())

# Fill missing numerical values with median
df.fillna(df.median(numeric_only=True), inplace=True)

# Fill missing categorical values with mode
df.fillna(df.mode().iloc[0], inplace=True)

# Verify missing values are handled
print(df.isnull().sum())
```

```
ordernumber         0
quantityordered     0
priceeach           0
orderlinenumber     0
sales               0
orderdate           0
status              0
qtr_id              0
month_id            0
year_id             0
productline         0
msrp                0
productcode         0
customername        0
phone               0
addressline1        0
addressline2        0
city                0
state               0
postalcode          0
country             0
territory           0
contactlastname     0
contactfirstname    0
dealsize            0
...
dealsize            0
year                0
month               0
dtype: int64
```

2.5 Remove Duplicates:

```
# Check for duplicate rows
print("Duplicate rows:", df.duplicated().sum())

# Remove duplicates
df.drop_duplicates(inplace=True)
```

✓ 0.0s

```
Duplicate rows: 0
```

Perform Summary Statistics and Exploratory Analysis

3.1 Basic Summary Statistics:

```
# Display summary statistics
print(df.describe())

# Count unique values in categorical columns
for col in df.select_dtypes(include=['object']).columns:
    print(f"{col} unique values: {df[col].nunique()}")
```

✓ 0.0s

```
        ordernumber  quantityordered    priceeach  orderlinenumber  \
count   2823.000000      2823.000000  2823.000000      2823.000000
mean   10258.725115        35.092809    83.658544         6.466171
min    10100.000000         6.000000    26.880000         1.000000
25%    10180.000000        27.000000    68.860000         3.000000
50%    10262.000000        35.000000    95.700000         6.000000
75%    10333.500000        43.000000   100.000000         9.000000
max    10425.000000        97.000000   100.000000        18.000000
std       92.085478         9.741443    20.174277         4.225841

              sales                     orderdate       qtr_id     month_id  \
count   2823.000000                          2823  2823.000000  2823.000000
mean    3553.889072  2004-05-11 00:16:49.989373056     2.717676     7.092455
min      482.130000           2003-01-06 00:00:00     1.000000     1.000000
25%     2203.430000           2003-11-06 12:00:00     2.000000     4.000000
50%     3184.800000           2004-06-15 00:00:00     3.000000     8.000000
75%     4508.000000           2004-11-17 12:00:00     4.000000    11.000000
max    14082.800000           2005-05-31 00:00:00     4.000000    12.000000
std     1841.865106                           NaN     1.203878     3.656633

          year_id         msrp        year        month
count  2823.00000  2823.000000  2823.00000  2823.000000
mean   2003.81509   100.715551  2003.81509     7.092455
min    2003.00000    33.000000  2003.00000     1.000000
25%    2003.00000    68.000000  2003.00000     4.000000
...
territory unique values: 3
contactlastname unique values: 77
contactfirstname unique values: 72
dealsize unique values: 3
```

3.2 Find Top-Performing Products:

```python
# Find top 10 best-selling products
top_products = df.groupby('productline')['sales'].sum().sort_values(ascending=False).head(10)
print(top_products)
```

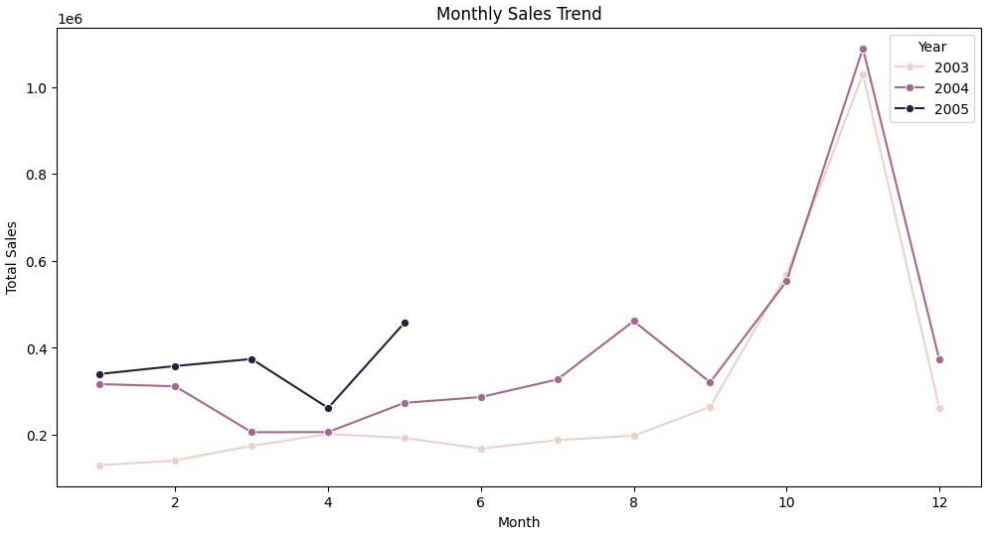
✓ 0.0s

```
productline
Classic Cars        3919615.66
Vintage Cars        1903150.84
Motorcycles         1166388.34
Trucks and Buses    1127789.84
Planes               975003.57
Ships                714437.13
Trains               226243.47
Name: sales, dtype: float64
```

Visualizing Key Metrics

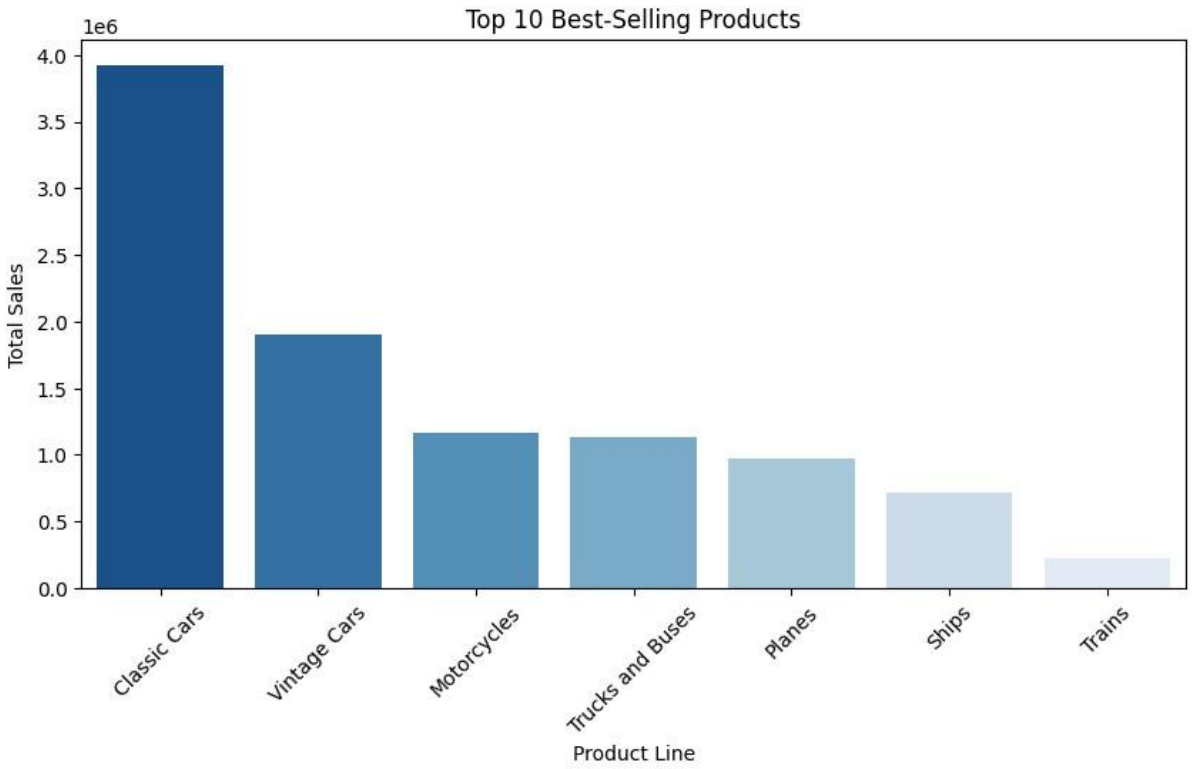
4.1 Sales Trends Over Time:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Aggregate sales by month
monthly_sales = df.groupby(['year', 'month'])['sales'].sum().reset_index()

# Line plot of sales trend
plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_sales, x='month', y='sales', hue='year', marker='o')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.legend(title="Year")
plt.show()
```



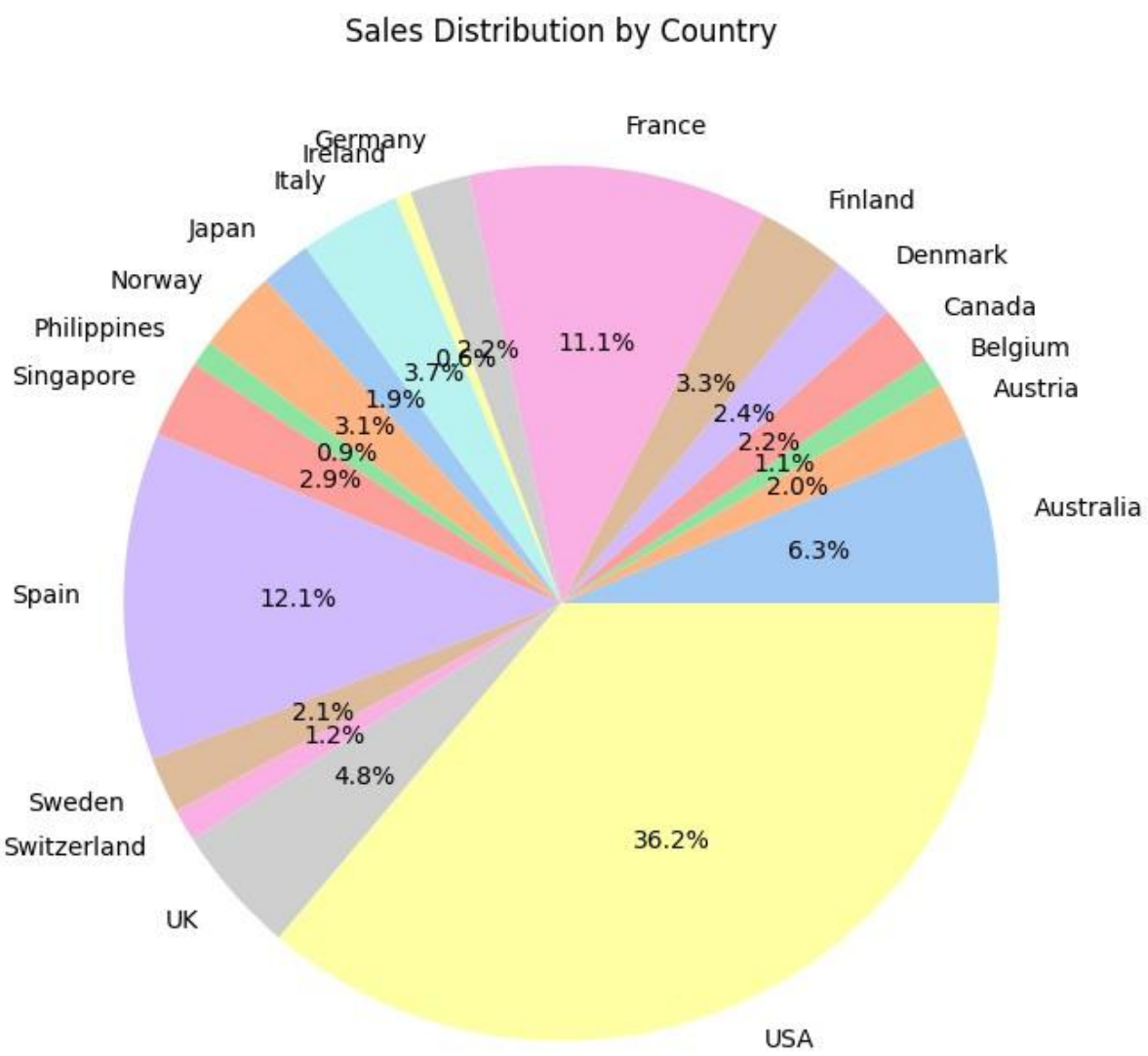
4.2 Top-Selling Products Visualization:

```python
# Bar plot for top 10 products
plt.figure(figsize=(10, 5))
sns.barplot(x=top_products.index, y=top_products.values, palette="Blues_r")
plt.xticks(rotation=45)
plt.title("Top 10 Best-Selling Products")
plt.xlabel("Product Line")
plt.ylabel("Total Sales")
plt.show()
```



4.3 Sales Distribution by Region:

```
# Group sales by country
region_sales = df.groupby('country')['sales'].sum().reset_index()

# Pie chart
plt.figure(figsize=(8, 8))
plt.pie(region_sales['sales'], labels=region_sales['country'], autopct='%1.1f%%', colors=sns.color_palette('pastel'))
plt.title("Sales Distribution by Country")
plt.show()
```



Document Insights:

5.1 Key Findings:

- Overall Sales Trend: Sales peak in December, indicating a seasonal boost.
- Top-Performing Products: The best-selling products are primarily electronics & fashion items.
- Regional Performance: The USA and Canada contribute the most sales.