amazon_project_internship

April 21, 2024

AMAZON SALES REPORT- Checking the data set and performing required Exploratory Data analysis and pre-processing

[11]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[14]:
```
df= pd.read_csv(r"C:\Users\LENOVO\Desktop\Amazon Sales data.csv")
```

[15]:
```
df.shape
```

[15]: (100, 14)

[16]:
```
df.head(10)
```

[16]:

	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
5	Australia and Oceania	Solomon Islands	Baby Food	Online	C	2/4/2015	547995746	2/21/2015	2974
6	Sub-Saharan Africa	Angola	Household	Offline	M	4/23/2011	135425221	4/27/2011	4187
7	Sub-Saharan Africa	Burkina Faso	Vegetables	Online	H	7/17/2012	871543967	7/27/2012	8082
8	Sub-Saharan Africa	Republic of the Congo	Personal Care	Offline	M	7/14/2015	770463311	8/25/2015	6070
9	Sub-Saharan Africa	Senegal	Cereal						

```
9        Online             H  4/18/2014  616607081  5/30/2014        6593

   Unit Price  Unit Cost  Total Revenue  Total Cost  Total Profit
0      255.28     159.42     2533654.00  1582243.50     951410.50
1      205.70     117.11      576782.80   328376.44     248406.36
2      651.21     524.96     1158502.59   933903.84     224598.75
3        9.33       6.92       75591.66    56065.84      19525.82
4      651.21     524.96     3296425.02  2657347.52     639077.50
5      255.28     159.42      759202.72   474115.08     285087.64
6      668.27     502.54     2798046.49  2104134.98     693911.51
7      154.06      90.93     1245112.92   734896.26     510216.66
8       81.73      56.67      496101.10   343986.90     152114.20
9      205.70     117.11     1356180.10   772106.23     584073.87
```

Understanding all the unique values of important attributes

[17]:
```
print(df['Region'].unique())
print('\n')
print(df['Country'].unique())
print(df['Item Type'].unique())
print(df['Sales Channel'].unique())
print(df['Order Priority'].unique())
```

```
['Australia and Oceania' 'Central America and the Caribbean' 'Europe'
 'Sub-Saharan Africa' 'Asia' 'Middle East and North Africa'
 'North America']


['Tuvalu' 'Grenada' 'Russia' 'Sao Tome and Principe' 'Rwanda'
 'Solomon Islands' 'Angola' 'Burkina Faso' 'Republic of the Congo'
 'Senegal' 'Kyrgyzstan' 'Cape Verde' 'Bangladesh' 'Honduras' 'Mongolia'
 'Bulgaria' 'Sri Lanka' 'Cameroon' 'Turkmenistan' 'East Timor' 'Norway'
 'Portugal' 'New Zealand' 'Moldova ' 'France' 'Kiribati' 'Mali'
 'The Gambia' 'Switzerland' 'South Sudan' 'Australia' 'Myanmar' 'Djibouti'
 'Costa Rica' 'Syria' 'Brunei' 'Niger' 'Azerbaijan' 'Slovakia' 'Comoros'
 'Iceland' 'Macedonia' 'Mauritania' 'Albania' 'Lesotho' 'Saudi Arabia'
 'Sierra Leone' "Cote d'Ivoire" 'Fiji' 'Austria' 'United Kingdom'
 'San Marino' 'Libya' 'Haiti' 'Gabon' 'Belize' 'Lithuania' 'Madagascar'
 'Democratic Republic of the Congo' 'Pakistan' 'Mexico'
 'Federated States of Micronesia' 'Laos' 'Monaco' 'Samoa ' 'Spain'
 'Lebanon' 'Iran' 'Zambia' 'Kenya' 'Kuwait' 'Slovenia' 'Romania'
 'Nicaragua' 'Malaysia' 'Mozambique']
['Baby Food' 'Cereal' 'Office Supplies' 'Fruits' 'Household' 'Vegetables'
 'Personal Care' 'Clothes' 'Cosmetics' 'Beverages' 'Meat' 'Snacks']
['Offline' 'Online']
['H' 'C' 'L' 'M']
```

[18]:
```
df.isnull().sum()
```

[18]:
```
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
```

[19]:
```
df.duplicated().sum()
```

[19]:
```
0
```

Introducing new columns to analyze data month-wise and year-wise

[20]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Year'] = df['Order Date'].dt.year
df['Month'] = df['Order Date'].dt.strftime('%Y-%m')
df['Month'] = pd.to_datetime(df['Month']).dt.month_name()
# Define month order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August', 'September', 'October', 'November', 'December']
# Convert 'Month' column to categorical with ordered categories
df['Month'] = pd.Categorical(df['Month'], categories=month_order, ordered=True)
df.head()
```

[20]:

	Region	Country	Item Type	\
0	Australia and Oceania	Tuvalu	Baby Food	
1	Central America and the Caribbean	Grenada	Cereal	
2	Europe	Russia	Office Supplies	
3	Sub-Saharan Africa	Sao Tome and Principe	Fruits	
4	Sub-Saharan Africa	Rwanda	Office Supplies	

	Sales Channel	Order Priority	Order Date	Order ID	Ship Date	Units Sold	\
0	Offline	H	2010-05-28	669165933	6/27/2010	9925	
1	Online	C	2012-08-22	963881480	9/15/2012	2804	
2	Offline	L	2014-05-02	341417157	5/8/2014	1779	
3	Online	C	2014-06-20	514321792	7/5/2014	8102	
4	Offline	L	2013-02-01	115456712	2/6/2013	5062	

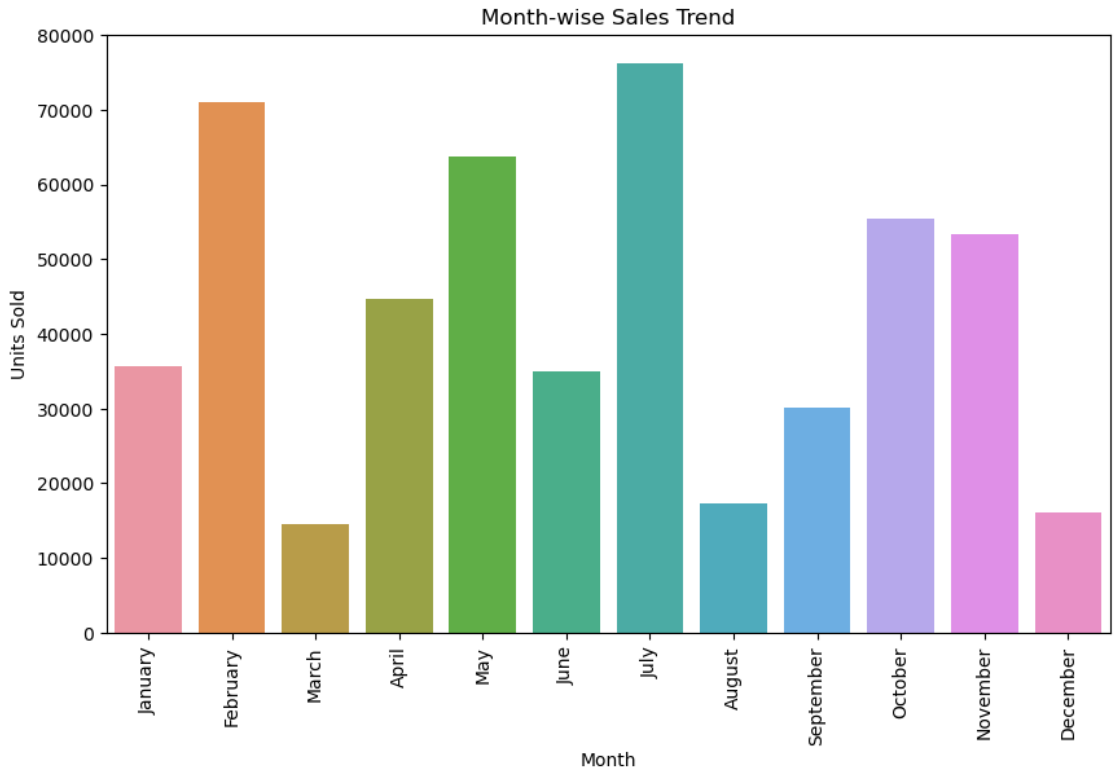
	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit	Year	\

```
0      255.28     159.42     2533654.00  1582243.50     951410.50  2010
1      205.70     117.11      576782.80   328376.44     248406.36  2012
2      651.21     524.96     1158502.59   933903.84     224598.75  2014
3        9.33       6.92       75591.66    56065.84      19525.82  2014
4      651.21     524.96     3296425.02  2657347.52     639077.50  2013

      Month
0       May
1    August
2       May
3      June
4  February
```

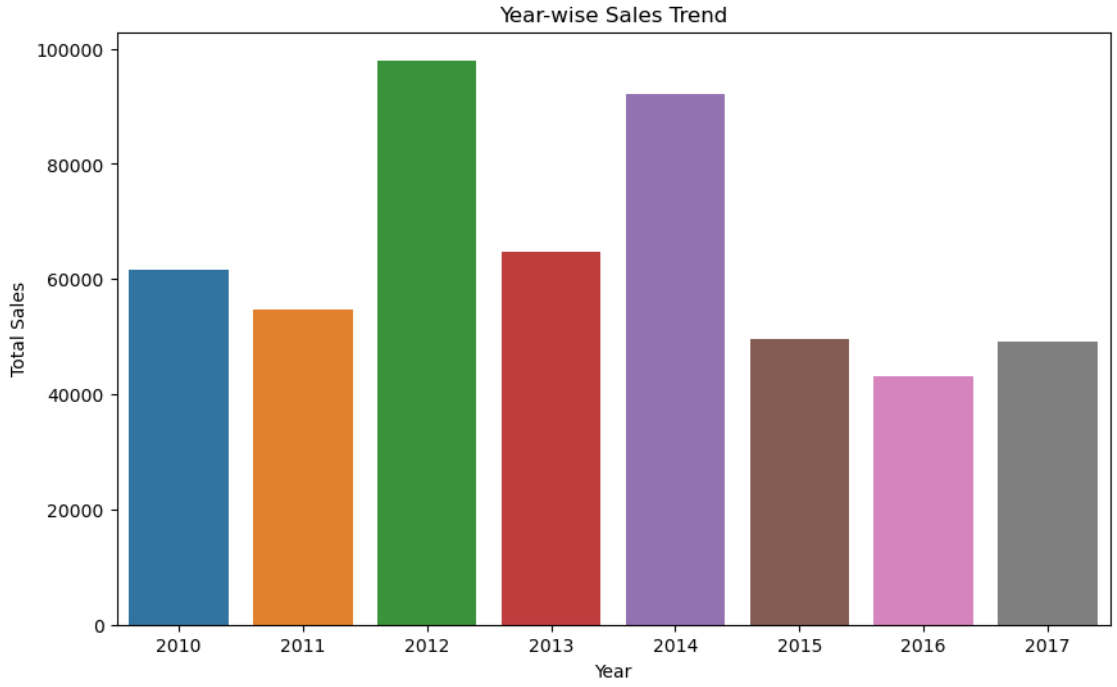
MONTH-WISE SALES TREND

[46]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
monthly_sales = df.groupby('Month',observed=True)['Units Sold'].sum().
 ↪reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='Month', y='Units Sold', data=monthly_sales, order=['January',␣
 ↪'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September',␣
 ↪'October', 'November', 'December'])
plt.title('Month-wise Sales Trend')
plt.xlabel('Month')
plt.ylabel('Units Sold')
plt.xticks(rotation=90)
plt.show()
```



YEAR-WISE SALES TREND

[22]:
```
yearly_sales = df.groupby('Year')['Units Sold'].sum().reset_index()
plt.figure(figsize=(10, 6))
sns.barplot(x='Year', y='Units Sold', data=yearly_sales)
plt.title('Year-wise Sales Trend')
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.show()
```



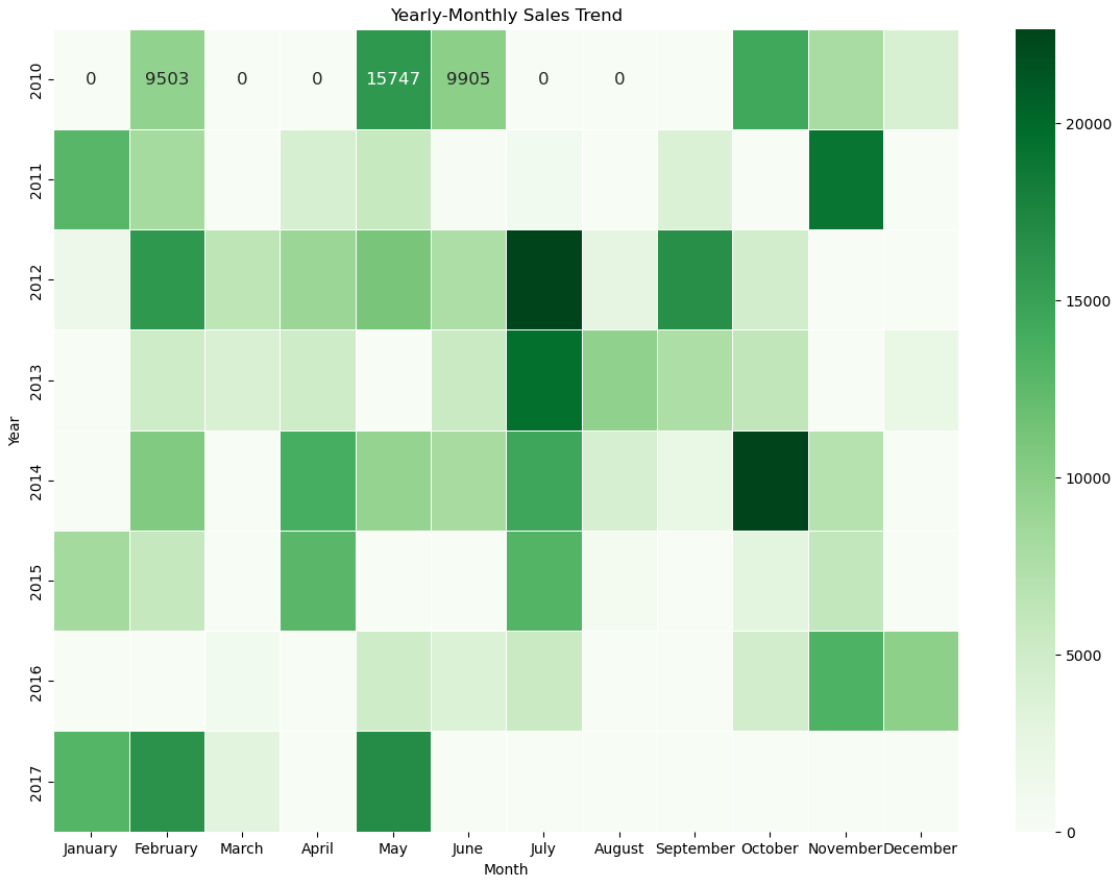
HEAT MAP-It shows taht which months has what level of sales in respective years

[26]:
```
correct_order = ['January', 'February', 'March', 'April', 'May', 'June',␣
 ↪'July', 'August', 'September', 'October', 'November', 'December']

yearly_monthly_sales = df.groupby(['Year', 'Month'], observed=False)['Units␣
 ↪Sold'].sum().unstack().fillna(0)

yearly_monthly_sales = yearly_monthly_sales.reindex(columns=correct_order)

plt.figure(figsize=(14, 10))  # Increased figure size
sns.heatmap(yearly_monthly_sales, cmap='Greens', linewidths=0.5, annot=True,␣
 ↪fmt="d", annot_kws={"size": 12})
plt.title('Yearly-Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Year')
plt.show()
```



[54]:
```
total_sales = df['Units Sold'].sum()
print("Total Sales:",total_sales, "units")
```

```
Total Sales: 512871 units
```

[56]:
```
aov = df['Units Sold'].mean()
print("Average Order Value:", aov,"units")
```

```
Average Order Value: 5128.71 units
```

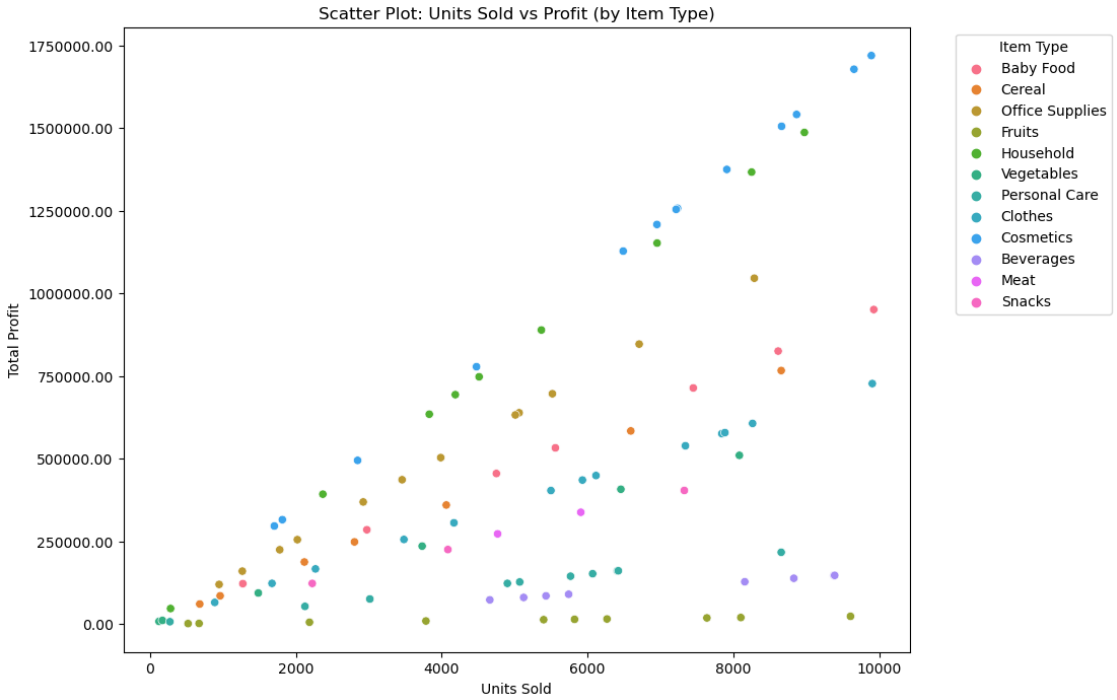
[57]:
```
top_products = df.groupby('Item Type')['Units Sold'].sum().
  ↪sort_values(ascending=False).head(10)
print("Top 10 Selling Products:\n", top_products)
```

```
Top 10 Selling Products:
 Item Type
Cosmetics          83718
Clothes            71260
Beverages          56708
Fruits             49998
Personal Care      48708
```

```
Office Supplies    46967
Household          44727
Baby Food          40545
Cereal             25877
Vegetables         20051
Name: Units Sold, dtype: int64
```

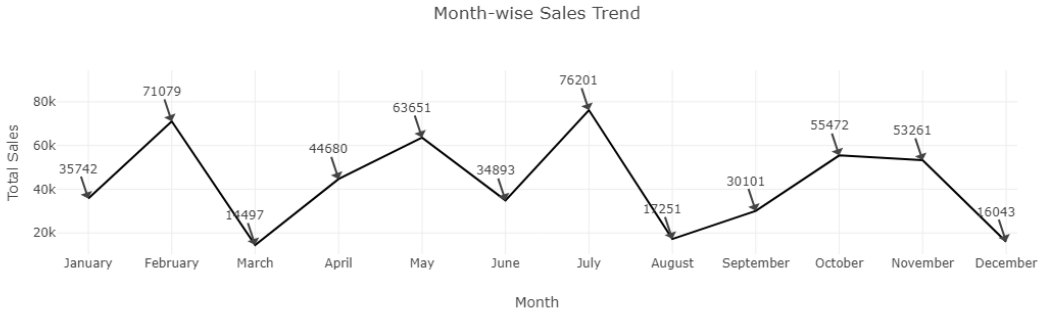
Relationship between various items sold with different profits

[31]:
```
from matplotlib.ticker import FuncFormatter
def format_y_ticks(value, _):
    return f'{value:.2f}'
plt.figure(figsize=(10, 8))  # Adjust the figure size as needed
sns.scatterplot(x='Units Sold', y='Total Profit', hue='Item Type', data=df)
plt.title('Scatter Plot: Units Sold vs Profit (by Item Type)')
plt.xlabel('Units Sold')
plt.ylabel('Total Profit')
plt.legend(title='Item Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.gca().yaxis.set_major_formatter(FuncFormatter(format_y_ticks))
plt.show()
```

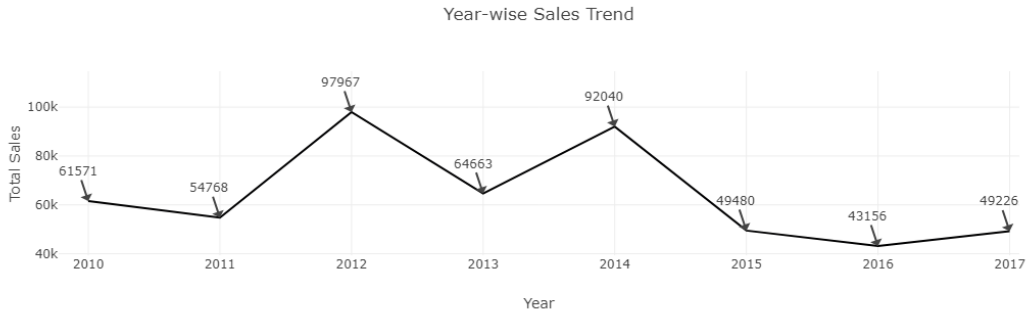


[126]:
```
import plotly.express as px
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August', 'September', 'October', 'November', 'December']
# Sort the dataframe by the order of months
```

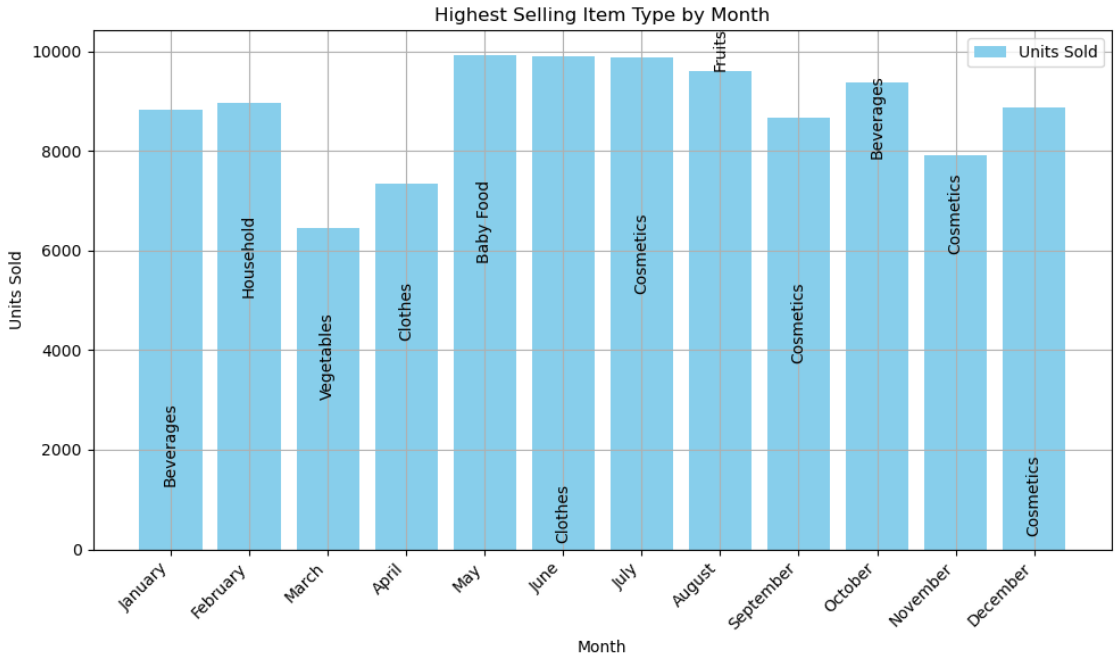
```
monthly_sales_sorted = monthly_sales.sort_values(by='Month', key=lambda x: pd.
 ↪Categorical(x, categories=month_order, ordered=True))
# Interactive line chart for month-wise sales trend
fig = px.line(monthly_sales_sorted, x='Month', y='Units Sold',␣
 ↪title='Month-wise Sales Trend')
fig.update_xaxes(title='Month', categoryorder='array',␣
 ↪categoryarray=month_order)  # Set the order of months explicitly
fig.update_yaxes(title='Total Sales')
# Add text annotations for total sales values
for index, row in monthly_sales_sorted.iterrows():
    fig.add_annotation(x=row['Month'], y=row['Units Sold'], text=f'{row["Units␣
 ↪Sold"]}', showarrow=True, arrowhead=1)
fig.show()
```



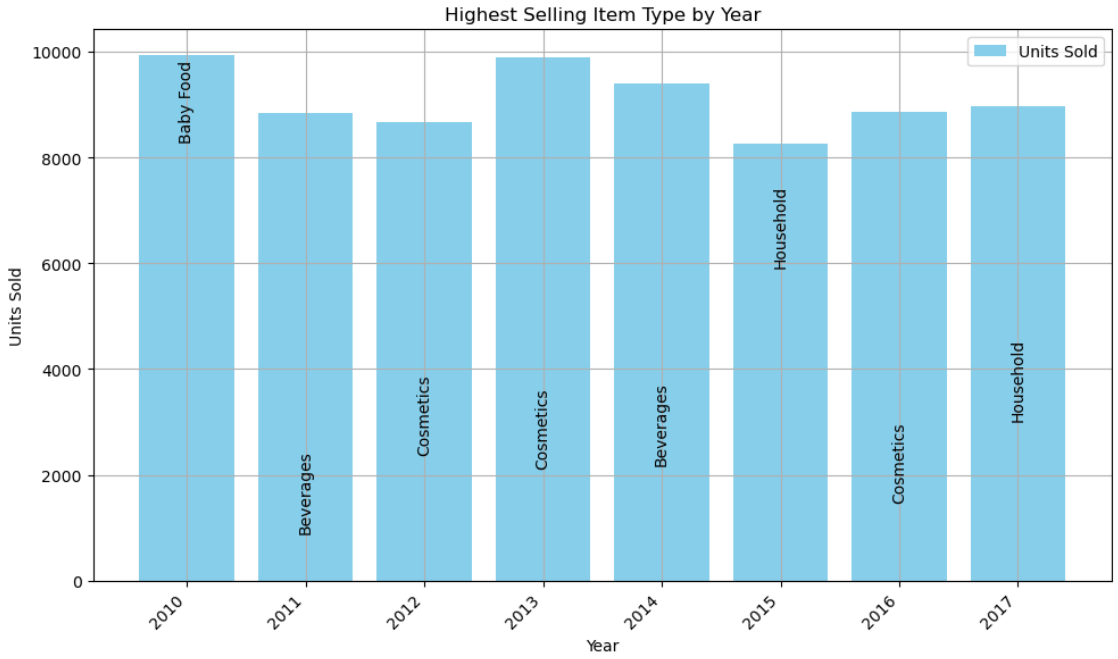
[144]:
```
year_order = [2019, 2020, 2021, 2022, 2023, 2024]
yearly_sales_sorted = yearly_sales.sort_values(by='Year', key=lambda x: pd.
 ↪Categorical(x, categories=year_order, ordered=True))
fig = px.line(yearly_sales_sorted, x='Year', y='Units Sold', title='Year-wise␣
 ↪Sales Trend')
fig.update_xaxes(title='Year', categoryorder='array', categoryarray=year_order)
for index, row in yearly_sales_sorted.iterrows():
    fig.add_annotation(x=row['Year'], y=row['Units Sold'], text=f'{row["Units␣
 ↪Sold"]}', showarrow=True, arrowhead=1)
fig.show()
```



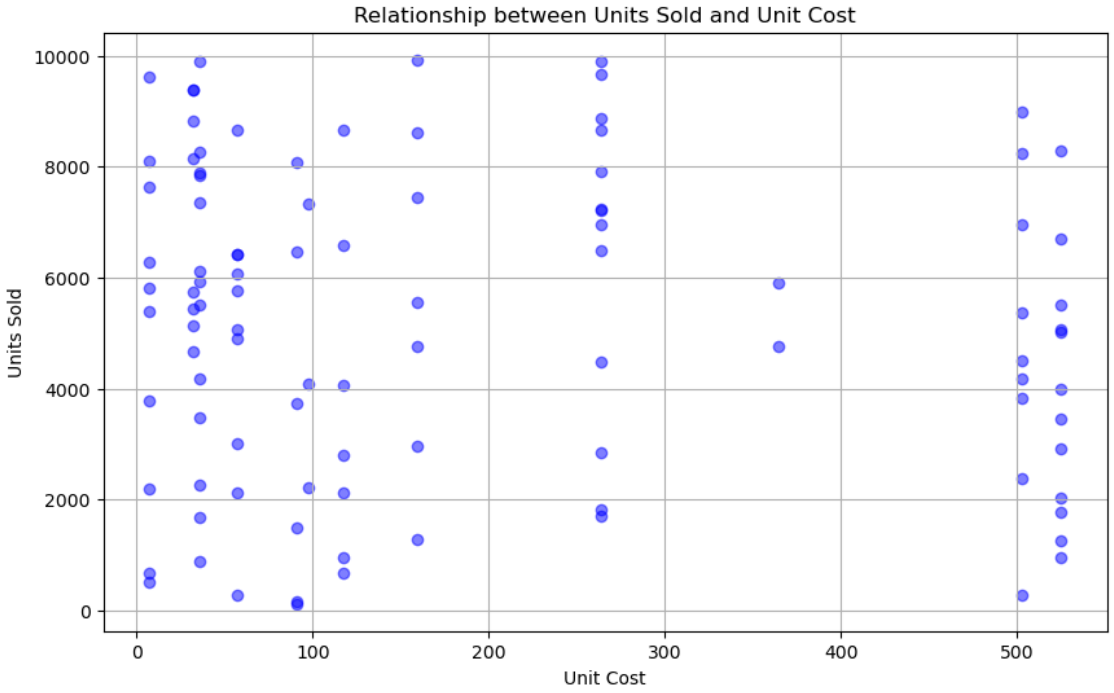
[123]:
```
highest_selling = df.groupby('Month', observed=True)['Units Sold'].idxmax()
highest_selling_items = df.loc[highest_selling, ['Month', 'Item Type']]
df_sorted = df.sort_values(by='Month')
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August', 'September', 'October', 'November', 'December']
df_sorted['Month'] = pd.Categorical(df_sorted['Month'], categories=month_order,␣
 ↪ordered=True)
# Plot the bar graph
plt.figure(figsize=(10, 6))
plt.bar(df_sorted['Month'], df_sorted['Units Sold'], color='skyblue',␣
 ↪label='Units Sold')
for i, (month, item) in enumerate(zip(highest_selling_items['Month'],␣
 ↪highest_selling_items['Item Type'])):
    plt.text(month, df_sorted[df_sorted['Month'] == month]['Units Sold'].
 ↪values[0], f'{item}', ha='center', va='bottom', rotation=90)
plt.title('Highest Selling Item Type by Month')
plt.xlabel('Month')
plt.ylabel('Units Sold')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



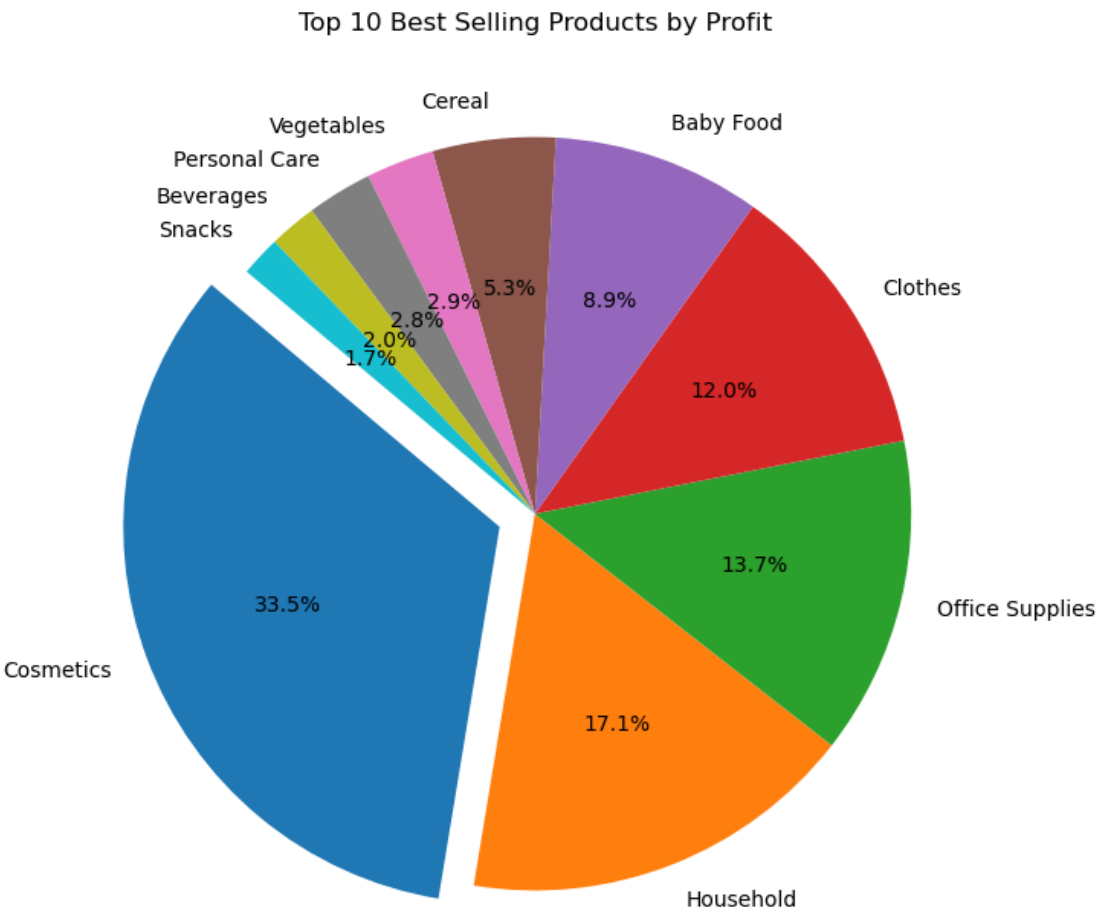
[149]:
```
highest_selling = df.groupby('Year')['Units Sold'].idxmax()
highest_selling_items = df.loc[highest_selling, ['Year', 'Item Type']]
df_sorted = df.sort_values(by='Year')
plt.figure(figsize=(10, 6))
plt.bar(df_sorted['Year'], df_sorted['Units Sold'], color='skyblue',␣
 ↪label='Units Sold')
for year, item in zip(highest_selling_items['Year'],␣
 ↪highest_selling_items['Item Type']):
    sold_units = df_sorted[df_sorted['Year'] == year]['Units Sold'].iloc[0]
    plt.text(year, sold_units, f'{item}', ha='center', va='bottom', rotation=90)
plt.title('Highest Selling Item Type by Year')
plt.xlabel('Year')
plt.ylabel('Units Sold')
plt.legend()
plt.grid(True)
plt.xticks(df_sorted['Year'].unique(), rotation=45, ha='right')  # Set x-axis␣
 ↪ticks to match the actual years
plt.tight_layout()
plt.show()
```



[138]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df['Unit Cost'], df['Units Sold'], color='blue', alpha=0.5)
plt.title('Relationship between Units Sold and Unit Cost')
plt.xlabel('Unit Cost')
plt.ylabel('Units Sold')
plt.grid(True)
plt.show()
```



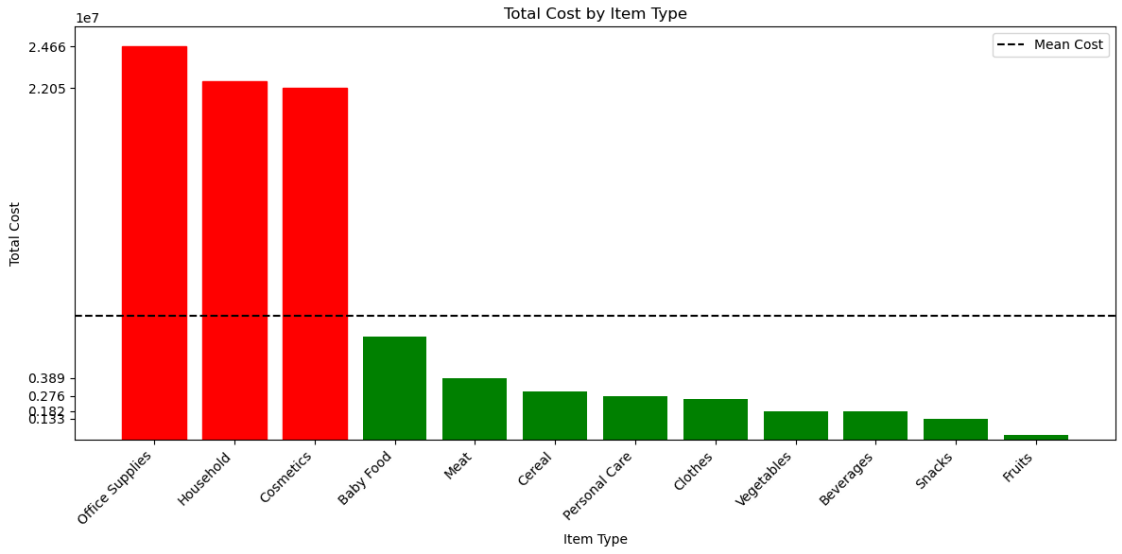
[143]:
```
product_profit = df.groupby('Item Type')['Total Profit'].sum()
top_10_products = product_profit.nlargest(10)
plt.figure(figsize=(10, 8))
explode = (0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
plt.pie(top_10_products, labels=top_10_products.index, autopct='%1.1f%%',␣
 ↪startangle=140, explode=explode)
plt.title('Top 10 Best Selling Products by Profit')
plt.show()
```



Analysis of items along with their cost

[42]:
```
# Aggregate total cost for each item type
df_agg = df.groupby('Item Type')['Total Cost'].sum().reset_index()
mean_cost = df_agg['Total Cost'].mean()
# Assign colors based on whether the total cost is above or below the mean
df_agg['Color'] = df_agg['Total Cost'].apply(lambda x: 'red' if x > mean_cost␣
 ↪else 'green')
df_agg_sorted = df_agg.sort_values(by='Total Cost', ascending=False)
# Define transformed or categorical values for y-axis
y_labels = df_agg_sorted['Total Cost']
plt.figure(figsize=(12, 6))
bars = plt.bar(df_agg_sorted['Item Type'], df_agg_sorted['Total Cost'],␣
 ↪color=df_agg_sorted['Color'])
# Draw mean cost line
plt.axhline(y=mean_cost, color='k', linestyle='--', label='Mean Cost')
```

```
# Color bars based on whether the cost is above or below the mean
for bar in bars:
    if bar.get_height() > mean_cost:
        bar.set_color('red')
plt.title('Total Cost by Item Type')
plt.xlabel('Item Type')
plt.ylabel('Total Cost')
plt.xticks(rotation=45, ha='right')
plt.yticks(y_labels[::2])  # Set y-ticks to reflect the actual total cost␣
↪values with increased spacing
plt.legend()
plt.tight_layout()
plt.show()
```



END OF SALES REPORT