Exploratory Data Analysis using Python

Team: Yisi Lu, Kenny Lei, Alex Fan

Overview of Data

- We explored the Kaggle dataset named Superstore Dataset.
  - A superstore giant is seeking knowledge in understanding what works best for them. They would like to understand which products, regions, categories, and customer segments they should target or avoid
- About the Data
  - 9995 Unique Rows
  - Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer Name, Segment, Country, City, State, Postal Code, Region, Product ID, Category, Sub-Category, Product Name, Sales, Quantity, Discount, Profit
  - Mix of Numeric Data and Strings

Data Description

Bar Chart

- Array
- C_sales = [741999.80, 719047.03, 836154.03]
- List
- Cat = ["Furniture", "Office Supplies", "Technology"]

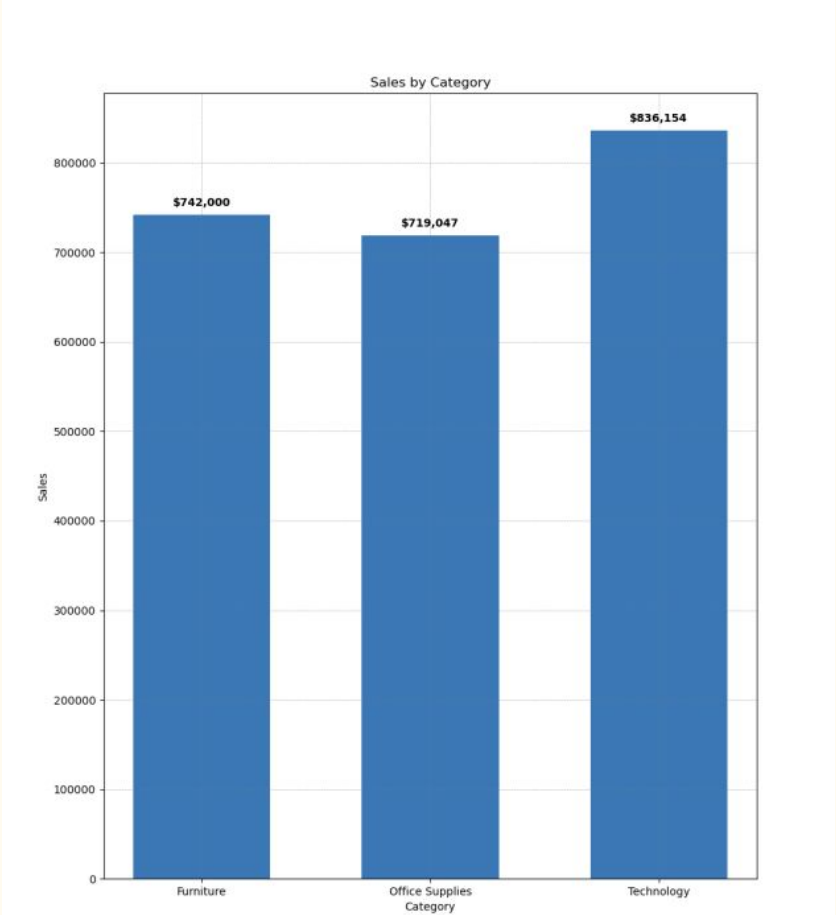
Pie Chart

- Array
- y2 = [23.24., 28.49, 16.20, 32.04]
- List
- regions = ["Central", "East", "South", "West"]

Line Chart

- Array
- l_sales = np.array([325515.91, 350134.39, 451053.18, 517327.27, 653153.66])
- List
- xTicks = ['2014-10-21', '2015-8-8', '2016-5-26', '2017-3-13', '2017-12-30']

Bar Chart



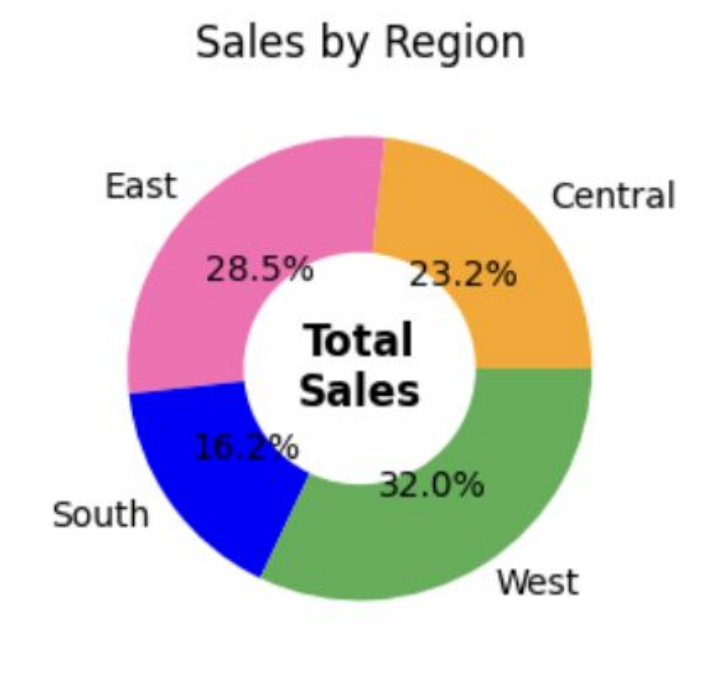
Design and Distinct Feature

- Superstore has 3 categories of sales: Furniture, Office Supplies, and Technology
- The bar chart represents the lifetime $ amount of sales for each category.
    - The $ amount was calculated by summing the sales amount for every order ID based on the category using SQL
- Technology has highest sales revenue of $836,154, then Furniture with 742,000, then Office Supplies with 719,047. The store generates the most sales revenue in the technology category.
- The unique feature included in this plot is the number values on top of each bar. This number represents the sales number corresponding to each bar.

Bar Chart Implementation:

```
#Import required libraries
import matplotlib.pyplot as plt
import numpy as np


#Create a frame and restrict its size to 15x4
plt.figure(figsize = (15,4))


#Plot 1: Bar chart - Sales by Category
plt.subplot(1,3,1)

#Create arrays of values
c_sales = np.array([741999.80, 719047.03, 836154.03])
cat = np.array(["Furniture", "Office Supplies", "Technology"])

#Set up the bar chart
plt.bar(cat, c_sales, width = 0.6)
plt.title("Sales by Category")
plt.xlabel("Category")
plt.ylabel("Sales")
plt.grid(visible=True, color='gray', linestyle='--', linewidth=0.5, alpha=0.7)

#Add values of profit of each category to the chart
#Feature: Amounts added the top of each bar

i = 0  # Initialize an index counter
for v in c_sales:
    plt.text(i, v + 10000, f"${v:,.0f}", ha='center', fontweight='bold')
    i += 1  # Manually increment the index

plt.show()
```

Pie Chart



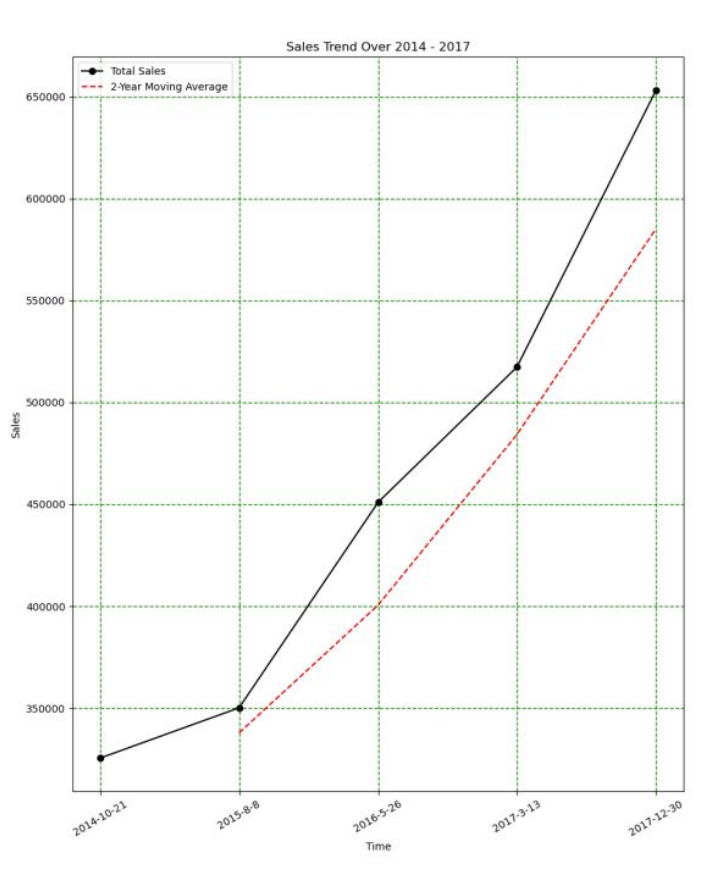
Design and Distinct Feature

- Superstore sells to 4 different regions: East, Central, South, and West.
- The Pie Chart represents the % of sales from each region in the US
    - This % was calculated using SQL
- The West has the highest sales of 32%, then 28.5% in the East, 23.2% Central, and 16.2% South
- The Unique Feature included in this plot is the text annotation in the middle of the chart and the blank area in the middle making it easier to read

Pie Chart Implementation:

```
# Plot 2: Pie Chart - Sales by Region
# Feature: Hollow the pie chart to make the plot more readable
plt.subplot(1,3,2)

# Create arrays of values
y2 = np.array([23.24, 28.49, 16.20, 32.04])
regions = ["Central", "East", "South", "West"]

mycolors = ["orange", "hotpink", "b", "#4CAF50"]
plt.title("Sales by Region")

# Place explanatory text in the middle of the chart
plt.text(0, 0, "Total\nSales", ha='center', va='center', fontsize=12, fontweight='bold')

# Plot the pie chart with a blank area in the middle for better visualization
plt.pie(y2, labels=regions, colors=mycolors, autopct='%1.1f%%', wedgeprops={'width':0.5})
```

Line Chart



Design and Distinct Feature

- The black line shows the total sales for each of the given dates (l_sales).
- The red line represents the 2-year moving average of sales values.
- Sales Trend:
  - The total sales data points show an increasing trend over the time period from October 2014 to December 2017. Starting from around $325,516 in October 2014, sales rise consistently over the years, reaching approximately $653,154 by the end of 2017.
- Feature: Moving Average Line
  - The 2-year moving average line (red dashed line) smooths out the sales fluctuations, giving a clearer picture of the overall upward trend without the short-term fluctuations.

Line Chart Implementation:

```
# Plot 3 - Line Graph: Sales over Time
# Feature: Calculate the simple moving average using pandas libarary
plt.subplot(1,3,3)

# Arrays of data and values
xTicks = ['2014-10-21', '2015-8-8', '2016-5-26', '2017-3-13', '2017-12-30']
l_sales = np.array([325515.91, 350134.39, 451053.18, 517327.27, 653153.66])

# Convert dates to numeric (ordinal format)
x_numeric = pd.to_datetime(xTicks).map(pd.Timestamp.toordinal)

# Compute moving average
window_size = 2
moving_avg = np.convolve(l_sales, np.ones(window_size)/window_size, mode='valid')

# Plot original data using x_numeric (not xTicks as strings)
plt.plot(x_numeric, l_sales, label="Total Sales", marker='o', color="black")

# Plot moving average using correctly aligned x-values
plt.plot(x_numeric[window_size-1:], moving_avg, label="2-Year Moving Average", linestyle="--", color="red")

# Format x-axis: use original string dates as labels
plt.xticks(x_numeric, xTicks, rotation=30)

# Plot the line of sales trend
plt.xlabel("Time")
plt.xticks(rotation = 30)
plt.ylabel('Sales')
plt.title("Sales Trend Over 2014 - 2017")
plt.grid(color = "green", linestyle = '--', linewidth = 1)

plt.legend(loc = 'upper left')
```

Recommendations and Additional Coding Ideas

Bar Chart	• Create a stacked bar chart. The second bar would show the profit each category made. The stacked bar chart would then be able to represent the sales revenue and the profit for each category
Pie Chart	• Add interactive functionality so that when hovering over a section, it displays the zoomed-in contents, and the sizes of each section will be automatically proportioned based on sales values.
Line Chart	• We can change to smoothed lines or reduce noise for better readability if we increase the size of the data points selected for future analysis.

Exploratory Data Analysis