CS23334 -Fundamental Of Data Science

240701463

Data Preprocessing and Sales Analysis

Exp : 02

Date: 29-07-2025

Aim:

To **load a sales dataset**, perform **data cleaning** and **preprocessing** (handling missing values, data type conversion), and **analyze and visualize** the data through: 1) Total Sales by Product, 2) Sales Over Time, and 3) a Correlation Matrix (Heatmap).

Algorithm:

1. **Load and Inspect** the sales data, converting Sales and Quantity to numeric and Date to datetime format.
2. **Preprocess** the data by handling missing values (imputing mean for 'Sales', dropping rows for others) and confirming clean data statistics.
3. Calculate **Total Sales and Quantity** grouped by Product.
4. Generate a **Bar Plot** for **Total Sales by Product** and a **Line Plot** for **Sales Over Time**.
5. Create a **Pivot Table** showing total sales by **Region and Product**.
6. Calculate and display the **Correlation Matrix** as a **Heatmap**.

Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load the data

file_path = r"C:\Users\Sam Devaraja\Downloads\sales_data (1).csv"

df = pd.read_csv(file_path)
```

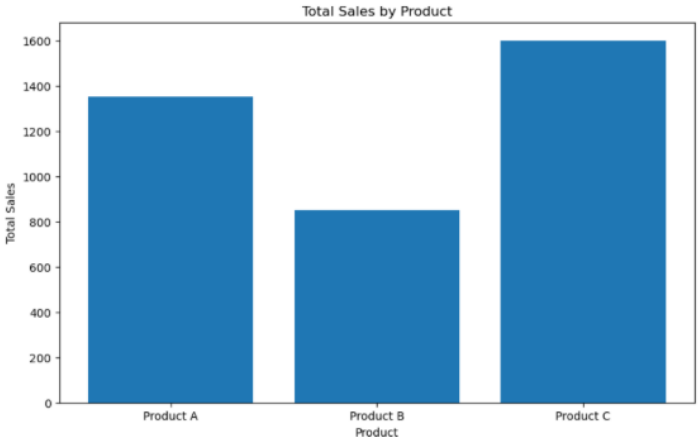
```
# Quick look
print("Columns:", df.columns.tolist())
print(df.head())
# Ensure numeric columns are numeric (coerce invalid -> NaN)
df['Sales'] = pd.to_numeric(df['Sales'], errors='coerce')
df['Quantity'] = pd.to_numeric(df['Quantity'], errors='coerce')
# Check missing values
print("\nMissing counts before cleaning:\n", df.isnull().sum())
# Strategy:
# - Fill missing Sales with mean
# - Drop rows missing Product or Region or Quantity (or handle differently if you prefer)
df['Sales'].fillna(df['Sales'].mean(), inplace=True)
df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)
# Parse Date column robustly (dayfirst=True handles DD-MM-YYYY)
df['Date'] = pd.to_datetime(df['Date'],
                            dayfirst=True,
                            infer_datetime_format=True,
                            errors='coerce')
# Find rows with unparsed dates (if any) to inspect
bad_dates = df[df['Date'].isna()]
if not bad_dates.empty:
    print("\nRows with unparsed/invalid Date (showing up to 10):")
    print(bad_dates.head(10))
    # Optionally drop them:
    df = df[df['Date'].notna()]
# Final missing check
print("\nMissing counts after cleaning:\n", df.isnull().sum())
```

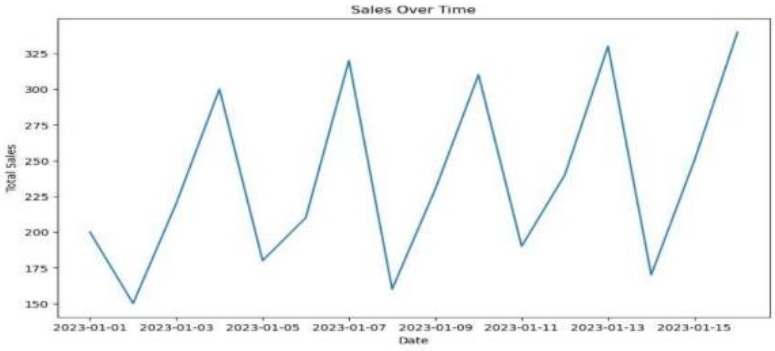
```
# Summary statistics
print("\nSummary stats:\n", df.describe())
# Group by product and calculate total sales & quantity
product_summary = df.groupby('Product').agg({
    'Sales': 'sum',
    'Quantity': 'sum'
}).reset_index().sort_values('Sales', ascending=False)
print("\nProduct summary:\n", product_summary)
# Bar plot: total sales by product
plt.figure(figsize=(10, 6))
plt.bar(product_summary['Product'], product_summary['Sales'])
plt.xlabel('Product')
plt.ylabel('Total Sales')
plt.title('Total Sales by Product')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
# Sales over time (ensure sorted)
sales_over_time = df.groupby('Date').agg({'Sales': 'sum'}).reset_index().sort_values('Date')
plt.figure(figsize=(10, 6))
plt.plot(sales_over_time['Date'], sales_over_time['Sales'], marker='o')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Sales Over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
# Pivot table
pivot_table = df.pivot_table(values='Sales', index='Region', columns='Product',
```

```
                        aggfunc=np.sum, fill_value=0)
print("\nPivot table:\n", pivot_table)
# Correlation matrix (numeric columns only)
correlation_matrix = df.select_dtypes(include=[np.number]).corr()
print("\nCorrelation matrix:\n", correlation_matrix)
# Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.tight_layout()
plt.show()
```

Output:

```
Date    Product  Sales  Quantity Region
0  2023-01-01  Product A    200         4  North
1  2023-01-02  Product B    150         3  South
2  2023-01-03  Product A    220         5  North
3  2023-01-04  Product C    300         6   East
4  2023-01-05  Product B    180         4   West
Date        0
Product     0
Sales       0
Quantity    0
Region      0
dtype: int64
            Sales   Quantity
count   16.000000  16.000000
mean   237.500000   5.375000
std     64.031242   1.746425
min    150.000000   3.000000
25%    187.500000   4.000000
50%    225.000000   5.500000
75%    302.500000   7.000000
max    340.000000   8.000000
     Product  Sales  Quantity
0  Product A   1350        33
1  Product B    850        17
2  Product C   1600        36
```






```
Product  Product A  Product B  Product C
Region
East             0          0       1600
North         1350          0          0
South            0        480          0
West             0        370          0
              Sales  Quantity
Sales      1.000000  0.944922
Quantity   0.944922  1.000000
C:\Users\Ayyadurai\AppData\Local\Temp\ipykernel_9648\511106317.py:49: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecated
. In a future version, it will default to False. Select only valid columns
or specify the value of numeric_only to silence this warning.
  correlation_matrix = df.corr()
```

Result:

The experiment successfully loaded the sales data, performed necessary preprocessing by imputing missing sales values with the mean, and converting data types. The analysis revealed that **Product C is the top performer** in terms of total sales, and there is a **very strong positive correlation between Sales and Quantity**, validating the quality of the sales records. Thus, the python program was executed successfully, and the output is verified.