DATA ANALYSIS OF SUPERMARKET SALES

TABLE OF CONTENT

- **Introduction**
- **Motive**
- **Libraries used**
- **Exploring data**
- **Analysis**
- **Conclusion**
- **References**

INTRODUCTION

In this Project we will analyse the dataset of a supermarket sales. We will use a variety of methods to explore the dataset including descriptive statistics, data manipulation, data visualization.

Our analysis should provide valuable insight into customers preferences, needs etc,. additionally this project will demostrate our understanding of data analysis, techniques such as cleaning and exploring data.

MOTIVE

The main goal of this data visualization project is to make it easier to identify patterns, trends & outliers in large dataset and to gain insight into customer behaviour and preferences inorder to take decisions about product design.

Through analysing the data we hope to uncover the trends that can help the supermarket in understanding and serving the customer in a better way.

LIBRARIES USED

- **Pandas # for Data Manipulation**
- **Matplotlib # for Data Visualization**
- **Seaborn # for Data Visualization**

In [1]:

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

EXPLORING DATA

- Firstly we have to import the supermarket sales csv dataset using read method.

In [2]:

```
Supermarket_Sales=pd.read_csv("MarketSales1.csv")
Supermarket_Sales
```

Out[2]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gr ma percent
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	01-05-2019	13:08	Ewallet	522.83	4.761
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	03-08-2019	10:29	Cash	76.40	4.761
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	03-03-2019	13:23	Credit card	324.31	4.761
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	02-08-2019	10:37	Ewallet	604.17	4.761
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
494	437-53-3084	B	Mandalay	Normal	Male	Fashion accessories	99.89	2	9.9890	209.7690	2/26/2019	11:48	Ewallet	199.78	4.761
495	632-32-4574	B	Mandalay	Normal	Male	Sports and travel	75.92	8	30.3680	637.7280	3/20/2019	14:14	Cash	607.36	4.761
496	556-97-7101	C	Naypyitaw	Normal	Female	Electronic accessories	63.22	2	6.3220	132.7620	01-01-2019	15:51	Cash	126.44	4.761
497	862-59-8517	C	Naypyitaw	Normal	Female	Food and beverages	90.24	6	27.0720	568.5120	1/27/2019	11:17	Cash	541.44	4.761
498	401-18-8016	B	Mandalay	Member	Female	Sports and travel	98.13	1	4.9065	103.0365	1/21/2019	17:36	Cash	98.13	4.761

499 rows × 17 columns

To display all the rows in a dataset, we are using the below mentioned syntax

In [3]:

```
pd.options.display.max_rows=500
```

In [4]:

```
Supermarket_Sales
```

Out[4]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gros incom
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	01-05-2019	13:08	Ewallet	522.83	4.761905	26.141
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	03-08-2019	10:29	Cash	76.40	4.761905	3.820
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	03-03-2019	13:23	Credit card	324.31	4.761905	16.215
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.288
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	02-08-2019	10:37	Ewallet	604.17	4.761905	30.208

To view the top 5 rows of a DataFrame, we use head() method

In [5]:

```
Supermarket_Sales.head()
```

Out[5]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gros margi percentag
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	01-05-2019	13:08	Ewallet	522.83	4.76190
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	03-08-2019	10:29	Cash	76.40	4.76190
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	03-03-2019	13:23	Credit card	324.31	4.76190
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.76190
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	02-08-2019	10:37	Ewallet	604.17	4.76190

To view the bottom 5 rows of a DataFrame, we use tail() method

In [6]:

```
Supermarket_Sales.tail()
```

Out[6]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gr mai percent
494	437-53-3084	B	Mandalay	Normal	Male	Fashion accessories	99.89	2	9.9890	209.7690	2/26/2019	11:48	Ewallet	199.78	4.761
495	632-32-4574	B	Mandalay	Normal	Male	Sports and travel	75.92	8	30.3680	637.7280	3/20/2019	14:14	Cash	607.36	4.761
496	556-97-7101	C	Naypyitaw	Normal	Female	Electronic accessories	63.22	2	6.3220	132.7620	01-01-2019	15:51	Cash	126.44	4.761
497	862-59-8517	C	Naypyitaw	Normal	Female	Food and beverages	90.24	6	27.0720	568.5120	1/27/2019	11:17	Cash	541.44	4.761
498	401-18-8016	B	Mandalay	Member	Female	Sports and travel	98.13	1	4.9065	103.0365	1/21/2019	17:36	Cash	98.13	4.761

To view all the column titles of a DataFrame, we use columns method

In [7]:

```
Supermarket_Sales.columns
```

Out[7]:

```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
       'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',
       'Rating'],
      dtype='object')
```

To view the complete information of the DataFrame, we use info() method

In [8]:

```
Supermarket_Sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               499 non-null    object 
 1   Branch                   499 non-null    object 
 2   City                     499 non-null    object 
 3   Customer type            499 non-null    object 
 4   Gender                   499 non-null    object 
 5   Product line             499 non-null    object 
 6   Unit price               499 non-null    float64
 7   Quantity                 499 non-null    int64  
 8   Tax 5%                   499 non-null    float64
 9   Total                    499 non-null    float64
 10  Date                     499 non-null    object 
 11  Time                     499 non-null    object 
 12  Payment                  499 non-null    object 
 13  cogs                     499 non-null    float64
 14  gross margin percentage  499 non-null    float64
 15  gross income             499 non-null    float64
 16  Rating                   499 non-null    float64
dtypes: float64(7), int64(1), object(9)
memory usage: 66.4+ KB
```

ANALYSIS

1Q. How many male customers in Yangon city bought products using Credit cards?

- We used loc method to return a single or multiple specified rows.

In [9]:

```
Male_customers=(Supermarket_Sales.loc[(Supermarket_Sales.City=="Yangon")&(Supermarket_Sales.Gender=="Male")&(Supermarket_Sales.Pa
Male_customers
```

Out[9]:

	City	Gender	Payment	Quantity	cogs
2	Yangon	Male	Credit card	7	324.31
17	Yangon	Male	Credit card	6	435.66
18	Yangon	Male	Credit card	3	164.01
33	Yangon	Male	Credit card	2	193.16
63	Yangon	Male	Credit card	10	158.10
87	Yangon	Male	Credit card	7	345.66
149	Yangon	Male	Credit card	8	259.68
152	Yangon	Male	Credit card	9	749.16
155	Yangon	Male	Credit card	5	461.45
162	Yangon	Male	Credit card	7	320.53
167	Yangon	Male	Credit card	10	989.80
169	Yangon	Male	Credit card	7	486.64
176	Yangon	Male	Credit card	8	177.36
194	Yangon	Male	Credit card	5	163.55
215	Yangon	Male	Credit card	1	18.28
240	Yangon	Male	Credit card	9	224.46
241	Yangon	Male	Credit card	2	119.54
248	Yangon	Male	Credit card	4	310.88
256	Yangon	Male	Credit card	1	66.35
268	Yangon	Male	Credit card	4	282.96
279	Yangon	Male	Credit card	10	440.20
324	Yangon	Male	Credit card	6	129.12
331	Yangon	Male	Credit card	3	98.70
333	Yangon	Male	Credit card	2	46.96
360	Yangon	Male	Credit card	8	647.68
369	Yangon	Male	Credit card	9	193.50
380	Yangon	Male	Credit card	4	329.32
405	Yangon	Male	Credit card	4	269.04
406	Yangon	Male	Credit card	5	68.95
437	Yangon	Male	Credit card	6	203.94
464	Yangon	Male	Credit card	5	256.70
472	Yangon	Male	Credit card	10	431.30
475	Yangon	Male	Credit card	3	195.54

From the above DataFrame we can observe that a total of 33 male customers have bought the products using Credit cards.

2Q. Give a brief sketch of products bought by customers in all the cities?

- We used groupby() method which allows us to group your data and execute functions on these groups.
- We used count() method that returns the count of values of all the columns at once.
- We used unstack() method that converts the specified row levels to column levels.

In [10]:

```
Supermarket_Sales.groupby(["City","Gender","Product line"])["City"].count().unstack()
```

Out[10]:

	Product line	Electronic accessories	Fashion accessories	Food and beverages	Health and beauty	Home and lifestyle	Sports and travel
City	**Gender**						
Mandalay	**Female**	11	20	14	12	6	14
	Male	12	13	8	16	16	16
Naypyitaw	**Female**	19	17	22	13	12	15
	Male	13	19	17	10	10	8
Yangon	**Female**	14	8	7	11	17	13
	Male	17	8	18	15	20	18

In the above DataFrame, it is clearly displayed the complete sketch of the products brought by the customers in all the cities from different category of accessories.

3Q. What is the average gross income of all the branches?

- We used mean() method which returns the mean of values over requested axis.
- We used groupby() method which allows us to group your data and execute functions on these groups.

In [11]:

```
Supermarket_Sales.groupby("Branch")[["gross income"]].mean()
```

Out[11]:

	gross income
Branch	
A	15.216750
B	15.885835
C	16.002800

From the above DataFrame, we can observe that the average gross income of Branch A is 15.2167, Branch B is 15.8858 and Branch B is 16.0028.

4Q. Give a Detailed analysis of all the payment methods used by male and female customers?

- We used groupby() method which allows us to group your data and execute functions on these groups.
- We used describe() method which returns some statistical details like percentage, mean, standard deviation, count, maximum and minimum values etc.

In [12]:

```
Supermarket_Sales.groupby(["Payment","Gender"])[["Unit price","Quantity"]].describe()
```

Out[12]:

		Unit price								Quantity							
		count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Payment	**Gender**																
Cash	**Female**	96.0	55.312708	28.681538	12.03	28.8225	54.705	81.4700	99.69	96.0	5.729167	2.943250	1.0	4.00	6.0	8.0	10.0
	Male	90.0	53.659667	23.758991	11.81	32.8450	52.540	74.5475	99.78	90.0	5.588889	2.879195	1.0	4.00	6.0	8.0	10.0
Credit card	**Female**	76.0	51.773158	27.424834	10.59	29.3650	45.820	74.0475	99.73	76.0	6.131579	2.945469	1.0	3.75	6.5	9.0	10.0
	Male	77.0	53.284675	25.494503	13.79	33.2000	49.040	71.8600	99.96	77.0	5.623377	2.860985	1.0	3.00	6.0	8.0	10.0
Ewallet	**Female**	73.0	58.746438	26.778614	10.96	34.7000	60.880	79.7400	99.71	73.0	5.808219	2.826745	1.0	4.00	6.0	8.0	10.0
	Male	87.0	56.413678	28.830931	12.78	27.6750	53.440	85.7150	99.89	87.0	5.310345	2.954344	1.0	3.00	5.0	8.0	10.0

In the above DataFrame we can clearly observe the complete analysis of the payment methods used by all the customers.

5Q. Give the statistical data of total Unit price?

- We used describe() method which returns some statistical details like percentage, mean, standard deviation, count, maximum and minimum values etc.

In [13]:

```
Supermarket_Sales["Unit price"].describe()
```

Out[13]:

```
count    499.000000
mean      54.856814
std       26.875044
min       10.590000
25%       30.510000
50%       52.590000
75%       77.825000
max       99.960000
Name: Unit price, dtype: float64
```

In the above DataFrame, it is clearly displayed the statistics of Unit price.

6Q. List out the top ten sales with highest gross income?

- We used sort_values() method that sorts the DataFrame by the specified label.

In [14]:

```
highest_gross_income=Supermarket_Sales.sort_values("gross income",ascending=False).head(10)
highest_gross_income
```

Out[14]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	g ma percen
350	860-79-0874	C	Naypyitaw	Member	Female	Fashion accessories	99.30	10	49.6500	1042.6500	2/15/2019	14:53	Credit card	993.00	4.76
167	687-47-8271	A	Yangon	Normal	Male	Fashion accessories	98.98	10	49.4900	1039.2900	02-08-2019	16:20	Credit card	989.80	4.76
422	271-88-8734	C	Naypyitaw	Member	Female	Fashion accessories	97.21	10	48.6050	1020.7050	02-08-2019	13:00	Credit card	972.10	4.76
166	234-65-2137	C	Naypyitaw	Normal	Male	Home and lifestyle	95.58	10	47.7900	1003.5900	1/16/2019	13:32	Cash	955.80	4.76
357	554-42-2417	C	Naypyitaw	Normal	Female	Sports and travel	95.44	10	47.7200	1002.1200	01-09-2019	13:45	Cash	954.40	4.76
429	325-77-6186	A	Yangon	Member	Female	Home and lifestyle	90.65	10	45.3250	951.8250	03-08-2019	10:53	Ewallet	906.50	4.76
141	280-17-4359	C	Naypyitaw	Member	Male	Health and beauty	90.50	10	45.2500	950.2500	1/25/2019	13:48	Cash	905.00	4.76
122	219-22-9386	B	Mandalay	Member	Male	Sports and travel	99.96	9	44.9820	944.6220	03-09-2019	17:26	Credit card	899.64	4.76
140	731-81-9469	C	Naypyitaw	Member	Female	Sports and travel	89.80	10	44.9000	942.9000	1/23/2019	13:00	Credit card	898.00	4.76
209	817-69-8206	B	Mandalay	Normal	Female	Electronic accessories	99.73	9	44.8785	942.4485	03-02-2019	19:42	Credit card	897.57	4.76

In the above DataFrame, we can see that it is clearly sorted the top ten sales with highest gross income.

7Q. What is the total Unit price, Quantity, Gross income in each city?

- We used groupby() method which allows us to group your data and execute functions on these groups.
- We used sum() method that returns the total sum of values over requested axis.

In [15]:

```
city_sales=Supermarket_Sales.groupby("City").sum()
city_sales[["Unit price","Quantity","gross income"]]
```

Out[15]:

City	Unit price	Quantity	gross income
Mandalay	8813.28	898	2509.9620
Naypyitaw	9758.62	976	2800.4900
Yangon	8801.65	964	2525.9805

From the above DataFrame, we observe the total Unit prices, Quantity, Gross income of the 3 cities.

8Q. List out the total no.of sales in each category?

- We used groupby() method which allows us to group your data and execute functions on these groups.
- We used count() method that returns the count of values of all the columns at once.

In [16]:

```
Catagory_of_Accessories=Supermarket_Sales.groupby("Product line").count()
Catagory_of_Accessories[["Invoice ID","Total"]]
```

Out[16]:

Product line	Invoice ID	Total
Electronic accessories	86	86
Fashion accessories	85	85
Food and beverages	86	86
Health and beauty	77	77
Home and lifestyle	81	81
Sports and travel	84	84

In the above DataFrame, it clearly shows the list of total no.of sales in each category.

9Q. Plot the Gross income of top 15 sales of the DataFrame using line graph?

- Here head() method returns the first 15 sales of the DataFrame.

In [17]:

```
First_15_sales=Supermarket_Sales["gross income"].head(15)
First_15_sales
```

Out[17]:

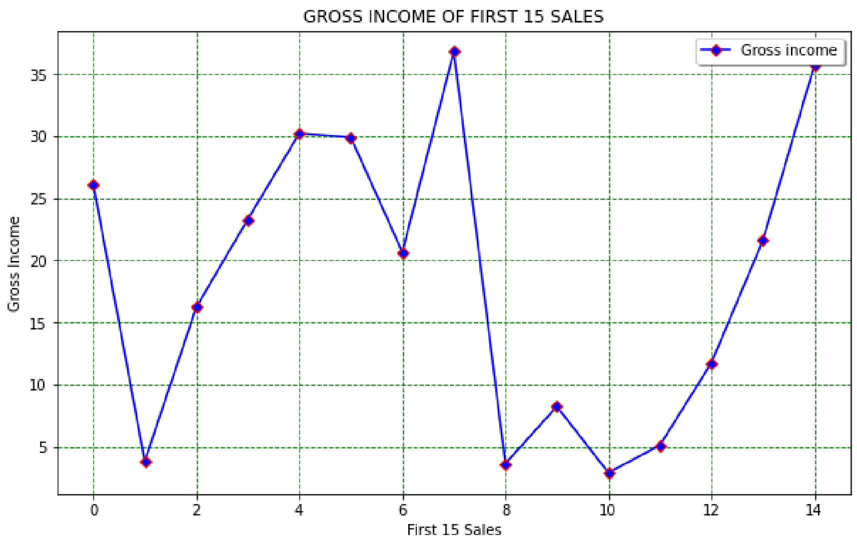
```
0     26.1415
1      3.8200
2     16.2155
3     23.2880
4     30.2085
5     29.8865
6     20.6520
7     36.7800
8      3.6260
9      8.2260
10     2.8960
11     5.1020
12    11.7375
13    21.5950
14    35.6900
Name: gross income, dtype: float64
```

- We used figsize() method for adjusting the size of the graph
- We used plot() method for plotting the graph

- In plot() method we have used the parameters like; marker for styling the point in the graph, color to adjust the color of the lines in the graph, mec to give the edge color of the point in the graph.
- Here xlabel(), ylabel() methods are used to label the x-axis and y-axis.
- We used title() method that returns the title of the graph.
- We used grid() method that adds the grid lines to the graph.
- we used legend() method that adds the list of explaination for each wedge.
- show() method is used to display the graph.

In [18]:

```
plt.figure(figsize=(10,6))
plt.plot(First_15_sales,marker="D",color="b",mec="r")
plt.xlabel("First 15 Sales")
plt.ylabel("Gross Income")
plt.title("GROSS INCOME OF FIRST 15 SALES")
plt.grid(color="g",linestyle="--")
plt.legend(["Gross income"],shadow=True,)
plt.show()
```



10Q. Plot all the category of sales based on the total no.of units sold.

- Here unique() method finds the uniques elements of an array and returns these unique elements as a sorted array.

In [19]:

```
catagory_of_Accessories=Supermarket_Sales["Product line"].unique()
catagory_of_Accessories
```

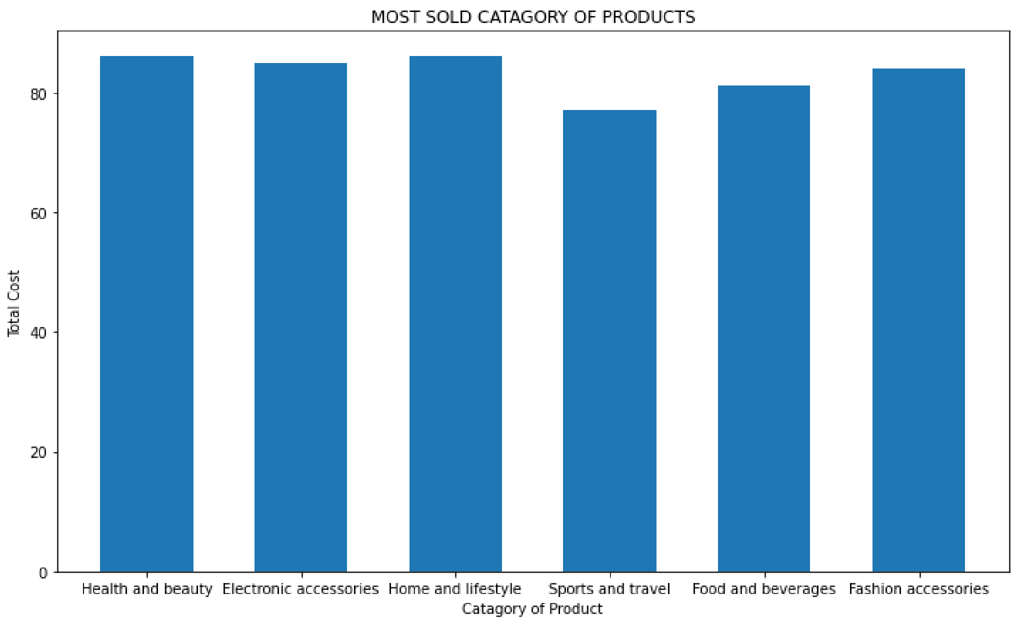
Out[19]:

```
array(['Health and beauty', 'Electronic accessories',
       'Home and lifestyle', 'Sports and travel', 'Food and beverages',
       'Fashion accessories'], dtype=object)
```

- We used figsize() method for adjusting the size of the graph.
- Here we used bar() method to represent the data in the form ofs bar graph.
- Here xlabel(), ylabel() methods are used to label the x-axis and y-axis.
- We used title() method that returns the title of the graph.
- show() method is used to display the graph.

In [20]:

```
plt.figure(figsize=(12,7))
plt.bar(catagory_of_Accessories,Catagory_of_Accessories["Total"],width=0.6)
plt.xlabel("Catagory of Product")
plt.ylabel("Total Cost")
plt.title("MOST SOLD CATAGORY OF PRODUCTS")
plt.show()
```



11Q. Plot the percentage of shares of Branches' A, B & C using Pie Chart.

- We used loc method to return a single or multiple specified rows.
- We used count() method that returns the count of values of all the columns at once.

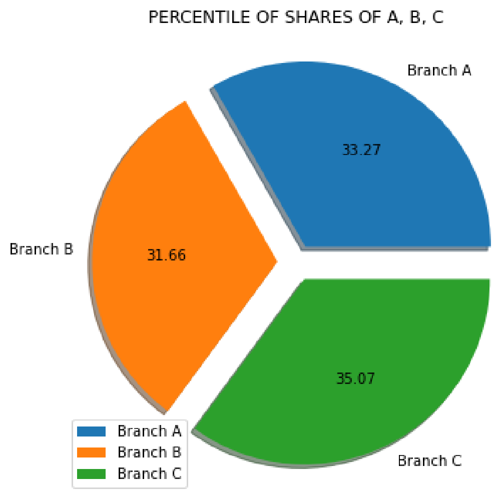
In [21]:

```
data1=Supermarket_Sales.loc[Supermarket_Sales["Branch"]=="A"].count()[0]
data2=Supermarket_Sales.loc[Supermarket_Sales["Branch"]=="B"].count()[0]
data3=Supermarket_Sales.loc[Supermarket_Sales["Branch"]=="C"].count()[0]
```

- we used figsize() method for adjusting the size of the graph
- Here we used pie() method to represent the data in the form of pie chart.
- we used title() method that returns the title of the graph.
- we used legend() method that adds the list of explaination for each wedge.
- show() method is used to display the graph.

In [22]:

```
names=["Branch A","Branch B","Branch C"]
expand=[.1,.1,.1]
plt.figure(figsize=(8,6))
plt.pie([data1,data2,data3],labels=names,explode=expand,autopct="%.2f",shadow=True)
plt.title("PERCENTILE OF SHARES OF A, B, C")
plt.legend()
plt.show()
```



12Q. Compare the Gross income of Naypyitaw and Yangon using Box plot.

- we are using loc method to return a single or multiple specified rows.

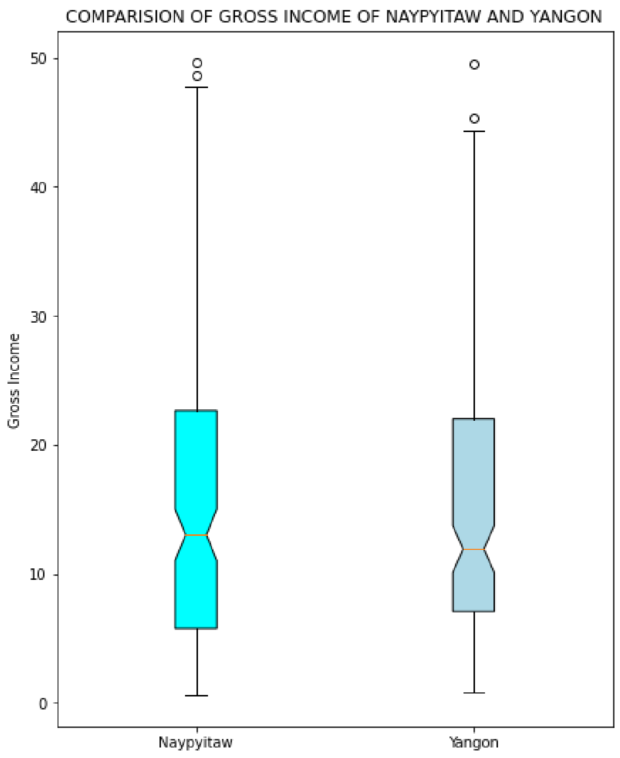
In [23]:

```
Naypyitaw=Supermarket_Sales.loc[Supermarket_Sales["City"]=="Naypyitaw"]["gross income"]
Yangon=Supermarket_Sales.loc[Supermarket_Sales["City"]=="Yangon"]["gross income"]
```

- We used figsize() method for adjusting the size of the graph.
- Here we used boxplot() method to represent the data in the form of box plot.
- In boxplot() method, we have used the parameters like; labels to label the plots, patch_artist, which is used to fill the boxplot with colors and notch used to narrow the box around the median of the box plot.
- We used title() method that returns the title of the graph.
- show() method is used to display the graph.

In [24]:

```
labels=["Naypyitaw","Yangon"]
plt.figure(figsize=(7,9))
box=plt.boxplot([Naypyitaw,Yangon],labels=labels,patch_artist=True,notch=True)
plt.ylabel("Gross Income")
plt.title("COMPARISION OF GROSS INCOME OF NAYPYITAW AND YANGON")
colors = ['cyan', 'lightblue']
for patch, color in zip(box['boxes'], colors):
    patch.set_facecolor(color)
plt.show()
```



13Q. Plot the top 5 sales based on total revenue in bar plot.

- Here, we used sort_values() method that sorts the DataFrame by the specified label.

In [25]:

```
Highest_Citysales=Supermarket_Sales[["Invoice ID","Branch","City","Quantity","Total"]].sort_values(by="Total",ascending=False)[0:
Highest_Citysales
```

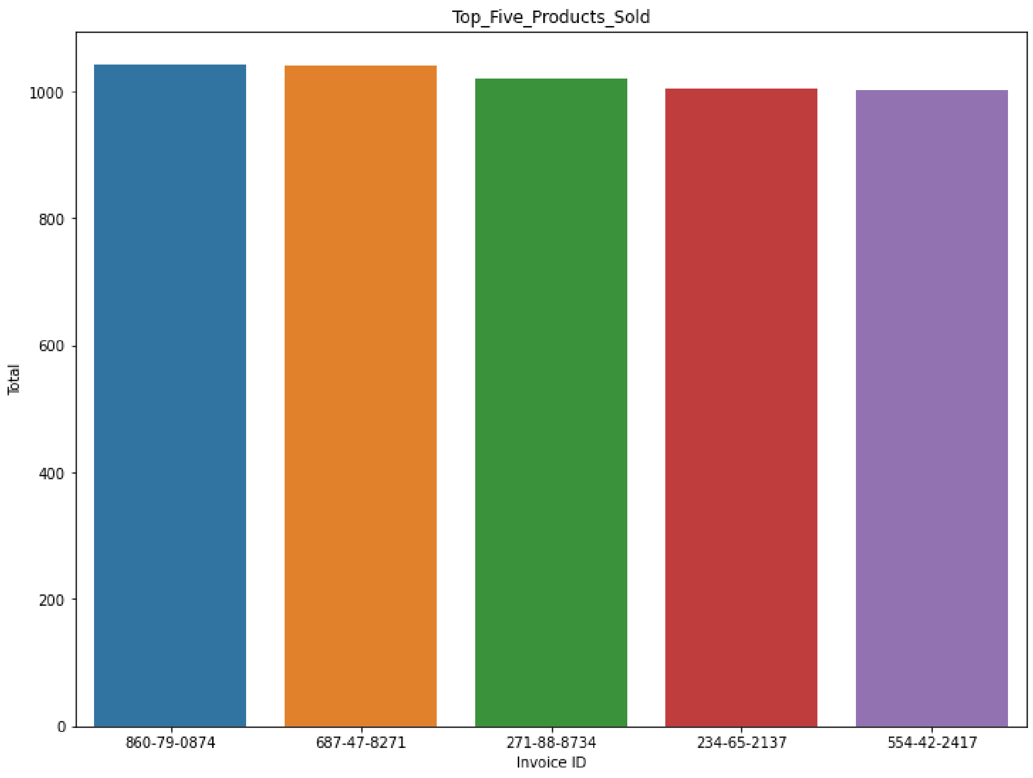
Out[25]:

	Invoice ID	Branch	City	Quantity	Total
350	860-79-0874	C	Naypyitaw	10	1042.650
167	687-47-8271	A	Yangon	10	1039.290
422	271-88-8734	C	Naypyitaw	10	1020.705
166	234-65-2137	C	Naypyitaw	10	1003.590
357	554-42-2417	C	Naypyitaw	10	1002.120

- We used figsize() method for adjusting the size of the graph.
- We used title() method that returns the title of the graph.
- show() method is used to display the graph.

In [26]:

```
plt.figure(figsize=(12,9))
sns.barplot(y="Total",x="Invoice ID",data=Highest_Citysales)
plt.title("Top_Five_Products_Sold")
plt.show()
```

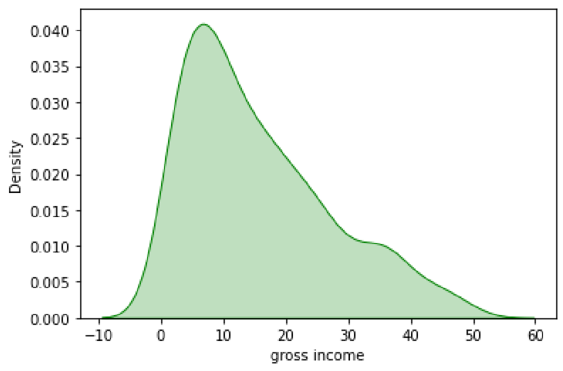


14Q. Plot the Gross income using Kdeplot.

- Here we used kdeplot() method to represent the graph in the form of kdeplot.
- In method kdeplot(), we have used the parameters like color used to give the color to the plot and shade is used to fill the area covered by curve in the graph.
- show() method is used to display the graph.

In [27]:

```
sns.kdeplot(x="gross income",data=Supermarket_Sales,color="green",shade=True)
plt.show()
```



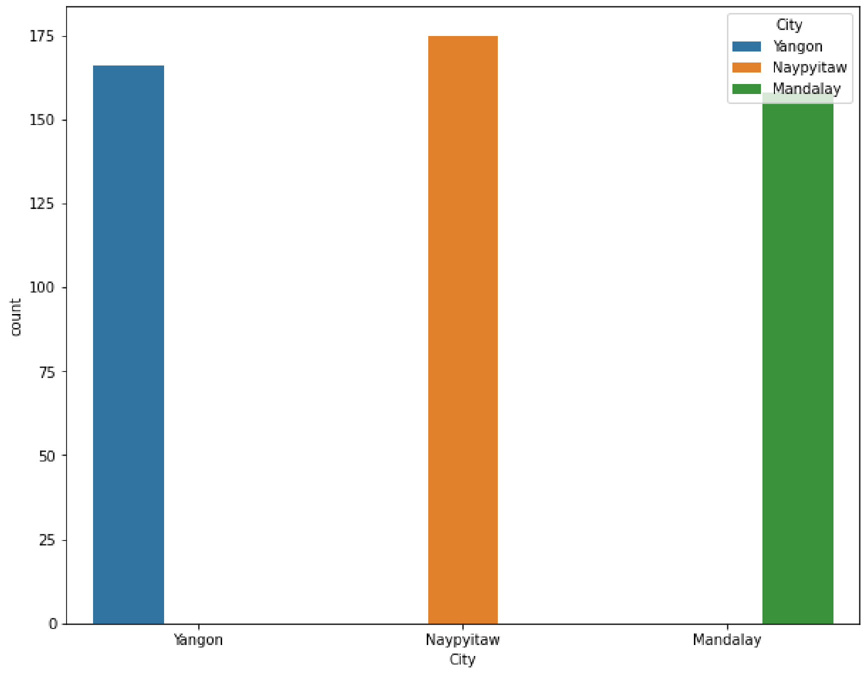
15Q. Plot the total no.of sales of each city using Count plot.

- We used figsize() method for adjusting the size of the graph.
- We used countplot() method to represent the data in the form of Count Plot.
- show() method is used to display the graph.

In [28]:

```
plt.figure(figsize=(10,8))
sns.countplot("City",data=Supermarket_Sales,hue="City")
plt.show()
```

```
C:\Users\Sanjay\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
s a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

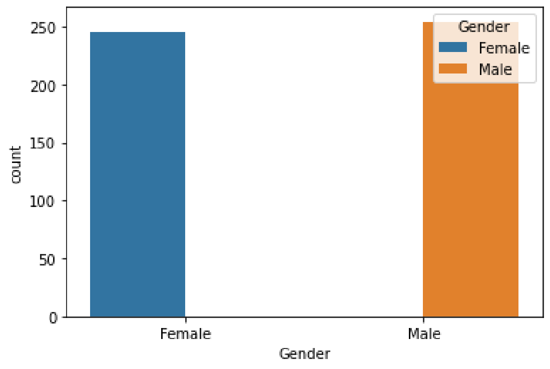


16Q. Plot the total no. of sales done by male and female customers using Count plot.

- We used countplot() method to represent the data in the form of Count Plot.
- show() method is used to display the graph.

In [29]:

```
sns.countplot(x="Gender",data=Supermarket_Sales,hue="Gender")
plt.show()
```



CONCLUSION

By this Data Analysis Project of Supermarket Sales using python, we have been able to provide a valuable insights into the sales pattern of source.

We have been able to identify the most profitable items, least profitable items and regional variations according to the Branchs', Payment method, City, Customer Type, Quantity, Product Line, Gender, and ratings of customers and it has also been able to identify future sales using this project.

With this project, the store can adjust the pricing, category of sales depending upon the locality and they can likewise further develop their marketing strategies to maximize the sales and profits.

This data analysis helped them to understanding their sales and made them to take portful choices for what's to come.

Finally, as opposed to using various technologies, this project can help you in getting the precise report in a brief time frame on the most proficient method to continue further.

REFERENCES

- **https://www.google.co.in (https://www.google.co.in)**
- **https://www.kaggle.com (https://www.kaggle.com)**
- **https://github.com (https://github.com)**