DATA MINING

For Sales Product Ecommerce

Team member

Name	ID	Role
Salsabil Mohamed Asker	22010110	Preprocessing , Report
Marihan Emad Eldeen Mahmoud	22011531	Fuzzy Logic Algorithm
Salma Tarek Abd El meged Mohamed	22011629	Hierarchical Clustering Algorithm
Rowan Kamal Fayad	22010340	Visualization
Sarah Sameh Ahmed Mohamed	22011968	K-medoids Algorithm

Data set about Sales Product Ecommerce

Link for data (https://www.kaggle.com/datasets/ronakkantariya/e-commerce-salses-dataset)

This data explains about sales for some product such as (LG Dryer, Vareebadd Phone, Flatscreen TV, Google Phone, ThinkPad Laptop, …etc)

Columes in data (Order Date, Product Quantity, Ordered Price, Each Order Date, Customer Shipping Address, City Store, Category, Customer Gender, Customer Age Range, Discount)

Sample from data =>

Order ID	Product	Quantity Ordered	Price Each	Order Date	Customer Shipping Address	City Store	Category	Customer Gender	Customer Age Range	Discount
236670	Wired Headphon	16	11.99	8/31/2019 22:21	359 Spruce St, Seattle, WA 98101	Dallas	Headphone	Male	18-20	0,18
236671	Bose SoundSport	9	99.99	8/15/2019 15:11	492 Ridge St, Dallas, TX 75001	Los Angeles	Headphone	Male	21-25	0,21
236672	iPhone	8	700	8/6/2019 14:40	149 7th St, Portland, OR 97035	New York City	Phone	Male	26-30	0,05
236673	AA Batteries (4-p	12	3.84	8/29/2019 20:59	631 2nd St, Los Angeles, CA 90001	San Francisco	Batteries	Female	31-40	0,08
236674	AA Batteries (4-p	16	3.84	8/15/2019 19:53	736 14th St, New York City, NY 1000	Boston	Batteries	Female	41-50	0,14
236675	Wired Headphon	16	11.99	8/2/2019 23:54	470 Hill St, San Francisco, CA 94016	Dallas	Headphone	Female	50	0,22
236676	34in Ultrawide M	5	379.99	8/4/2019 19:52	470 Cherry St, Los Angeles, CA 9000	Los Angeles	Monitor	Female	18-20	0,17
236677	20in Monitor	3	109.99	8/13/2019 7:16	918 6th St, San Francisco, CA 94016	New York City	Monitor	Male	21-25	0,30
236678	Wired Headphon	4	11.99	8/25/2019 20:11	58 9th St, San Francisco, CA 94016	San Francisco	Headphone	Male	26-30	0,25
236679	Macbook Pro Lapt	20	1700	8/7/2019 15:43	239 Spruce St, Los Angeles, CA 9000	Boston	Laptop	Male	31-40	0,01
236680	LG Washing Mach	9	600	8/9/2019 19:38	967 Willow St, San Francisco, CA 94	Dallas	Machine	Female	41-50	0,04
236681	AA Batteries (4-p	9	3.84	8/26/2019 20:52	295 1st St, Boston, MA 02215	Los Angeles	Batteries	Female	50	0,27
236682	AA Batteries (4-p	12	3.84	8/19/2019 12:40	118 Johnson St, Portland, OR 97035	New York City	Batteries	Female	18-20	0,01
236683	27in FHD Monitor	10	149.99	8/31/2019 15:47	196 West St, Dallas, TX 75001	San Francisco	Monitor	Female	21-25	0,23
236684	Lightning Chargin	3	14.95	8/9/2019 16:50	669 12th St, New York City, NY 1000	Boston	Cable	Male	26-30	0,22
236685	Apple Airpods He	10	150	8/23/2019 19:29	238 Highland St, Atlanta, GA 30301	Dallas	Headphone	Male	31-40	0,23
236686	AAA Batteries (4-	8	2.99	8/15/2019 19:13	766 Maple St, Dallas, TX 75001	Los Angeles	Batteries	Male	41-50	0,24
236687	USB-C Charging C	2	11.95	8/23/2019 12:54	668 Meadow St, New York City, NY	New York City	Cable	Female	50	0,15
236688	34in Ultrawide M	12	379.99	8/8/2019 16:06	821 7th St, Los Angeles, CA 90001	San Francisco	Monitor	Female	18-20	0,00
236689	AAA Batteries (4-	12	2.99	8/21/2019 10:55	13 Cedar St, San Francisco, CA 9401	Boston	Batteries	Female	21-25	0,20
236690	AAA Batteries (4-	20	2.99	8/8/2019 12:00	139 River St, San Francisco, CA 9401	Dallas	Batteries	Female	26-30	0,13

Preprocessing for data

Read data into python =>

```
[2]: # read data
     data=pd.read_csv("Sales Product Ecommerce.csv")
     data.head()
```

[2]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Customer Shipping Address	City Store	Category	Customer Gender	Customer Age Range	Discount
0	236670	Wired Headphones	16	11.99	8/31/2019 22:21	359 Spruce St, Seattle, WA 98101	Dallas	Headphones	Male	18-20	0,18
1	236671	Bose SoundSport Headphones	9	99.99	8/15/2019 15:11	492 Ridge St, Dallas, TX 75001	Los Angeles	Headphones	Male	21-25	0,21
2	236672	iPhone	8	700	8/6/2019 14:40	149 7th St, Portland, OR 97035	New York City	Phone	Male	26-30	0,05
3	236673	AA Batteries (4-pack)	12	3.84	8/29/2019 20:59	631 2nd St, Los Angeles, CA 90001	San Francisco	Batteries	Female	31-40	0,08
4	236674	AA Batteries (4-pack)	16	3.84	8/15/2019 19:53	736 14th St, New York City, NY 10001	Boston	Batteries	Female	41-50	0,14

```
[3]: data.shape
[3]: (185950, 11)

[4]: data.size
[4]: 2045450

[5]: data.duplicated().sum()
[5]: 0

[6]: data.isnull().sum()
[6]: Order ID                   0
     Product                    0
     Quantity Ordered           0
     Price Each                 0
     Order Date                 0
     Customer Shipping Address  0
     City Store                 0
     Category                   0
     Customer Gender            0
     Customer Age Range         0
     Discount                   0
     dtype: int64
```

<= Shape of data (185950, 11)

<= Size of data 2045450

<= Data dose not have duplicated

<= Data dose not have null value

Information of data =>

```
[3]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 185950 entries, 0 to 185949
Data columns (total 11 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Order ID                   185950 non-null  int64
 1   Product                    185950 non-null  object
 2   Quantity Ordered           185950 non-null  int64
 3   Price Each                 185950 non-null  object
 4   Order Date                 185950 non-null  object
 5   Customer Shipping Address  185950 non-null  object
 6   City Store                 185950 non-null  object
 7   Category                   185950 non-null  object
 8   Customer Gender            185950 non-null  object
 9   Customer Age Range         185950 non-null  object
 10  Discount                   185950 non-null  object
dtypes: int64(2), object(9)
memory usage: 15.6+ MB
```

uiques data

```
[11]: data.nunique()

[11]: Order ID                     178437
      Product                          19
      Quantity Ordered                 48
      Price Each                       17
      Order Date                   142395
      Customer Shipping Address    140787
      City Store                        5
      Category                          9
      Customer Gender                   2
      Customer Age Range                6
      Discount                         31
      dtype: int64
```

Change type for columns (price each and quantity ordered) to can use them

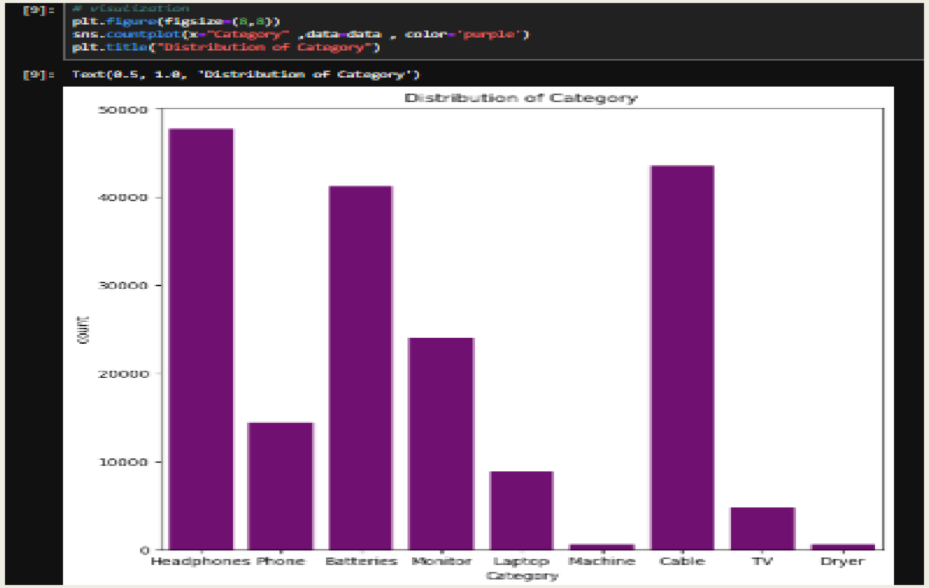
```
[7]: data['Price Each'] = pd.to_numeric(data['Price Each'], errors='coerce')
     data['Quantity Ordered'] = pd.to_numeric(data['Quantity Ordered'], errors='coerce')
     data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 185950 entries, 0 to 185949
Data columns (total 11 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Order ID                   185950 non-null  int64
 1   Product                    185950 non-null  object
 2   Quantity Ordered           185950 non-null  int64
 3   Price Each                 146668 non-null  float64
 4   Order Date                 185950 non-null  object
 5   Customer Shipping Address  185950 non-null  object
 6   City Store                 185950 non-null  object
 7   Category                   185950 non-null  object
 8   Customer Gender            185950 non-null  object
 9   Customer Age Range         185950 non-null  object
 10  Discount                   185950 non-null  object
dtypes: float64(1), int64(2), object(8)
memory usage: 15.6+ MB
```

minimum and maximum for price each

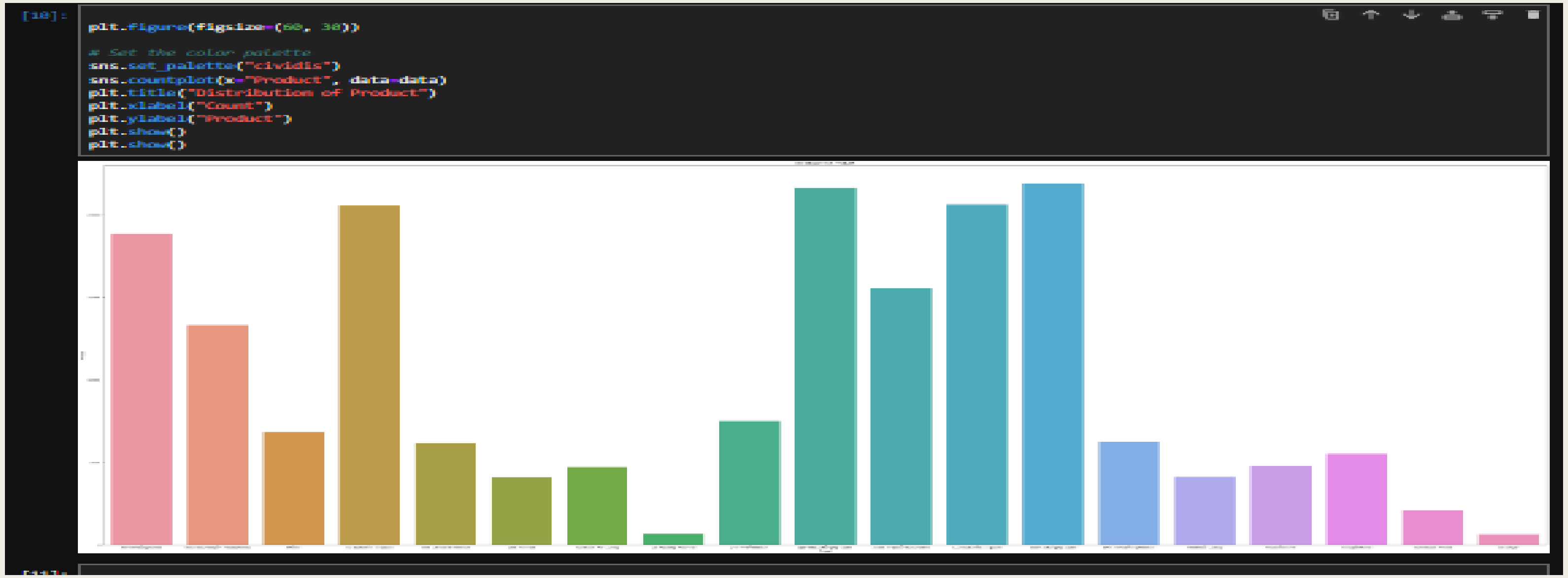
```
[9]: data['Price Each'].min()

[9]: 2.99

[10]: data['Price Each'].max()

[10]: 1700.0
```

Visualization for Data

Visualization For Category=> From this we predicted the Headphones and Cable have High category sales

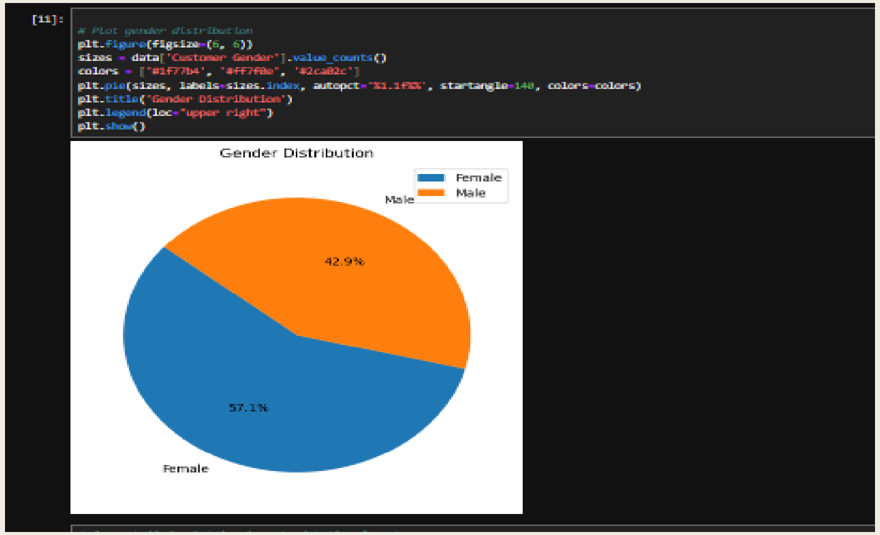


Visualization For Product => From this we predicted the Lightning Charging Cable and USB-C Charging Cable have High product sales and AAA Batteries (4-pack) have Less product sales

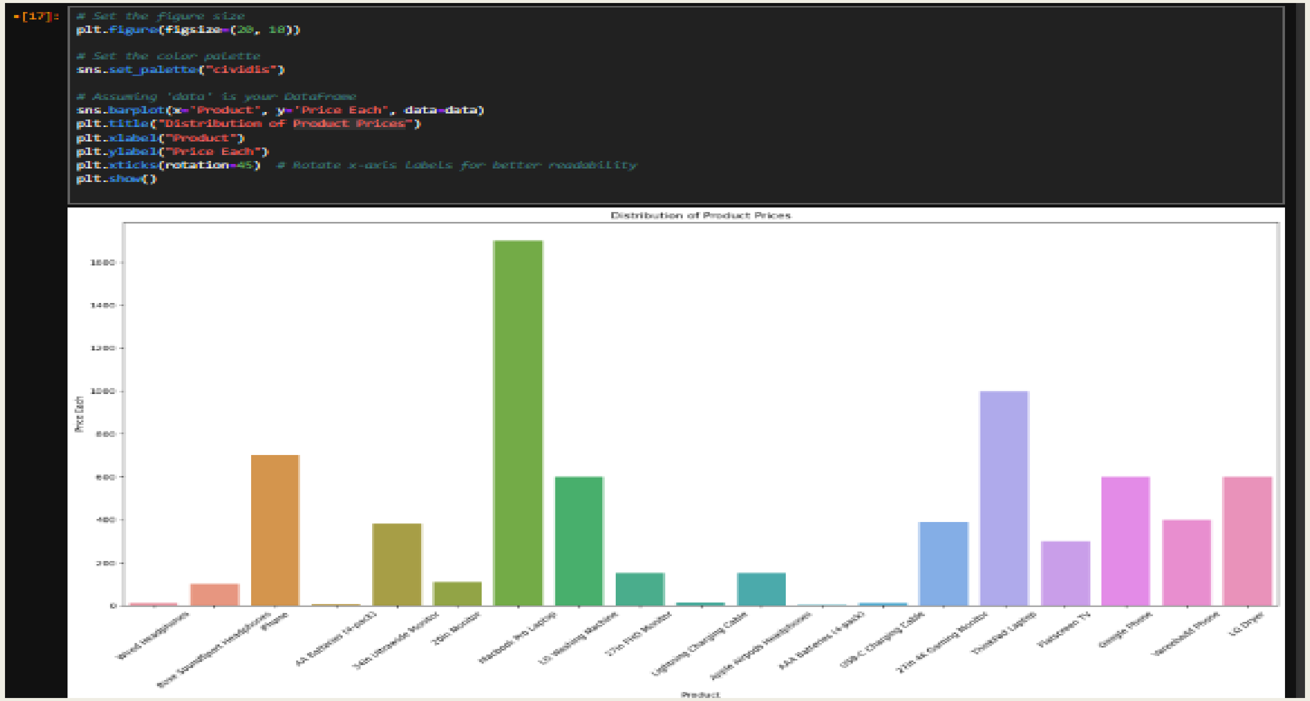


Visualization for Data

Pie chart For Customer Gender =>
From this we predicted the
Female buy more than men



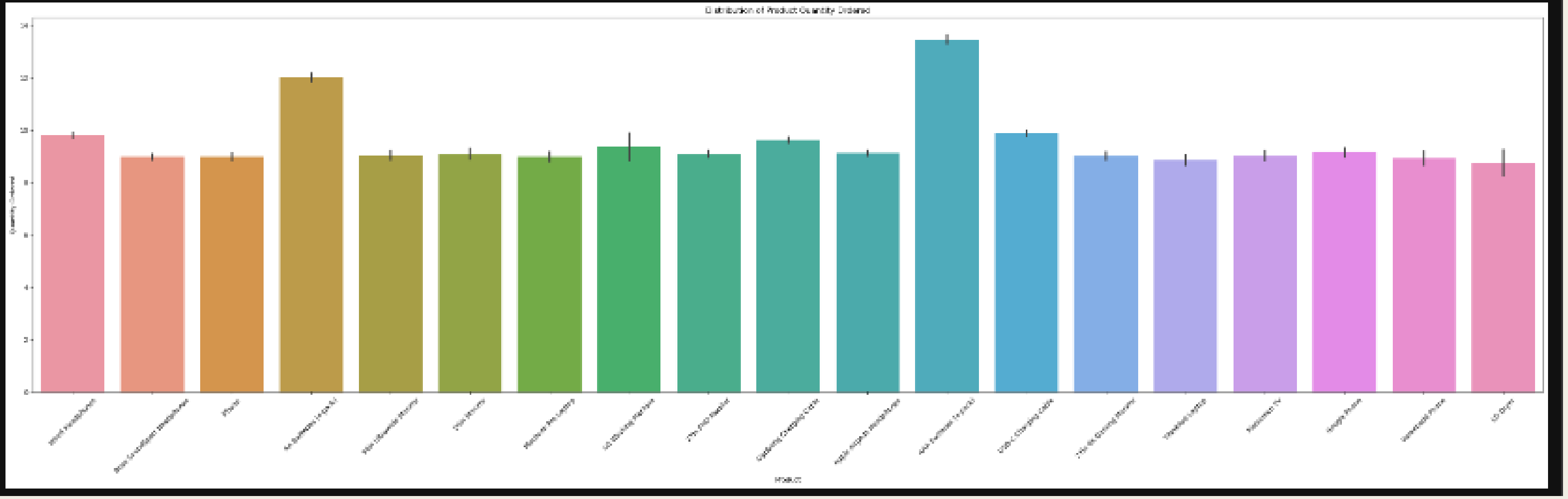
Histogram For Product Prices =>
From this we predicted the
Macbook Pro Laptop have
High price



Visualization for Data

Histogram For Product and Quantity Ordered
From this we predicted the AAA Batteries (4-pack) have a high Quantity Ordered

[15]:
```
# Set the figure size
plt.figure(figsize=(40, 10))

# Set the color palette
sns.set_palette("cividis")

# Assuming 'data' is your DataFrame
sns.barplot(x='Product', y='Quantity Ordered', data=data)
plt.title("Distribution of Product Quantity Ordered")
plt.xlabel("Product")
plt.ylabel("Quantity Ordered")
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```

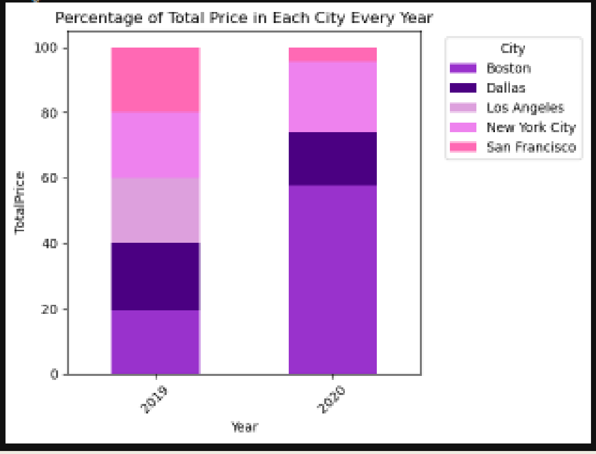


Visualization for Data

Stacked bar chart For Total Price in Each City Every Year =>
From this we predicted in 2019 all city have same sales
Or in 2020 Boston city have a high sales

```python
[12]: # Convert 'Order Date' column to datetime format
data['Order Date'] = pd.to_datetime(data['Order Date'])

# Extract just the year from the 'Order Date' column
data['Year'] = data['Order Date'].dt.year

# Calculate total price for each row, replace NaN values in 'Discount' with 0
data['TotalPrice'] = data['Price Each'] * data['Quantity Ordered']

# Group data by 'Year' and 'City Store' and calculate total price
grouped_data = data.groupby(['Year', 'City Store'])['TotalPrice'].sum().reset_index()

# Pivot the table to have 'Year' as rows, 'City Store' as columns, and 'TotalPrice' as values
pivot_table = grouped_data.pivot(index='Year', columns='City Store', values='TotalPrice')

# Calculate the percentage of total price for each city every year
percentage_per_city_every_year = pivot_table.div(pivot_table.sum(axis=1), axis=0) * 100

# Define custom colors for the bars
colors = ['darkorchid', 'indigo', 'plum', 'violet', 'hotpink']  # Add more colors as needed

# Plot the results with custom colors
plt.figure(figsize=(12, 8))
percentage_per_city_every_year.plot(kind='bar', stacked=True, color=colors)
plt.title('Percentage of Total Price in Each City Every Year')
plt.xlabel('Year')
plt.ylabel('TotalPrice')
plt.xticks(rotation=45)
plt.legend(title='City', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

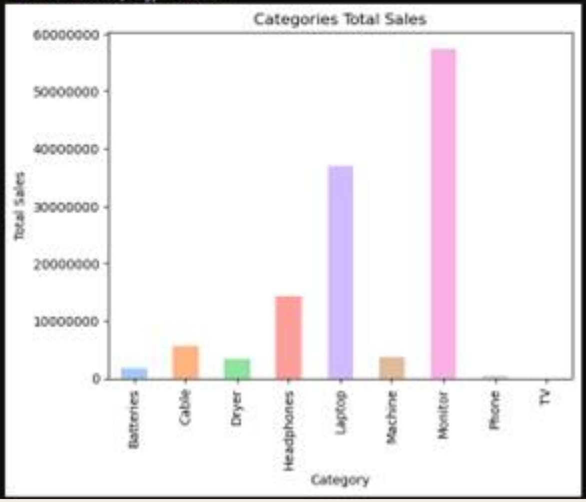
```
<Figure size 1200x800 with 0 Axes>
```



Histogram For Categories
Total Sales =>
From this we predicted
The Monitor category is
The highest sales return
And TV is lowest sales return

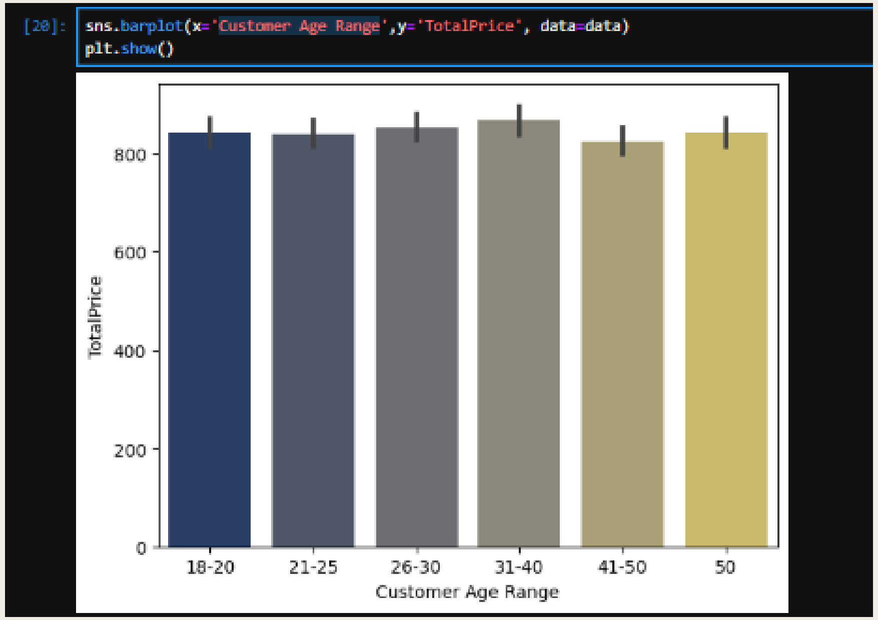
```python
[13]: categories_total_sales = data.groupby('Category')['TotalPrice'].sum()
print(categories_total_sales)

# show Total Sales for each Category

grouped_data = data.groupby('Category')['TotalPrice'].sum()

num_shades = len(grouped_data)
colors = sns.color_palette("pastel", num_shades)

grouped_data.plot(kind='bar', color=colors)
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.title('Categories Total Sales')
# Display actual values on the y-axis
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```

```
Category
Batteries      1780609.11
Cable          5703903.30
Dryer          3389400.00
Headphones    14311681.38
Laptop        37015733.97
Machine        3746400.00
Monitor       57435295.30
Phone           468300.00
TV               54300.00
Name: TotalPrice, dtype: float64
```

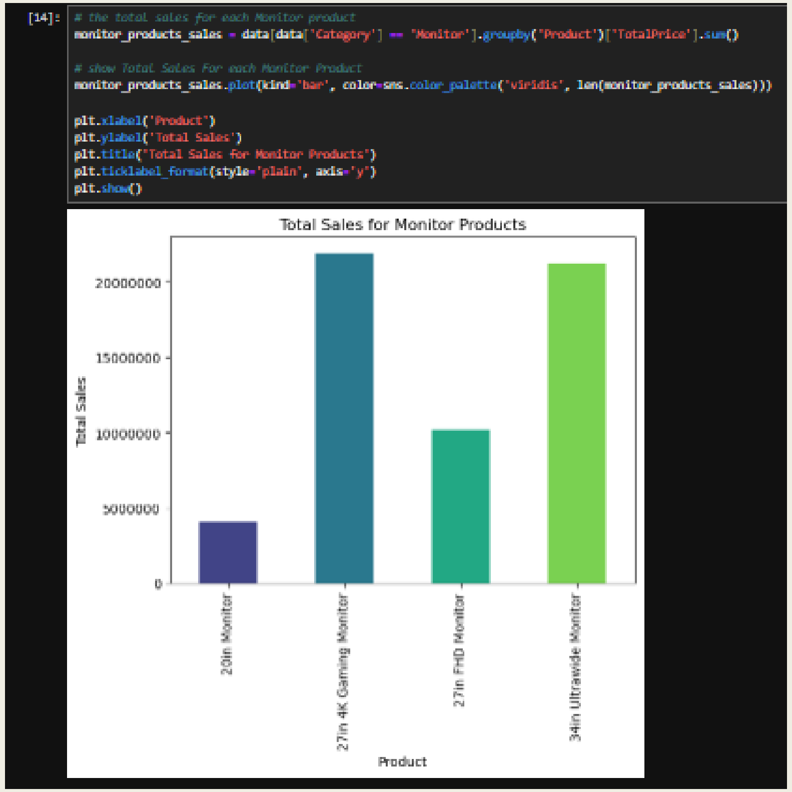


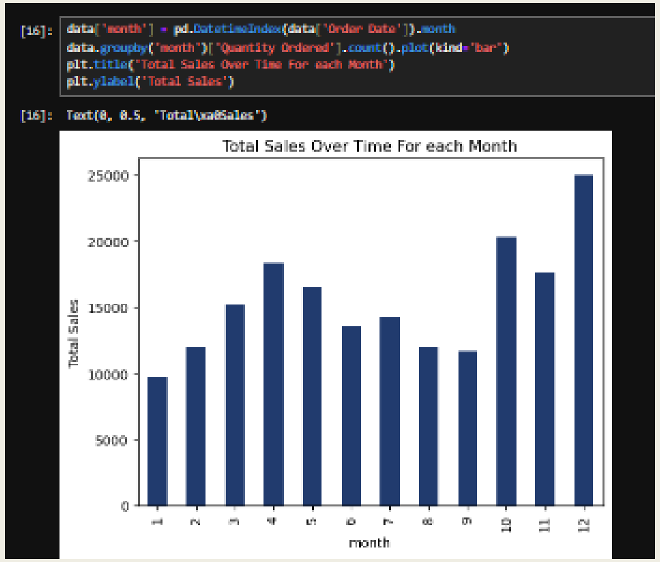
Visualization for Data

Histogram For Total Sales for Customer Age Range
From this we predicted the age between 31 and 40
Is the highest purchasing power



Histogram For Total Sales for Monitor Products
From this we predicted
The 27in 4K Gaming Monitor fron Monitor category is The highest sales return
And 20in Monitor is lowest sales return





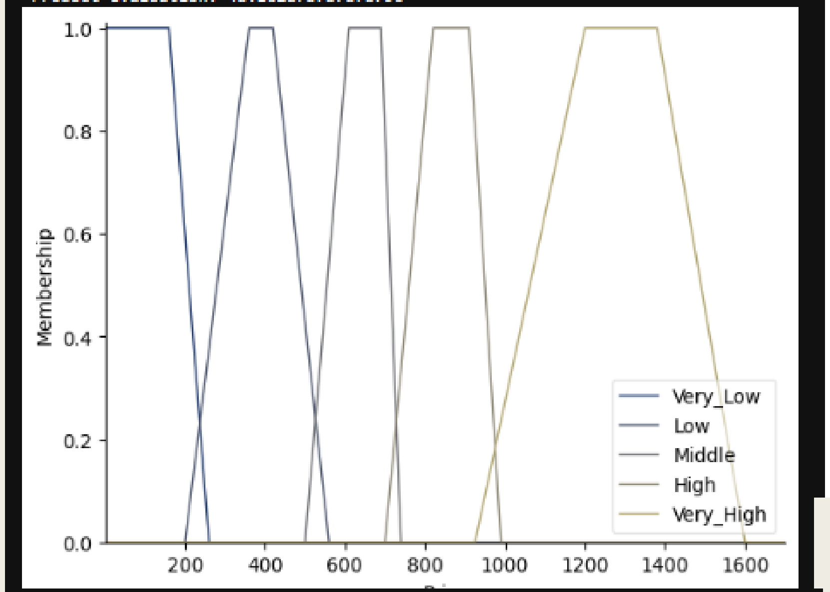
<= Histogram Total Sales Over Time For each Month
From this we predicted in December the highest sales and January is lowest

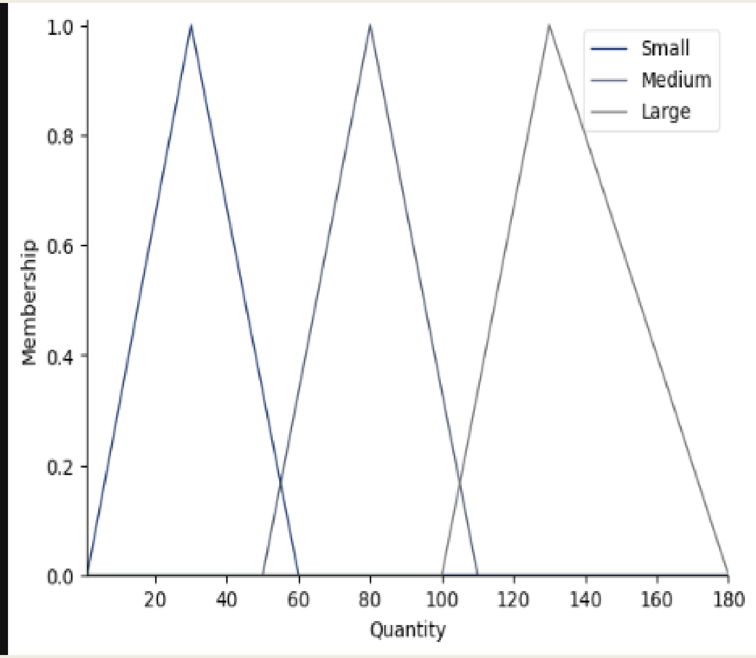
Fuzzy Logic Algorithm for data

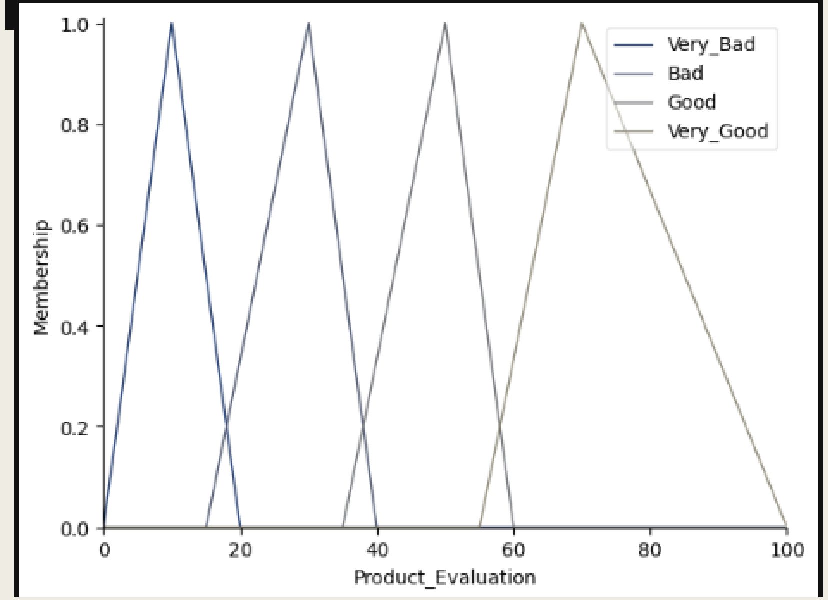
```
[32]: # Fuzzy Logic Algorithm
print ( " the smallest quantity = ",data['Quantity Ordered'].min())
print ( " the smallest price = ",data['Price Each'].min())
print ("the biggest quantity = ",data['Quantity Ordered'].max())
print ("the largest price = ",data['Price Each'].max())

 the smallest quantity =  1
 the smallest price =  2.99
the biggest quantity =  100
the largest price =  1700.0
```

```
[33]: Price = ctrl.Antecedent(np.arange(2.99, 1701.0, 1), 'Price')
Quantity = ctrl.Antecedent(np.arange(1, 181, 1), 'Quantity')

Product_Evaluation = ctrl.Consequent(np.arange(0.0, 101.0, 1.0), 'Product_Evaluation')

Price['Very_Low'] = fuzz.trapmf(Price.universe, [2.99, 2.99, 160, 260])
Price['Low'] = fuzz.trapmf(Price.universe, [200.0, 360.0, 420.0, 560.0])
Price['Middle'] = fuzz.trapmf(Price.universe, [500.0, 610.0, 690.0, 740.0])
Price['High'] = fuzz.trapmf(Price.universe, [700.0, 820.0, 910.0, 990.0])
Price['Very_High'] = fuzz.trapmf(Price.universe, [925.0, 1200.0, 1380.0, 1600.0])

Quantity['Small'] = fuzz.trimf(Quantity.universe, [1.0, 30.0, 60.0])
Quantity['Medium'] = fuzz.trimf(Quantity.universe, [50.0, 80.0, 110.0])
Quantity['Large'] = fuzz.trimf(Quantity.universe, [100.0, 130.0, 180.0])

Product_Evaluation["Very_Bad"] = fuzz.trimf(Product_Evaluation.universe, [0.0, 10.0, 20.0])
Product_Evaluation['Bad'] = fuzz.trimf(Product_Evaluation.universe, [15.0, 30.0, 40.0])
Product_Evaluation["Good"] = fuzz.trimf(Product_Evaluation.universe, [35.0, 50.0, 60.0])
Product_Evaluation["Very_Good"] = fuzz.trimf(Product_Evaluation.universe, [55.0, 70.0, 100.0])


Price.view()
Quantity.view()
Product_Evaluation.view()
# Define rules
rule1 = ctrl.Rule(Price['Very_Low'] & Quantity['Small'], Product_Evaluation['Very_Bad'])
rule2 = ctrl.Rule(Price['Very_Low'] & Quantity['Medium'], Product_Evaluation['Bad'])
rule3 = ctrl.Rule(Price['Very_Low'] & Quantity['Large'], Product_Evaluation['Good'])

rule4 = ctrl.Rule(Price['Low'] & Quantity['Small'], Product_Evaluation['Very_Bad'])
rule5 = ctrl.Rule(Price['Low'] & Quantity['Medium'], Product_Evaluation['Good'])
rule6 = ctrl.Rule(Price['Low'] & Quantity['Large'], Product_Evaluation['Good'])

rule7 = ctrl.Rule(Price['Middle'] & Quantity['Small'], Product_Evaluation['Bad'])
rule8 = ctrl.Rule(Price['Middle'] & Quantity['Medium'], Product_Evaluation['Good'])
rule9 = ctrl.Rule(Price['Middle'] & Quantity['Large'], Product_Evaluation['Very_Good'])

rule10 = ctrl.Rule(Price['High'] & Quantity['Small'], Product_Evaluation['Bad'])
rule11 = ctrl.Rule(Price['High'] & Quantity['Medium'], Product_Evaluation['Very_Good'])
rule12 = ctrl.Rule(Price['High'] & Quantity['Large'], Product_Evaluation['Very_Good'])

rule13 = ctrl.Rule(Price['Very_High'] & Quantity['Small'], Product_Evaluation['Very_Bad'])
rule14 = ctrl.Rule(Price['Very_High'] & Quantity['Medium'], Product_Evaluation['Very_Good'])
rule15 = ctrl.Rule(Price['Very_High'] & Quantity['Large'], Product_Evaluation['Very_Good'])

# Create control system
product_evaluation_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5,
                                              rule6, rule7, rule8, rule9, rule10,
                                              rule11, rule12, rule13, rule14, rule15])

product_evaluation = ctrl.ControlSystemSimulation(product_evaluation_ctrl)

# Input values
product_evaluation.input['Price'] = 300
product_evaluation.input['Quantity'] = 70

# Compute output
product_evaluation.compute()

# Output
print("Product Evaluation:", product_evaluation.output['Product_Evaluation'])
```

```
Product Evaluation: 48.162878787878796
```







Performing Fuzzy Logic Algorithm based on Price Each and Quantity Ordered to observation Product Evaluation to make easier to make decision

Hierarchical Clustering Algorithm for data

The final conclusion would typically involve identifying the number of clusters in data all of data point belong to 2 clusters , as well as understanding the relationships between different data points within each cluster. Additionally, hierarchical clustering can also provide insights into the overall structure and patterns in the data we make Standardize of data and perform hierarchical clustering using the linkage() function with method='ward'.

K-Medoids Clustering Algorithm for data

Performing K-medoids Clustering based on the Mean of each customer age range and their Total Sales per Purchase ,We decided to cluster them into 3 Clusters k = 3 .This Algorithm could help us in identifying the suitable offers or sales for each cluster in order to be able to maximize our Total sales profit.

```
[69]: #Array[Mean Age,Total Price]
      x=np.array(sample_data)[ : ,[13,12]]
      print(x)

[[35.5 4679.88]
 [25.0 17.94]
 [19.0 15.36]
 [28.0 89.69999999999999]
 [45.5 1499.85]
 [23.0 89.7]
 [25.0 3119.92]
 [25.0 107.55]
 [25.0 29.9]
 [28.0 11.99]
 [19.0 8.97]
 [23.0 35.97]
 [28.0 47.84]
 [25.0 1799.88]
 [25.0 599.9399999999999]
 [23.0 11.96]
 [19.0 143.88]
 [45.5 15.36]
 [19.0 449.97]
 [35.5 47.8]
 [25.0 399.96]
 [25.0 1319.8799999999999]
 [45.5 35.88]
 [28.0 2999.8]
 [23.0 5849.85]
 [19.0 14.950000000000001]
 [35.5 30.72]
 [23.0 149.5]
 [28.0 44.849999999999994]
```

```
[74]: plt.figure(figsize=(10,10))
      plt.scatter(x[:,0],x[:,1],c='r')

      plt.xlabel('Mean Age')
      plt.ylabel('Total Price')
      plt.title("Ages VS Sales")
      plt.grid()
      plt.show()
```