DATA ANALYSIS

DATA

ANALYTICS

Data Analysis on Furniture Brand EMEA

FEATURES OF DATASET

There are total 57 Features in the dataset. At very first we understand the meaning of each features.

- Name:- The order number as it appears in your store admin.
- Financial Status:- Whether the order has been paid, authorized, refunded, and so on.
- Paid At:- The date when the payment was captured for the order.
- Fulfillment Status:- Whether the order has been fulfilled or is still pending.
- Accepts Marketing:- Whether the customer has agreed to accept marketing from your store.
- Total:- The total cost of the order.
- Lineitem name:- The name of the line item.
- Billing Country:- The customer's billing country.

And so on.

All Features

```
df.columns
```

[3] ✓ 0.2s

```
Index(['Name', 'Financial Status', 'Paid at', 'Fulfillment Status',
       'Fulfilled at', 'Accepts Marketing', 'Currency', 'Subtotal', 'Shipping',
       'Taxes', 'Total', 'Discount Code', 'Discount Amount', 'Shipping Method',
       'Created at', 'Lineitem quantity', 'Lineitem name', 'Lineitem price',
       'Lineitem compare at price', 'Lineitem sku',
       'Lineitem requires shipping', 'Lineitem taxable',
       'Lineitem fulfillment status', 'Billing Zip', 'Billing Province',
       'Billing Country', 'Shipping Zip', 'Shipping Province',
       'Shipping Country', 'Notes', 'Note Attributes', 'Cancelled at',
       'Payment Method', 'Payment Reference', 'Refunded Amount', 'Vendor',
       'Outstanding Balance', 'Employee', 'Location', 'Device ID', 'Id',
       'Tags', 'Risk Level', 'Source', 'Lineitem discount', 'Tax 1 Name',
       'Tax 1 Value', 'Tax 2 Name', 'Tax 2 Value', 'Tax 3 Name', 'Tax 3 Value',
       'Tax 4 Name', 'Tax 4 Value', 'Tax 5 Name', 'Tax 5 Value', 'Phone',
       'Receipt Number'],
      dtype='object')
```

STATISTICAL ANALYSIS OF FEATURES

Descriptive Statistics

```
df.describe().T
```

[7] ✓ 0.3s

	count	mean	std	min	25%	50%	75%	max
Subtotal	95484.0	5.881073e+01	6.610937e+01	0.000000e+00	2.240000e+01	3.599000e+01	7.200000e+01	1.952900e+03
Shipping	95484.0	6.426888e-02	8.298321e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.600000e+01
Taxes	95484.0	9.809089e+00	1.103631e+01	0.000000e+00	3.750000e+00	6.000000e+00	1.200000e+01	3.254800e+02
Total	95484.0	5.887248e+01	6.621758e+01	0.000000e+00	2.250000e+01	3.600000e+01	7.200000e+01	1.952900e+03
Discount Amount	95484.0	1.025338e+01	1.605742e+01	0.000000e+00	0.000000e+00	5.990000e+00	1.399000e+01	9.734300e+02
Lineitem quantity	118900.0	1.025517e+00	2.649508e-01	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	3.700000e+01
Lineitem price	118900.0	5.474903e+01	5.994009e+01	0.000000e+00	2.499000e+01	3.499000e+01	6.999000e+01	9.999900e+02
Lineitem compare at price	97212.0	5.727834e+01	6.476138e+01	0.000000e+00	2.499000e+01	3.999000e+01	6.999000e+01	9.999900e+02
Shipping Province	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Note Attributes	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Refunded Amount	95484.0	2.973846e+00	2.253504e+01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.394950e+03
Outstanding Balance	95484.0	2.029743e-02	1.336411e+00	-2.995000e+01	0.000000e+00	0.000000e+00	0.000000e+00	2.398800e+02
Device ID	1.0	2.000000e+00	NaN	2.000000e+00	2.000000e+00	2.000000e+00	2.000000e+00	2.000000e+00
Id	95484.0	1.887133e+12	9.048625e+11	2.739127e+09	1.690000e+12	2.130000e+12	2.600000e+12	2.850000e+12
Lineitem discount	118900.0	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
Tax 1 Value	95430.0	9.814640e+00	1.103697e+01	1.700000e-01	3.750000e+00	6.000000e+00	1.200000e+01	3.254800e+02

DATA CLEANING

In this stage, we first split the Lineitem name feature into two separate features because this feature contains the name of the item and color. So we separate it into two different features.

I. Data Cleaning

1. Feature Lineitem name split into name and color

```
df[["Lineitem name", "Lineitem_color"]] = df['Lineitem name'].str.split('-', 1, expand = True)
df['Date'] = pd.to_datetime(df['Paid at'].str.split('+', 1, expand = True)[0], errors='coerce')
```

9] ✓ 1.1s Python

After that, we check the duplicate rows in the Dataset and we find that there are no duplicate rows in the dataset.

2. Finding Duplicate Rows

```
df.duplicated().sum()
```

[10] ✓ 0.8s Python

... 0

NULL VALUES IN DATASET

3. Handeling Null Values and Outliers

```
df.isnull().sum()
```

[11] ✓ 0.7s

Name	0
Financial Status	23416
Paid at	24928
Fulfillment Status	23416
Fulfilled at	33655
Accepts Marketing	23416
Currency	23416
Subtotal	23416
Shipping	23416
Taxes	23416
Total	23416
Discount Code	56869
Discount Amount	23416
Shipping Method	23416
Created at	0
Lineitem quantity	0
Lineitem name	0
Lineitem name	0
Lineitem price	0
Lineitem compare at price	21688
Lineitem sku	0
Lineitem requires shipping	0
Lineitem taxable	0
Lineitem fulfillment status	0
Billing Zip	23464
Billing Province	118592
Billing Country	23417
Shipping Zip	23416
Shipping Province	118900
Shipping Country	23416
Notes	118869
Note Attributes	118900
Cancelled at	117844
Payment Method	23436
Payment Reference	23475
Refunded Amount	23416
Vendor	0
Outstanding Balance	23416
Employee	118899
Location	118899
Device ID	118899
Id	23416
Tags	111891
Risk Level	23416
Source	23416
Lineitem discount	0
Tax 1 Name	23470
Tax 1 Value	23470
Tax 2 Name	118900
Tax 2 Value	118900
Tax 3 Name	118900
Tax 3 Value	118900
Tax 4 Name	118900
Tax 4 Value	118900
Tax 5 Name	118900
Tax 5 Value	118900
Phone	114096
Receipt Number	118900
Lineitem_color	13
Date	24928

dtype: int64

DATA CLEANING

Here we drop all those columns which contain above 90% null values. And after that, we drop fulfilled at, discount code, lineitem_color columns

Droping the columns which is contain above 90% null values

```python
for i in df.columns:
    if df[i].isnull().sum() >= 110000:
        df.drop(i, axis=1, inplace=True)
df.shape
```

[59] ✓ 2.4s Python

```
(118900, 40)
```

Droping the unwanted columns

```python
df1 = df.drop(columns = ['Fulfilled at', 'Discount Code', 'Lineitem_color'])
df1.shape
```

[14] ✓ 0.1s Python

```
(118900, 37)
```

REAMAINING FEATURES


```
df1.columns
```

[15] ✓ 0.1s

```
Index(['Name', 'Financial Status', 'Paid at', 'Fulfillment Status',
       'Accepts Marketing', 'Currency', 'Subtotal', 'Shipping', 'Taxes',
       'Total', 'Discount Amount', 'Shipping Method', 'Created at',
       'Lineitem quantity', 'Lineitem name', 'Lineitem price',
       'Lineitem compare at price', 'Lineitem sku',
       'Lineitem requires shipping', 'Lineitem taxable',
       'Lineitem fulfillment status', 'Billing Zip', 'Billing Country',
       'Shipping Zip', 'Shipping Country', 'Payment Method',
       'Payment Reference', 'Refunded Amount', 'Vendor', 'Outstanding Balance',
       'Id', 'Risk Level', 'Source', 'Lineitem discount', 'Tax 1 Name',
       'Tax 1 Value', 'Date'],
      dtype='object')
```

OUTLIERS HANDLING

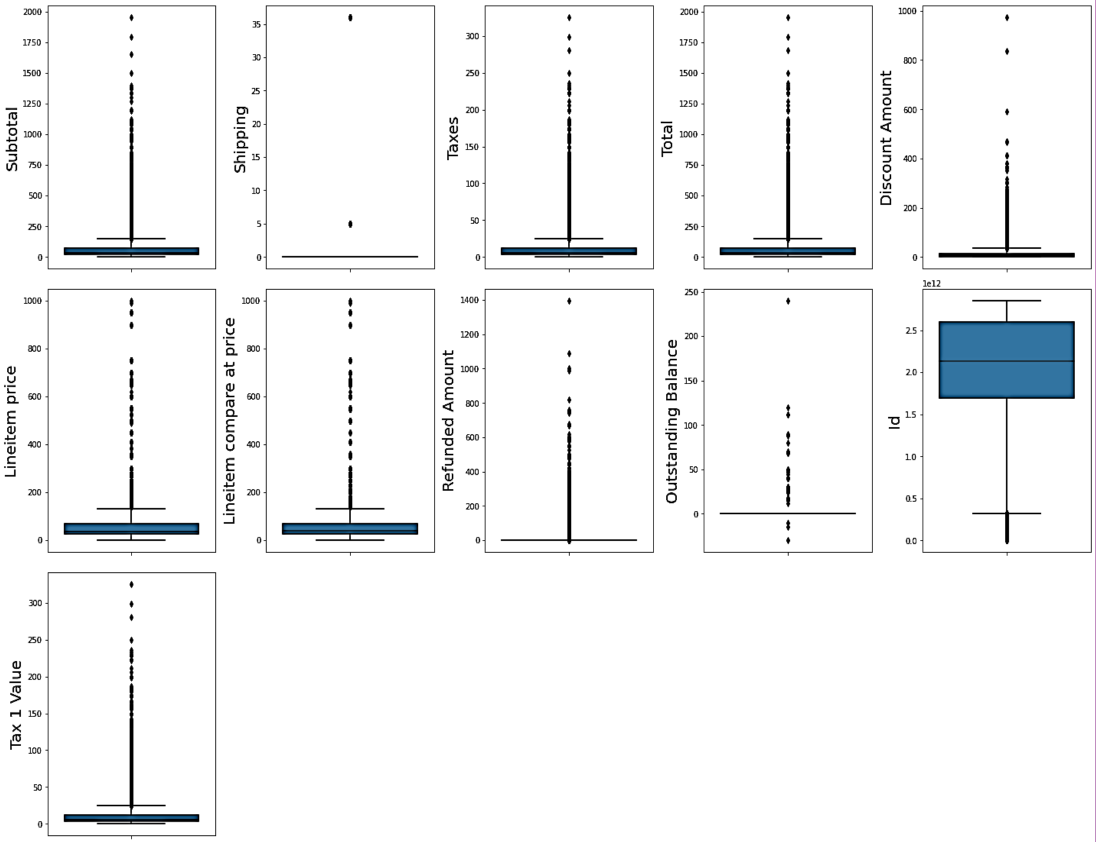
Here we find the outliers of all continuous features and remove the outliers or subtotal by the .99 percentile. After removing the outliers of subtotal we find that most of the other features outliers are also removed with it. Most of the outliers are due to null values, so we remove the outliers of subtotal by .99 percentile, null values and other outliers are also removed with it.

Removing Outliers

```
q = df1[df1.columns[6]].quantile(.99)
df1 = df1[df1.columns[6]]<q]
df1.shape
```

[17] ✓ 0.2s

```
(94528, 37)
```



DATA CLEANING

After all, we fill the null values of lineitem compare at price with their median because the feature is not normally distributed.

Fill null values with median, because that feature data is not normal distributed as shown below

[19]
```python
df1["Lineitem compare at price"].fillna(df1["Lineitem compare at price"].median, inplace = True)
```
Python

Droping rest of the null rows

[20]
```python
df2 = df1.dropna()
```
Python

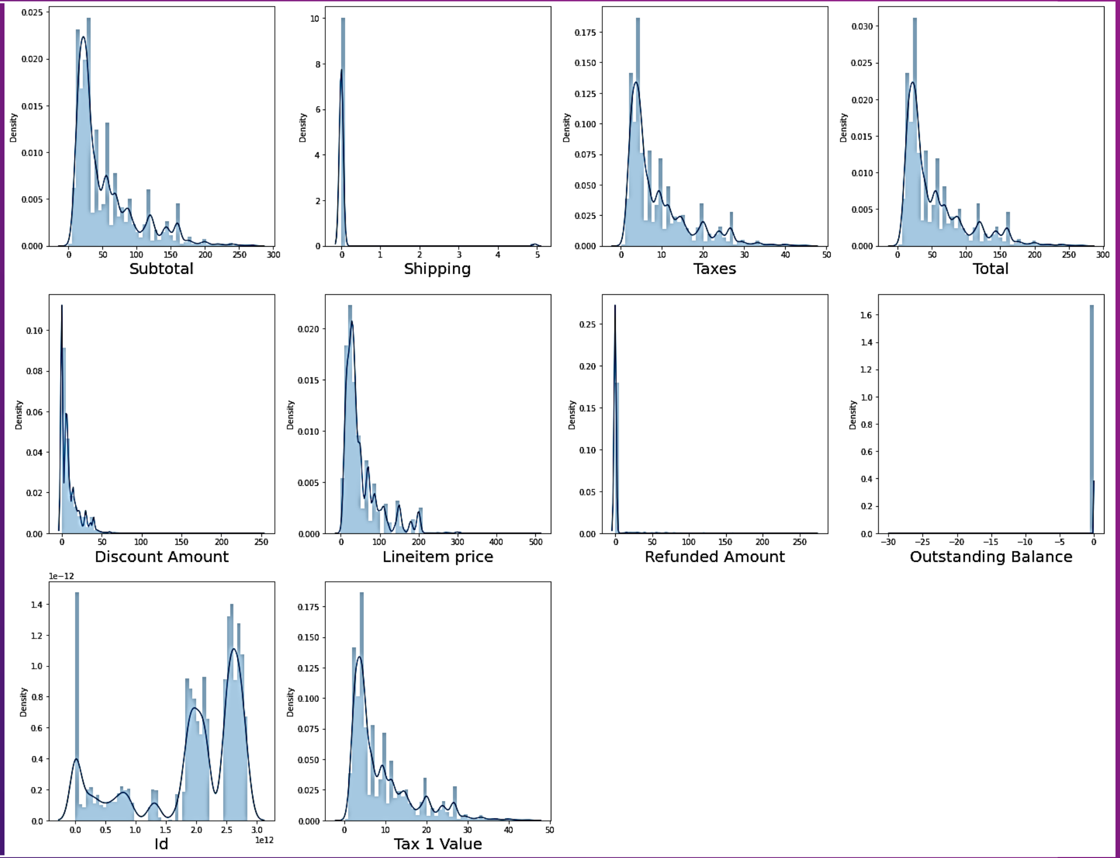
Shape of Dataset after Data Cleaning

[21]
```python
df2.shape
```
Python

```
(92990, 37)
```

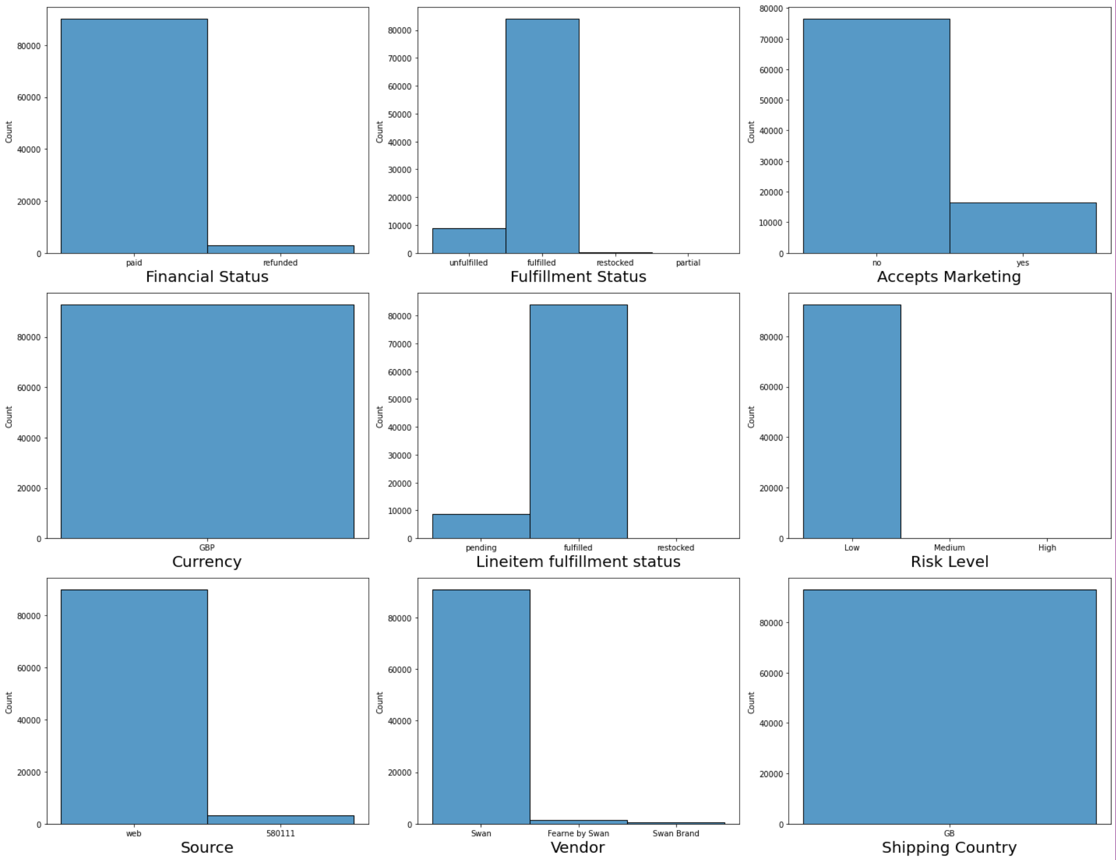
DATA VISUALIZATION

This graph shows the distribution of data of all continuous features.



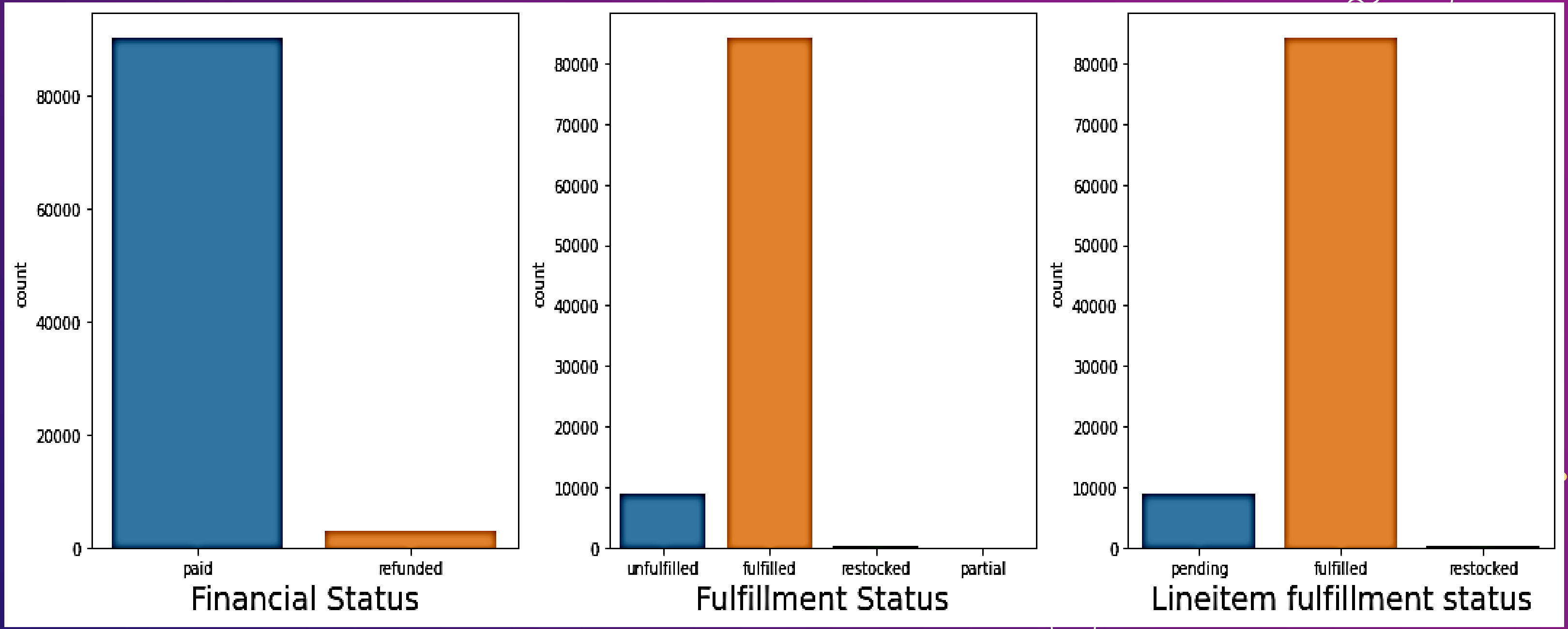
DATA VISUALIZATION

Here histogram shows the distribution of categorical features.



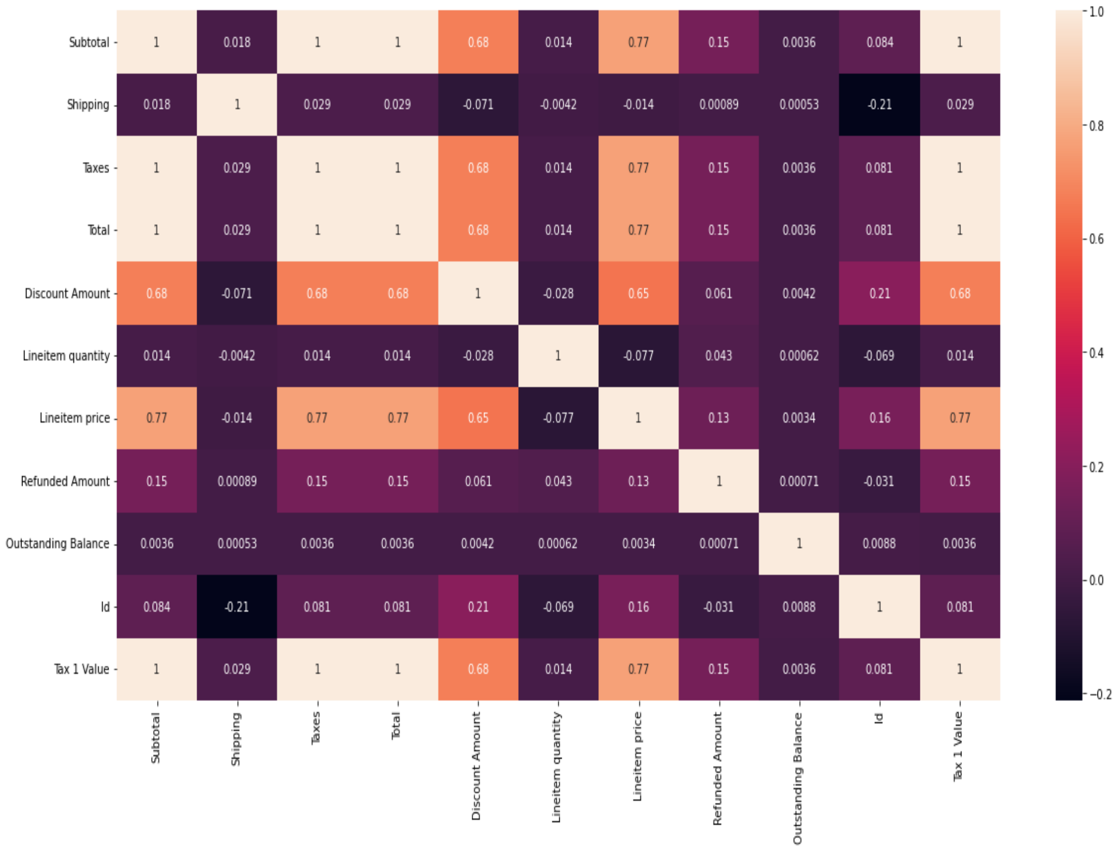
DATA VISUALIZATION

Here we clearly see how many products have been paid and fulfilled.



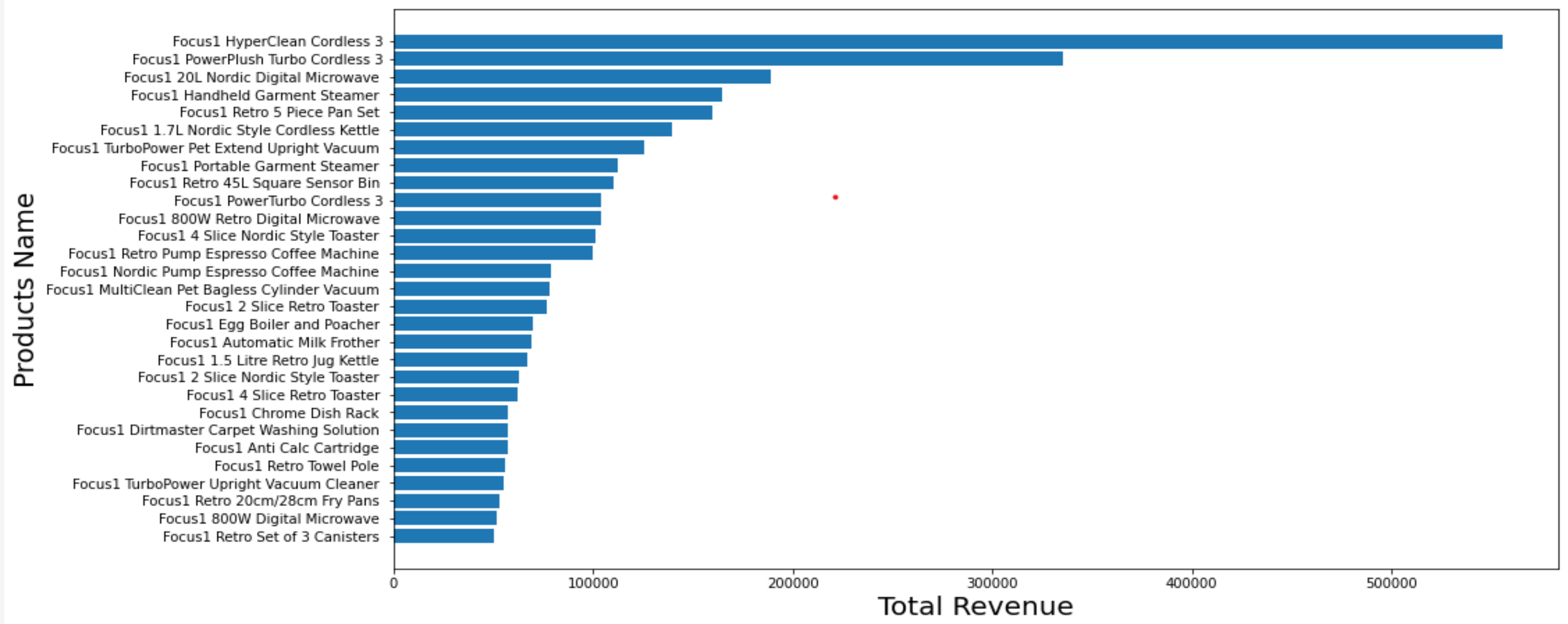
DATA VISUALIZATION

Here we plot a correlation plot that tells us which features are highly correlated to each other by coloring. In the correlation plot, light colors show high collinearity and dark colors show low collinearity. In this plot, collinearity is also represented by numbers between 0 to 1 where 0 is the lowest collinearity and 1 is the highest collinearity.



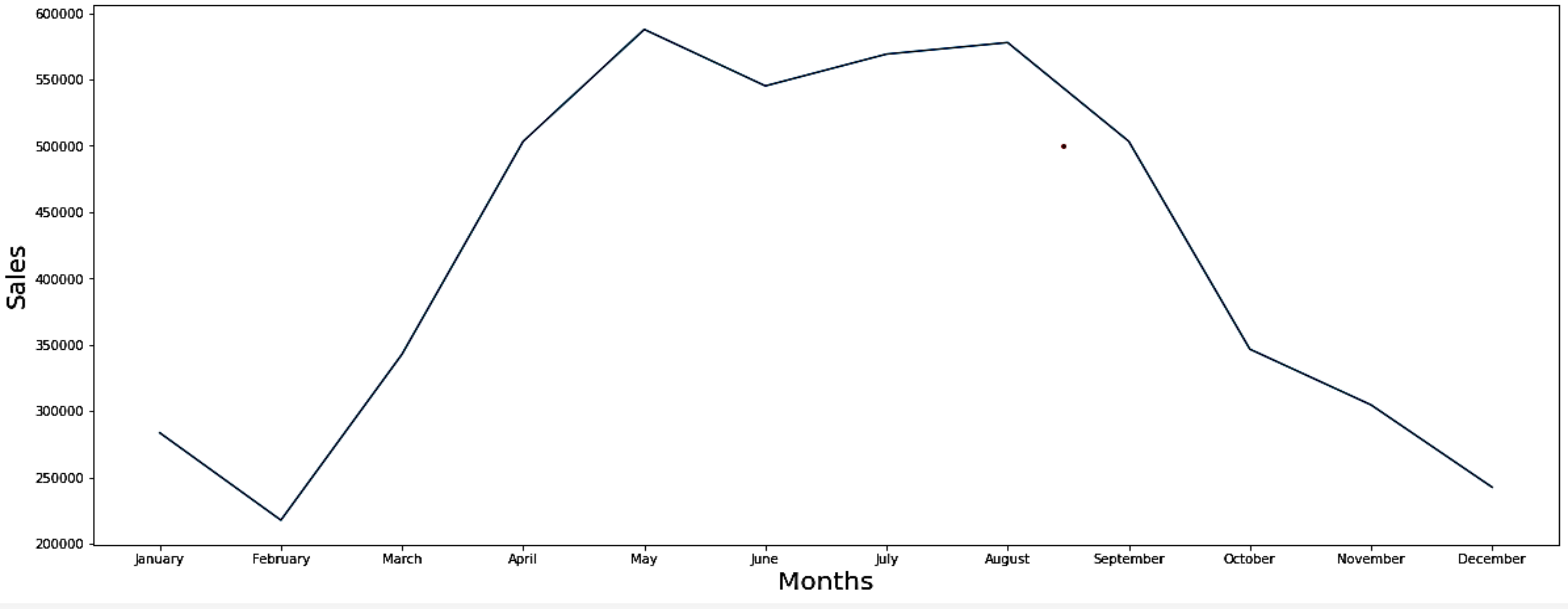
DATA VISUALIZATION

Here below graph shows the products which generate the highest revenue



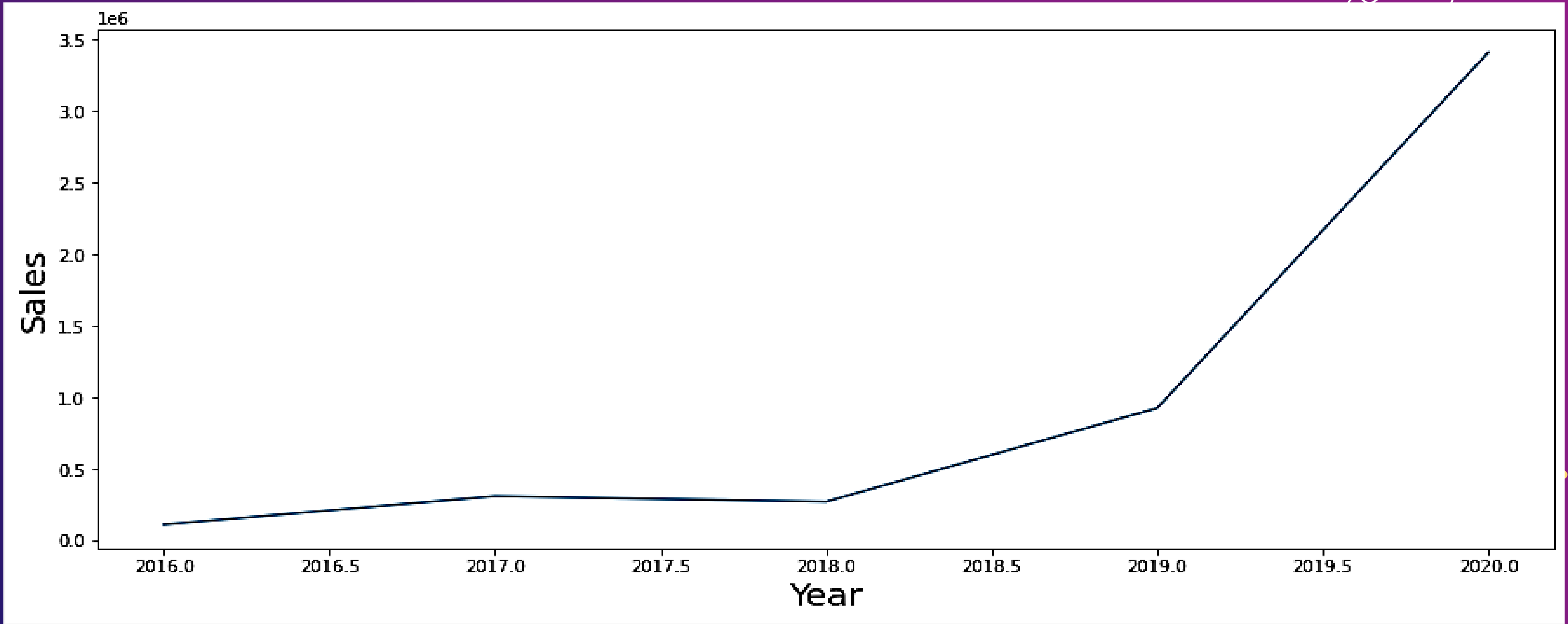
DATA VISUALIZATION

The below graph shows the seasonal trend of sales by months.



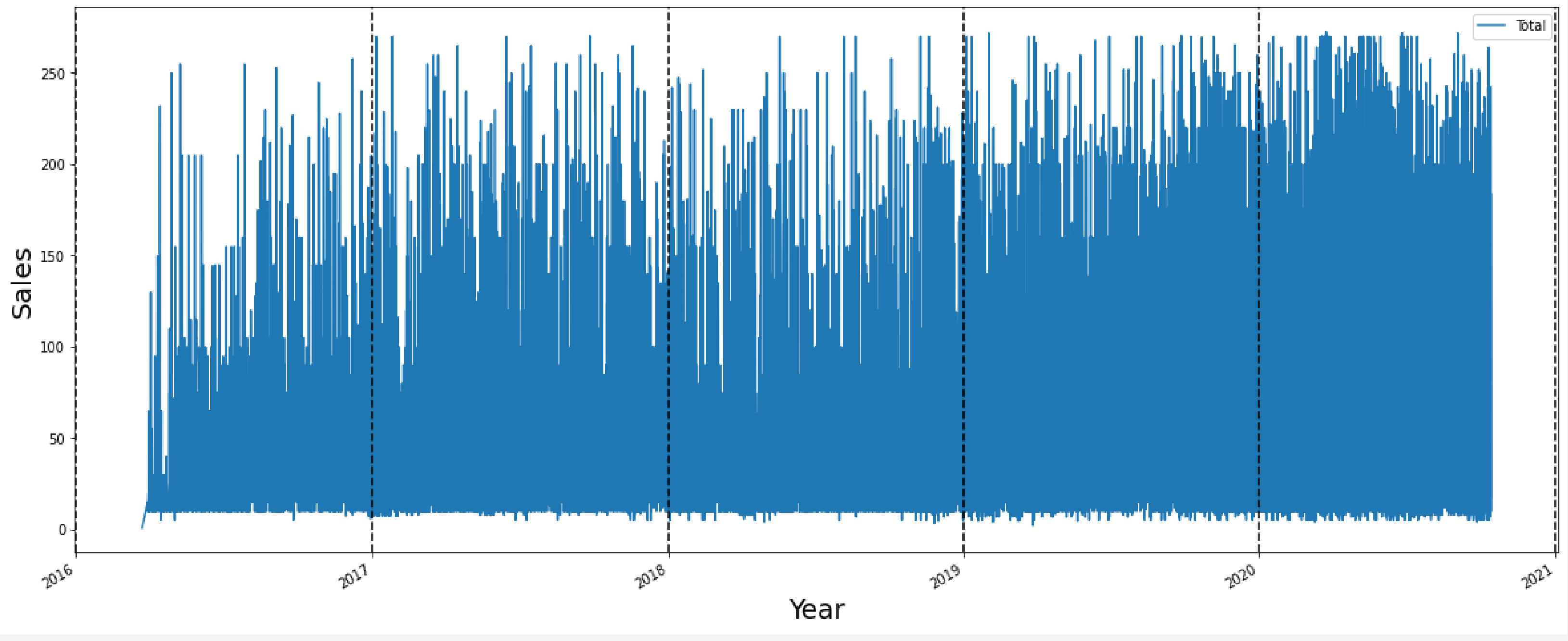
DATA VISUALIZATION

The below graph shows the seasonal trend of sales by Year.



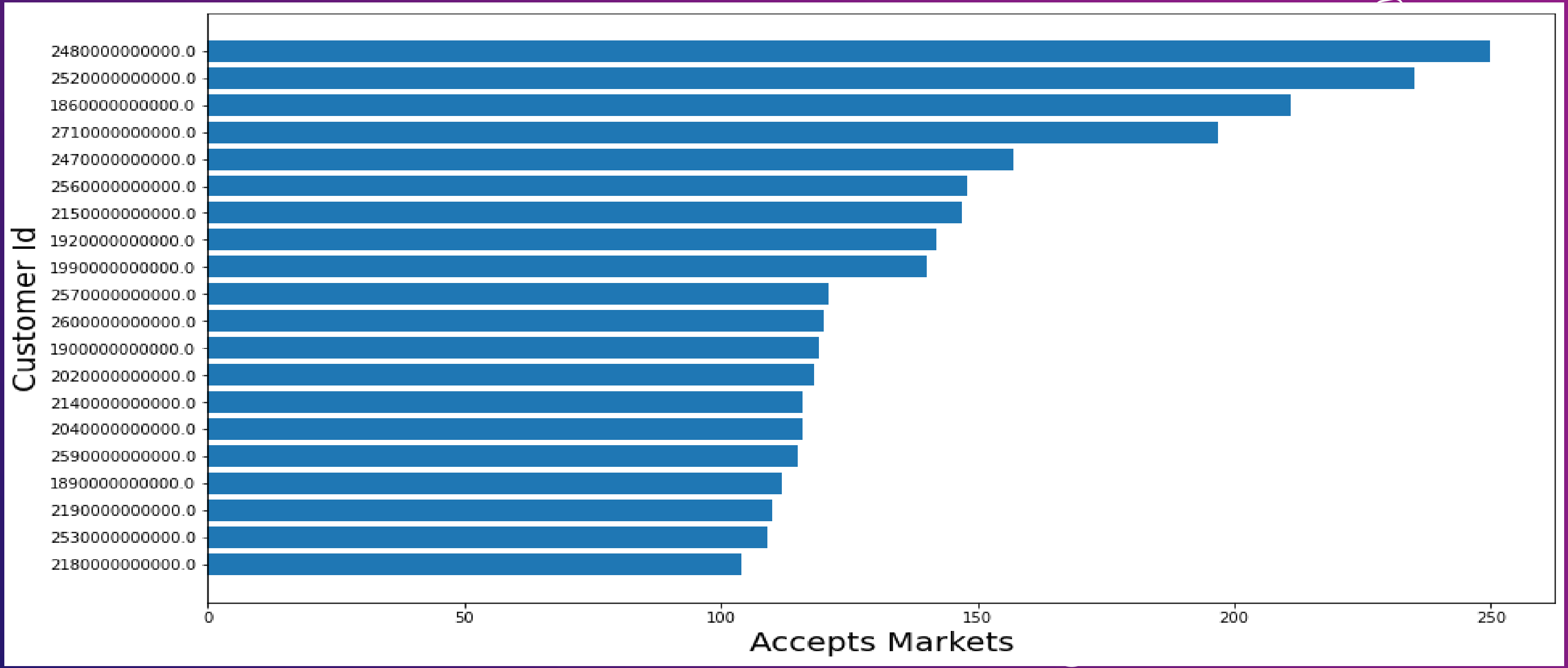
DATA VISUALIZATION

The below graph shows the seasonal trend of sales by Year.



DATA VISUALIZATION

The below graph shows the top customers which accepts markets and buy products



CUSTOMER SEGMENTATION

At this stage, we perform RFM Segmentation of Customers

- Recency: How recently has the customer made a transaction with us
- Frequency: How frequent is the customer in ordering/buying some product from us
- Monetary: How much does the customer spend on purchasing products from us.

	CustomerName	Recency	Frequency	Monetary
0	2.739127e+09	1664	1	0.99
1	2.786663e+09	1657	1	14.99
2	2.789790e+09	1657	1	9.99
3	2.791239e+09	1657	1	14.99
4	2.795784e+09	1657	1	39.96
5	2.796316e+09	1657	1	14.99
6	2.796482e+09	1657	1	14.99
7	2.796532e+09	1657	1	59.96
8	2.796830e+09	1657	1	64.90
9	2.797054e+09	1657	1	14.99

CUSTOMER SEGMENTATION

Here we are ranking Customer based upon their recency, frequency, and monetary score and normalizing the rank of the customers within a company to analyze the ranking.

	CustomerName	Recency	Frequency	Monetary	R_rank_norm	F_rank_norm	M_rank_norm
0	2.739127e+09	1664	1	0.99	0.010930	44.994536	44.994536
1	2.786663e+09	1657	1	14.99	0.065577	44.994536	44.994536
2	2.789790e+09	1657	1	9.99	0.065577	44.994536	44.994536
3	2.791239e+09	1657	1	14.99	0.065577	44.994536	44.994536
4	2.795784e+09	1657	1	39.96	0.065577	44.994536	44.994536

CUSTOMER SEGMENTATION

Calculating RFM score

RFM score is calculated based upon recency, frequency, monetary value normalize ranks. Based upon this score we divide our customers. Here we rate them on a scale of 5. Formula used for calculating rfm score is : 0.15*Recency score + 0.28*Frequency score + 0.57 *Monetary score

	CustomerName	RFM_Score
0	2.739127e+09	1.91
1	2.786663e+09	1.91
2	2.789790e+09	1.91
3	2.791239e+09	1.91
4	2.795784e+09	1.91
5	2.796316e+09	1.91
6	2.796482e+09	1.91

	CustomerName	RFM_Score	Customer_segment
0	2.739127e+09	1.91	Low Value Customers
1	2.786663e+09	1.91	Low Value Customers
2	2.789790e+09	1.91	Low Value Customers
3	2.791239e+09	1.91	Low Value Customers
4	2.795784e+09	1.91	Low Value Customers

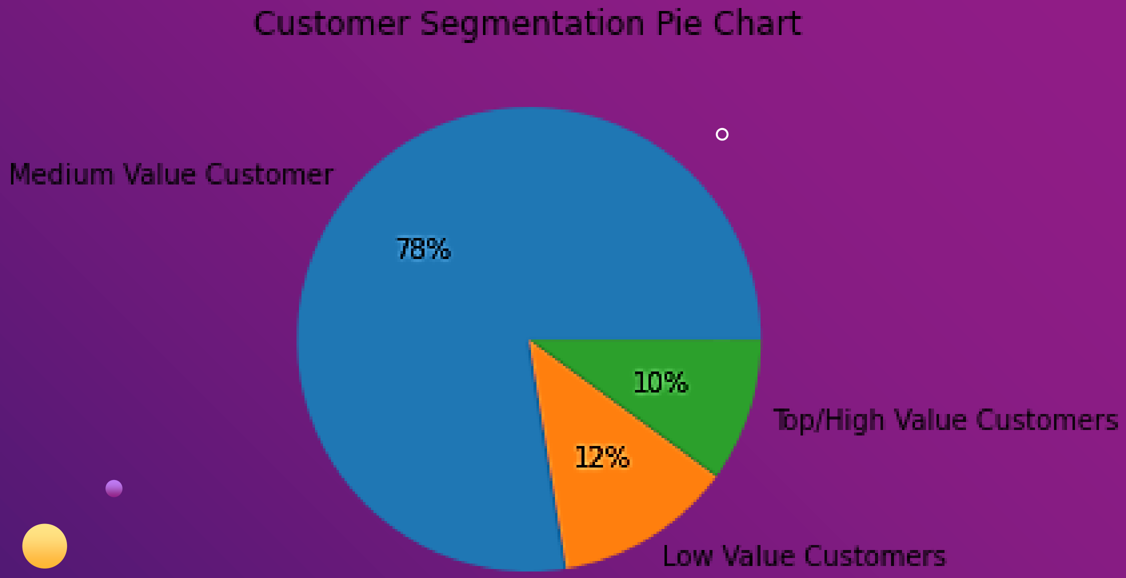
Rating Customer based upon the RFM score.

- rfm score > 3.5: High Value Customer
- 4>rfm score > 2 : Medium value customer
- 3>rfm score> .1 : Low-value customer

CUSTOMER SEGMENTATION

Visualizing the customer segments.

Here we will use a pie plot to display all segments of customers.



THANK YOU