Project Report

Bachelor Of Computer Applications

2 nd Semester

Exploratory Data Analysis Project

10,000 Sales Record Data Analysis

By

DIPIN ROKA

2411021240007

Githublink: https://github.com/DIPINROKA10/sds_ids_project_data_science/upload/main

Department Of Computer Application

Alliance University

Chandrapur a- Anekal Main Road,Anekal

Bengaluru – 56210

Introduction

The dataset used in this analysis is a sales dataset that contains various attributes related to product sales, including 'Units Sold', 'Unit Price', and other relevant features. The primary objective of this analysis is to explore and visualize the distribution of key variables, specifically 'Units Sold' and 'Unit Price'. By employing statistical visualizations such as histograms and boxplots, we aim to gain insights into the sales performance, identify trends, and detect any anomalies or outliers in the data. This understanding can inform business decisions, optimize pricing strategies, and enhance inventory management.

Objectives

1. Analyze Sales Distribution: To visualize and understand the distribution of 'Units Sold' and 'Unit Price' to identify patterns, trends, and central tendencies in the sales data.

2. Identify Outliers: To detect any outliers in the 'Units Sold' and 'Unit Price' data using boxplots, which can indicate unusual sales behavior or pricing strategies that may need further investigation.

3. Understand Variability: To assess the variability and spread of the data, helping to understand the range of sales performance and pricing strategies across different products.

4. Support Data-Driven Decisions: To provide insights that can inform business decisions related to inventory management, pricing strategies, and sales forecasting.

5. Enhance Reporting: To create visual representations of the data that can be used in reports and presentations, making it easier for stakeholders to grasp key insights quickly.

Libraries Used

1. **Pandas**: For data manipulation and analysis.
2. **Seaborn**: For creating visualizations like histograms and boxplots.
3. **Matplotlib**: For customizing and displaying plots.

These libraries facilitated effective analysis and visualization of the sales data.

Load the dataset

```
import pandas as pd import
seaborn as sns import
matplotlib.pyplot as plt
df=pd.read_csv(r"D:\IDS(Assignment)Dataset\100000 Sales Records.csv")
df
```

```
                                Region                Country  \
0                   Middle East and North Africa           Azerbaijan
1                   Central     America     and     the     Caribbean
                    Panama
2                   Sub-Saharan Africa  Sao Tome and Principe
3                   Sub-Saharan Africa  Sao Tome and Principe
4                   Central     America     and     the     Caribbean
                    Belize   ...                                  ...
                    ...
99995                           Sub-Saharan                    Africa
                                Niger
99996                           Europe                  Poland
99997                           Sub-Saharan                    Africa
                                Comoros
99998                           Middle   East   and   North   Africa
                                Kuwait
99999                           Sub-Saharan                    Africa
                                Tanzania

           Item Type Sales Channel Order Priority Order Date   Order
ID  \
0             Snacks        Online              C  10/8/2014
535113847
1           Cosmetics       Offline             L  2/22/2015
874708545
2             Fruits       Offline              M  12/9/2015
854349935
3          Personal Care        Online             M  9/17/2014
892836844
4          Household       Offline              H   2/4/2010
129280602   ...               ...           ...            ...        ...
..
.
99995          Cereal       Offline             L  8/26/2012
836322486
```

```
99996           Meat       Offline              C  12/3/2013
110449349
99997           Clothes        Online               M   8/7/2013
193128764
99998           Cosmetics        Online               L  6/28/2011
701597058
99999           Cosmetics       Offline              C   4/3/2012
423403060

        Ship Date  Units Sold  Unit Price  Unit Cost  Total Revenue  \
0       10/23/2014         934      152.58      97.44      142509.72
1       2/27/2015         4551      437.20     263.33     1989697.20
2       1/18/2016         9986        9.33       6.92       93169.38
3       10/12/2014         9118       81.73      56.67      745214.14
4        3/5/2010         5858      668.27     502.54     3914725.66
...           ...          ...         ...        ...             ...
99995    9/11/2012         5263      205.70     117.11
         1082599.10
99996    12/10/2013         3272      421.89     364.69
         1380424.08
99997    8/31/2013         9948      109.28      35.84
         1087117.44
99998    7/3/2011         7015      437.20     263.33     3066958.00
99999    4/30/2012         3229      437.20     263.33
         1411718.80
       Total Cost  Total Profit
0     91008.96       51500.76
1     1198414.83      791282.37
2     69103.12       24066.26
3     516717.06      228497.08
4     2943879.32      970846.34
...           ...            ...
99995   616349.93      466249.17
99996   1193265.68
        187158.40
99997   356536.32      730581.12
99998   1847259.95
        1219698.05
99999   850292.57      561426.23
[100000 rows x 14 columns]
```

it is a Pandas function in Python that displays the first 20 rows of a DataFrame named df.

```
df.head(20)

                              Region                 Country  \
0                               Middle   East   and   North   Africa
                                Azerbaijan
```

1	Central America and the Caribbean	Panama
2	Sub-Saharan Africa	Sao Tome and Principe
3	Sub-Saharan Africa	Sao Tome and Principe
4	Central America and the Caribbean	Belize
5	Europe	Denmark
6	Europe	Germany
7	Middle East and North Africa	Turkey
8	Europe	United Kingdom
9	Asia	Kazakhstan
10	Central America and the Caribbean	Haiti
11	Europe	Italy
12	Europe	Malta
13	Middle East and North Africa	Jordan
14	Asia	Cambodia
15	Central America and the Caribbean	Saint Kitts and Nevis
16	Sub-Saharan Africa	Cameroon
17	Middle East and North Africa	Bahrain
18	Australia and Oceania	Solomon Islands
19	Europe	Monaco

```
2     1/18/2016         9986         9.33        6.92        93169.38
69103.12
3     10/12/2014         9118        81.73       56.67       745214.14
516717.06
4     3/5/2010          5858       668.27      502.54      3914725.66
2943879.32
5     2/28/2013          1149       109.28       35.84       125562.72
41180.16
6     5/3/2013          7964       437.20      263.33      3481860.80
2097160.12
7     4/7/2012          6307         9.33        6.92        58844.31
43644.44
8     1/15/2013          8217       152.58       97.44      1253749.86
800664.48
9     9/18/2015          2758       437.20      263.33      1205797.60
726264.14
10    1/1/2014          1031       437.20      263.33       450753.20
271493.23
11    1/10/2014          1165       109.28       35.84       127311.20
41753.60
12    4/17/2015          3322       668.27      502.54      2219992.94
1669437.88
13    7/18/2014          4693       668.27      502.54      3136191.11
2358420.22
14    6/29/2017          4502       154.06       90.93       693578.12
409366.86
15    8/29/2011          9004       651.21      524.96      5863494.84
4726739.84
16    2/6/2016          6486         9.33        6.92        60514.38
44883.12
17    7/19/2016          2264       154.06       90.93       348791.84
205865.52
18    5/30/2015          3688        47.45       31.79       174995.60
117241.52
19    1/17/2012          5137       651.21      524.96      3345265.77
2696719.52

    Total Profit
0       51500.76
1       791282.37
2       24066.26
3       228497.08
4       970846.34
5       84382.56
6       1384700.68
7       15199.87
8       453085.38
9       479533.46
```

10 179259.97

```
11      85557.60
12      550555.06
13      777770.89
14      284211.26
15      1136755.00
16      15631.26
17      142926.32
18      57754.08
19      648546.25
```

it is a Pandas function in Python that displays the last 20 rows of a DataFrame.

```
df.tail(20)
```

```
                                Region                    Country  \
99980                           Europe                    Croatia
99981                   Sub-Saharan Africa     Sao Tome and
                                Principe
99982                             Asia                   Cambodia
99983                    North America                     Mexico
99984                             Asia               North Korea
99985          Central America and the Caribbean
                                Antigua and Barbuda
99986                  Australia       and       Oceania
                                New Zealand
99987          Central America and the Caribbean
                                Barbados
99988                    North America   United States of
                                America
99989                           Europe                  Slovenia
99990                   Sub-Saharan                     Africa
                                Sierra Leone
99991                           Europe            United Kingdom
99992                  Middle   East   and   North   Africa
                                Morocco
99993                             Asia                  Maldives
99994                             Asia                   Vietnam
99995                   Sub-Saharan                     Africa
                                Niger
99996                           Europe                    Poland
99997                   Sub-Saharan                     Africa
                                Comoros
99998                  Middle   East   and   North   Africa
                                Kuwait
99999                   Sub-Saharan                     Africa
                                Tanzania
```

	Item Type	Sales Channel	Order Priority	Order Date	Order ID
99980	Fruits	Online	M	10/15/2015	986272561
99981	Personal Care	Online	L	8/28/2013	587970530
99982	Fruits	Online	L	10/23/2011	777176129
99983	Cereal	Offline	L	1/7/2013	116438186
99984	Fruits	Online	C	7/17/2013	388618028
99985	Office Supplies	Online	L	7/1/2014	746889483
99986	Snacks	Online	H	6/26/2010	864738881
99987	Clothes	Offline	L	2/9/2016	485965049
99988	Vegetables	Online	L	9/29/2016	774679315
99989	Clothes	Offline	M	3/12/2010	265305507
99990	Baby Food	Online	M	6/18/2010	712579953
99991	Cereal	Offline	C	9/12/2013	341551482
99992	Household	Online	L	2/9/2017	597581422
99993	Snacks	Offline	H	3/7/2015	296343600
99994	Baby Food	Offline	H	1/13/2011	573824346
99995	Cereal	Offline	L	8/26/2012	836322486
99996	Meat	Offline	C	12/3/2013	110449349
99997	Clothes	Online	M	8/7/2013	193128764
99998	Cosmetics	Online	L	6/28/2011	701597058
99999	Cosmetics	Offline	C	4/3/2012	423403060

	Ship Date	Units Sold	Unit Price	Unit Cost	Total Revenue
99980	10/19/2015	8823	9.33	6.92	82318.59
99981	10/6/2013	2401	81.73	56.67	196233.73
99982	11/2/2011	2542	9.33	6.92	23716.86

99983	2/10/2013	5502	205.70	117.11	1131761.40
99984	8/24/2013	9565	9.33	6.92	89241.45
99985	8/6/2014	51	651.21	524.96	33211.71
99986	7/6/2010	4306	152.58	97.44	657009.48
99987	3/7/2016	6712	109.28	35.84	733487.36
99988	10/17/2016	3679	154.06	90.93	566786.74
99989	3/14/2010	8650	109.28	35.84	945272.00
99990	6/30/2010	7098	255.28	159.42	1811977.44
99991	10/1/2013	1528	205.70	117.11	314309.60
99992	2/14/2017	6477	668.27	502.54	4328384.79
99993	4/22/2015	3512	152.58	97.44	535860.96
99994	2/7/2011	6230	255.28	159.42	1590394.40
99995	9/11/2012	5263	205.70	117.11	1082599.10
99996	12/10/2013	3272	421.89	364.69	1380424.08
99997	8/31/2013	9948	109.28	35.84	1087117.44
99998	7/3/2011	7015	437.20	263.33	3066958.00
99999	4/30/2012	3229	437.20	263.33	1411718.80

	Total Cost	Total Profit
99980	61055.16	21263.43
99981	136064.67	
	60169.06	
99982	17590.64	6126.22
99983	644339.22	
	487422.18	
99984	66189.80	23051.65
99985	26772.96	6438.75
99986	419576.64	
	237432.84	
99987	240558.08	
	492929.28	
99988	334531.47	
	232255.27	
99989	310016.00	
	635256.00	
99990	1131563.16	
	680414.28	
99991	178944.08	
	135365.52	
99992	3254951.58	
	1073433.21	
99993	342209.28	
	193651.68	
99994	993186.60	
	597207.80	

```
99995    616349.93
         466249.17
99996    1193265.68
         187158.40
99997    356536.32
         730581.12
99998    1847259.95
         1219698.05
99999    850292.57
         561426.23
```

df.describe() is a Pandas method that provides a quick summary of the statistical properties of the numerical columns in a DataFrame. which includes mean,median,std,count and much more

```
df.describe()
           Order ID     Units Sold     Unit Price      Unit Cost  \
count  1.000000e+05  100000.000000  100000.000000  100000.000000
mean   5.503956e+08    5001.446170     266.703989     188.019711
std    2.593219e+08    2884.575424     216.940081     175.706023
min    1.000089e+08       1.000000       9.330000       6.920000
25%    3.260464e+08    2505.000000     109.280000      56.670000
50%    5.477185e+08    5007.000000     205.700000     117.110000
75%    7.750785e+08    7495.250000     437.200000     364.690000
max    9.999965e+08   10000.000000     668.270000     524.960000

       Total Revenue    Total Cost  Total Profit
count   1.000000e+05  1.000000e+05  1.000000e+05
mean    1.336067e+06  9.419755e+05  3.940912e+05
std     1.471768e+06  1.151828e+06  3.795986e+05
min     1.866000e+01  1.384000e+01  4.820000e+00
25%     2.797533e+05  1.629283e+05  9.590000e+04
50%     7.898916e+05  4.679374e+05  2.836575e+05
```

```
75%     1.836490e+06  1.209475e+06  5.683841e+05  max
6.682700e+06  5.249075e+06  1.738700e+06
```

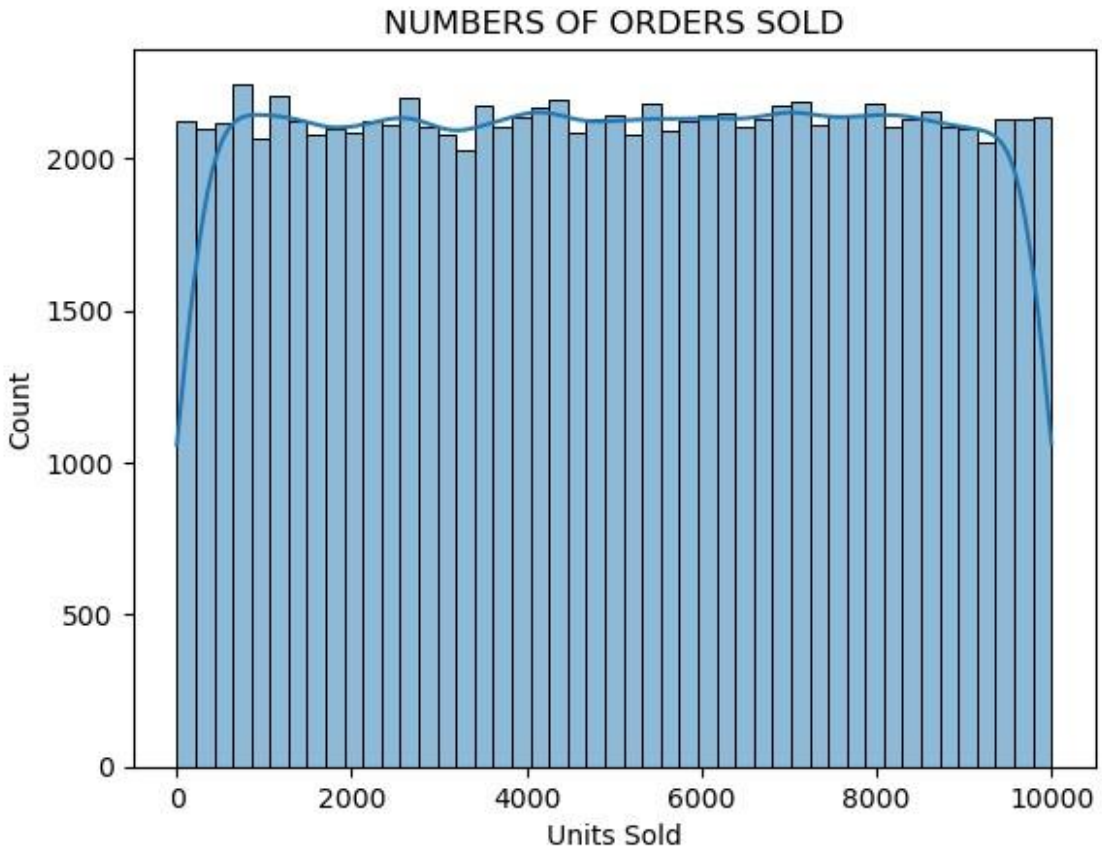
it is a Pandas method that provides a concise summary of a DataFrame's structure.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999 Data
columns (total 14 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
0   Region          100000 non-null  object
1   Country         100000 non-null  object
2   Item Type       100000 non-null  object
3   Sales Channel   100000 non-null  object
4   Order Priority  100000 non-null  object
5   Order Date      100000 non-null  object
6   Order ID        100000 non-null  int64
7   Ship Date       100000 non-null  object
8   Units Sold      100000 non-null  int64
9   Unit Price      100000 non-null  float64
10  Unit Cost       100000 non-null  float64
11  Total Revenue   100000 non-null  float64
12  Total Cost      100000 non-null  float64 13  Total Profit    100000
    non-null   float64 dtypes: float64(5), int64(2), object(7) memory
    usage: 10.7+ MB
```

This line of code creates a histogram using the Units Sold column from the dataset. It uses Seaborn's histplot() function to display the frequency of units sold. The kde=True option overlays a smooth curve to show the data distribution. .set_title() adds a custom title to the chart for clarity. This plot helps visualize how many orders fall within different quantity ranges.

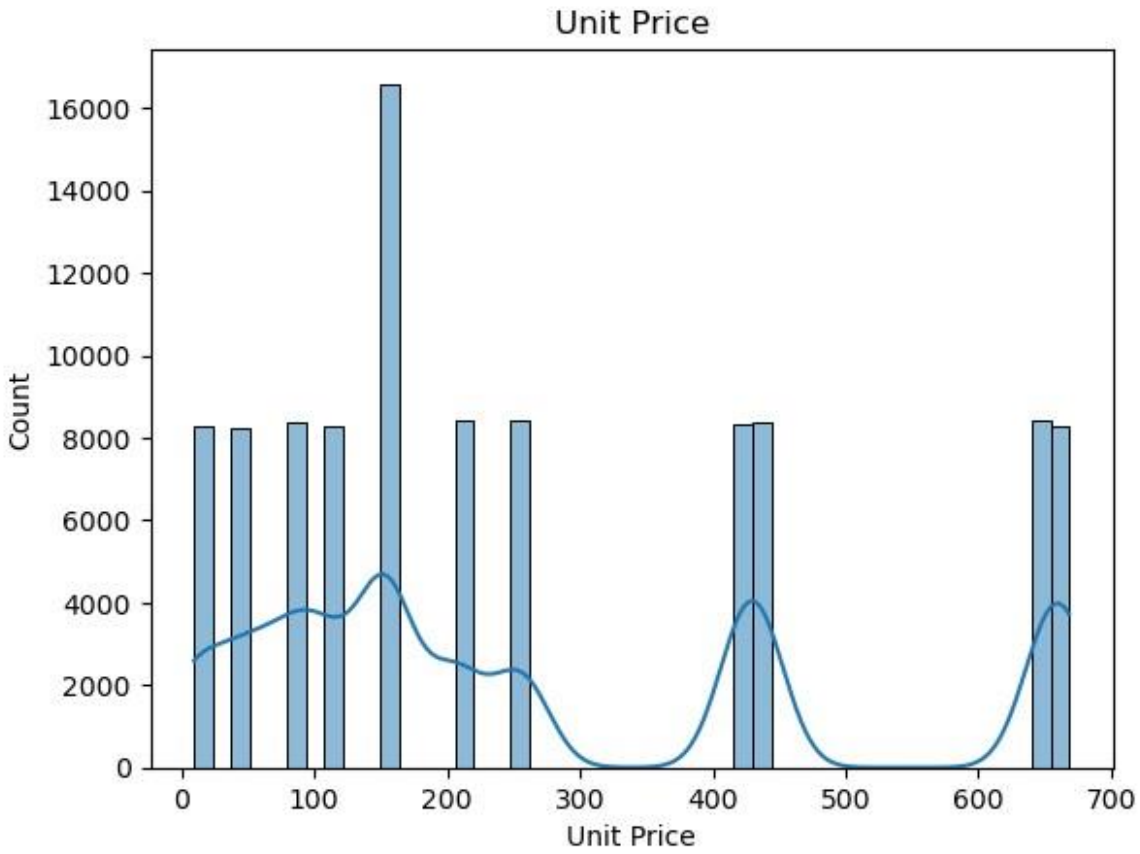
```
sns.histplot(df["Units Sold"],kde=True).set_title("NUMBERS OF ORDERS
SOLD")
```

```
Text(0.5, 1.0, 'NUMBERS OF ORDERS SOLD')
```



This code performs univariate analysis on the Unit Price column. It uses sns.histplot() to plot a histogram showing the distribution of unit prices. The kde=True argument adds a smooth density curve over the bars. .set_title('Unit Price') sets the title of the plot. plt.show() displays the final visualization

```
# Univariate Analysis
sns.histplot(df['Unit Price'], kde=True).set_title('Unit Price')
plt.show()
```

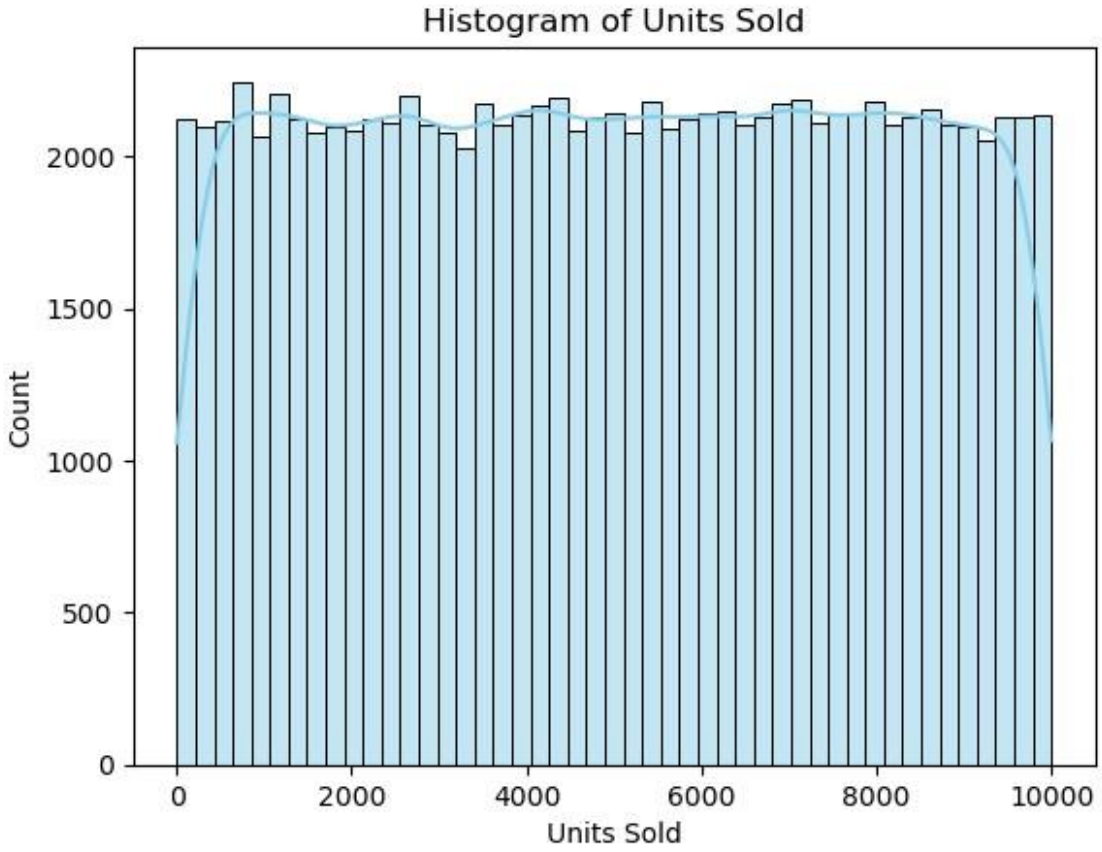


This code calculates the correlation matrix for selected numerical columns in the dataset.

```
# Correlation matrix
correlation = df[['Units Sold', 'Unit Price', 'Unit Cost', 'Total
Revenue', 'Total Profit']].corr() print("Correlation Matrix:\n",
correlation)

Correlation Matrix:
                Units Sold  Unit Price  Unit Cost  Total Revenue
Total Profit
Units Sold        1.000000    0.003453   0.003167       0.525322
0.601624
Unit Price        0.003453    1.000000   0.986030       0.739258
0.577093
Unit Cost         0.003167    0.986030   1.000000       0.729055
0.504763
Total Revenue     0.525322    0.739258   0.729055       1.000000
0.880186
Total Profit      0.601624    0.577093   0.504763       0.880186
1.000000
```

The code snippet you provided for creating a histogram using Seaborn's histplot

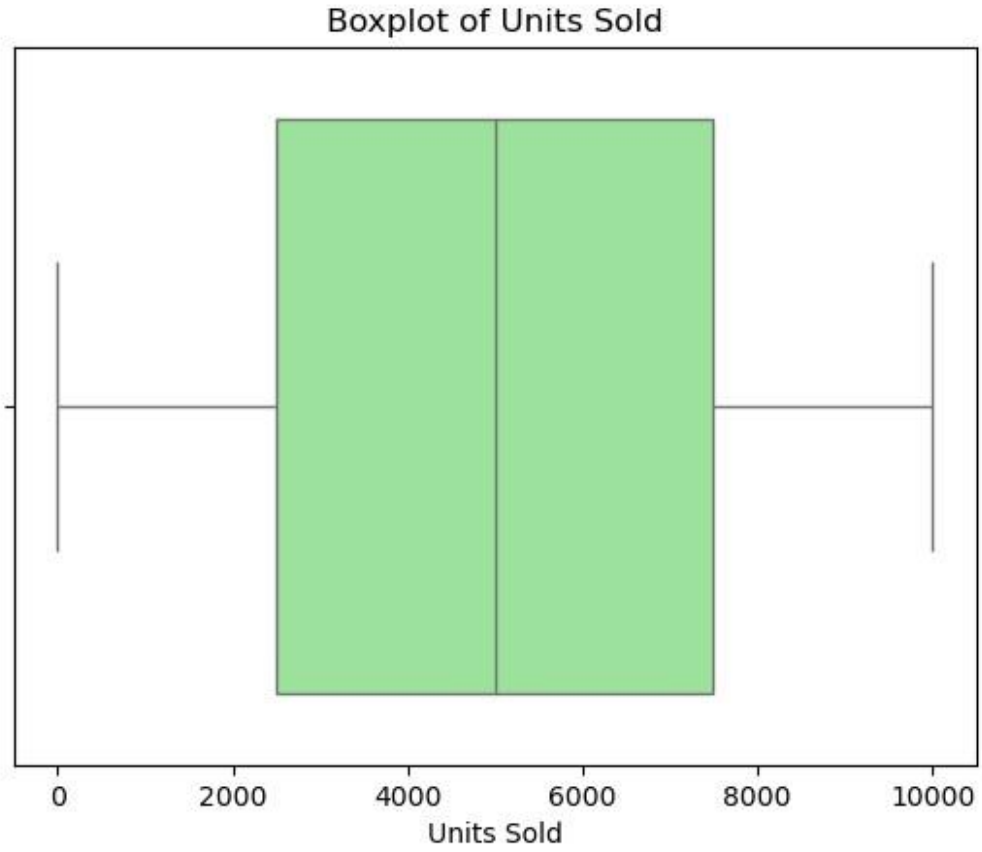
```
# Histogram for Units Sold
sns.histplot(df['Units Sold'], kde=True,
color='skyblue').set_title('Histogram of Units Sold')

Text(0.5, 1.0, 'Histogram of Units Sold')
```



The provided code creates a boxplot for the 'Units Sold' variable using Seaborn, which visually summarizes the distribution of the data. A boxplot displays the median, quartiles, and potential outliers, providing insights into the central tendency and variability of the data

```
# Boxplot for Units Sold
sns.boxplot(x=df['Units Sold'], color='lightgreen').set_title('Boxplot
of Units Sold')
```

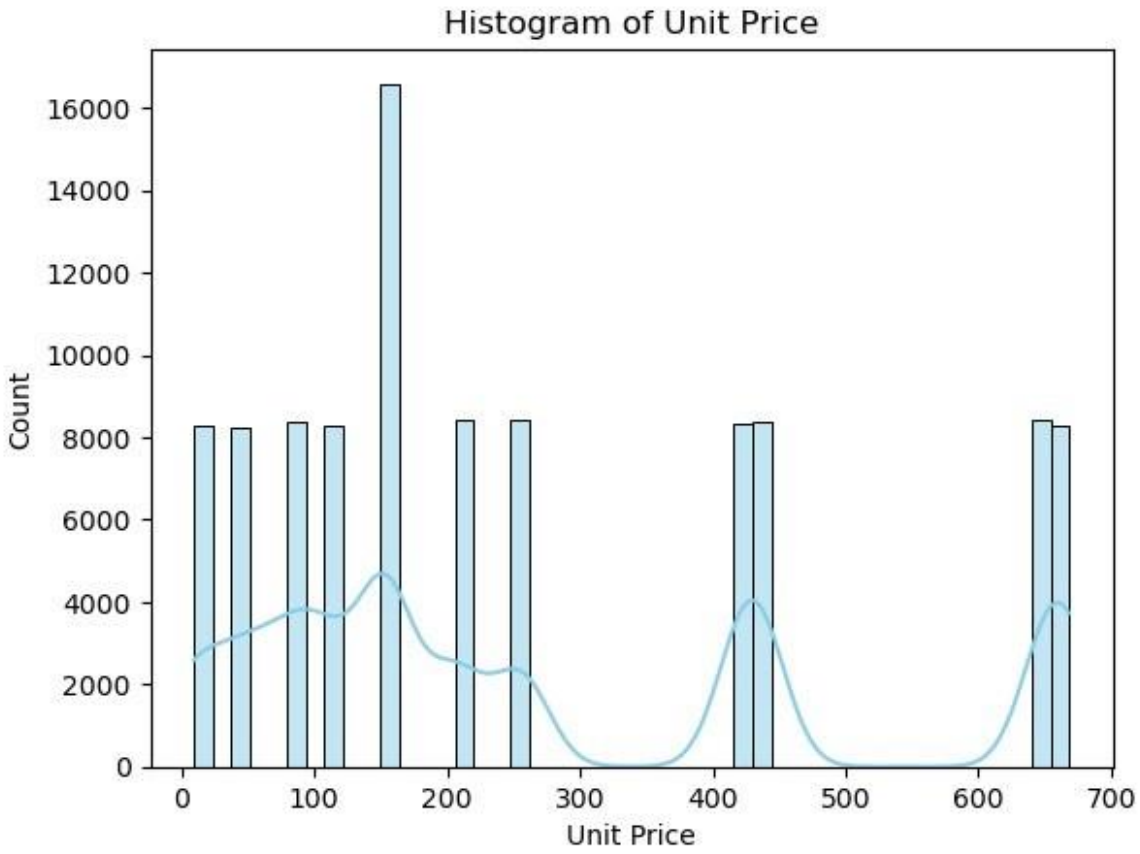
```
Text(0.5, 1.0, 'Boxplot of Units Sold')
```



The provided code creates a boxplot for the 'Units Sold' variable using Seaborn, which visually summarizes the distribution of the data

```
# Histogram for Unit Price
sns.histplot(df['Unit Price'], kde=True,
color='skyblue').set_title('Histogram of Unit Price')
```

```
Text(0.5, 1.0, 'Histogram of Unit Price')
```



The code creates a boxplot to display the distribution of values in the "Unit Price" column.

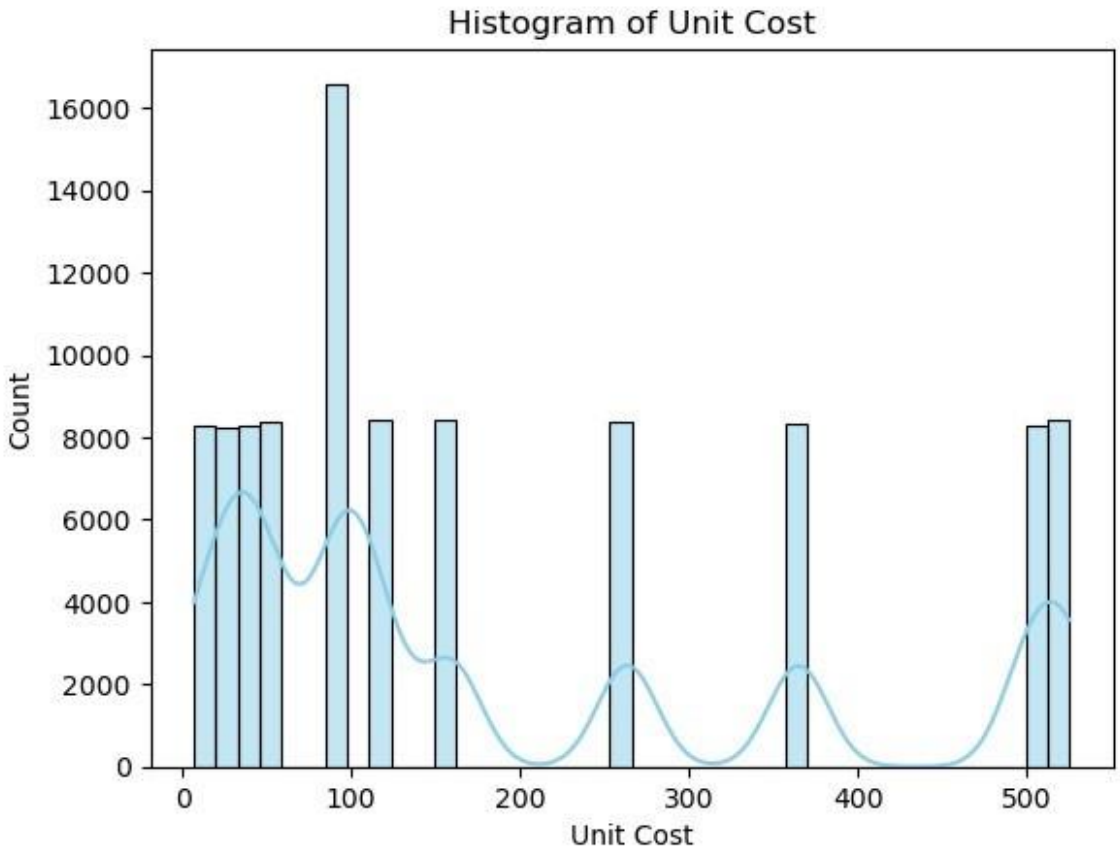
```
# Boxplot for Unit Price
sns.boxplot(x=df['Unit Price'], color='lightgreen').set_title('Boxplot
of Unit Price')
```

```
Text(0.5, 1.0, 'Boxplot of Unit Price')
```

The code creates a histogram to visualize the distribution of values in the "Unit Cost" column.

```
# Histogram for Unit Cost
sns.histplot(df['Unit Cost'], kde=True,
color='skyblue').set_title('Histogram of Unit Cost')
```

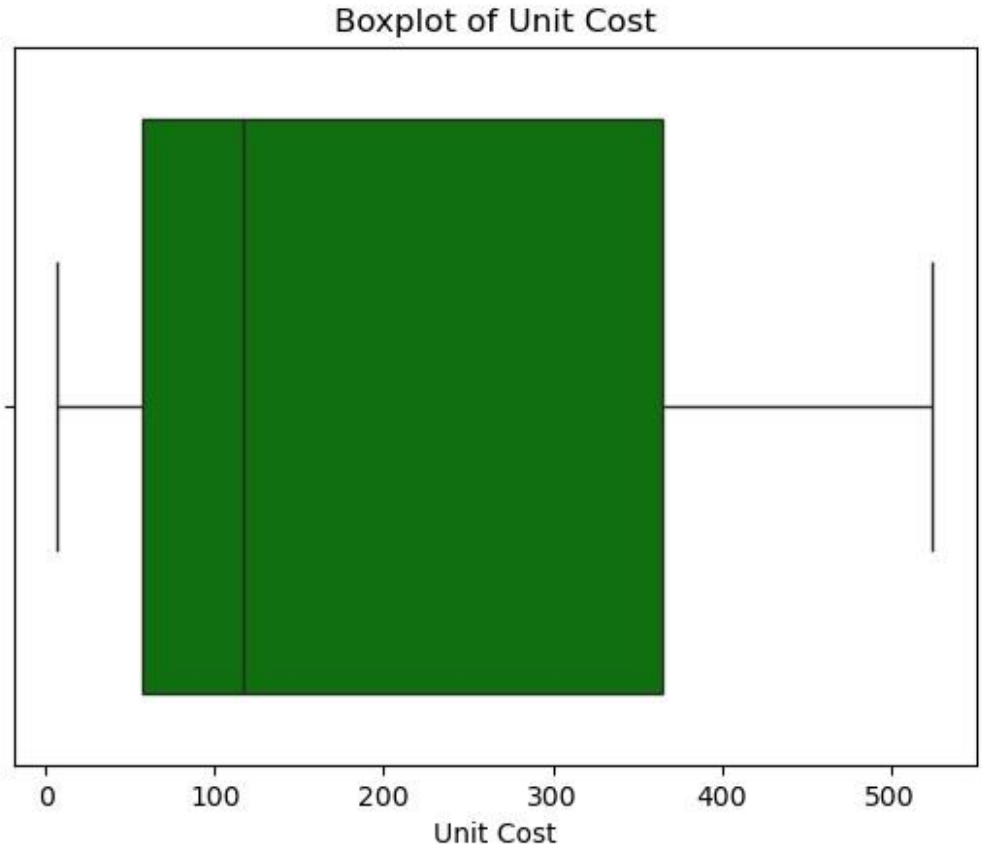
```
Text(0.5, 1.0, 'Histogram of Unit Cost')
```



he code creates a boxplot to visualize the distribution of the "Unit Cost" values in the DataFrame. The sns.boxplot() function displays key statistics such as the median, quartiles, and potential outliers.

```
# Boxplot for Unit Cost
sns.boxplot(x=df['Unit Cost'],  color='green').set_title('Boxplot of
Unit Cost')
```

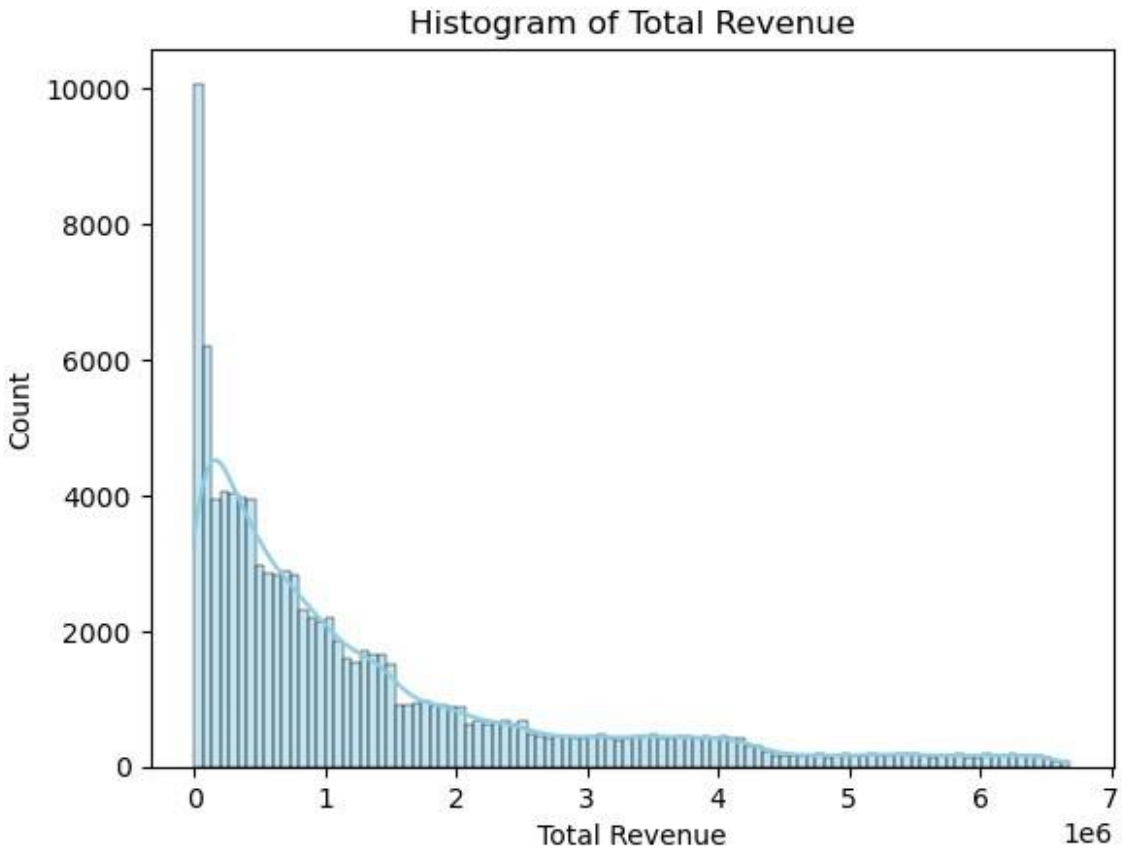
```
Text(0.5, 1.0, 'Boxplot of Unit Cost')
```



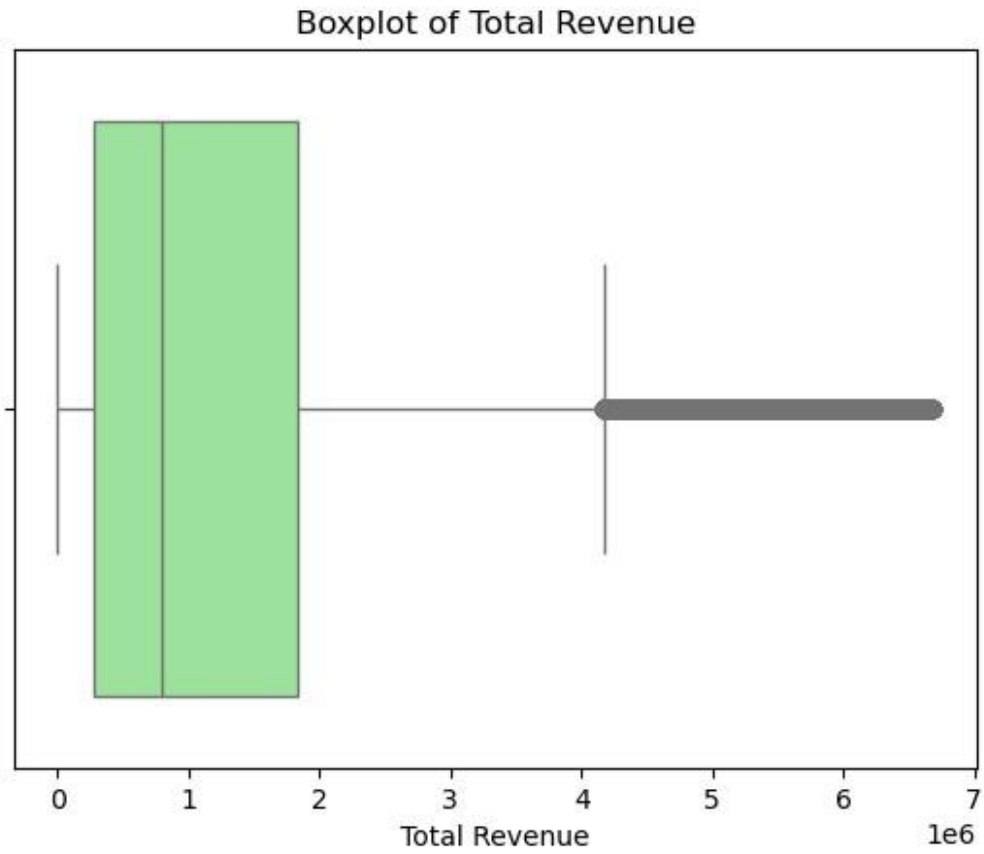
The code generates a histogram to display the distribution of values in the "Total Revenue" column. The sns.histplot() function plots the frequency of different revenue values. A kernel density estimate (KDE) curve is included to provide a smooth representation of the data distribution.

```
# Histogram for Total Revenue
sns.histplot(df['Total Revenue'], kde=True,
color='skyblue').set_title('Histogram of Total Revenue')
```

```
Text(0.5, 1.0, 'Histogram of Total Revenue')
```

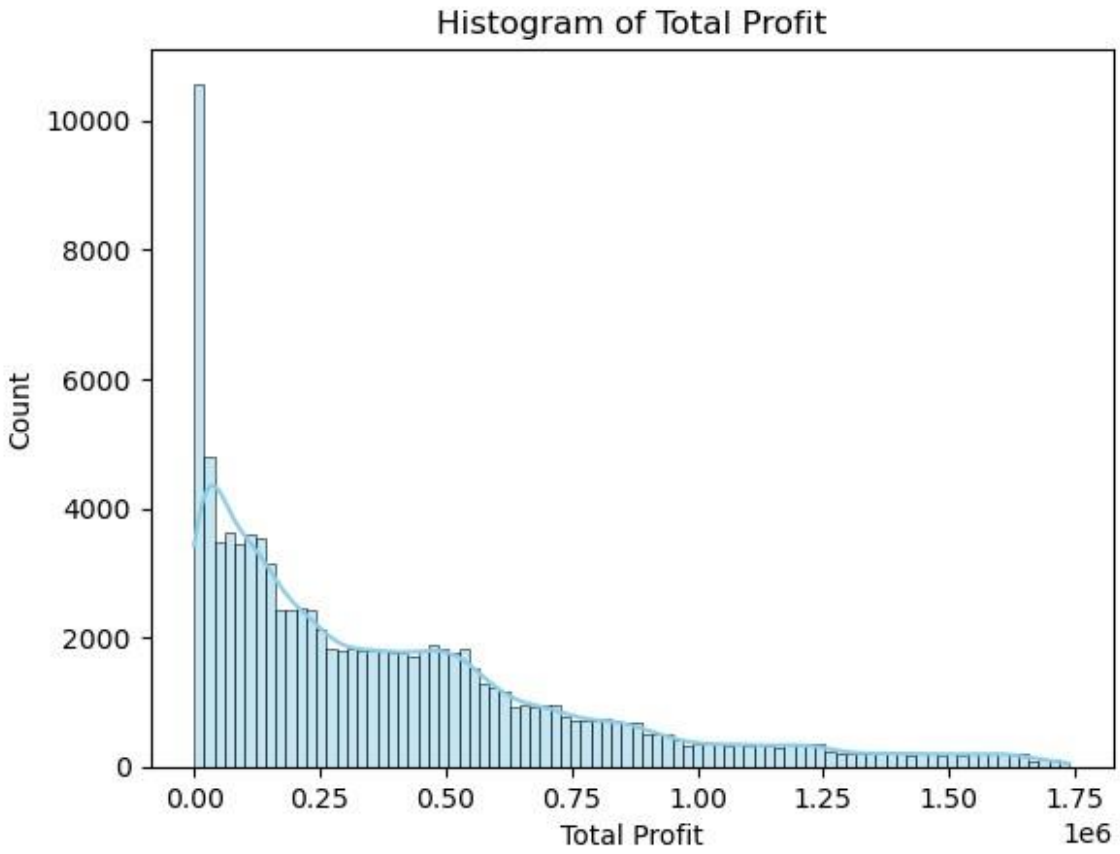



```
# Boxplot for Total Revenue
sns.boxplot(x=df['Total Revenue'],
color='lightgreen').set_title('Boxplot of Total Revenue')

Text(0.5, 1.0, 'Boxplot of Total Revenue')
```

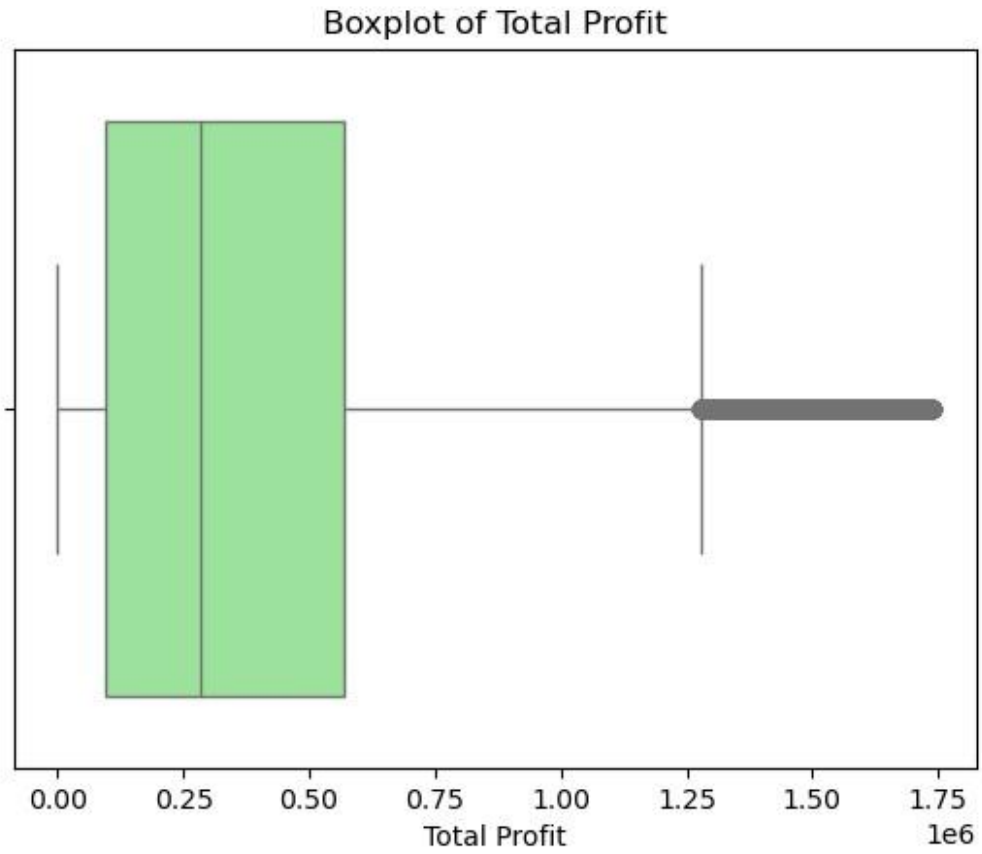



```
# Histogram for Total Profit sns.histplot(df['Total Profit'],
kde=True,  color='skyblue').set_title('Histogram of Total
Profit')
```

```
Text(0.5, 1.0, 'Histogram of Total Profit')
```




```
# Boxplot for Total Profit
sns.boxplot(x=df['Total Profit'],
color='lightgreen').set_title('Boxplot of Total Profit')

Text(0.5, 1.0, 'Boxplot of Total Profit')
```




```
# Adjust layout
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

The code you've written performs several steps:

1 Loading the Dataset: The first line loads a dataset from a CSV file located at D:\ IDS(Assignment)Dataset\100000 Sales Records.csv into a DataFrame (df) using pd.read_csv().

2 Displaying Data Types: df.dtypes is used to print the data types of each column in the dataset, helping to understand what type of data (numeric, object, etc.) each column contains.

3 Selecting Numeric Columns: df.select_dtypes(include=['number']) selects only the numeric columns from the DataFrame (df). This is useful because correlation calculations can only be performed on numerical data.

4 Calculating Correlation Matrix: The numeric_df.corr() method calculates the correlation matrix, which measures the relationship between numeric variables in the dataset. Values range from -1 (perfect negative correlation) to 1 (perfect positive correlation).

5 Displaying the Correlation Matrix: Finally, the code prints out the correlation matrix, which helps identify how closely different numeric variables are related to each other.

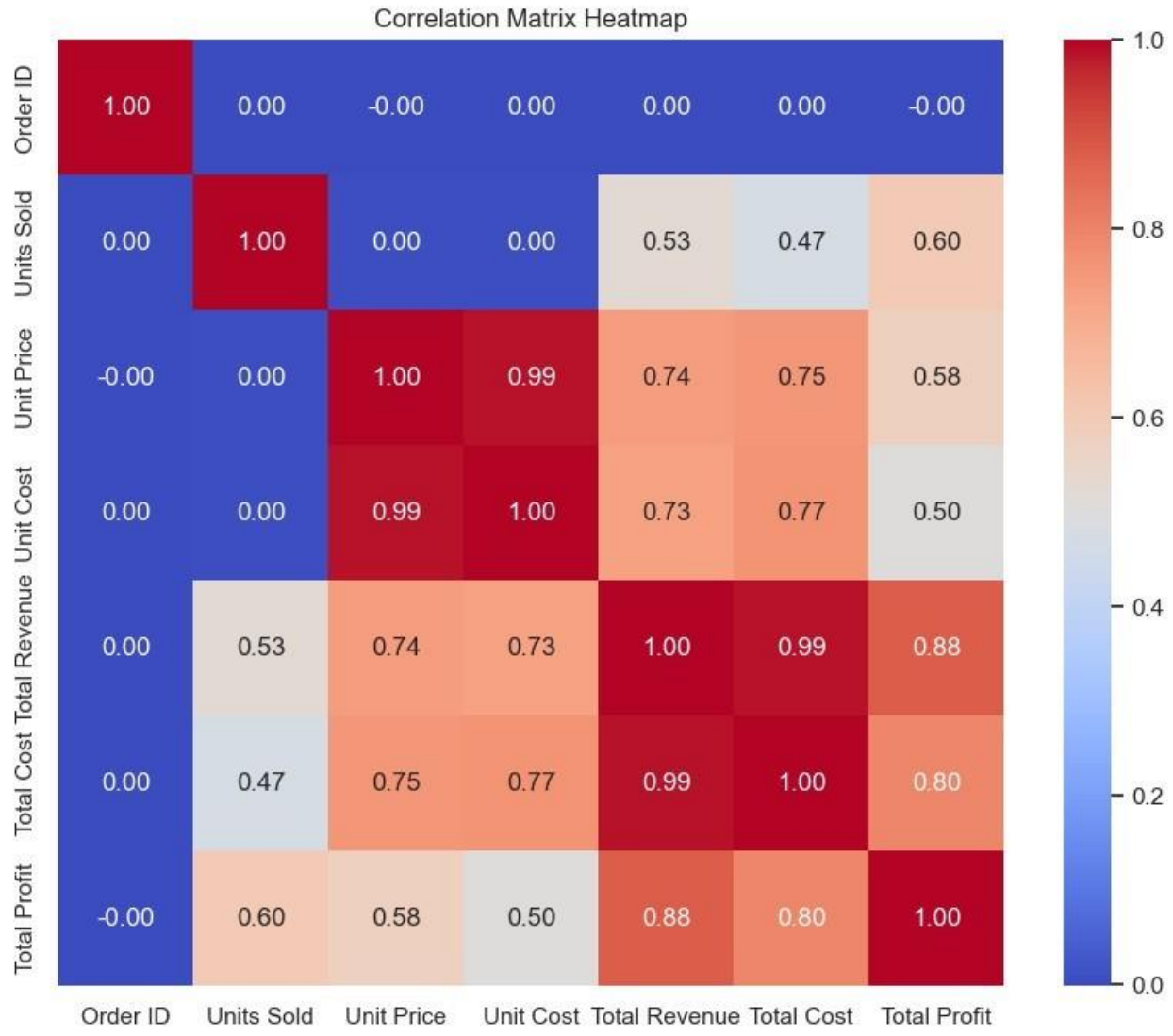
```python
import pandas as pd

# Load your dataset
df=pd.read_csv(r"D:\IDS(Assignment)Dataset\100000 Sales Records.csv")

# Display the data types of each column
print(df.dtypes)

# Select only numeric columns for correlation
numeric_df = df.select_dtypes(include=['number'])

# Calculate the correlation matrix
correlation_matrix = numeric_df.corr()

# Display the correlation matrix
print(correlation_matrix)
```

```
Region              object
Country             object
Item Type           object
Sales Channel       object
Order Priority      object
Order Date          object
Order ID             int64
Ship Date           object
Units Sold           int64
Unit Price         float64
Unit Cost          float64
Total Revenue      float64
Total Cost         float64 Total
Profit       float64
dtype: object
                Order ID  Units Sold  Unit Price  Unit Cost  Total
Revenue  \
Order ID        1.000000    0.000583   -0.000751   0.000005
0.001699
Units Sold      0.000583    1.000000    0.003453   0.003167
0.525322
Unit Price     -0.000751    0.003453    1.000000   0.986030
0.739258
Unit Cost       0.000005    0.003167    0.986030   1.000000
0.729055
Total Revenue   0.001699    0.525322    0.739258   0.729055
1.000000
Total Cost      0.002334    0.472966    0.754412   0.765211
0.987691
Total Profit   -0.000497    0.601624    0.577093   0.504763
0.880186
```

```
                Total Cost  Total Profit
Order ID          0.002334     -0.000497
Units Sold        0.472966      0.601624
Unit Price        0.754412      0.577093
Unit Cost         0.765211      0.504763
Total Revenue     0.987691      0.880186
Total Cost        1.000000      0.795110
Total Profit      0.795110      1.000000
```

this code creates a heatmap to visually represent the correlation matrix of numeric variables in the dataset. First, sns.set(style='white') sets the background style of the plot to white. A figure size of 10x8 inches is set using plt.figure(figsize=(10, 8)). The sns.heatmap() function generates the heatmap, where annot=True displays correlation values within the cells, and fmt=".2f" formats them to two decimal places. The cmap='coolwarm' argument specifies a color scheme, with cooler colors indicating weaker correlations and warmer colors indicating stronger ones. The square=True option ensures the heatmap is square-shaped. Finally, plt.title() adds a title, and plt.show() renders the plot for visualization.

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

# Set the style for the heatmap
sns.set(style='white')

# Create a heatmap for the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f",
cmap='coolwarm', square=True) plt.title('Correlation
Matrix Heatmap') plt.show()
```



This code creates a scatter plot to visualize the relationship between "Units Sold" and "Total Revenue" in the dataset. Here's a breakdown of the steps:

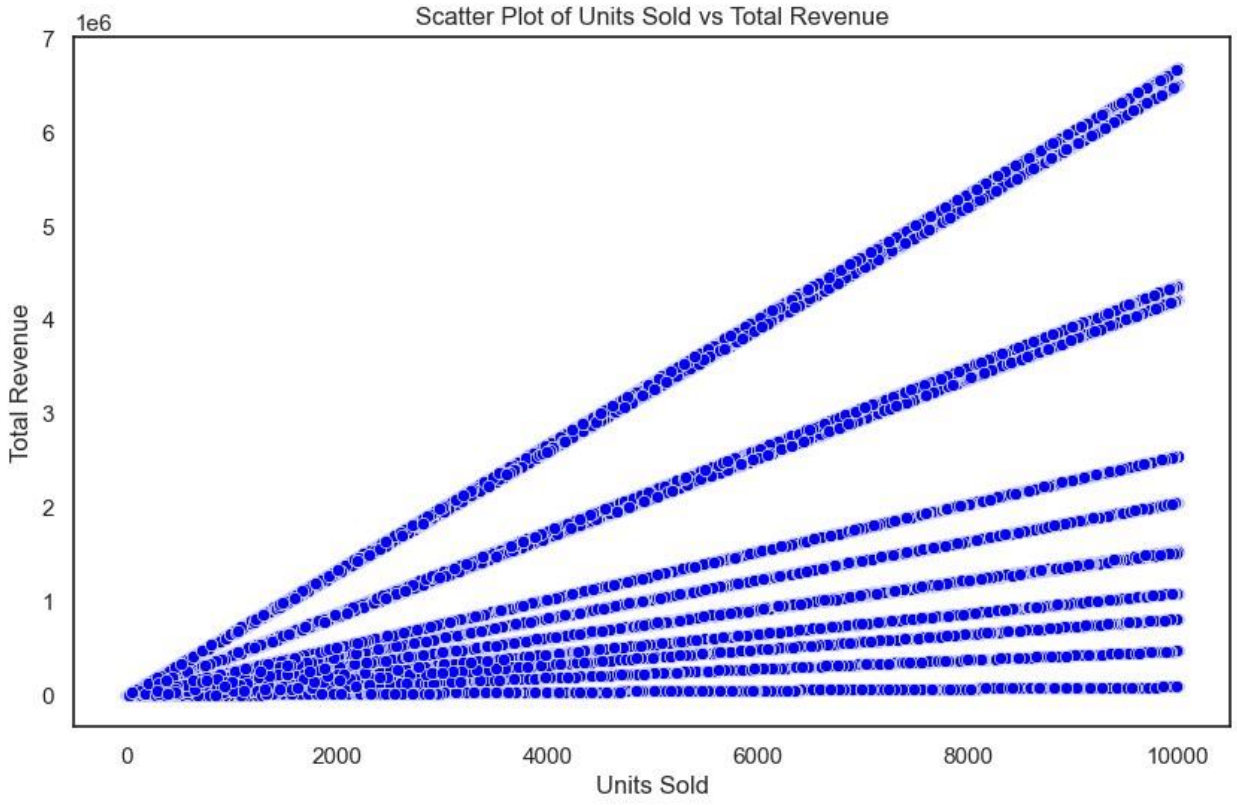
1 Loading the Dataset 2 Setting the Plot Size 3 Creating the Scatter Plot 4 Adding Title and Labels 5 Displaying the Plot: Finally, plt.show() renders and displays the scatter plot.

The scatter plot helps in identifying any potential correlation or patterns between units sold and the total revenue generated.

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

# Load your dataset
df=pd.read_csv(r"D:\IDS(Assignment)Dataset\100000 Sales Records.csv")
```

```
# Print the column names to verify
print(df.columns)


plt.figure(figsize=(10, 6))
sns.scatterplot(x='Units Sold', y='Total Revenue', data=df,
color='blue')
plt.title('Scatter Plot of Units Sold vs Total Revenue')
plt.xlabel('Units Sold')
plt.ylabel('Total Revenue')
plt.show()
```

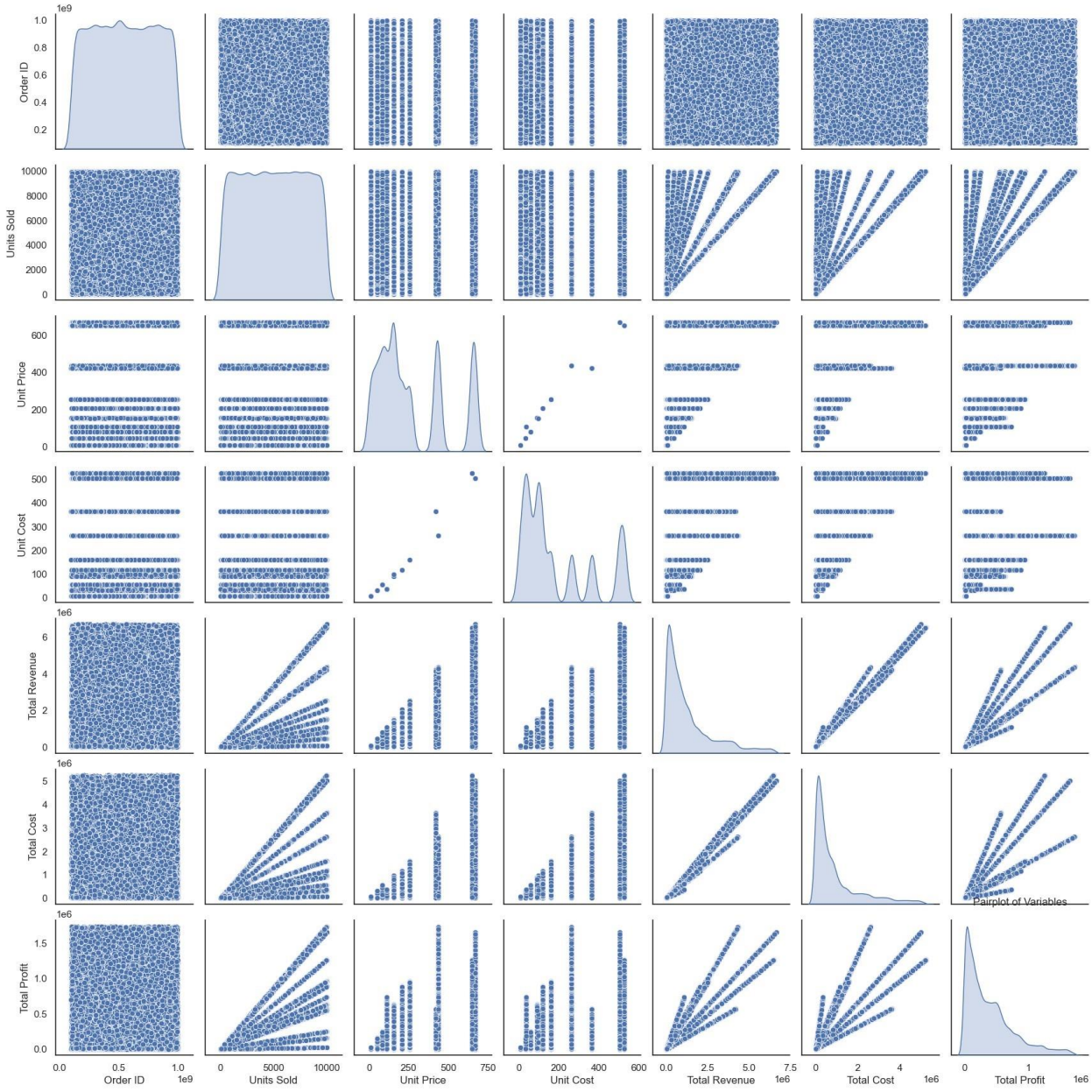
```
Index(['Region', 'Country', 'Item Type', 'Sales Channel', 'Order
Priority',
       'Order Date', 'Order ID', 'Ship Date', 'Units Sold', 'Unit
Price',
       'Unit Cost', 'Total Revenue', 'Total Cost', 'Total Profit'],
      dtype='object')
```



The code generates a pairplot to visualize relationships between multiple variables in the dataset. `sns.pairplot()` creates a grid of scatter plots for every combination of numeric variables, and `diag_kind='kde'` adds Kernel Density Estimate plots on the diagonal to show the distribution of individual variables. The `markers='o'` argument ensures the points are represented by circles. A title "Pairplot of Variables" is added with `plt.title()`, and

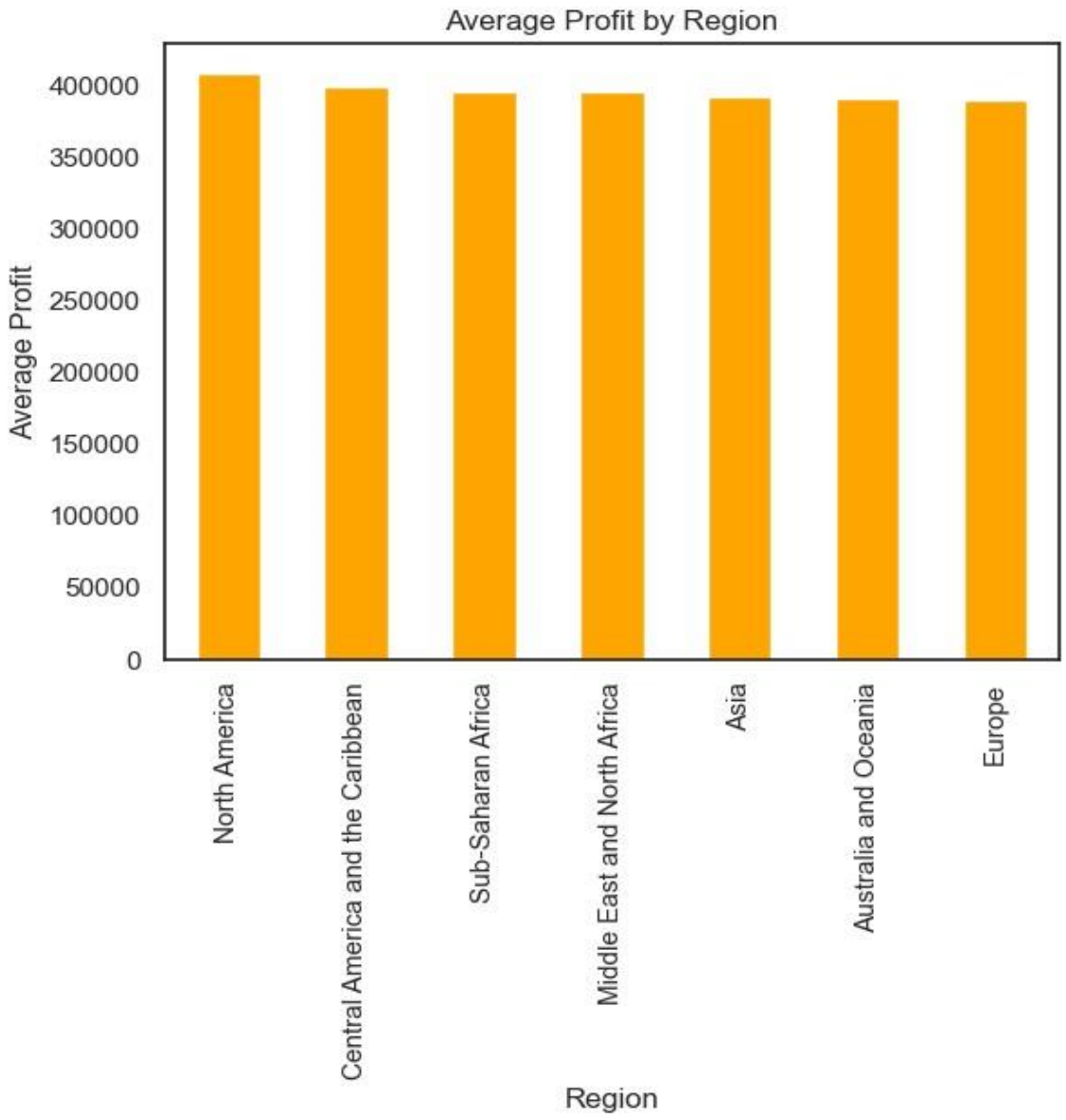
`plt.show()` displays the plot. This visualization helps in identifying patterns, correlations, and distributions in the data.

```
# Pairplot for multiple variables
sns.pairplot(df, diag_kind='kde', markers='o')
plt.title('Pairplot of Variables') plt.show()
```



The code groups the dataset by "Region" and calculates the average "Total Profit" for each region. It then sorts the average profits in descending order and creates a bar plot with orange bars. The y-axis is labeled "Average Profit" for clarity. Finally, plt.show() displays the plot, allowing for comparison of profits across regions.

```
region_profit = df.groupby('Region')['Total
Profit'].mean().sort_values(ascending=False)
region_profit.plot(kind='bar', title='Average Profit by Region',
color='orange')
plt.ylabel("Average Profit")
plt.show()
```



Conclusion

In this analysis, we looked closely at a sales dataset by creating visual representations of important variables like 'Units Sold' and 'Unit Price' using histograms and boxplots. The histogram for 'Unit Price' showed us how prices are spread out across different products, helping us see where most prices fall and if there are any unusual pricing patterns. The boxplot for 'Unit Price' highlighted the average price, the range of prices, and any outliers that might indicate pricing issues. Similarly, examining 'Units Sold' gave us insights into which products are popular and helped us spot any unusual sales figures that might need further attention.

These visual tools not only helped us understand the data better but also provided a basis for making smart business decisions. By recognizing trends in sales and pricing, companies can adjust their strategies to meet customer needs, manage inventory more effectively, and improve overall sales performance.

Final Thoughts

Understanding how products are selling and how they are priced is essential for making good business choices. The insights from this analysis can guide businesses in managing their stock, changing prices, and boosting sales. For example, if we find that some products are selling well at low prices, it might be a good idea to raise those prices to increase profits. On the other hand, if some products are priced high but not selling well, they might need discounts or promotions to attract buyers.

Looking ahead, we could dive deeper into the data by exploring relationships between different factors, breaking down sales by product categories, or analyzing sales trends over time. Adding information about customer preferences and buying habits could also enhance our understanding and help create more effective marketing strategies.

Overall, using visual tools to analyze data has shown to be a powerful way to uncover important insights. By continually examining and interpreting sales data, businesses can stay flexible and responsive to changes in the market, ultimately leading to growth and success in a competitive environment.