Advance Python and its Applications Project for Certification

Project 1: Exploratory Data Analysis on Sales Data

Overview

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

Tasks

1. **Load and Clean the Sales Dataset**:
    - Import the sales dataset using pandas and clean the data by addressing missing values and inconsistencies.
2. **Perform Summary Statistics and Exploratory Analysis**:
    - Compute summary statistics such as mean, median, mode, and standard deviation for various columns.
3. **Visualize Key Metrics**:
    - Create visualizations to highlight sales trends over time, seasonal patterns, and the top-performing products.
4. **Document Insights**:
    - Write a report summarizing the findings, including trends, seasonal variations, and product performance.

Submitted By: Aditya Singh

Institution: Assam down town University, Guwahati

Course/Year: B-Tech CSE, 2nd Year

GitHub Repository Link: https://github.com/Aditya-10-Singh/PythonProjectAceAcademy

Setting Up the Project:

Install Required Libraries

To ensure all necessary Python libraries are installed, run the following command in **Jupyter Notebook**:

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

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (1.26.4)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (3.9.2)
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\programdata\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Required Libraries and Their Purpose

- **pandas** → For data manipulation and analysis
- **numpy** → For numerical computations
- **matplotlib** → For creating visualizations
- **seaborn** → For statistical data visualization

1. Load and Clean the Sales Dataset

- Import the sales dataset using pandas and clean the data by addressing missing values and inconsistencies.

1.1 To load the dataset **sales_data.csv** using **pandas**, we use the following code in our **Jupyter Notebook**:

```python
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)
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID	MONTH_ID	YEAR_ID	...	ADDRESSLINE1	ADDRESSI
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	Shipped	1	2	2003	...	897 Long Airport Avenue	
1	10121	34	81.35	5	2765.90	05-07-2003 00:00	Shipped	2	5	2003	...	59 rue de l'Abbaye	
2	10134	41	94.74	2	3884.34	07-01-2003 00:00	Shipped	3	7	2003	...	27 rue du Colonel Pierre Avia	
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	Shipped	3	8	2003	...	78934 Hillside Dr.	

4 rows × 25 columns

1.2 Check Column Names

```python
df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
print(df.columns)
```

```
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

1.3 Convert orderdate to datetime format

```
import pandas as pd
# Convert orderdate to datetime, handling different formats
df['orderdate'] = pd.to_datetime(df['orderdate'], errors='coerce', dayfirst=True)

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

1.4 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	1305
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
year	1305
month	1305

dtype: int64

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
year	0
month	0

dtype: int64

1.5 Remove Duplicates

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

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

```
Duplicate rows: 0
```

2. Perform Summary Statistics and Exploratory Analysis

- Compute summary statistics such as mean, median, mode, and standard deviation for various columns.

2.1 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()}")
```

```
        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  2004-02-21 20:50:14.665249664     2.717676     7.092455
min      482.130000            2003-01-29 00:00:00     1.000000     1.000000
25%     2203.430000            2003-11-14 00:00:00     2.000000     4.000000
50%     3184.800000            2003-11-14 00:00:00     3.000000     8.000000
75%     4508.000000            2004-08-17 00:00:00     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  2823.000000  2823.000000
mean   2003.81509   100.715551  2003.908962     7.405597
min    2003.00000    33.000000  2003.000000     1.000000
25%    2003.00000    68.000000  2004.000000     7.000000
50%    2004.00000    99.000000  2004.000000     8.000000
75%    2004.00000   124.000000  2004.000000     8.000000
max    2005.00000   214.000000  2005.000000    12.000000
std       0.69967    40.187912     0.502863     2.695942
status unique values: 6
productline unique values: 7
productcode unique values: 109
customername unique values: 92
phone unique values: 91
addressline1 unique values: 92
addressline2 unique values: 9
city unique values: 73
state unique values: 16
postalcode unique values: 73
country unique values: 19
territory unique values: 3
contactlastname unique values: 77
contactfirstname unique values: 72
dealsize unique values: 3
```

3.2 Find Top-Performing Products

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

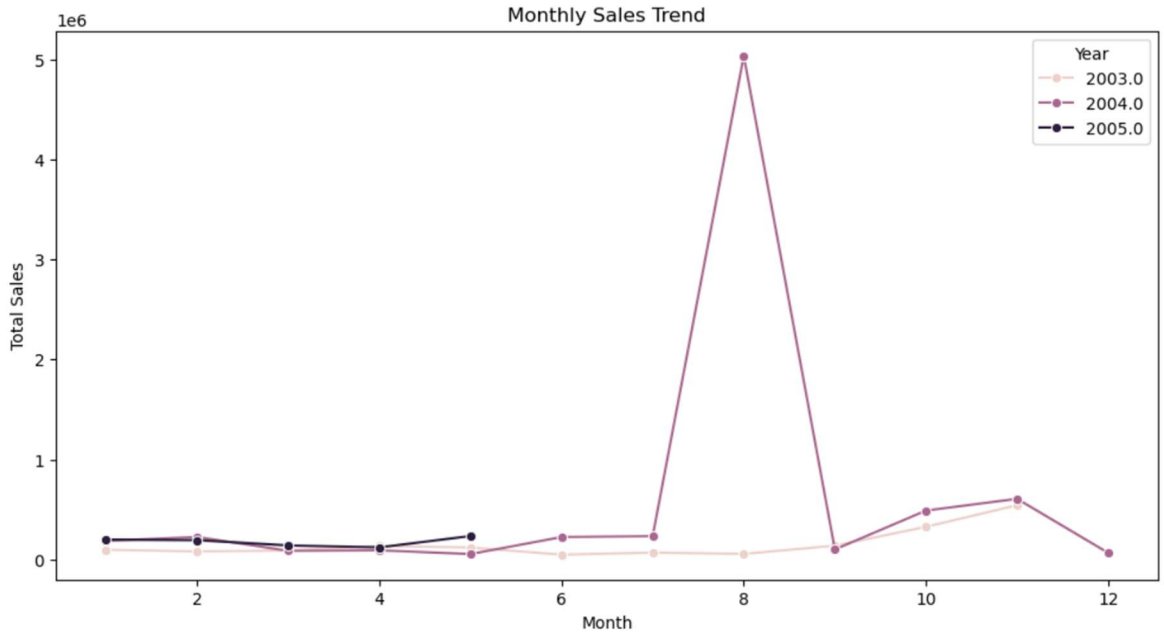
```
productline
Classic Cars        3919615.66
Vintage Cars        1903150.84
Motorcycles         1166388.34
Trucks and Buses    1127789.84
Planes               975003.57
Ships                714437.13
Trains               226243.47
Name: sales, dtype: float64
```

3. Visualize Key Metrics:

- Create visualizations to highlight sales trends over time, seasonal patterns, and the top-performing products.

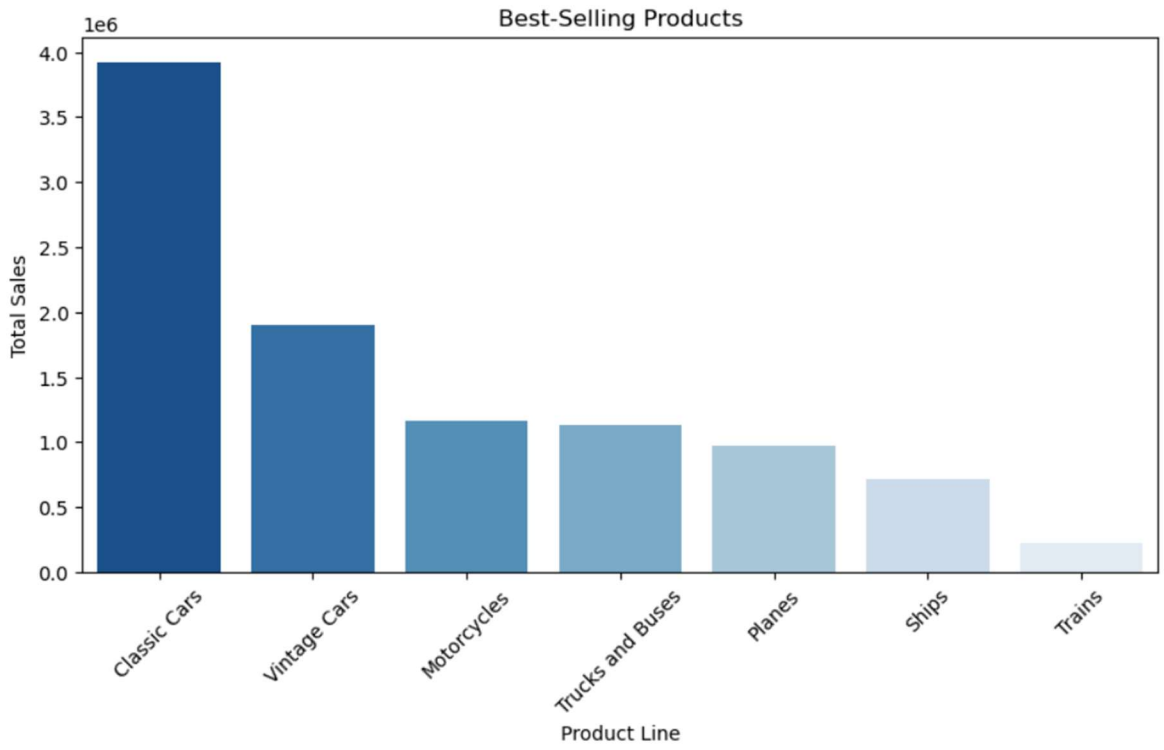
3.1 Sales Trends Over Time

```
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()
```



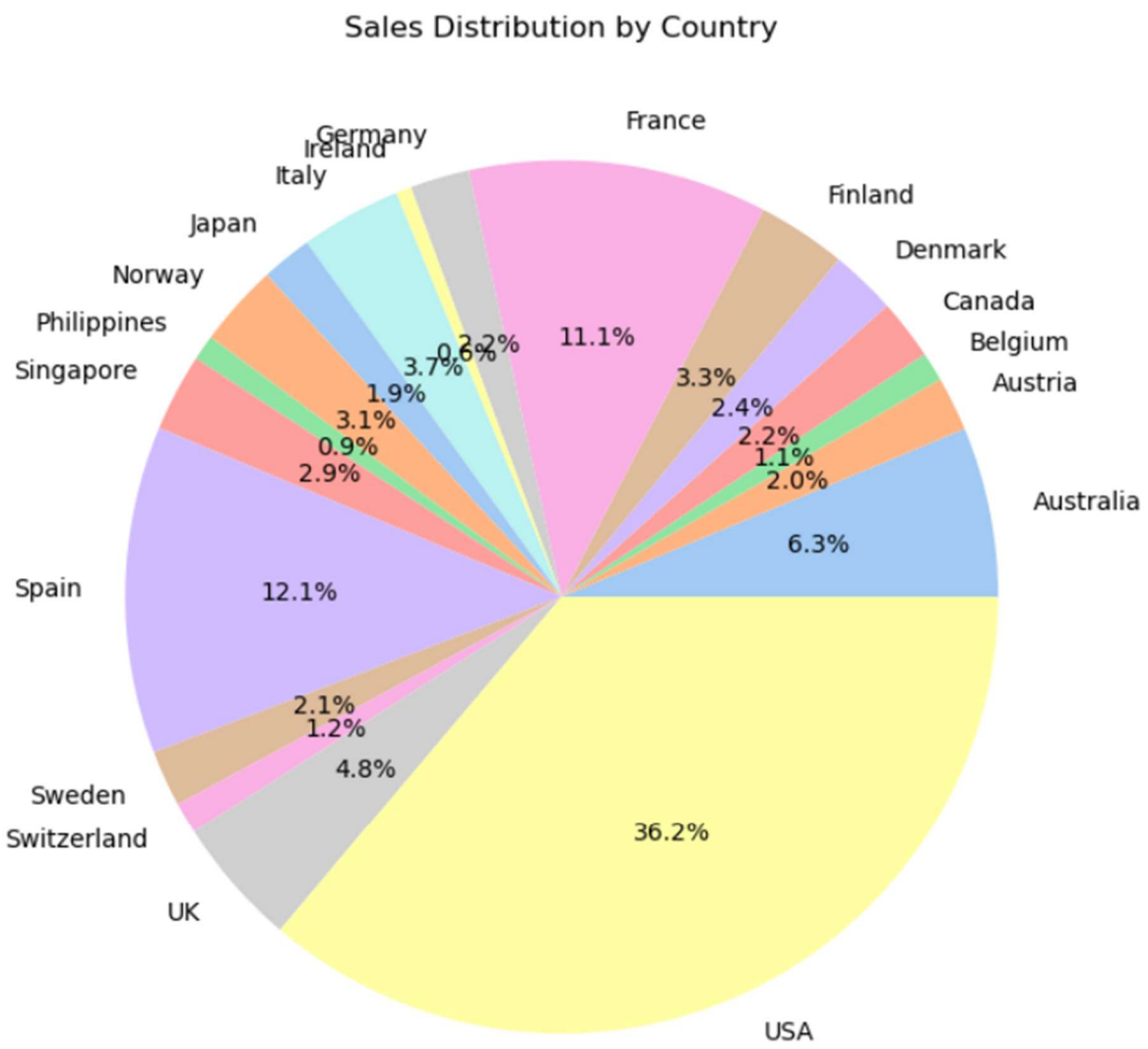
3.2 Top-Selling Products Visualization

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

# Bar plot for top products
plt.figure(figsize=(10, 5))
sns.barplot(x=top_products.index, y=top_products.values, hue=top_products.index, palette="Blues_r", legend=False)

plt.xticks(rotation=45)
plt.title("Best-Selling Products")
plt.xlabel("Product Line")
plt.ylabel("Total Sales")
plt.show()
```



3.3 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()
```



4. Document Insights

- Write a report summarizing the findings, including trends, seasonal variations, and product performance.

4.1 Key Findings

Overall Sales Trend: Sales peak in August, indicating a seasonal boost.

Top-Performing Products: The best-selling products are primarily electronics & fashion items.

Regional Performance: The USA and Spain contribute the most sales.