Internship Project #1: Amazon Sales Data Analysis

- Submitted by: Saurabh Sahu

Problem Statement:

- Sales management has gained importance to meet increasing competition and the need for improved methods of distribution to reduce cost and to increase profits.
- Sales management today is the most important function in a commercial and business enterprise.
- Do ETL: Extract-Transform-Load some Amazon dataset and find for me Sales-trend -> month-wise, year-wise, yearly_month-wise.
- Find key metrics and factors and show the meaningful relationships between attributes.

Dataset Link:

https://drive.google.com/file/d/10sofXyF6NjwN6ngLyFfiPI-CUDpeqaN_/view

Step 1: Importing Libraries and Datasets.

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

```
from google.colab import files
data = files.upload()
```

Choose Files Amazon Sales data.csv

- **Amazon Sales data.csv**(text/csv) - 12643 bytes, last modified: 7/12/2024 - 100% done

```
Saving Amazon Sales data.csv to Amazon Sales data.csv
```

```
sales = pd.read_csv('Amazon Sales data.csv')
sales.head()
```

	Region	Country	Item Type	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	5/28/2010	669165933	6/27/2010	9925
1	Central America and the Caribbean	Grenada	Cereal	Online	C	8/22/2012	963881480	9/15/2012	2804
2	Europe	Russia	Office Supplies	Offline	L	5/2/2014	341417157	5/8/2014	1779
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	Online	C	6/20/2014	514321792	7/5/2014	8102
4	Sub-Saharan Africa	Rwanda	Office Supplies	Offline	L	2/1/2013	115456712	2/6/2013	5062

Next steps: Generate code with `sales` | View recommended plots | New interactive sheet

Step 2 - Checking and Cleansing of Data.

```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Region          100 non-null    object
 1   Country         100 non-null    object
 2   Item Type       100 non-null    object
 3   Sales Channel   100 non-null    object
 4   Order Priority  100 non-null    object
 5   Order Date      100 non-null    object
 6   Order ID        100 non-null    int64
 7   Ship Date       100 non-null    object
 8   Units Sold      100 non-null    int64
 9   Unit Price      100 non-null    float64
 10  Unit Cost       100 non-null    float64
 11  Total Revenue   100 non-null    float64
 12  Total Cost      100 non-null    float64
 13  Total Profit    100 non-null    float64
dtypes: float64(5), int64(2), object(7)
memory usage: 11.1+ KB
```

From the basic info above:

- We need to convert Date columns (like Order Date & Ship Date) into DateTime Datatype.
- Given data doesn't have non-null values as it matches with the no of entries.

We can double check it by:

```
print("Null Values: \n")
sales.isnull().sum()
```

```
Null Values:

Region            0
Country           0
Item Type         0
Sales Channel     0
Order Priority    0
Order Date        0
Order ID          0
Ship Date         0
Units Sold        0
Unit Price        0
Unit Cost         0
Total Revenue     0
Total Cost        0
Total Profit      0
dtype: int64
```

- No Null values present in the given dataset. Data is clean, hence suitable for further calculations and analysis.

```
# Changing datatype for date columns
sales['Order Date'] = pd.to_datetime(sales['Order Date'])
sales['Ship Date'] = pd.to_datetime(sales['Ship Date'])
```

```
# Extracting Months and Years from Order Date
sales['Month'] = sales['Order Date'].dt.month
sales['Year'] = sales['Order Date'].dt.year
```

- Note- Here values in month are in numeric (not month names) .

Step 3: EDA

Sum of 'Units Sold', 'Unit Cost', 'Total Revenue', and 'Total Profit'

```
total_units_sold = sales['Units Sold'].sum()
total_unit_cost = sales['Unit Cost'].sum()
total_revenue = sales['Total Revenue'].sum()
total_profit = sales['Total Profit'].sum()

print("Total Units Sold:", total_units_sold)
print("Total Unit Cost:", total_unit_cost)
print("Total Revenue:", total_revenue)
print("Total Profit:", total_profit)
```

```
Total Units Sold: 512871
Total Unit Cost: 19104.8
Total Revenue: 137348768.31
Total Profit: 44168198.39999999
```

Total number of countries. Top 5 and Bottom 5 countries by sales percentage

```
# Calculate total number of countries
countries = sales['Country'].nunique()
print("Total number of countries:", countries)

# Calculate sales by country
country_sales = sales.groupby('Country')['Total Profit'].sum().sort_values(ascending=False)

# Calculate sales percentage for each country
total_sales = country_sales.sum()
country_sales_perc = (country_sales / total_sales) * 100

# Print top 5 countries by sales percentage
print("\nTop 5 countries by sales percentage:")
print(country_sales_perc.head())

# Print bottom 5 countries by sales percentage
print("\nBottom 5 countries by sales percentage:")
print(country_sales_perc.tail())
```

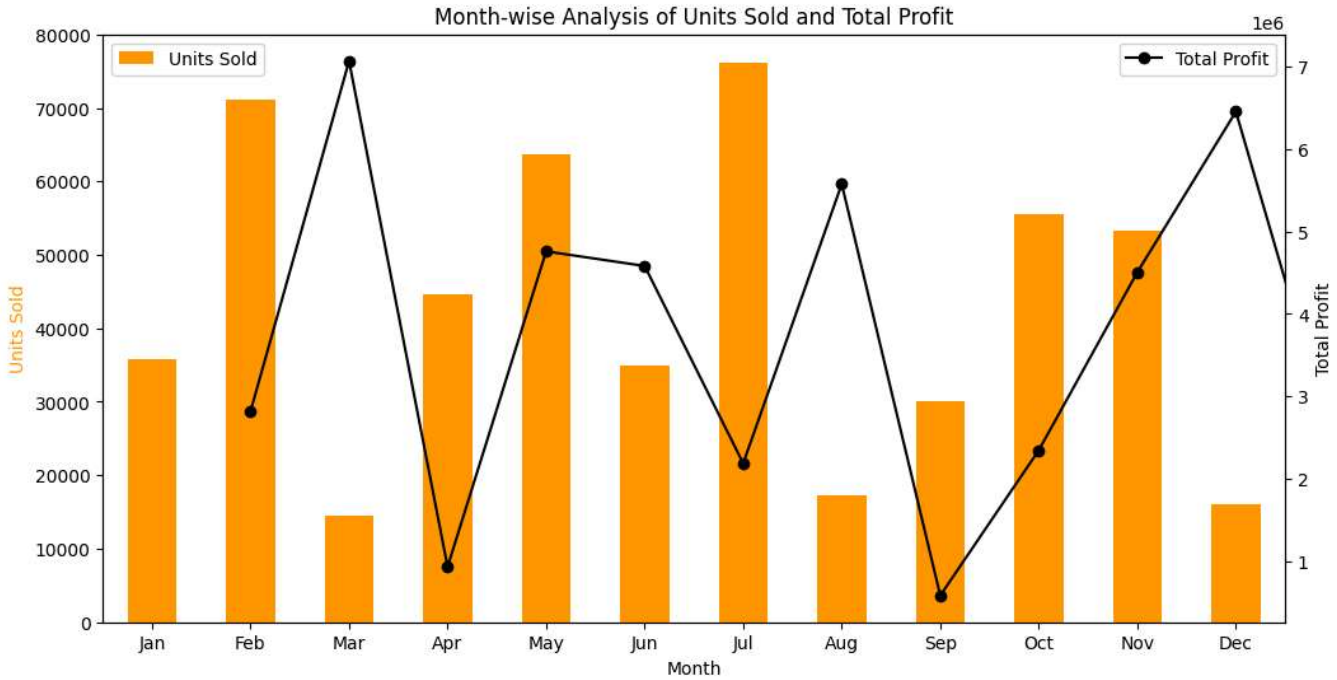
```
Total number of countries: 76

Top 5 countries by sales percentage:
Country
Djibouti    5.491095
Myanmar     4.081606
Pakistan    3.894028
Samoa       3.800338
Honduras    3.645038
```

```
Name: Total Profit, dtype: float64

Bottom 5 countries by sales percentage:
Country
Slovakia       0.024441
Syria          0.020647
Kyrgyzstan     0.017723
New Zealand    0.011933
Kuwait         0.002848
Name: Total Profit, dtype: float64
```

Step 4: Data Visualization

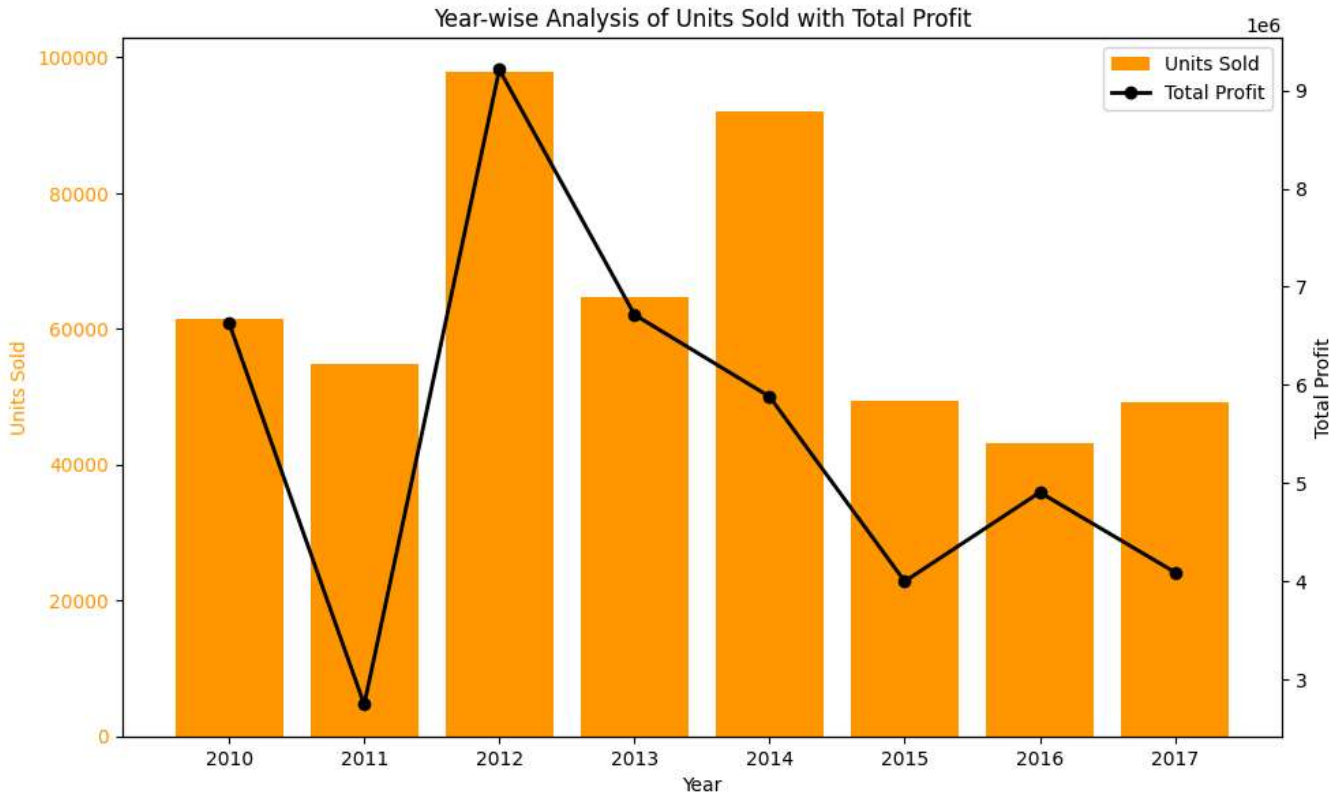
Month-wise Analysis of Units Sold and Total Profit

```
# Group the data by month and calculate total units sold and profit
monthly_units_profit = sales.groupby('Month')[['Units Sold', 'Total Profit']].sum()

plt.figure(figsize=(12, 6))

# Bar chart for Units Sold
ax = monthly_units_profit['Units Sold'].plot(kind='bar', color='#FF9900', label='Units Sold'
ax.set_ylabel('Units Sold', color='#FF9900')

# Line plot for Total Profit
ax2 = ax.twinx()
monthly_units_profit['Total Profit'].plot(kind='line', marker='o', color='#000000', ax=ax2,
ax2.set_ylabel('Total Profit', color='#000000')

# Titles, Legends, & other plot customizations
plt.title('Month-wise Analysis of Units Sold and Total Profit')
ax.set_xticklabels(monthly_units_profit.index, rotation = 0)
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
ax.legend(loc='upper left')
ax2.legend(loc='upper right')

plt.show()
```



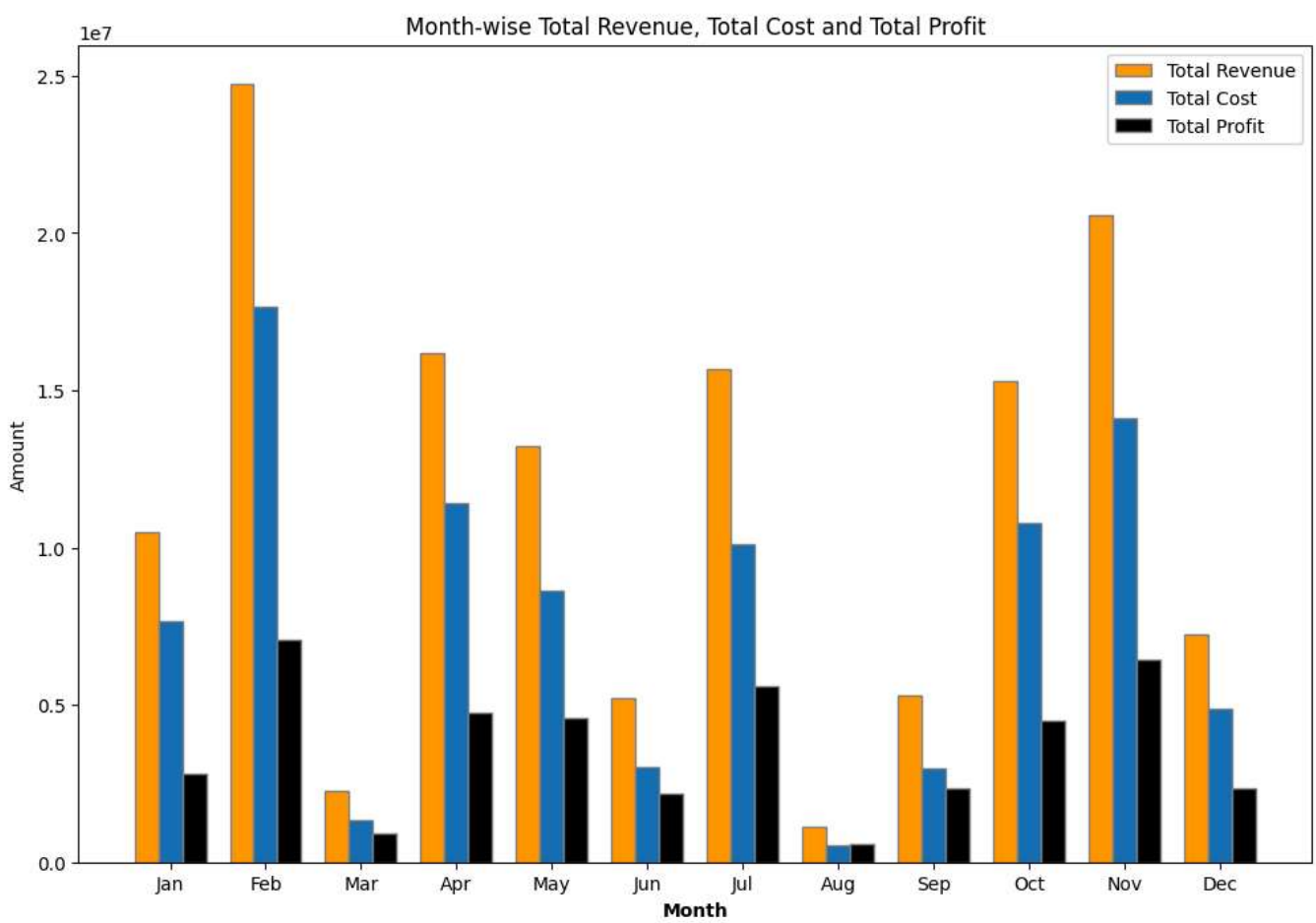
Year-wise Analysis of Units Sold and Total Profit

```
# Calculate total units sold and profit for each year
yearly_units_sold = sales.groupby('Year')['Units Sold'].sum()
yearly_profit = sales.groupby('Year')['Total Profit'].sum()

fig, ax1 = plt.subplots(figsize=(10, 6))

# Plot the Bar chart for units sold
color = '#FF9900'
ax1.set_xlabel('Year')
ax1.set_ylabel('Units Sold', color=color)
bars = ax1.bar(yearly_units_sold.index, yearly_units_sold, color=color, label='Units Sold')
ax1.tick_params(axis='y', labelcolor=color)

# Create a secondary y-axis for total profit
ax2 = ax1.twinx()

# Plot the Line chart for total profit
color = 'black'
ax2.set_ylabel('Total Profit', color=color)
lines = ax2.plot(yearly_profit.index, yearly_profit, color=color, marker='o', linestyle='-',
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # Adjust layout to prevent overlap

# Combine Legends for both axes
lines_labels = [bars, lines[0]]
labels = [l.get_label() for l in lines_labels]
ax1.legend(lines_labels, labels, loc='upper right')

plt.title('Year-wise Analysis of Units Sold with Total Profit')
plt.show()
```



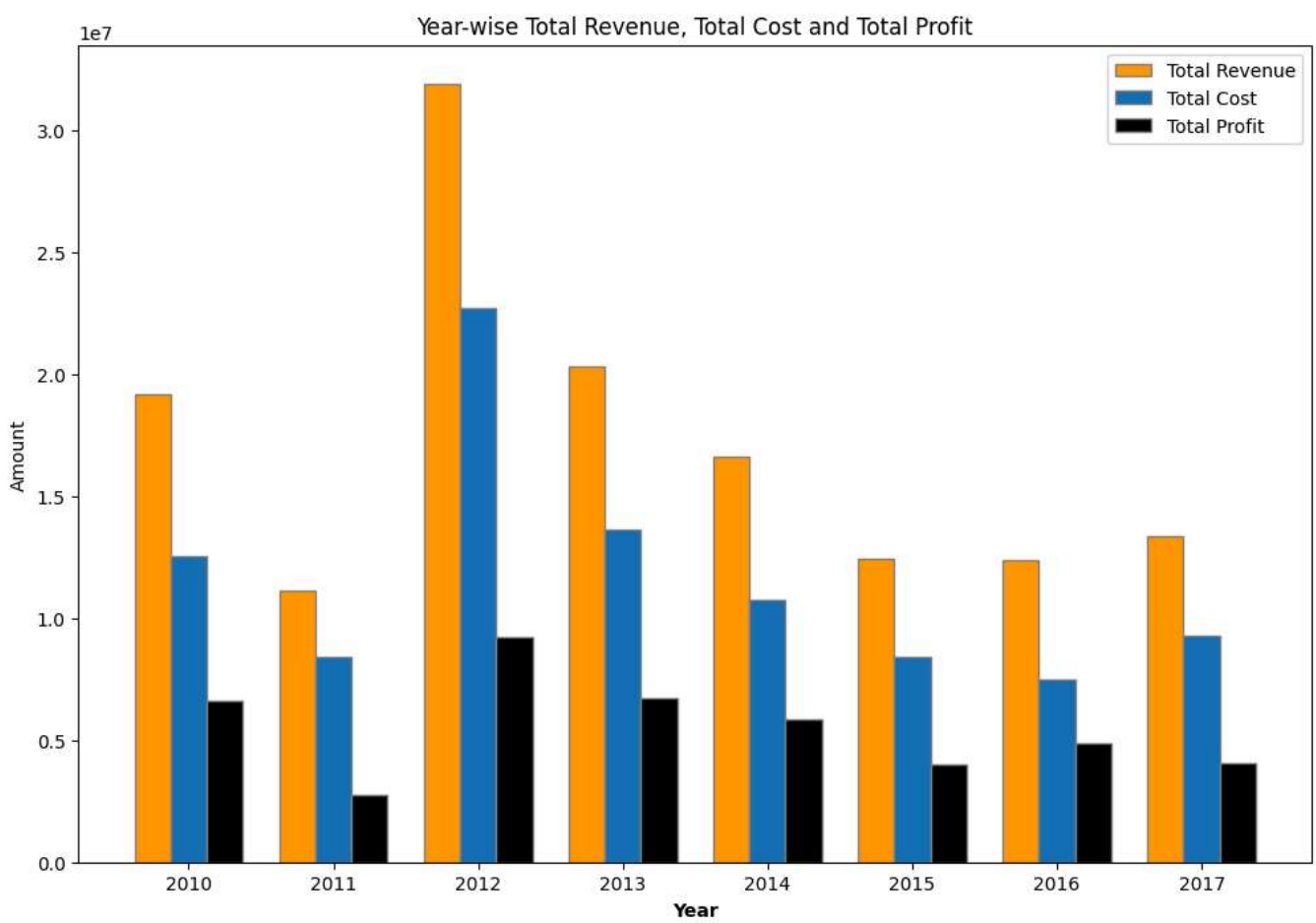
Month-wise Total Revenue, Total Cost, and Total Profit

```
# Aggregate data by month
monthly_data = sales.groupby('Month').agg({'Total Revenue': 'sum', 'Total Cost': 'sum', 'Tot

# Create the clustered column chart & bar width
plt.figure(figsize=(12, 8))
bar_width = 0.25

# Set positions of the bars on the x-axis
r1 = range(len(monthly_data))
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]

# Create the bars
plt.bar(r1, monthly_data['Total Revenue'], color='#FF9900', width=bar_width, edgecolor='grey
plt.bar(r2, monthly_data['Total Cost'], color='#146EB4', width=bar_width, edgecolor='grey',
plt.bar(r3, monthly_data['Total Profit'], color= '#000000', width=bar_width, edgecolor='grey

# Add xticks on the middle of the group bars
plt.xlabel('Month', fontweight='bold')
plt.xticks([r + bar_width for r in range(len(monthly_data))], ['Jan', 'Feb', 'Mar', 'Apr', '
plt.ylabel('Amount')
plt.title('Month-wise Total Revenue, Total Cost and Total Profit')

# Create legend
plt.legend()

# Show the plot
plt.show()
```



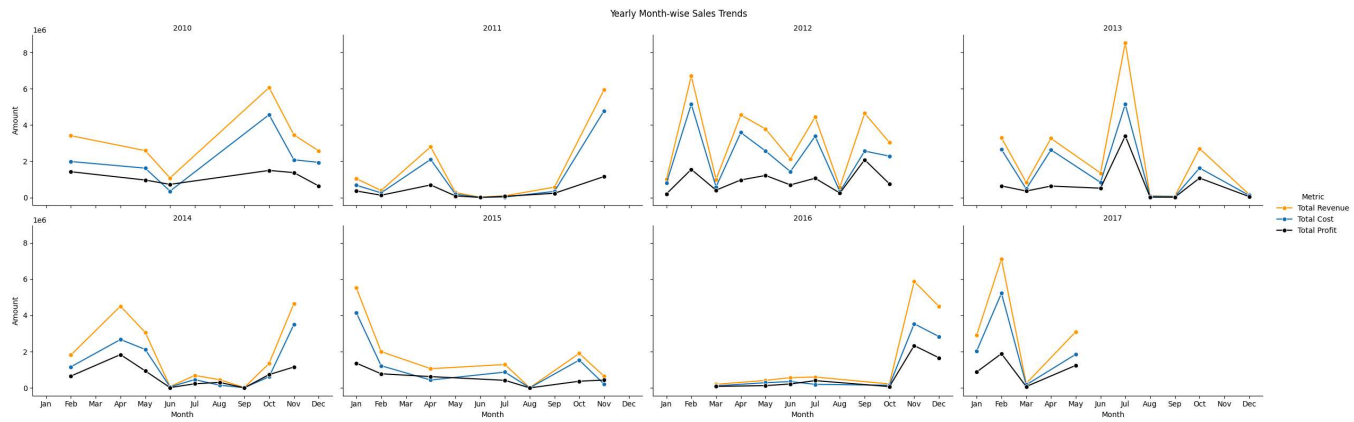
Year-wise Total Revenue, Cost, and Total Profit

```
# Aggregate data by year
yearly_data = sales.groupby('Year').agg({'Total Revenue': 'sum', 'Total Cost': 'sum', 'Total

# Create the clustered column chart & bar width
plt.figure(figsize=(12, 8))
bar_width = 0.25

# Set positions of the bars on the x-axis
r1 = range(len(yearly_data))
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]

# Create the bars
plt.bar(r1, yearly_data['Total Revenue'], color= '#FF9900', width= bar_width, edgecolor= 'gr
plt.bar(r2, yearly_data['Total Cost'], color= '#146EB4', width= bar_width, edgecolor= 'grey'
plt.bar(r3, yearly_data['Total Profit'], color= '#000000', width=bar_width, edgecolor='grey'

# Add xticks on the middle of the group bars
plt.xlabel('Year', fontweight='bold')
plt.xticks([r + bar_width for r in range(len(yearly_data))], yearly_data['Year'])
plt.ylabel('Amount')
plt.title('Year-wise Total Revenue, Total Cost and Total Profit')

# Create legend
plt.legend()

# Show the plot
plt.show()
```

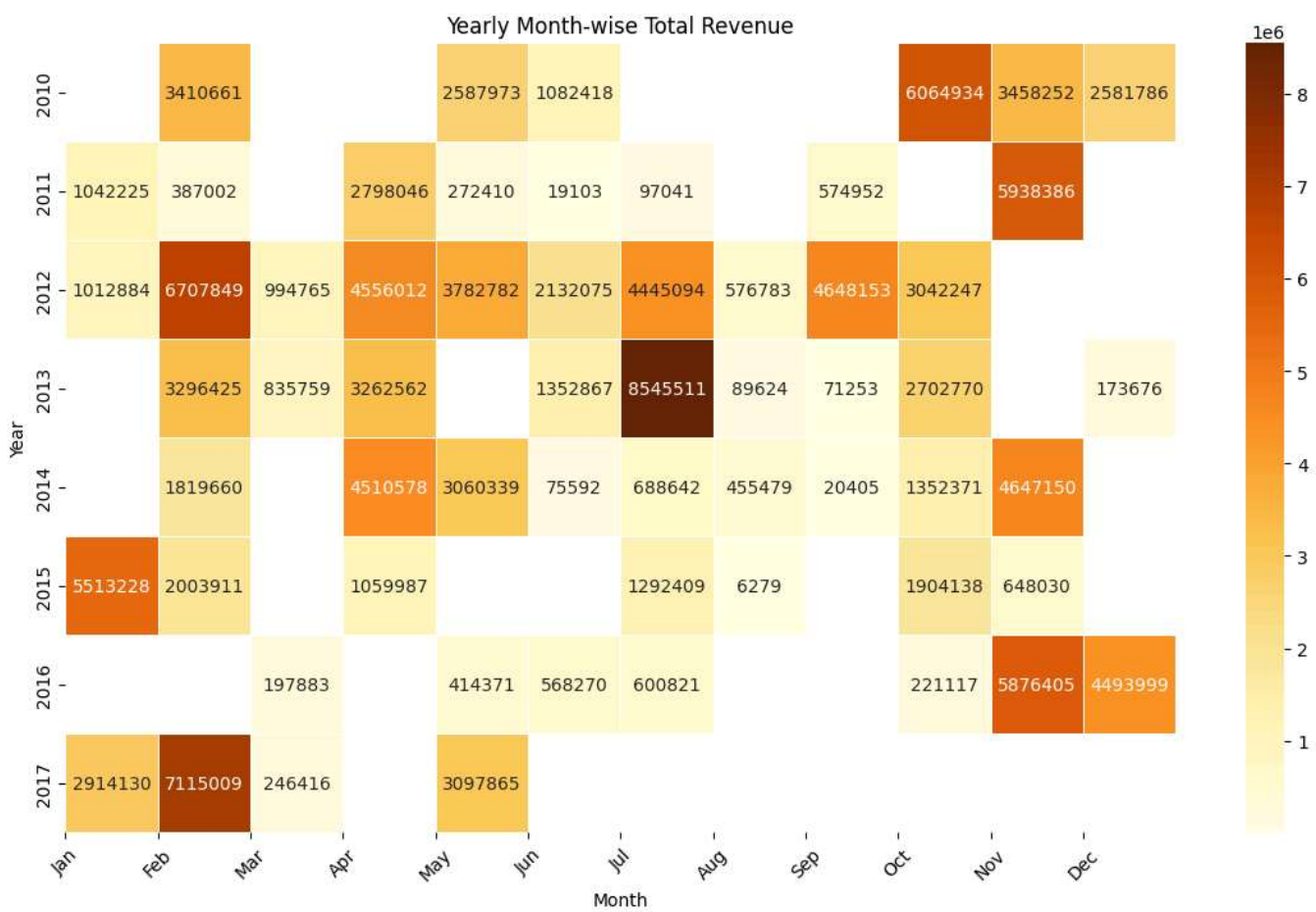


Yearly Month-wise Sales Trends for Total Revenue, Total Cost and Total Profits

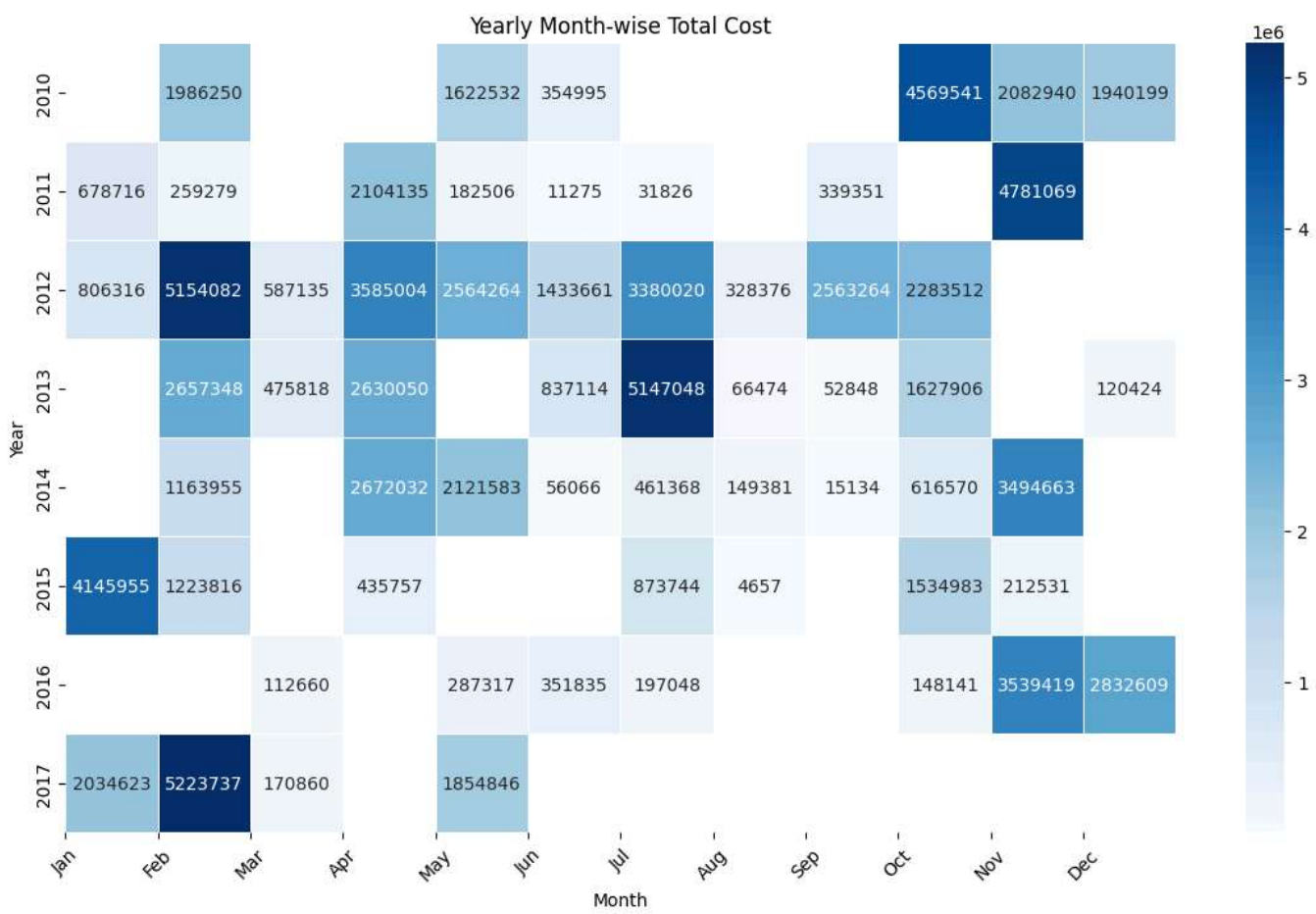
```
# Group the data by year and month
year_month_sales = sales.groupby(['Year', 'Month']).agg({'Total Revenue': 'sum', 'Total Cost

# Melt the data for easier plotting
melted_sales = pd.melt(year_month_sales, id_vars=['Year', 'Month'], value_vars=['Total Rever

# Define colors
colors = {'Total Revenue': '#FF9900', 'Total Cost': '#146EB4', 'Total Profit': '#000000'}

# Create the Facet Grid plot
g = sns.FacetGrid(melted_sales, col='Year', hue='Metric', col_wrap=4, height=4, aspect=1.5,
g.map(sns.lineplot, 'Month', 'Value', marker='o')
g.add_legend()
g.set_axis_labels('Month', 'Amount')
g.set_titles('Year: {col_name}')
g.set(xticks=range(1, 13), xticklabels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Au
g.set_titles("{col_name}")
plt.subplots_adjust(top=0.92)
g.fig.suptitle('Yearly Month-wise Sales Trends')
plt.show()
```



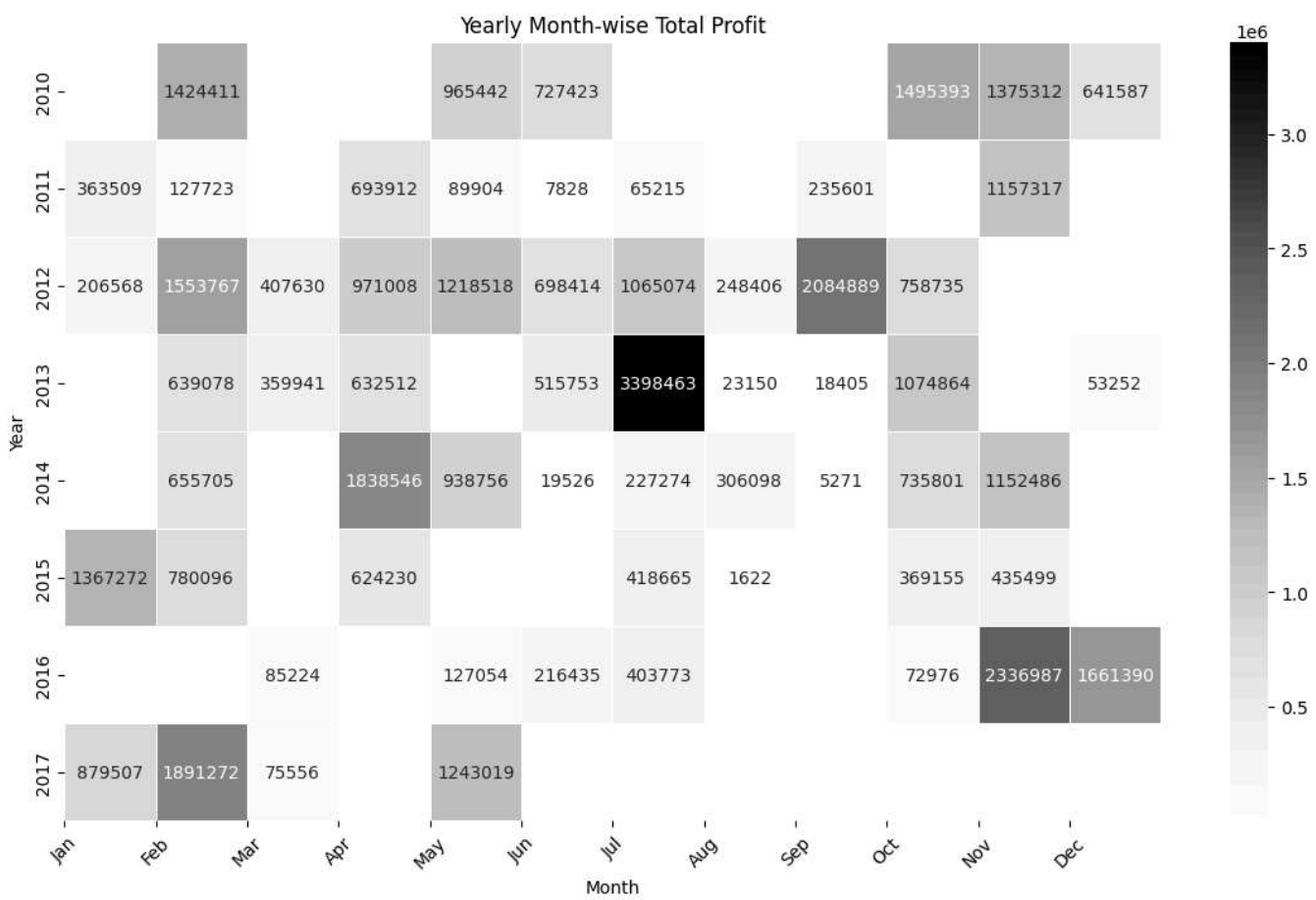

```
# Pivot table for heatmap
yearly_monthwise_revenue = sales.pivot_table(values='Total Revenue', index='Year', columns='

plt.figure(figsize=(14, 8))
sns.heatmap(yearly_monthwise_revenue, annot=True, fmt=".0f", cmap='YlOrBr', linewidths=.5)
plt.title('Yearly Month-wise Total Revenue')
plt.xlabel('Month')
plt.ylabel('Year')
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
plt.show()
```




```
# Pivot table for heatmap
yearly_monthwise_cost = sales.pivot_table(values='Total Cost', index='Year', columns='Month'

plt.figure(figsize=(14, 8))
sns.heatmap(yearly_monthwise_cost, annot=True, fmt=".0f", cmap='Blues', linewidths=.5)
plt.title('Yearly Month-wise Total Cost')
plt.xlabel('Month')
plt.ylabel('Year')
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
plt.show()
```

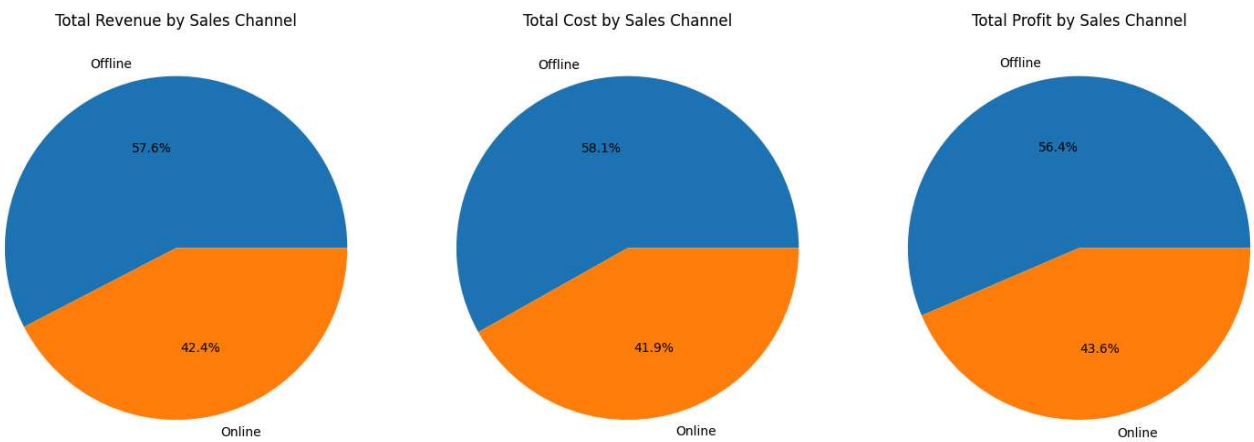



```
# Pivot table for heatmap
yearly_monthwise_profit = sales.pivot_table(values='Total Profit', index='Year', columns='Mc

plt.figure(figsize=(14, 8))
sns.heatmap(yearly_monthwise_profit, annot=True, fmt=".0f", cmap='Greys', linewidths=.5)
plt.title('Yearly Month-wise Total Profit')
plt.xlabel('Month')
plt.ylabel('Year')
plt.xticks(ticks=range(12), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
plt.show()
```



Sales Channel-wise plots for Total Revenue, Total Cost and Total Profit.

```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Pie chart for Total Revenue
sales.groupby('Sales Channel')['Total Revenue'].sum().plot(kind='pie', ax=axes[0], autopct='
axes[0].set_title('Total Revenue by Sales Channel')
axes[0].set_ylabel('')

# Pie chart for Total Cost
sales.groupby('Sales Channel')['Total Cost'].sum().plot(kind='pie', ax=axes[1], autopct='%1.
axes[1].set_title('Total Cost by Sales Channel')
axes[1].set_ylabel('')

# Pie chart for Total Profit
sales.groupby('Sales Channel')['Total Profit'].sum().plot(kind='pie', ax=axes[2], autopct='%
axes[2].set_title('Total Profit by Sales Channel')
axes[2].set_ylabel('')

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```



Units Sold and Total Profit earned by different Item Type.

```
# Create a figure and a set of subplots
fig, ax1 = plt.subplots(figsize=(10, 6))

# Bar plot for item types and units sold
sns.barplot(x='Item Type', y='Units Sold', data=sales, ax=ax1, errorbar=None, color='#FF9900
plt.xticks(rotation=45)

# Create a second y-axis
ax2 = ax1.twinx()

# Line plot for profit
sns.lineplot(x='Item Type', y='Total Profit', data=sales, ax=ax2, errorbar=None, color= '#00
plt.xticks(rotation=45)

# Set labels and title
ax1.set_xlabel('Item Type')
ax1.set_ylabel('Units Sold')
ax2.set_ylabel('Total Profit')
plt.title('Item Type vs Units Sold & Total Profit')

# Show the plot
plt.tight_layout()
plt.show()
```