Exploratory Data Analysis

Dune is a reputable retailer offering a diverse selection of products, including accessories, clothing and phones. With growing presence in 14 countries and a team of over 70,000 employees, the company prides itself on providing affordable options for everyone. From fashion-forward trendletters to multi-generational families. Dune strives to offer great quality essentials and standout styles that cater to a wide range of customers.

As the newly appointed Data Scientist, your first task is to analyze the company's sales data from the previous year and provide actionable insights and recommendations. this analysis will help identify areas of opportunity and inform future business decisions aimed at improving performance and increasing profitability.

Exploratory Data Analysis Exploratory Data Analysis (EDA) is the process of analyzing and summarizing data in order to gain insights and understanding of the underlxying patterns and relationships. The main objective of EDA is to identify and explore the main characteristics and patterns of the data, and to identify any anomales or outliers that may impact subsequent analysis.

EDA typically involves a number of steps, including:

1. Data Cleaning - Data cleaning involves removing or correcting any errors or inconsistences in the data, such as missing values or incorrect values.
2. Data Visualization - Data Visualization techniques are then used to graphically represent the data and identify any trends or patterns.
3. Statistical Analysis - It is used to identify any relationships between variables and to test hypotheses about the data. This may involve calculating summary statistics such as mean and standard deviation and performing tests such as correlation analysis and hypothesis testing.

In [1]:
```
# Import necessary Libraries

# For data analysis
import pandas as pd # for data ptocessing
import numpy as np

# For data visualization
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno # for missing data visualization
from collections import Counter # for counting
```

In [2]:
```
# Load the dataset
df = pd.read_csv(r"C:\Users\ADMIN\Desktop\New folder (2)\10Alytics Data Science\Python\WMD6\Dune
df.head()
```

Out[2]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payment Option	C
0	19-Feb-16	High	Segun	29	F	Abia	Accessories	Keyboard	Online	
1	20-Feb-16	High	Segun	29	F	Abia	Clothing	Polo shirts	Online	
2	27-Feb-16	High	Segun	29	F	Abia	Accessories	Keyboard	Online	
3	12-Mar-16	High	Segun	29	F	Abia	Accessories	Keyboard	Online	
4	12-Mar-16	High	Segun	29	F	Abia	Accessories	Keyboard	Online	

In [3]:
```
df.tail() # bottom 5 row
```

Out[3]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Paym Opt
34862	7-Feb-16	High	Kenny	38	M	Ebonyi	Phones	IPhone	On
34863	13-Mar-15	High	Kenny	38	M	Ebonyi	Phones	IPhone	On
34864	5-Apr-15	High	Kenny	38	M	Ebonyi	Phones	IPhone	On
34865	30-Aug-15	High	Kenny	38	M	Ebonyi	Phones	IPhone	On
34866	NaN	NaN	NaN	38	NaN	NaN	NaN	NaN	N

In [4]:
```
# Dimentionality of the data - The number of rows and columns
df.shape
```

Out[4]:
```
(34867, 12)
```

In [5]:
```
# Examine the columns/features of the data
df.columns
```

Out[5]:
```
Index(['Date', 'Customer', 'Sales Person', 'Customer_Age', 'Customer_Gender',
       'State', 'Product_Category', 'Sub_Category', 'Payment Option',
       'Quantity', 'Unit_Cost', 'Unit_Price'],
      dtype='object')
```

In [6]:
```
# Investigate the dataset for anomalies and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34867 entries, 0 to 34866
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Date              34866 non-null  object 
 1   Customer          34866 non-null  object 
 2   Sales Person      34866 non-null  object 
 3   Customer_Age      34867 non-null  int64  
 4   Customer_Gender   34866 non-null  object 
 5   State             34866 non-null  object 
 6   Product_Category  34866 non-null  object 
 7   Sub_Category      34866 non-null  object 
 8   Payment Option    34866 non-null  object 
 9   Quantity          34866 non-null  float64
 10  Unit_Cost         34866 non-null  float64
 11  Unit_Price        34866 non-null  float64
dtypes: float64(3), int64(1), object(8)
memory usage: 3.2+ MB
```

In [7]:
```
# Numerical Statistical Analysis
df.describe()
```

Out[7]:

	Customer_Age	Quantity	Unit_Cost	Unit_Price
count	34867.000000	34866.000000	34866.000000	34866.000000
mean	36.382683	2.002524	349.880567	389.232473
std	11.112813	0.813936	490.015846	525.319097
min	17.000000	1.000000	0.670000	0.670000
25%	28.000000	1.000000	45.000000	53.670000
50%	35.000000	2.000000	150.000000	179.000000
75%	44.000000	3.000000	455.000000	521.000000
max	87.000000	3.000000	3240.000000	5082.000000

In [8]:
```
# Categorical Statistical Analysis
df.describe(include=["object", "bool"])
```

Out[8]:

	Date	Customer	Sales Person	Customer_Gender	State	Product_Category	Sub_Category	Payment Option
count	34866	34866	34866	34866	34866	34866	34866	34866
unique	576	4	7	2	36	3	17	3
top	1-Mar-16	Low	Remota	F	Lagos	Accessories	Keyboard	Cash
freq	196	13041	6667	17439	10332	22534	11112	15911

Dealing with missing data - 1 MCAR (Missing completely at random) These are values that are randomly missing and do not depend on any other values. 2 MAR (Missing at Random) These values are independent

on some additional features. 3 MNAT (Missing not at Random) There is a reason behind why these values are missing.

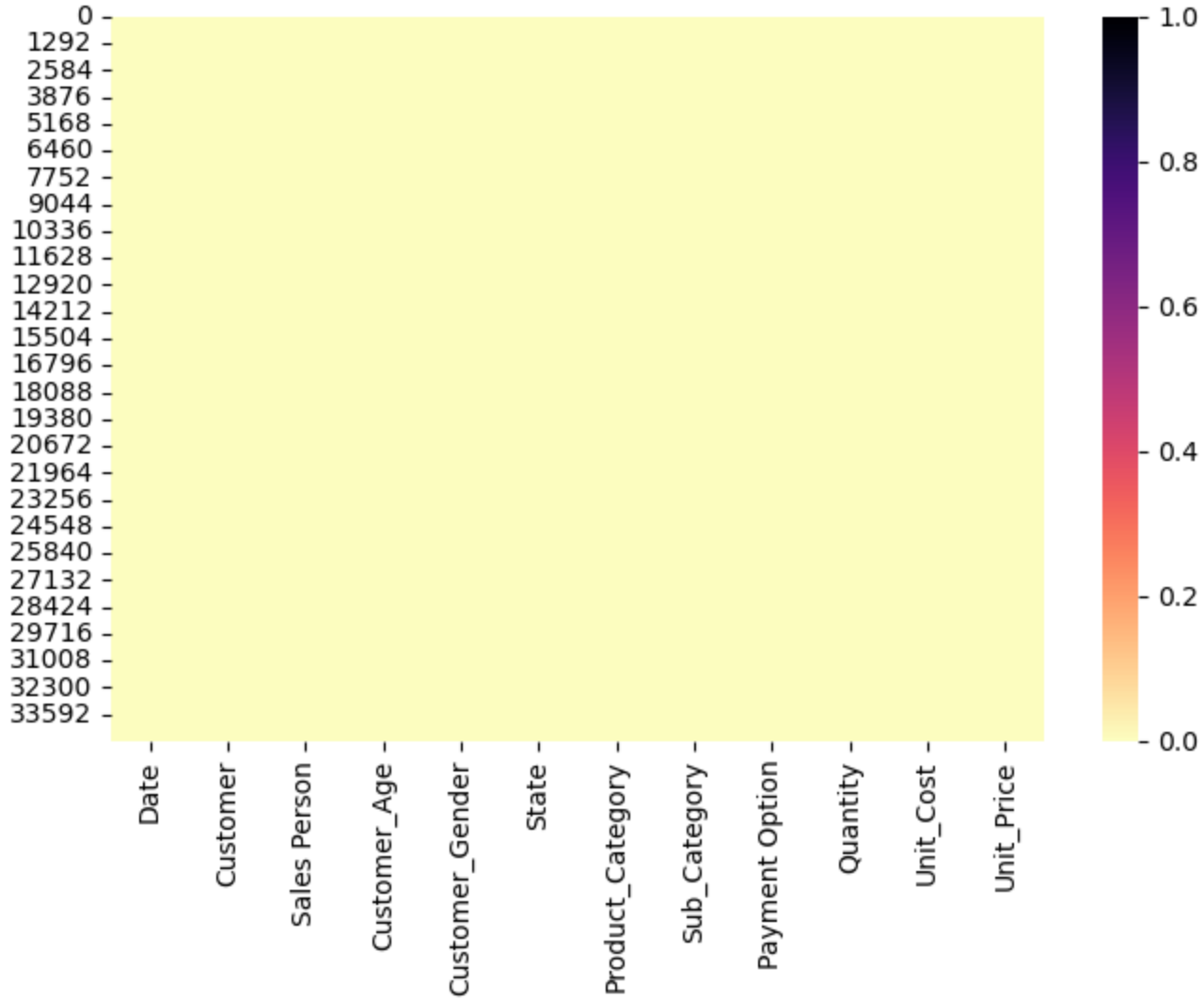
There are several methods for imputting missing data, including the measure of Central Tendency, regression imputation and multiple imputation, measure of central tendency involves replacing missing values with either the Mean, median and Mode of the variable while regression imputation involves using other variables in the dataset to predict missing values.

In [9]:
```
# Investigate the missing data
null_vals = df.isnull().sum()
null_vals
```

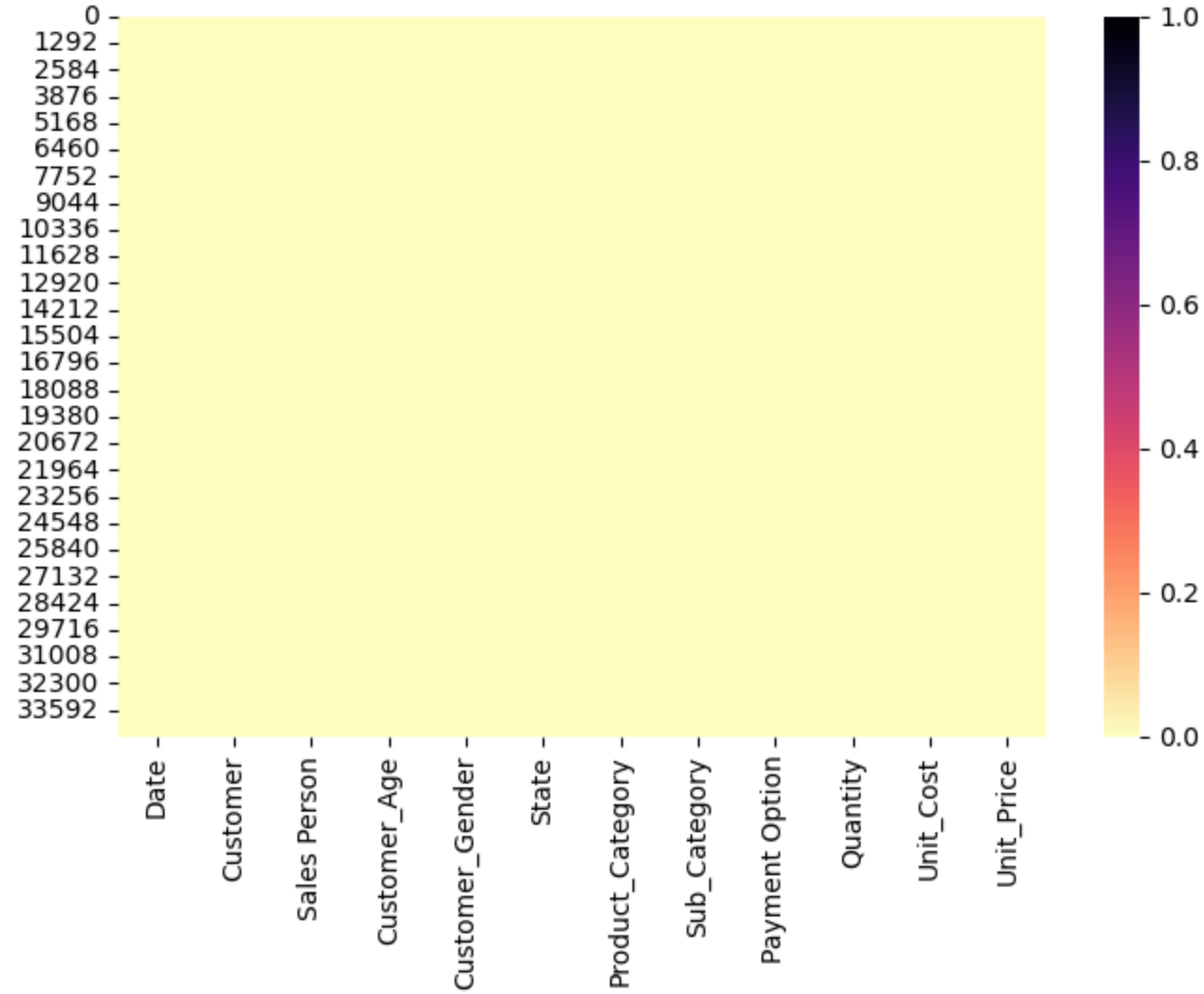
Out[9]:
```
Date                1
Customer            1
Sales Person        1
Customer_Age        0
Customer_Gender     1
State               1
Product_Category    1
Sub_Category        1
Payment Option      1
Quantity            1
Unit_Cost           1
Unit_Price          1
dtype: int64
```

In [10]:
```
# Visualize the missing data - Explore the missing data through visualization
plt.figure(figsize = (8, 5))
sns.heatmap(df.isnull(), cmap="magma_r", cbar=True)
```

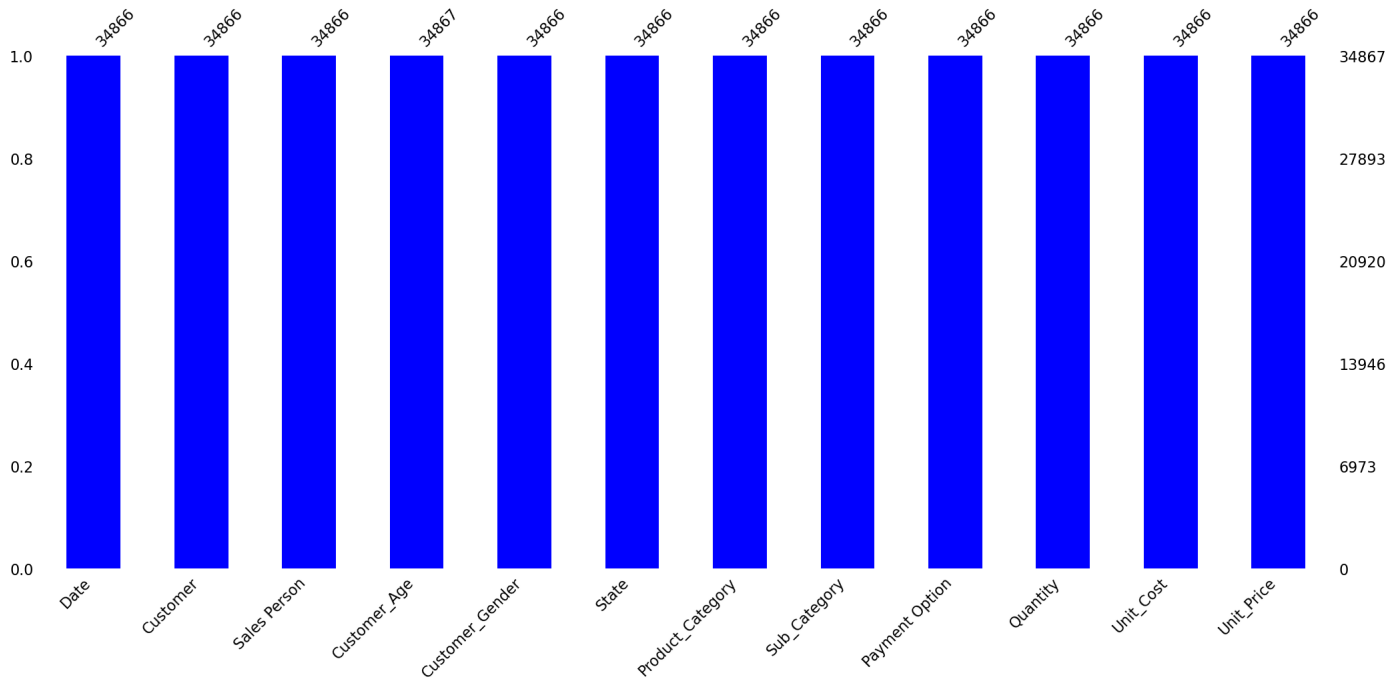
Out[10]:
```
<Axes: >
```



In [11]:
```
# Visualize the missing data - Explore the missing data through visualization
plt.figure(figsize = (8, 5))
sns.heatmap(df.isnull(), cmap="magma_r", cbar=True);
```



In [12]:
```
msno.bar(df, color="blue");
```



In [13]:
```
# Display where the missing data exist in the data
df.isnull()
```

Out[13]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payme Opti
0	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	
34862	False	False	False	False	False	False	False	False	Fal
34863	False	False	False	False	False	False	False	False	Fal
34864	False	False	False	False	False	False	False	False	Fal
34865	False	False	False	False	False	False	False	False	Fal
34866	True	True	True	False	True	True	True	True	Tr

34867 rows × 12 columns

In [14]:
```
# Display where the missing data exist in the data
df[df.isnull().any(axis=1)]
```

Out[14]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payme Opti
34866	NaN	NaN	NaN	38	NaN	NaN	NaN	NaN	Na

In [15]:
```
# Drop the missing data
df.dropna(inplace=True)
```

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
Date                0
Customer            0
Sales Person        0
Customer_Age        0
Customer_Gender     0
State               0
Product_Category    0
Sub_Category        0
Payment Option      0
Quantity            0
Unit_Cost           0
Unit_Price          0
dtype: int64
```

In [17]:
```
# Datatime Analysis
df.head(2)
```

Out[17]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payment Option	C
0	19-Feb-16	High	Segun	29	F	Abia	Accessories	Keyboard	Online	
1	20-Feb-16	High	Segun	29	F	Abia	Clothing	Polo shirts	Online	

In [18]:
```
# Convert the date colum into a pandas datetime object
# df.info()
df["Date"] = pd.to_datetime(df["Date"])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 34866 entries, 0 to 34865
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   Date              34866 non-null  datetime64[ns]
 1   Customer          34866 non-null  object        
 2   Sales Person      34866 non-null  object        
 3   Customer_Age      34866 non-null  int64         
 4   Customer_Gender   34866 non-null  object        
 5   State             34866 non-null  object        
 6   Product_Category  34866 non-null  object        
 7   Sub_Category      34866 non-null  object        
 8   Payment Option    34866 non-null  object        
 9   Quantity          34866 non-null  float64       
 10  Unit_Cost         34866 non-null  float64       
 11  Unit_Price        34866 non-null  float64       
dtypes: datetime64[ns](1), float64(3), int64(1), object(7)
memory usage: 3.5+ MB
```

In [19]:
```
# Extract the Year, Month, Quarter
df['year'] = df["Date"].dt.year
df['month'] = df["Date"].dt.month
df['month_name'] = df["Date"].dt.month_name()
df['quarter'] = df["Date"].dt.quarter
df.head()
```

Out[19]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payment Option
0	2016-02-19	High	Segun	29	F	Abia	Accessories	Keyboard	Online
1	2016-02-20	High	Segun	29	F	Abia	Clothing	Polo shirts	Online
2	2016-02-27	High	Segun	29	F	Abia	Accessories	Keyboard	Online
3	2016-03-12	High	Segun	29	F	Abia	Accessories	Keyboard	Online
4	2016-03-12	High	Segun	29	F	Abia	Accessories	Keyboard	Online

In [20]:

```
# Group Customer Age

def age_group(x):
    if x <= 25:
        return "<=25 Young Adult"
    elif x <= 40:
        return "25-40 Adult"
    elif x <= 50:
        return "41-50 Old Adult"
    else:
        return ">=51 Elder"

    # Apply function to the data
df["age_group"] = df["Customer_Age"].apply(age_group)
df.head(2)
```

Out[20]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payment Option
0	2016-02-19	High	Segun	29	F	Abia	Accessories	Keyboard	Online
1	2016-02-20	High	Segun	29	F	Abia	Clothing	Polo shirts	Online

In [21]:

```
# Cost, revenue and Profit Calculation
df["cost"] = df["Quantity"]*df["Unit_Cost"]
df["revenue"] = df["Quantity"]*df["Unit_Price"]
df["profit"] = df["revenue"] - df["cost"]

df.head()
```

Out[21]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payment Option
0	2016-02-19	High	Segun	29	F	Abia	Accessories	Keyboard	Online
1	2016-02-20	High	Segun	29	F	Abia	Clothing	Polo shirts	Online
2	2016-02-27	High	Segun	29	F	Abia	Accessories	Keyboard	Online
3	2016-03-12	High	Segun	29	F	Abia	Accessories	Keyboard	Online
4	2016-03-12	High	Segun	29	F	Abia	Accessories	Keyboard	Online

In [22]:

```
# Profit/Loss grouping
def porl(x):
    if x >= 0:
        return "Profit"
    else:
        return "Loss"
df["profit_label"] = df['profit'].apply(porl)
df.head()
```

Out[22]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Payment Option
0	2016-02-19	High	Segun	29	F	Abia	Accessories	Keyboard	Online
1	2016-02-20	High	Segun	29	F	Abia	Clothing	Polo shirts	Online
2	2016-02-27	High	Segun	29	F	Abia	Accessories	Keyboard	Online
3	2016-03-12	High	Segun	29	F	Abia	Accessories	Keyboard	Online
4	2016-03-12	High	Segun	29	F	Abia	Accessories	Keyboard	Online

5 rows × 21 columns

Univariate Analysis

Univariate analysis involves analyzing the distribution and summary statistics of individual variable/column/feature.

- Numerical Column/Feature = Numerical Visualization techniques
- Categorucal Column/Features = Categorical Visualization Techniques

Take each column and examine each column

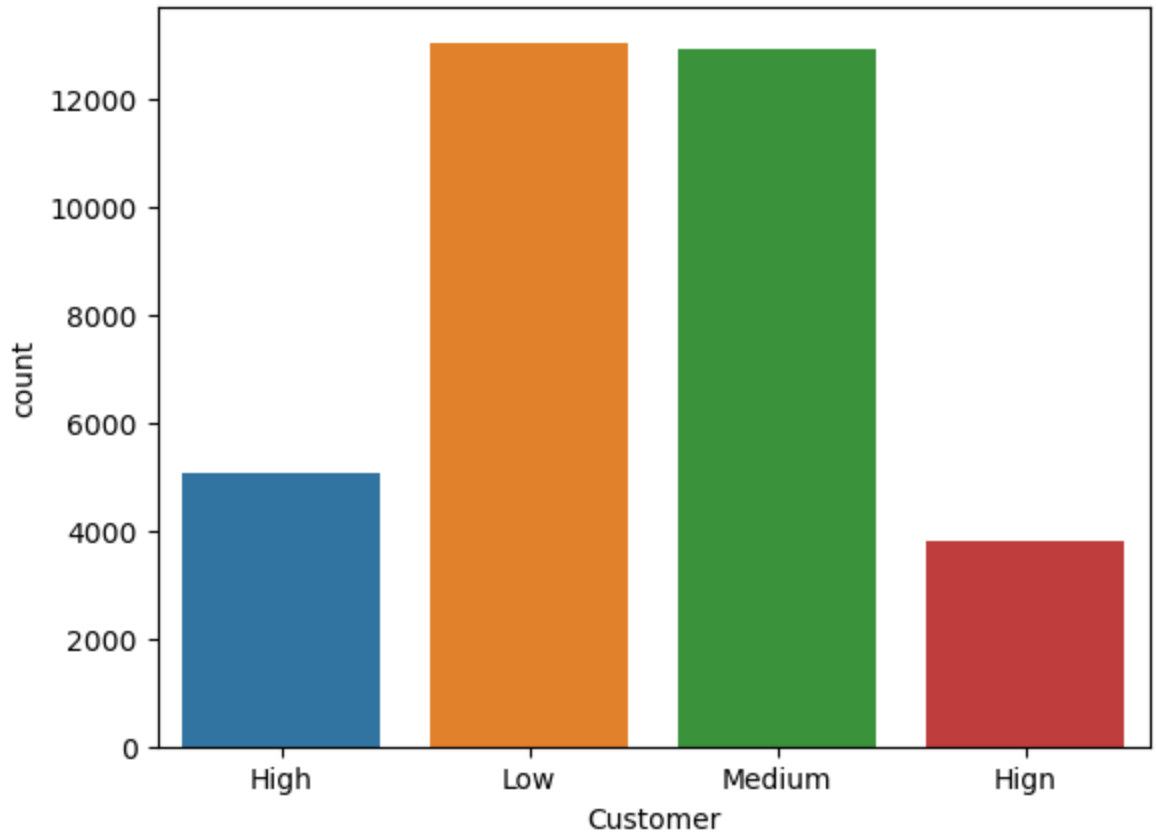
Categorical Data Visualization

In [23]:
```
df.columns
```

Out[23]:
```
Index(['Date', 'Customer', 'Sales Person', 'Customer_Age', 'Customer_Gender',
       'State', 'Product_Category', 'Sub_Category', 'Payment Option',
       'Quantity', 'Unit_Cost', 'Unit_Price', 'year', 'month', 'month_name',
       'quarter', 'age_group', 'cost', 'revenue', 'profit', 'profit_label'],
      dtype='object')
```

In [24]:
```
# How many customers below to each customer spec
sns.countplot(x="Customer", data=df)
```

Out[24]:
```
<Axes: xlabel='Customer', ylabel='count'>
```



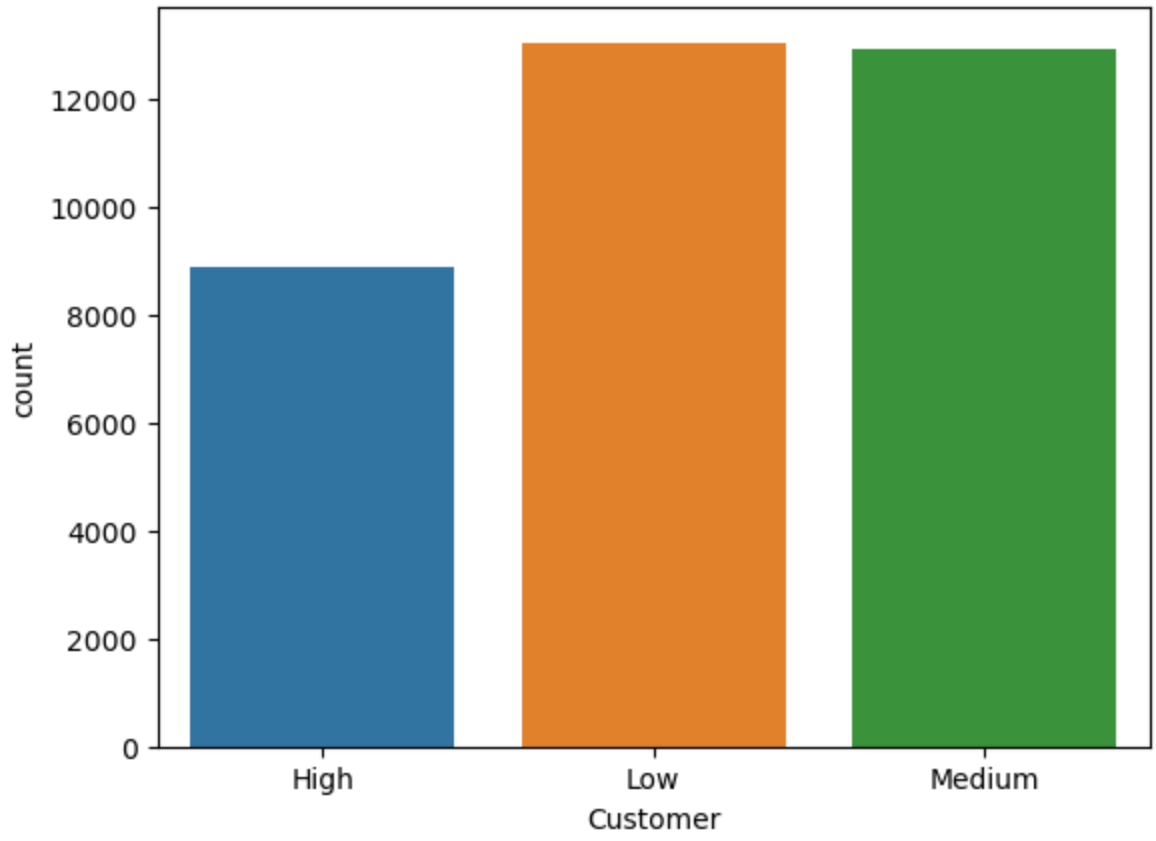
In [25]:
```
# Investigate the columns affected
df[df["Customer"] == "Hign"].head(3)
```

Out[25]:

	Date	Customer	Sales Person	Customer_Age	Customer_Gender	State	Product_Category	Sub_Category	Paym Opt
29770	2015-08-03	Hign	Derick	28	F	Lagos	Phones	IPhone	C
29771	2015-08-04	Hign	Derick	28	F	Lagos	Accessories	Keyboard	C
29772	2015-08-04	Hign	Derick	28	F	Lagos	Accessories	Keyboard	C

3 rows × 21 columns

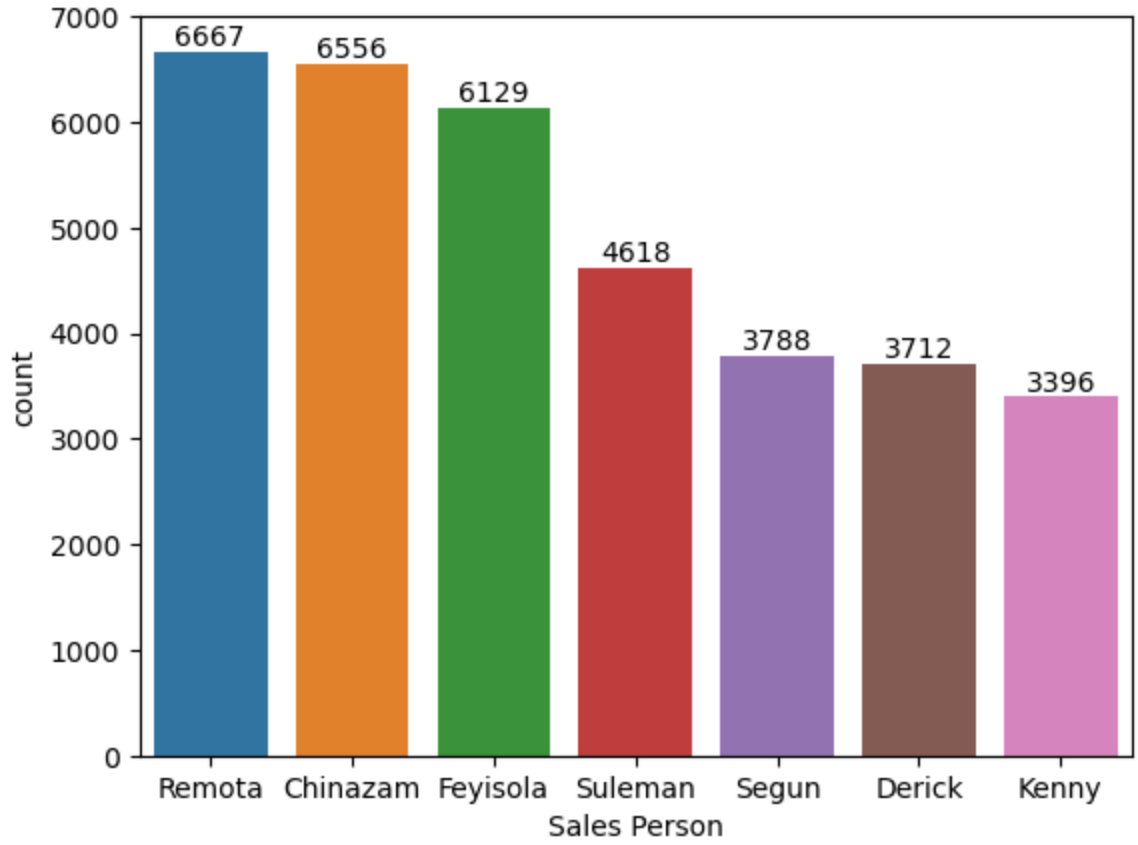
In [26]:
```
# Correct the spelling of HIGN
df.loc[df["Customer"] == "Hign", "Customer"] = "High"
sns.countplot(x="Customer", data=df);
```



In [27]:
```
df["Customer"].value_counts()
```

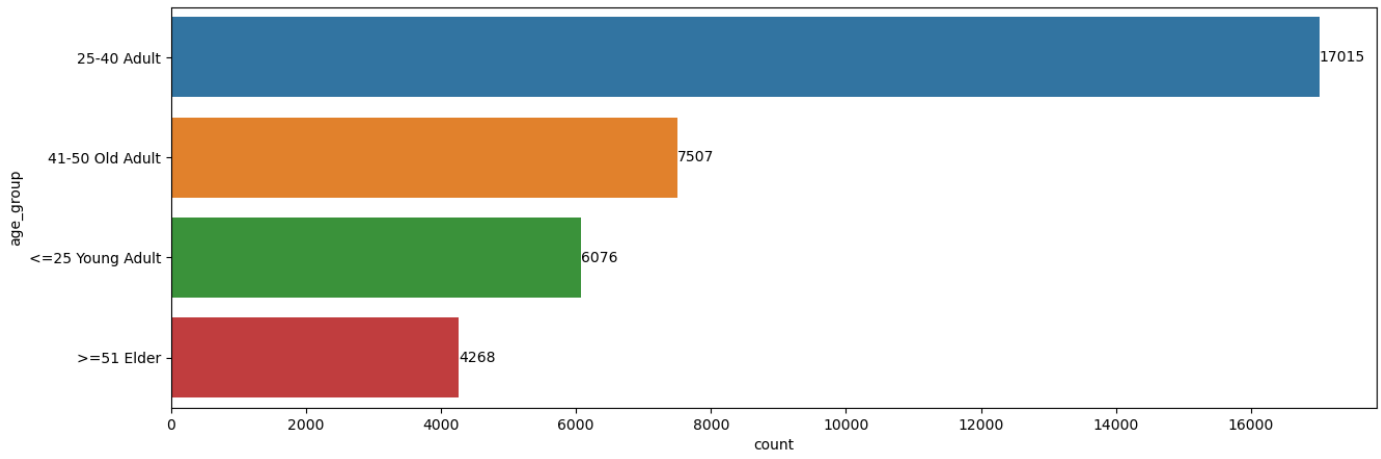
Out[27]:
```
Low       13041
Medium    12926
High       8899
Name: Customer, dtype: int64
```

In [28]:
```
# Sales Person - how many transactions by sales person
ax = sns.countplot(x=df["Sales Person"], order=df["Sales Person"].value_counts(ascending=False).
values = df["Sales Person"].value_counts(ascending=False).values
ax.bar_label(container=ax.containers[0], labels=values);
```



Sales person with highest transections is Ramota while the lowest is Kenny

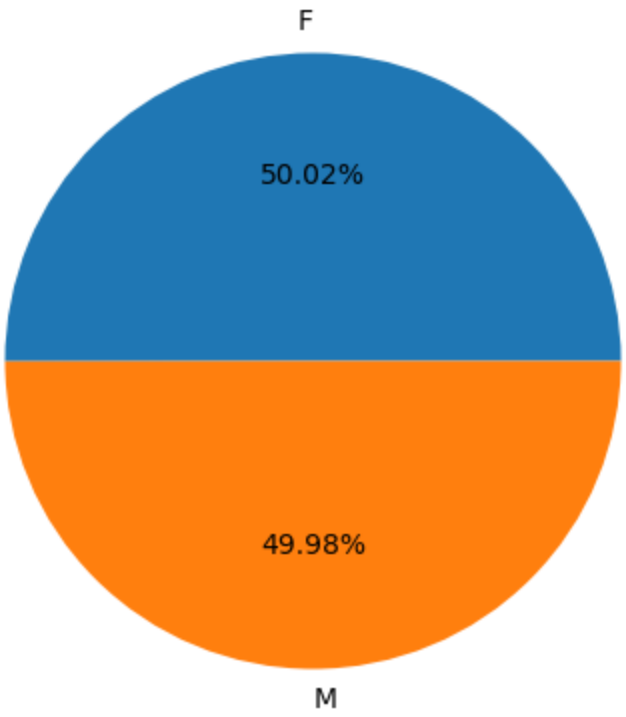
In [29]:
```
# Total transactions by customer Age Group
# Sales Person - how many transactions by sales person
plt.figure(figsize=(15,5))
ax = sns.countplot(y=df["age_group"], order=df["age_group"].value_counts(ascending=False).index)
values = df["age_group"].value_counts(ascending=False).values
ax.bar_label(container=ax.containers[0], labels=values);
```



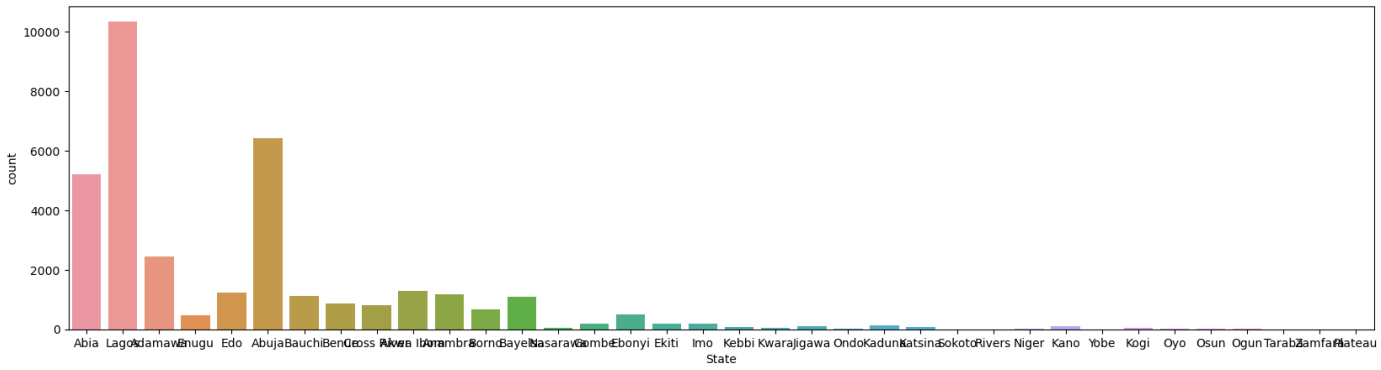
The age group with highest transactions is 25 - 40 whivh is adult category.

In [30]:
```
# Total transaction by Customer Gender
fig,ax = plt.subplots(figsize=(5,5))
count = Counter(df["Customer_Gender"])
```

```
ax.pie(count.values(), labels=count.keys(), autopct=lambda p: f'{p:.2f}%')
plt.show()
```

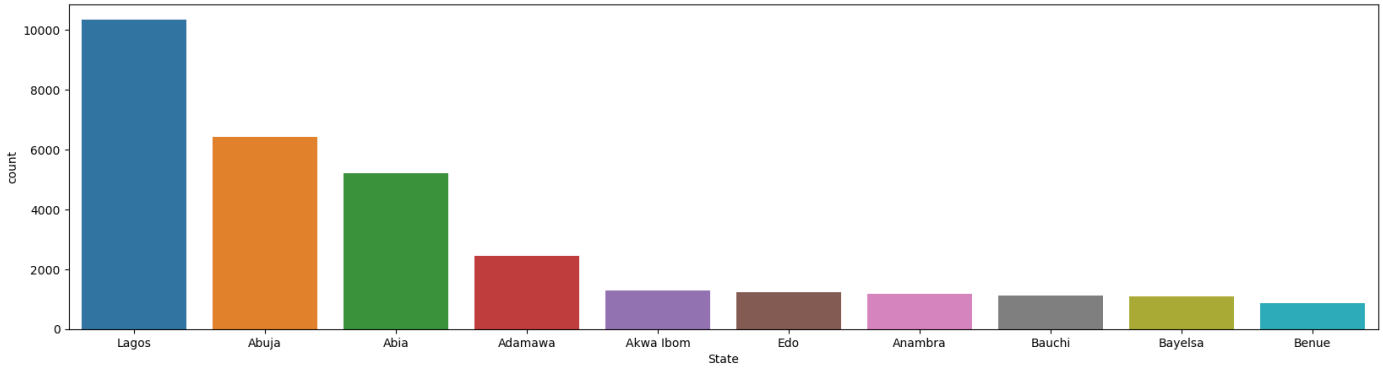


In [31]:
```
#Total transaction by state
plt.figure(figsize=(20,5))
sns.countplot(x="State", data=df);
```



In [32]:
```
#Total 10 transaction by state
plt.figure(figsize=(20,5))
topten = df["State"].value_counts().head(10)
sns.countplot(x="State", data=df, order=topten.index);
print(topten)
```

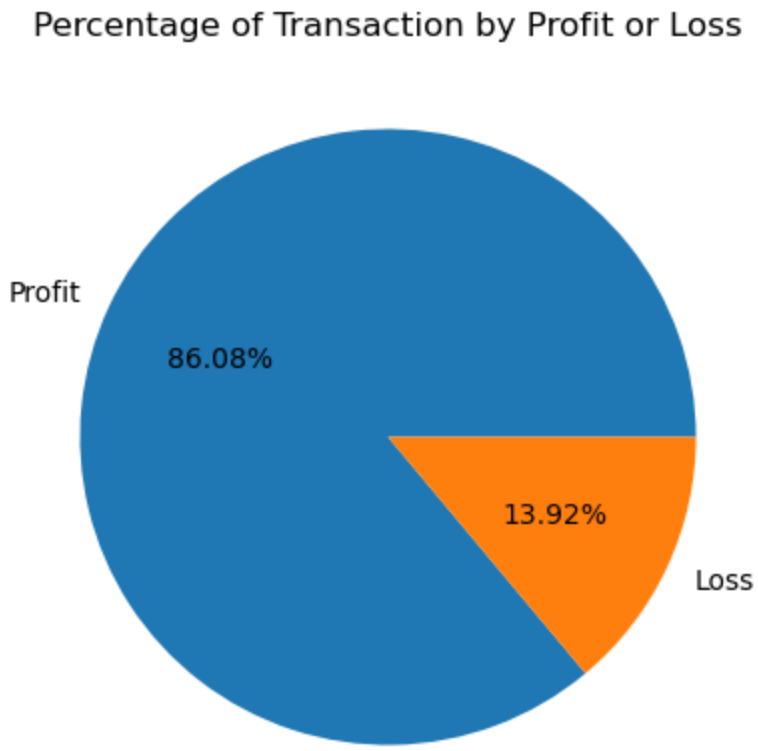
```
Lagos        10332
Abuja         6421
Abia          5206
Adamawa       2446
Akwa Ibom     1287
Edo           1229
Anambra       1171
Bauchi        1112
Bayelsa       1092
Benue          869
Name: State, dtype: int64
```



work on

- product category
- Sub Category
- Payment Option
- Month Name

In [33]:
```
# Total transaction by Profit or Loss
fig,ax = plt.subplots(figsize=(5,5))
count = Counter(df["profit_label"])
ax.pie(count.values(), labels=count.keys(), autopct=lambda p: f'{p:.2f}%')
ax.set_title("Percentage of Transaction by Profit or Loss")
plt.show();
```

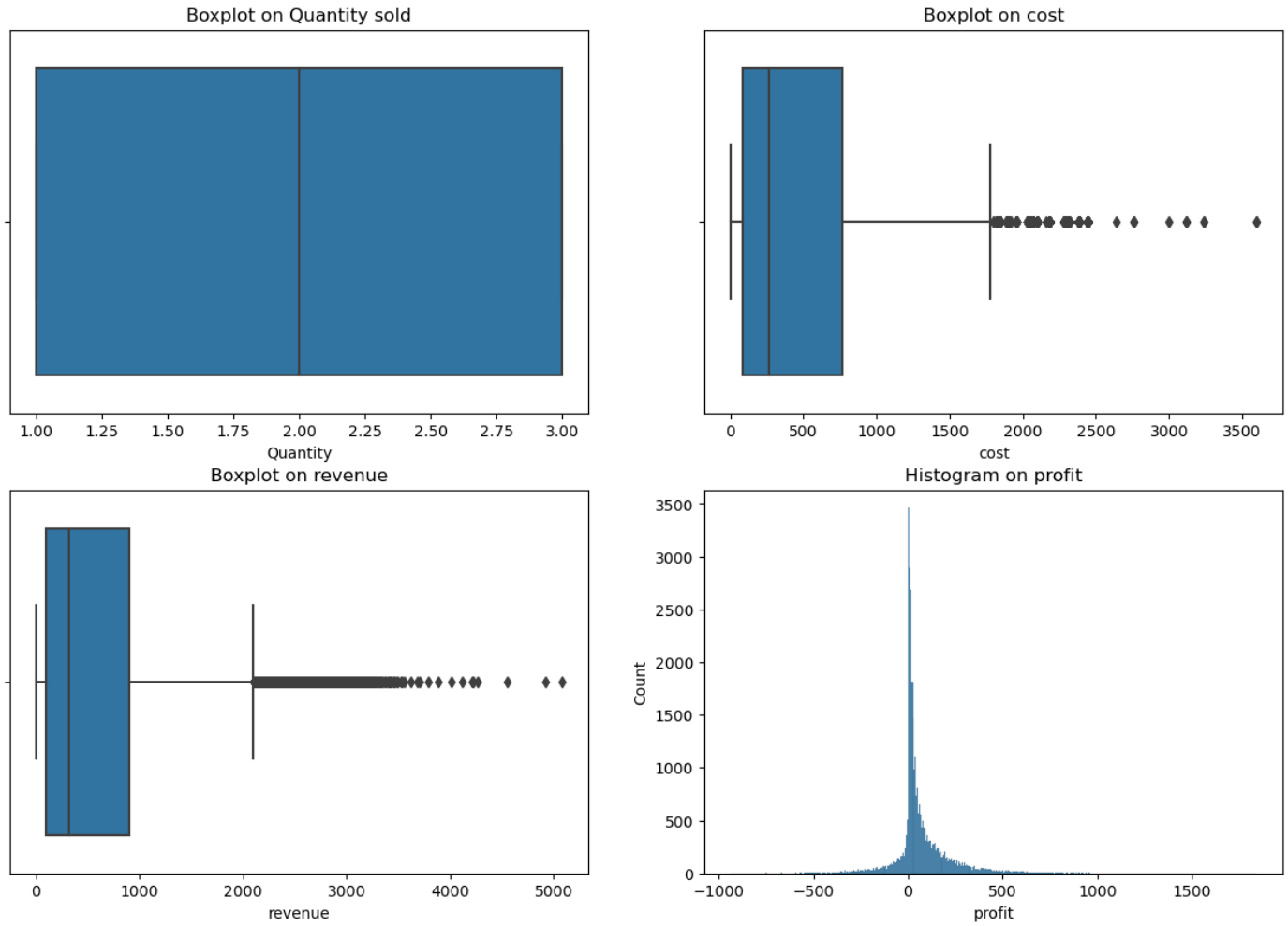


In [34]:
```
# Narration
```

Numerical Data Visualization

In [35]:
```
# Quantity, Cost, Revenue and Profit - dubplot

fig,axs = plt.subplots(nrows=2, ncols=2, figsize=(15,10))
sns.boxplot(x="Quantity", data=df, ax=axs[0,0])
axs[0,0].set_title("Boxplot on Quantity sold")

sns.boxplot(x="cost", data=df, ax=axs[0,1])
axs[0,1].set_title("Boxplot on cost")

sns.boxplot(x="revenue", data=df, ax=axs[1,0])
axs[1,0].set_title("Boxplot on revenue")

sns.histplot(x="profit", data=df, ax=axs[1,1])
axs[1,1].set_title("Histogram on profit")
```

Out[35]:
```
Text(0.5, 1.0, 'Histogram on profit')
```



Bivariate Analysis

Bivariate analysis involves analyzing the relationship bewteen two variables

- focus on profit

In [36]:
```
# Categorical Columns
```

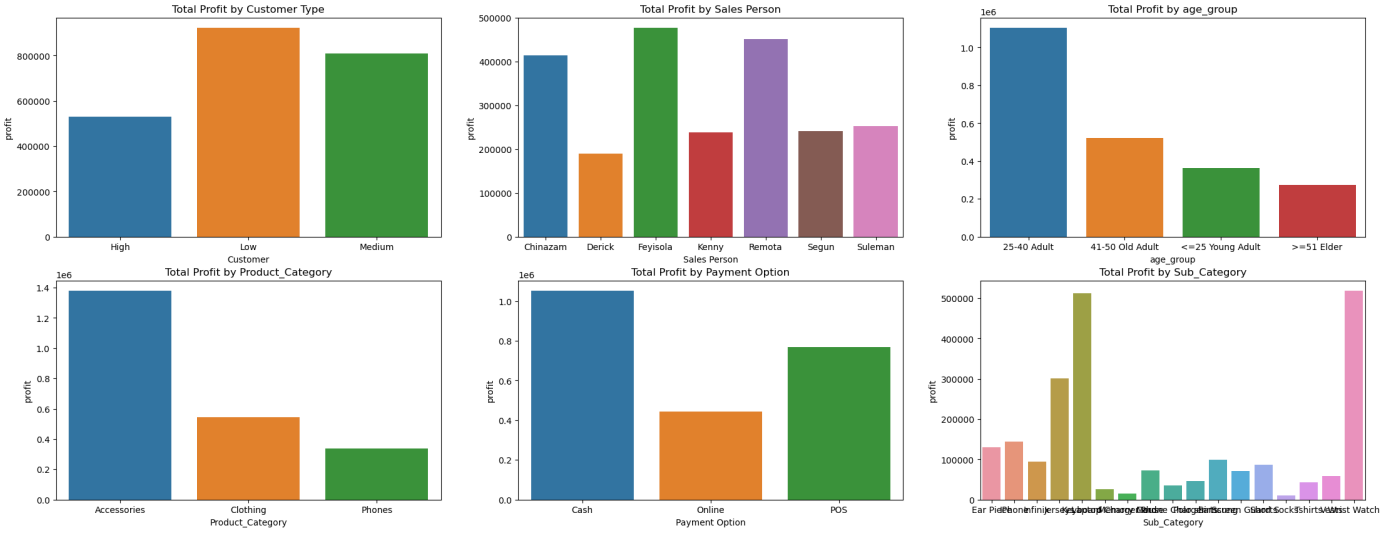
```
df.columns
```

Out[36]:

```
Index(['Date', 'Customer', 'Sales Person', 'Customer_Age', 'Customer_Gender',
       'State', 'Product_Category', 'Sub_Category', 'Payment Option',
       'Quantity', 'Unit_Cost', 'Unit_Price', 'year', 'month', 'month_name',
       'quarter', 'age_group', 'cost', 'revenue', 'profit', 'profit_label'],
      dtype='object')
```

In [37]:

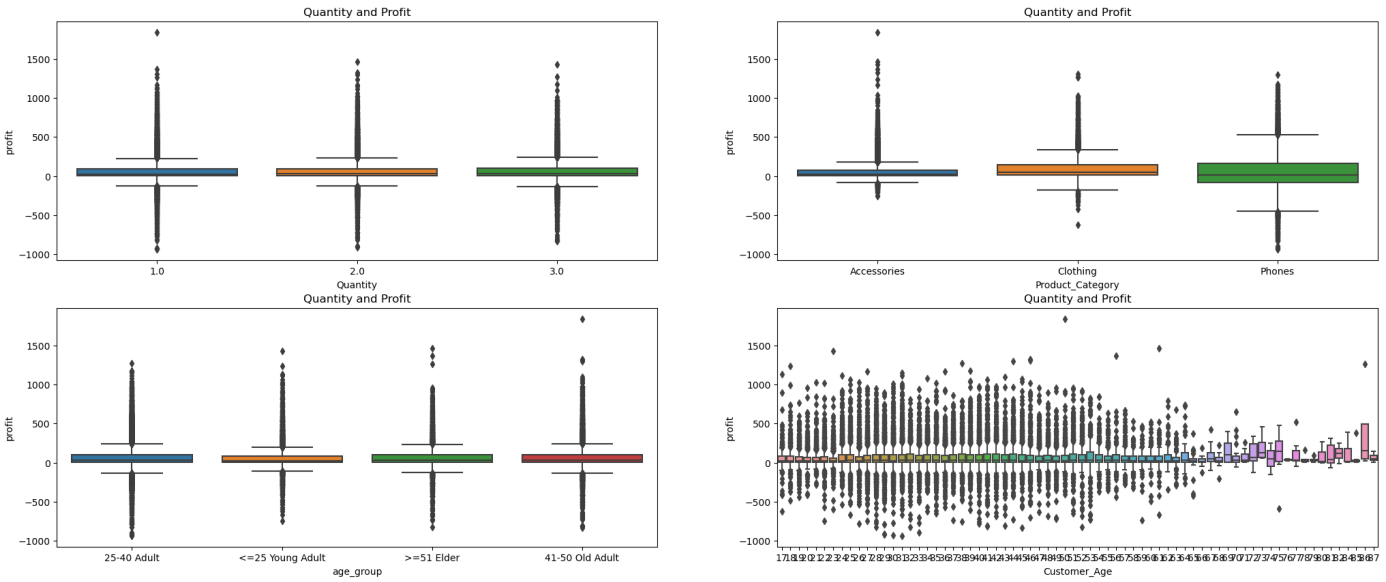
```
fig,axs = plt.subplots(nrows=2, ncols=3, figsize=(27,10))

cust_prof = df.groupby("Customer")["profit"].sum().reset_index()
sns.barplot(x='Customer', data=cust_prof, y='profit', ax=axs[0,0])
axs[0,0].set_title("Total Profit by Customer Type")

sp_prof = df.groupby("Sales Person")["profit"].sum().reset_index()
sns.barplot(x='Sales Person', data=sp_prof, y='profit', ax=axs[0,1])
axs[0,1].set_title("Total Profit by Sales Person")

ag_prof = df.groupby("age_group")["profit"].sum().reset_index()
sns.barplot(x='age_group', data=ag_prof, y='profit', ax=axs[0,2])
axs[0,2].set_title("Total Profit by age_group")

pc_prof = df.groupby("Product_Category")["profit"].sum().reset_index()
sns.barplot(x='Product_Category', data=pc_prof, y='profit', ax=axs[1,0])
axs[1,0].set_title("Total Profit by Product_Category")

po_prof = df.groupby("Payment Option")["profit"].sum().reset_index()
sns.barplot(x='Payment Option', data=po_prof, y='profit', ax=axs[1,1])
axs[1,1].set_title("Total Profit by Payment Option")

sc_prof = df.groupby("Sub_Category")["profit"].sum().reset_index()
sns.barplot(x='Sub_Category', data=sc_prof, y='profit', ax=axs[1,2])
axs[1,2].set_title("Total Profit by Sub_Category");
```



In [38]:

```
# Numerical Columns
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(25, 10))

sns.boxplot(x='Quantity', y='profit', data=df, ax=axs[0,0])
axs[0,0].set_title("Quantity and Profit")

sns.boxplot(x='Product_Category', y='profit', data=df, ax=axs[0,1])
axs[0,1].set_title("Quantity and Profit")

sns.boxplot(x='age_group', y='profit', data=df, ax=axs[1,0])
```

```
axs[1,0].set_title("Quantity and Profit")

sns.boxplot(x='Customer_Age', y='profit', data=df, ax=axs[1,1])
axs[1,1].set_title("Quantity and Profit");
```

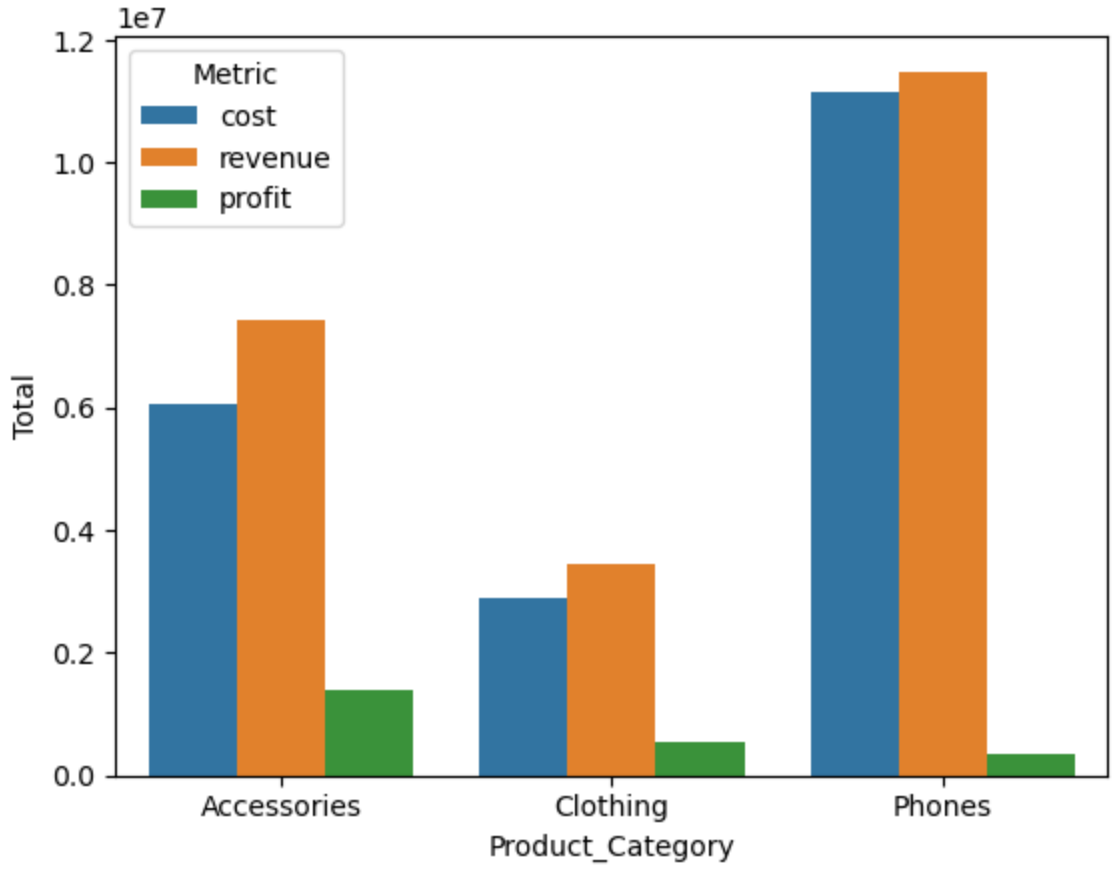


In [39]:
```
# Narration
```

Multivariate Analysis

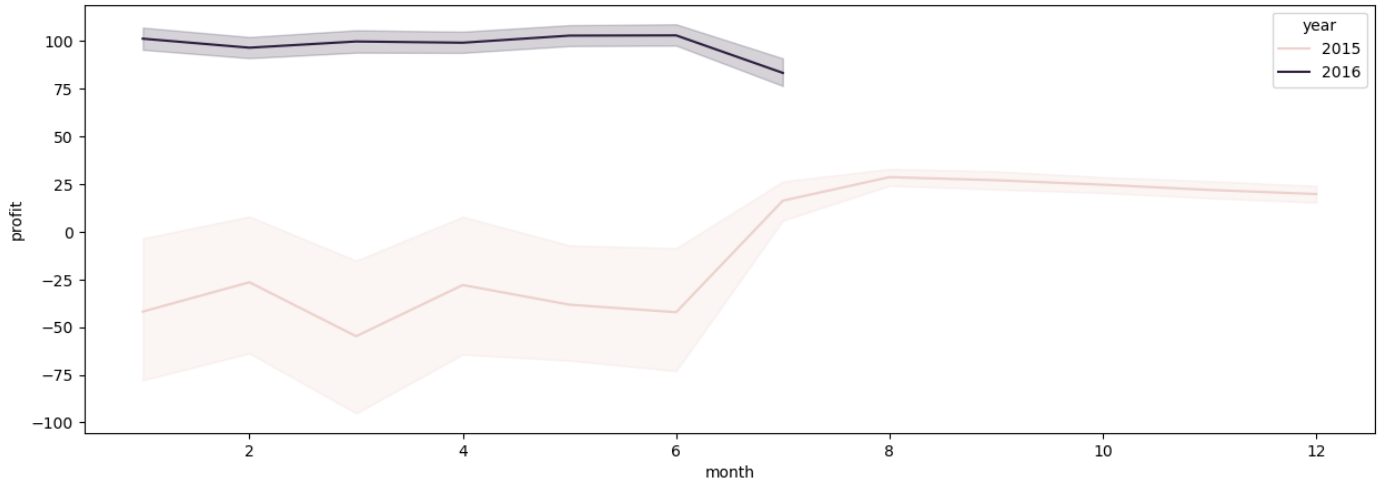
It involves analyzing the relationship between three or more variables.

In [40]:
```
# Product Category against cost, revenue and profit

procat = df.groupby("Product_Category")[["cost", "revenue", "profit"]].sum().reset_index()
procat = pd.melt(procat, id_vars="Product_Category", var_name="Metric", value_name="Total")
sns.barplot(data=procat, x='Product_Category', y="Total", hue="Metric");
```



In [41]:
```
# Narration
```

In [42]:
```
plt.figure(figsize=(15,5))
sns.lineplot(x='month', y="profit", data=df, hue='year');
```



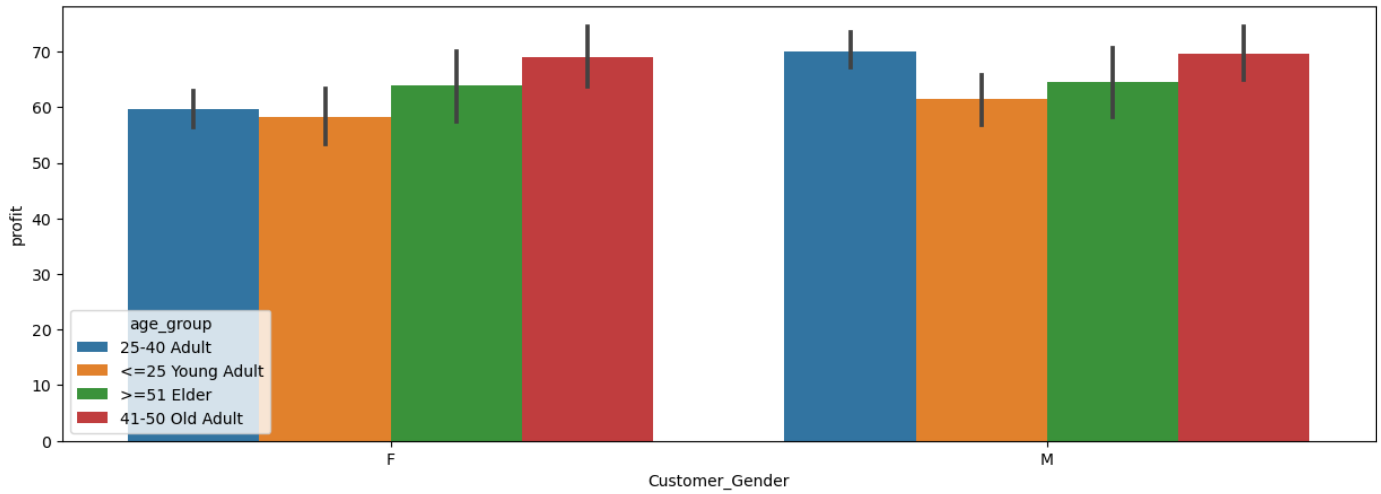
In [43]:
```
# Using pivot table
df.pivot_table(values='profit', index='year', columns='month', aggfunc='sum')
```

Out[43]:

month	1	2	3	4	5	6	7	8	9	10
year										
2015	-5778.70	-4079.65	-8163.79	-4945.69	-7801.45	-8690.61	15104.47	65926.24	62172.33	62950.66
2016	280204.25	263640.18	299777.67	307746.51	356915.59	357549.86	107151.79	NaN	NaN	NaN

In [44]:
```
# Narratiion
```

In [45]:
```
# Customer Gender, Age Group and Profit
plt.figure(figsize=(15,5))
sns.barplot(x="Customer_Gender", y='profit', data=df, hue="age_group");
```



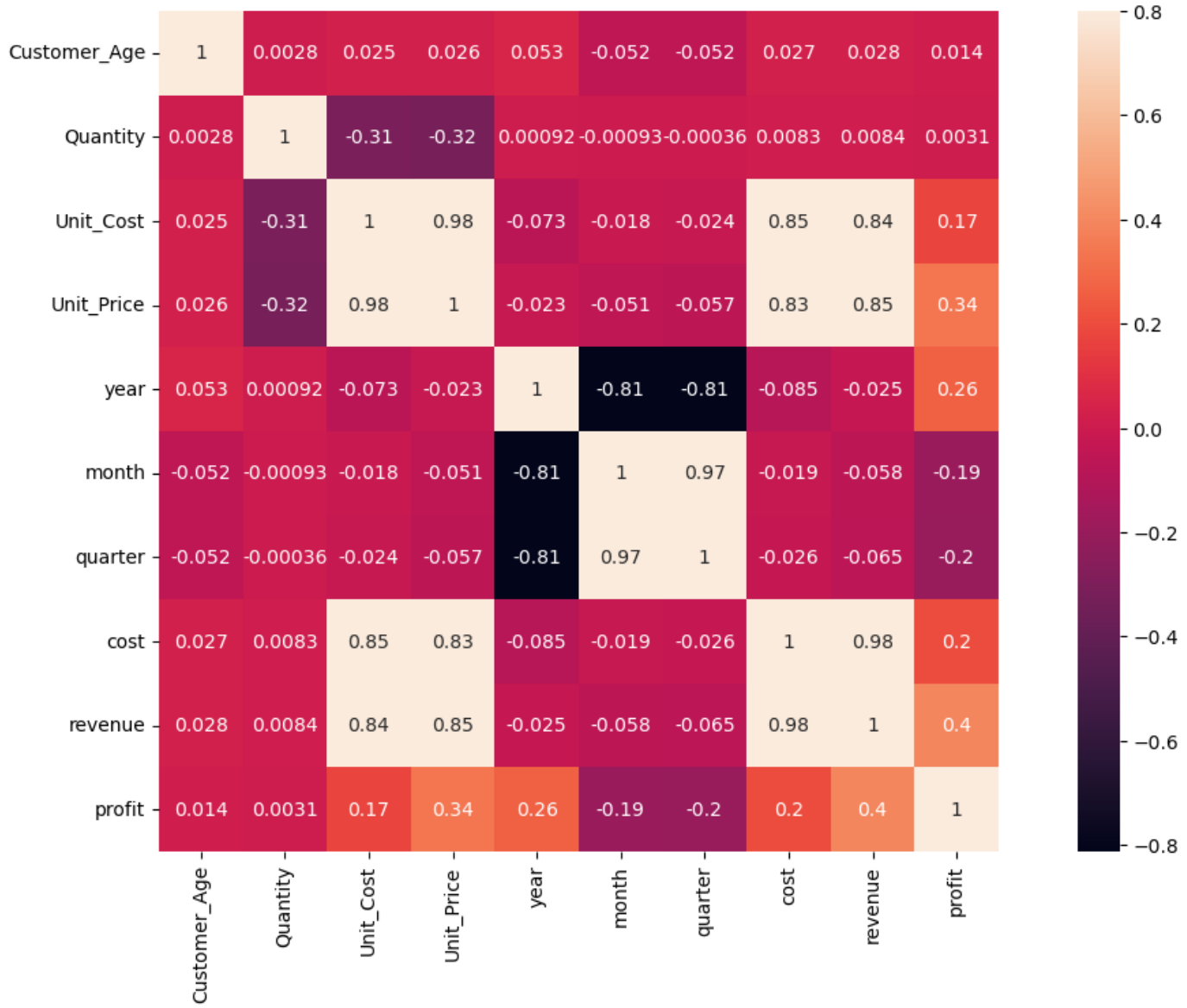
In [46]:
```
# Narration
```

In [47]:
```
# Correlation

import warnings
warnings.filterwarnings("ignore")

a = df.corr()
a
```

Out[47]:

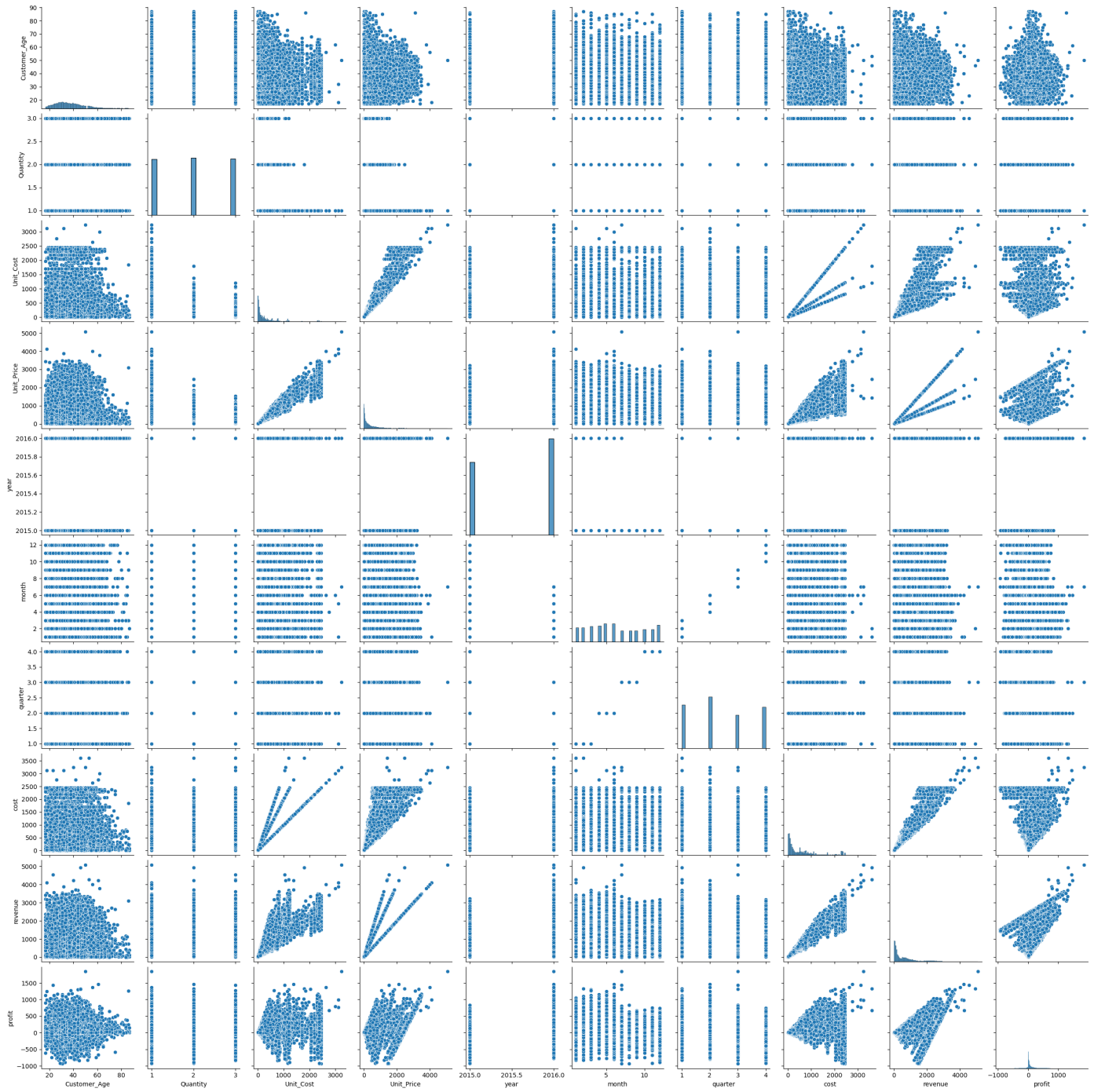
	Customer_Age	Quantity	Unit_Cost	Unit_Price	year	month	quarter	cost	revenu
Customer_Age	1.000000	0.002801	0.025360	0.026300	0.052688	-0.051610	-0.051932	0.026537	0.02776
Quantity	0.002801	1.000000	-0.312514	-0.324109	0.000919	-0.000925	-0.000358	0.008295	0.0084
Unit_Cost	0.025360	-0.312514	1.000000	0.981033	-0.073245	-0.018384	-0.024225	0.854908	0.8369
Unit_Price	0.026300	-0.324109	0.981033	1.000000	-0.022628	-0.051448	-0.057183	0.832969	0.8510
year	0.052688	0.000919	-0.073245	-0.022628	1.000000	-0.810662	-0.812493	-0.084566	-0.0253
month	-0.051610	-0.000925	-0.018384	-0.051448	-0.810662	1.000000	0.971628	-0.019345	-0.0584
quarter	-0.051932	-0.000358	-0.024225	-0.057183	-0.812493	0.971628	1.000000	-0.026190	-0.0652
cost	0.026537	0.008295	0.854908	0.832969	-0.084566	-0.019345	-0.026190	1.000000	0.9791
revenue	0.027762	0.008418	0.836957	0.851034	-0.025361	-0.058461	-0.065223	0.979119	1.0000
profit	0.013914	0.003097	0.171576	0.338499	0.259750	-0.194321	-0.195989	0.201260	0.3961

In [48]:
```
a = df.corr()
f, ax = plt.subplots(figsize=(15,8))
sns.heatmap(a, vmax=.8, square=True, annot=True);
```



In [49]:
```
# Narration
```

In [50]:
```
# Pairplot
sns.pairplot(df, size=2.5);
```


Customer_Age
Quantity
Unit_Cost
Unit_Price
year
month
quarter
cost
revenue
profit

In [ ]:

In [ ]: