RevoBank

Data Analysis with Python

By **AAN HISBULLAH**

Business Understanding

RevoBank, a **European bank provides credit cards** to customers who are working with **Revoshop** one of it's partners, and RevoBank implemented a new promotion exclusively for credit card users. The promotion involves **distributing RevoShop** vouchers to all RevoBank customers via email or SMS. so it is necessary to look for **segmentation** and **promo sensitivity** to optimize promotion costs.

Table of contents

01

Data Overview

Data Overview

RevoBank, a **European bank** provides credit cards to customers in working with **Revoshop**, one of it's partners.

Profit RevoShop
(over the past 6 months)

58.077.849 Euros

Total Revoshop Customer

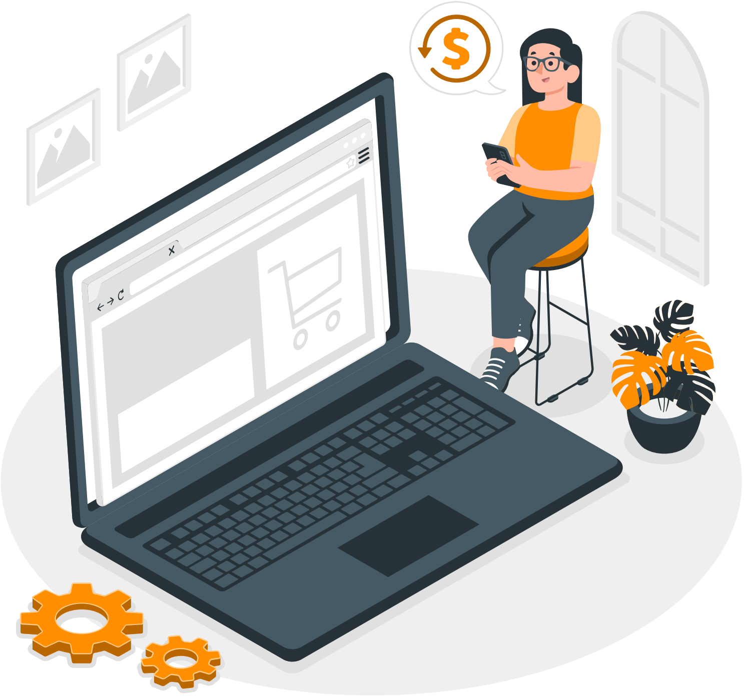
Total	: 111.133
Female	: 58.48%
Male	: 41.51%

Promo Sensitive

True	: 109306
False	: 1827

Merchant Name

REVOSHOP	: 111133
TOKTOKLIVE	: 1500
EL CORTE INGLES	: 1



02
Data Cleaning

Data Cleaning

- Check the type of each column. Convert data type register_date to datetime
- Check Missing Value Check
- Check Duplicate data
- Copy Dataframe to select Merchant Name Revoshop

```
# Convert data type from data frame

# df['ACCOUNT_ID'] = df['ACCOUNT_ID'].astype(str)
df['BIRTH_DATE'] = pd.to_datetime(df['BIRTH_DATE'])

df['BIRTH_DATE'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 112634 entries, 0 to 112633
Series name: BIRTH_DATE
Non-Null Count   Dtype
--------------   -----
112634 non-null  datetime64[ns]
dtypes: datetime64[ns](1)
```

Convert Data type

```
# to see each MERCHANT name and its counts
df['MERCHANT_NAME'] = df['MERCHANT_NAME'].replace('REVOSH MKTPLC', 'REVOSHOP')
df['MERCHANT_NAME'].value_counts()
```

```
REVOSHOP           111133
TOKTOKLIVE           1500
EL CORTE INGLES         1
Name: MERCHANT_NAME, dtype: int64
```

Select Merchant Name Revoshop

MERCHANT NAME

REVOSHOP	**: 111133**
TOKTOKLIVE	: 1500
EL CORTE INGLES	: 1

[Link to Syntax]

Find Promo Sensitive

```
# find promo sensitive
df2['PCT_PROMO_SALES'] = df2['AVG_PROMO_TXN_AMT_LTM'] / df2['AVG_TXN_AMT_LTM'] * 100
df2['PROMO_SENSITIVE'] = df2['PCT_PROMO_SALES'] > 50
df2['PROMO_SENSITIVE'].value_counts()
```

Customers are considered to be **promo-sensitive if more than 50%** of their total sales on revoshop are attributed to promos.

It turns out that for **promo-sensitive** it can be seen that the number of customers who are **promo-sensitive** is **109306 customers**, more than **98%** of the **total customers,** and customers who are not sensitive are **1827**.

```
[18] df2['PROMO_SENSITIVE'] = df2['PROMO_SENSITIVE'].astype(int)
     df2['PROMO_SENSITIVE'].value_counts()

1    109306
0      1827
Name: PROMO_SENSITIVE, dtype: int64
```

[Link to Syntax]

Insight into the promo Performance in the past 6 months

(Customer Demographics)

```
[20] # Calculate age based on the current date
current_date = datetime.now()
df2['AGE'] = (current_date - df2['BIRTH_DATE']).astype('<m8[Y]')

# Analyze age distribution
pd.DataFrame(df2['AGE'].describe())
```

	AGE
count	111133.000000
mean	40.507131
std	5.680929
min	17.000000
25%	37.000000
50%	40.000000
75%	44.000000
max	67.000000

Analyze age distribution

```
df2['TXN_AMT_L6M'] = (df2['AVG_TXN_AMT_L6M'] * df2['TXN_CNT_LTM']) / df2['TXN_CNT_L6M']
df2['TXN_AMT_L6M']

0          907.200000
1         1050.866667
2          789.900000
3          795.150000
4          862.400000
             ...
112628     600.000000
112629     880.000000
112630     211.600000
112631    1495.000000
112632     420.000000
Name: TXN_AMT_L6M, Length: 111133, dtype: float64
```

Find Total Revenue L6M

```
df2['MOB'].describe()

count    111133.000000
mean        101.489432
std          25.559089
min          21.000000
25%          81.000000
50%         105.000000
75%         125.000000
max         570.000000
Name: MOB, dtype: float64
```

Analyze month on book distribution

Descriptive Statistics

Description	Value
Number of **sales made in response** to the promo in **the last 6 months?**	**307.570 Euros**
Percentage of total sales attributed to the promo in **the lifetime?**	In the past 6 months: **53%** In the lifetime : **50%**
Average number of transactions **per customer?**	In the past 6 months : **4** In the Lifetime : **20**
The total cost of the **promotion** over **the past 6 months?**	**461.355 Euros**
The total revenue generated by the promo in **the past 6 months?**	**1.393.868.37 Euros**

[Link to Syntax]

03

Exploratory Data Analysis

Promo sensitivity between active and inactive

```
[33] # Mengganti nilai-nilai 'MAPP_ACTIVE_GROUP'
     df2['MAPP_ACTIVE_GROUP'] = df2['MAPP_ACTIVE_GROUP'].replace({'X': 'Active', 'Y': 'Average', 'Z': 'Inactive'})
     df2['MAPP_ACTIVE_GROUP'].value_counts()

     Active      103348
     Average       7301
     Inactive       484
     Name: MAPP_ACTIVE_GROUP, dtype: int64
```

Status of account activity

MAPP_ACTIVE_GROUP	Total_Customers	Promo-Sensitive_Customers	Average_Transaction_Amount	Average_Transaction_Count
Active	103348	101710	523.588484	4.488350
Average	7301	7171	499.798041	4.197781
Inactive	484	425	654.960331	5.183884

Select Merchant Name Revoshop

Differences in **transaction behavior** and **promo sensitivity** between **active** and **inactive** customers **in the last 6 months?** We find the distribution of the number of customers first with the **MAPP_ACTIVE_GROUP** column.

We are looking for **behavior**, **The average sales amount per transaction** attributed to each account over the last 6 months of **Inactive status** has a total of **654.8 Euros**, **The number of transactions** that occurred in the last 6 months of **Inactive status** have a total of **5.1**

[Link to Syntax]

Promo sensitivity between active and inactive in the past 6 Months

93%

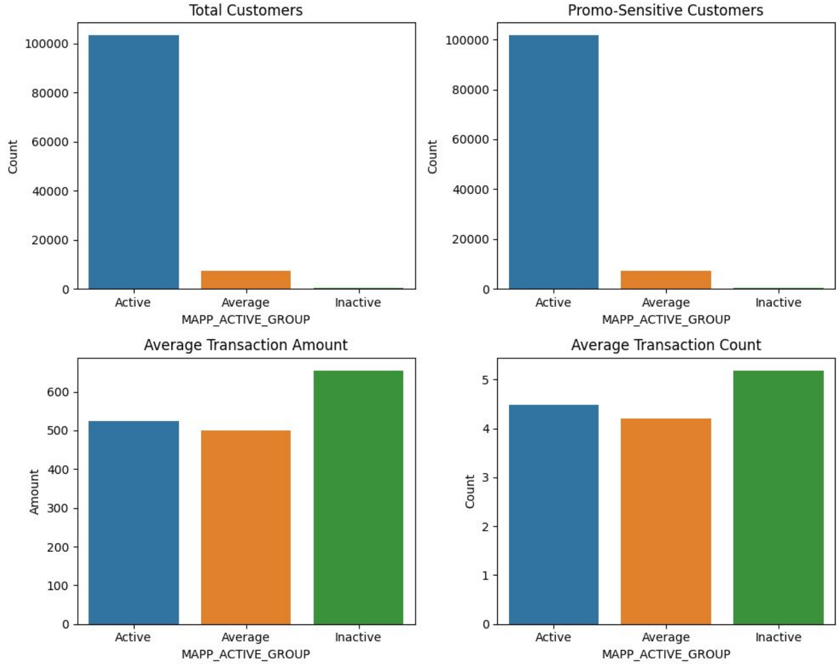
Total Customers

Customer **Active** have a total of **103348** customers

35%

Avg Transaction Amount

Average transaction amount **Inactive** have a total of **654.9 Euros**



[Link to Syntax]

93%

Promo-Sensitive Customers

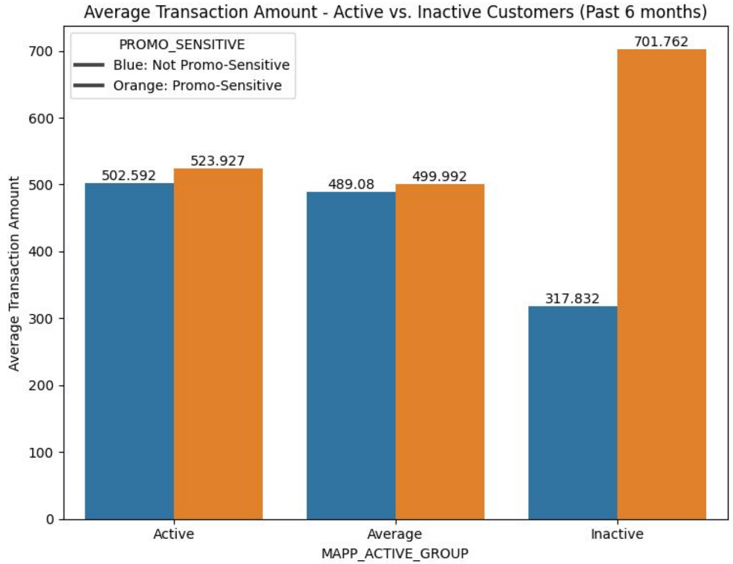
Promo-sensitive Customer **Active** have a total of **101710 customers**

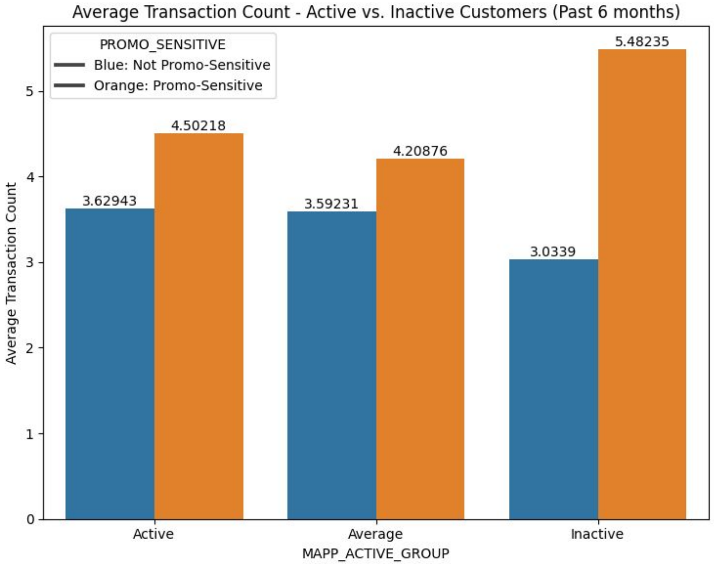
38%

Avg Transaction Count

Average transaction amount **Inactive** have a total of **5.2 Number of Transaction**

Promo sensitivity between active and inactive in the past 6 Months





	MAPP_ACTIVE_GROUP	PROMO_SENSITIVE	Average_Trans_Amount	Average_Trans_Count
0	Active	0	502.591880	3.629426
1	Active	1	523.926626	4.502183
2	Average	0	489.080000	3.592308
3	Average	1	499.992344	4.208757
4	Inactive	0	317.832203	3.033898
5	Inactive	1	701.761647	5.482353

We are looking for **behavior**, **The average sales amount per transaction** attributed to each account over the last 6 months of **Inactive status and Promo-Sensitive** has a total of **701.8 Euros**, **The number of transactions** that occurred in the last 6 months of **Inactive status and Promo-sensitive** have a total of **5.4**

Promo sensitivity between high-value and lower-value

Differences in **transaction behavior** and **promo sensitivity** between **high-value** and **lower-value** customers **in the last 6 months?** We find the distribution of the number of customers first with the **CUST_VALUE_GROUP** column the most profitable and creditworthy customers and the lowest customers.

```python
# Mengganti nilai-nilai 'MAPP_ACTIVE_GROUP'
df2['CUST_VALUE_GROUP'] = df2['CUST_VALUE_GROUP'].replace({
    'A': 'A: Best', 'B': 'B: Good', 'C': 'C: Average', 'D': 'D: Low', 'E': 'E: Worst'})
df2['CUST_VALUE_GROUP'].value_counts()
```

```
E: Worst      45958
C: Average    18233
B: Good       17619
A: Best       16878
D: Low        12445
Name: CUST_VALUE_GROUP, dtype: int64
```

Status of Customer Value

We are looking for **behavior**, **The average sales amount per transaction** attributed to each account over the last 6 months of **most profitable Best status** has a total of **554 Euros**, **The number of transactions** that occurred in the last 6 months of **Worst status** have a total of **4.6**

```python
# Group the data by CUST_VALUE_GROUP
grouped_value = df2.groupby('CUST_VALUE_GROUP').agg({
    'ACCOUNT_ID': 'nunique',
    'PROMO_SENSITIVE': 'sum',
    'TXN_AMT_L6M': 'mean',
    'TXN_CNT_L6M': 'mean'
})
grouped_value.columns = ['Total_Customers', 'Promo-Sensitive_Customers', 'Average_Transaction_Amount', 'Average_Transaction_Count']

# Reset the index
grouped_value = grouped_value.reset_index()

grouped_value
```

	CUST_VALUE_GROUP	Total_Customers	Promo-Sensitive_Customers	Average_Transaction_Amount	Average_Transaction_Count
0	A: Best	16878	16551	554.724505	4.182782
1	B: Good	17619	17301	522.040655	4.300471
2	C: Average	18233	17887	522.117068	4.486700
3	D: Low	12445	12207	528.192696	4.530655
4	E: Worst	45958	45360	509.688320	4.622960

Select Merchant Name Revoshop

[Link to Syntax]

Promo sensitivity between High-Value and Lower-Value in the past 6 Months

41%

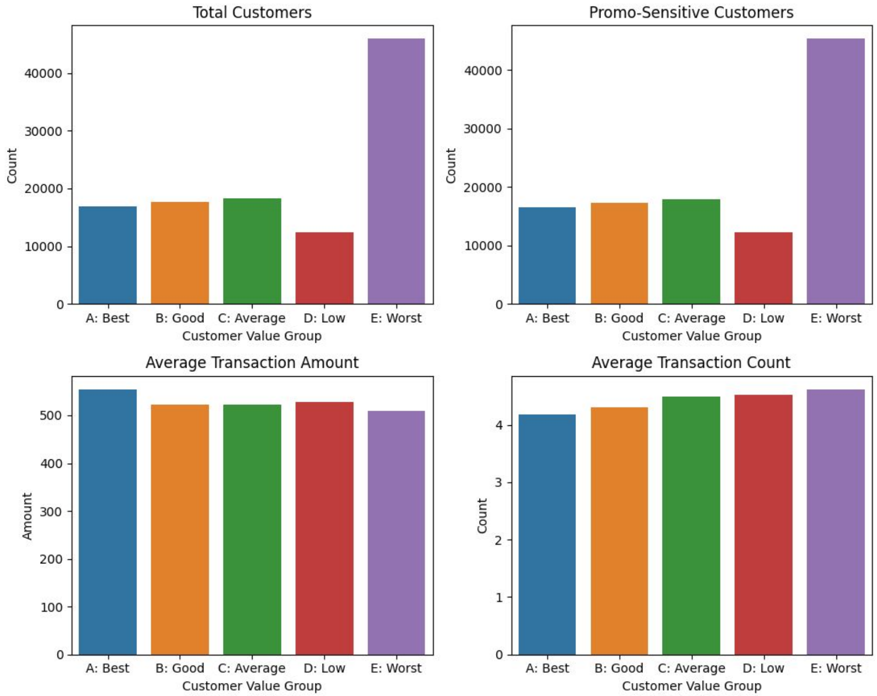
Total Customers

Customer value grup **Worst** have a total of **45958** customers

21%

Avg Transaction Amount

Average transaction amount **Best** have a total of **554 Euros**



41%

Promo-Sensitive Customers

Promo-sensitive Customer value group **Worst** have a total of **45360** customers

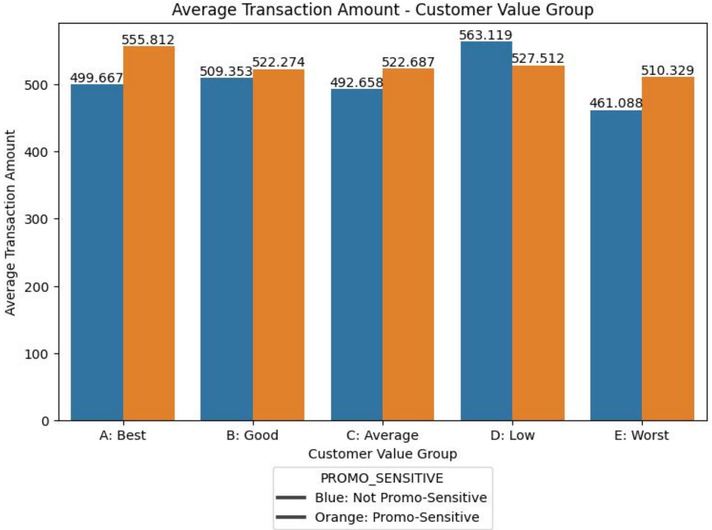
21%

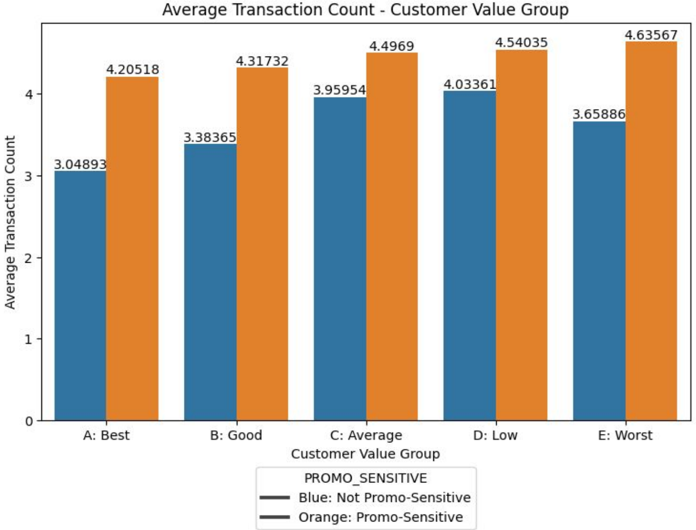
Avg Transaction Count

Average transaction amount **Best** have a total of **4.6 Number of Transaction**

[Link to Syntax]

Promo sensitivity between High-Value and Lower-Value in the past 6 Months

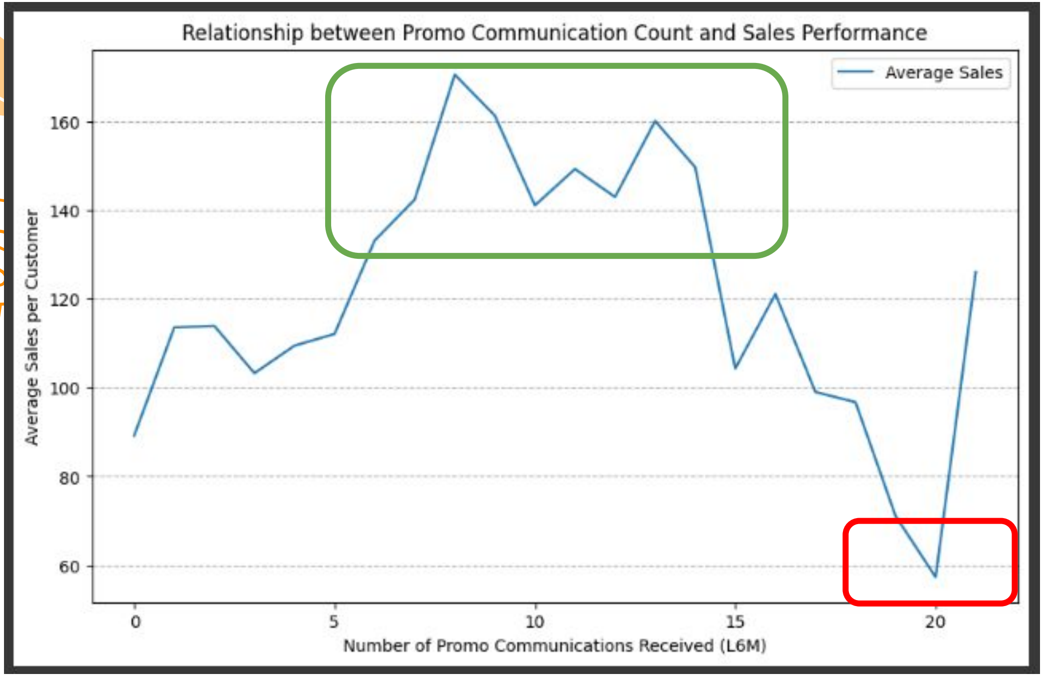




	CUST_VALUE_GROUP	PROMO_SENSITIVE	Total_Customer	Average_Transaction_Amount	Average_Transaction_Count
0	A: Best	0	327	499.666667	3.048930
1	A: Best	1	16551	555.812289	4.205184
2	B: Good	0	318	509.352830	3.383648
3	B: Good	1	17301	522.273863	4.317323
4	C: Average	0	346	492.657514	3.959538
5	C: Average	1	17887	522.686923	4.496897
6	D: Low	0	238	563.119328	4.033613
7	D: Low	1	12207	527.511731	4.540346
8	E: Worst	0	598	461.088462	3.658863
9	E: Worst	1	45360	510.329032	4.635670

We are looking for **behavior**, **The average sales amount per transaction** attributed to each account over the last 6 months of **Low status and Not Promo-Sensitive** has a total of **563.1 Euros**, **The number of transactions** that occurred in the last 6 months of **Worst status and Promo-sensitive** have a total of **4.6**

Relationship number of email and SMS messages and improved sales performance



The **relationship** between the increase in **the number of email and SMS messages** and the increase **in sales performance**, at **8 times** the receipt is seen to have the highest average sales per customer with an amount of **170 Euros** and a decrease in the amount of revenue after **8 times**, and in the **20th** receipt of the promo has the lowest opinion of **57 Euros** so it can be concluded that **optimizing the number of promotions** in 6 months is better with an **average of 8 times**

[Link to Syntax]

04

Customer Segmentation

Check Data Distribution

Use **RobustScaler** because there are outliers in our data, and look at the scatter distribution of the data

```
# visualize scaled data
robust_scatter = sns.scatterplot (data = robust_df , x =
'TXN_AMT_L6M' , y = 'TXN_CNT_L6M' )
plt.xlabel ('Transaction Amount L6M' )
plt.ylabel ('Transaction Count L6M' )
```

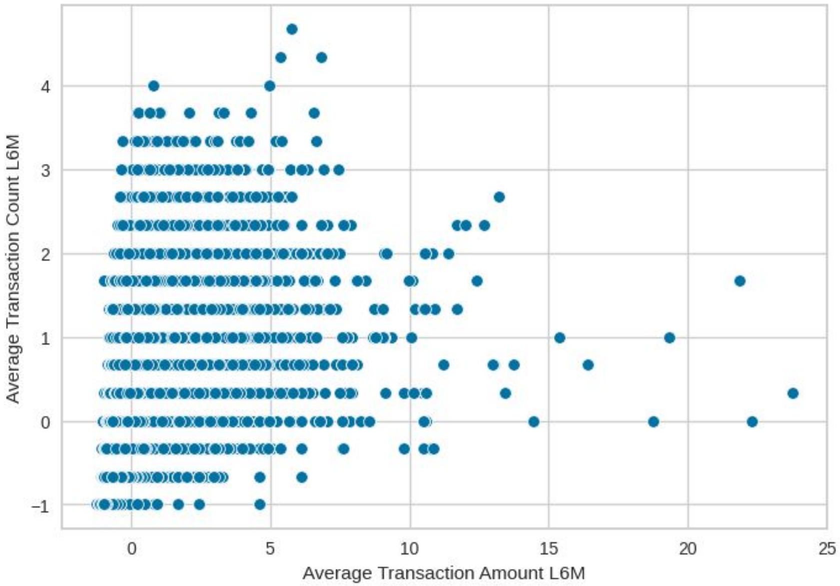
```
# we use RobustScaler because we have outliers in our data
scaler = RobustScaler()

# create new dataframe where we apply RobustScaler into our raw Data
robust_df = customer.copy()
robust_df[col_list] = scaler.fit_transform(robust_df[col_list])

# check the data
robust_df
```

	ACCOUNT_ID	TXN_CNT_L6M	TXN_AMT_L6M
0	100000004	0.333333	1.921742
1	100000008	-0.333333	0.009120
2	100000012	0.000000	1.842895
3	100000014	1.333333	4.289497
4	100000015	0.333333	0.185937
...	...	...	...
111128	101059832	-0.666667	-0.667255
111129	101059843	-0.333333	-0.196528
111130	101059857	1.000000	-0.166225
111131	101059860	-0.333333	0.465431
111132	101059866	0.333333	-0.020006

111133 rows × 3 columns



Clustering Process

We use the **elbow method** to adjust the elbow, we can see the sharpest elbow between **point 3 and 4.**

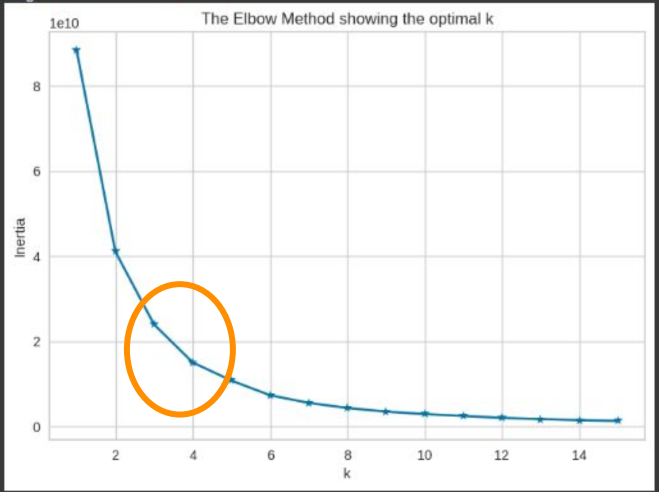
```
distortions = []
K = range(1,16)
for k in K:
    kmeanModel = cluster.KMeans(n_clusters=k, init = 'k-means++', n_init=10)
    kmeanModel.fit(customer[['TXN_AMT_L6M','TXN_CNT_L6M']]) # Ini yang diganti jadi df yang dipakai
    distortions.append(kmeanModel.inertia_)

distortions
```

```
[15637619925.652817,
 6938206479.638441,
 3957940012.915864,
 2623947411.180729,
 1762449035.342586,
 1272172703.0892267,
 966858595.828717,
 765153937.7006444,
 624841809.7167413,
 511686098.2087545,
 398579563.36334616,
 331168934.2724246,
 283627984.1643067,
 245656387.01645958,
 212581630.41465676]
```

```
plt.figure(figsize=(16,8))
plt.figure()
plt.plot(K, distortions, 'b*-')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

```
<Figure size 1600x800 with 0 Axes>
```



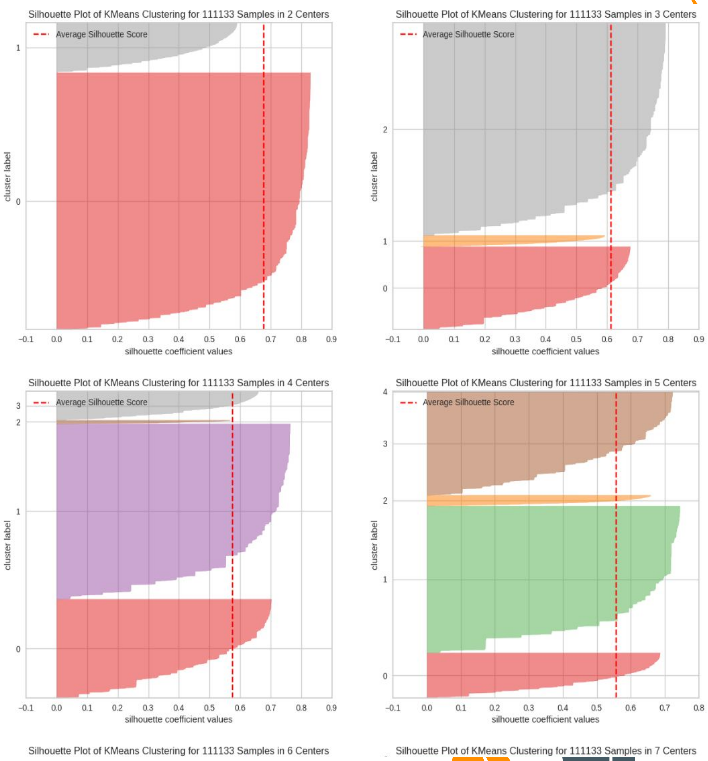
[Link to Syntax]

Silhouette Score Plot

Comparing the **elbow method** and **silhouette** coefficient to determine the appropriate number of clusters

The nearer silhouette_score to 1, the more optimal cluster number. Cluster **number= 3** is the most optimal. From elbow method and **silhouette analysis**, we can determine **3 cluster**.

```
For k=2, the average silhouette score is 0.6767450002719647
For k=3, the average silhouette score is 0.6128238122501103
For k=4, the average silhouette score is 0.5756151661781306
For k=5, the average silhouette score is 0.5565437815692782
For k=6, the average silhouette score is 0.552634649200282
For k=7, the average silhouette score is 0.5399812682144985
```



```
# We Group the data to see unique user characteristics
customer = df2.groupby('ACCOUNT_ID').agg({
    'TXN_CNT_L6M': 'sum',
    'TXN_AMT_L6M': 'sum',
}).reset_index()

customer.head()
```

	ACCOUNT_ID	TXN_CNT_L6M	TXN_AMT_L6M
0	100000004	5.0	1080.0
1	100000008	3.0	429.9
2	100000012	4.0	1053.2
3	100000014	8.0	1884.8
4	100000015	5.0	490.0

```
[76] # Initialize KMeans for 3 clusters
cluster_result = customer.copy()
kmeans = KMeans(n_clusters=3)
kmeans.fit(customer[['TXN_AMT_L6M','TXN_CNT_L6M']])
cluster_labels = kmeans.labels_
cluster_result['cluster'] = cluster_labels
```

```
cluster_result
```

	ACCOUNT_ID	TXN_CNT_L6M	TXN_AMT_L6M	cluster
0	100000004	5.0	1080.0	1
1	100000008	3.0	429.9	0
2	100000012	4.0	1053.2	1
3	100000014	8.0	1884.8	2
4	100000015	5.0	490.0	0
...	...	...	...	...
111128	101059832	2.0	200.0	0
111129	101059843	3.0	360.0	0
111130	101059857	7.0	370.3	0
111131	101059860	3.0	585.0	1
111132	101059866	5.0	420.0	0

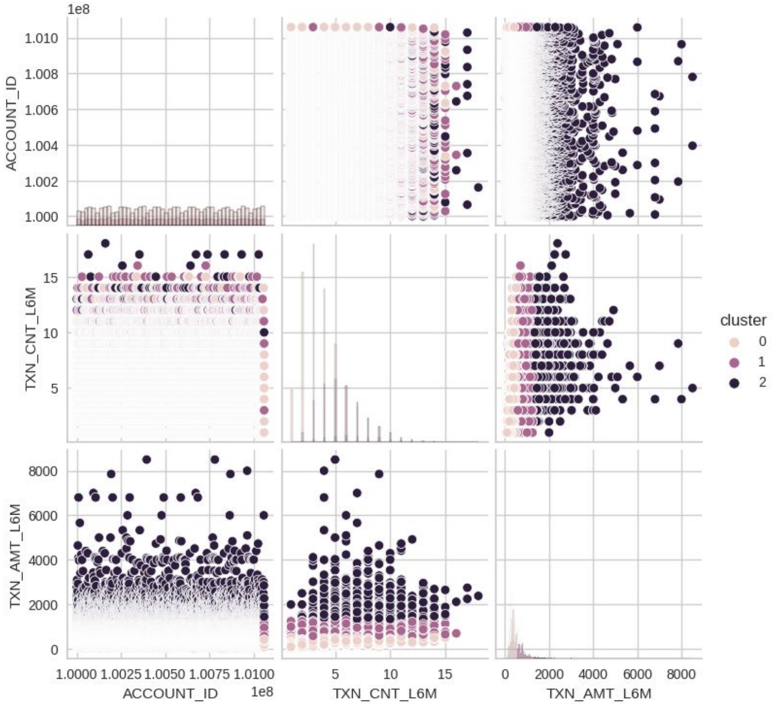
111133 rows × 4 columns

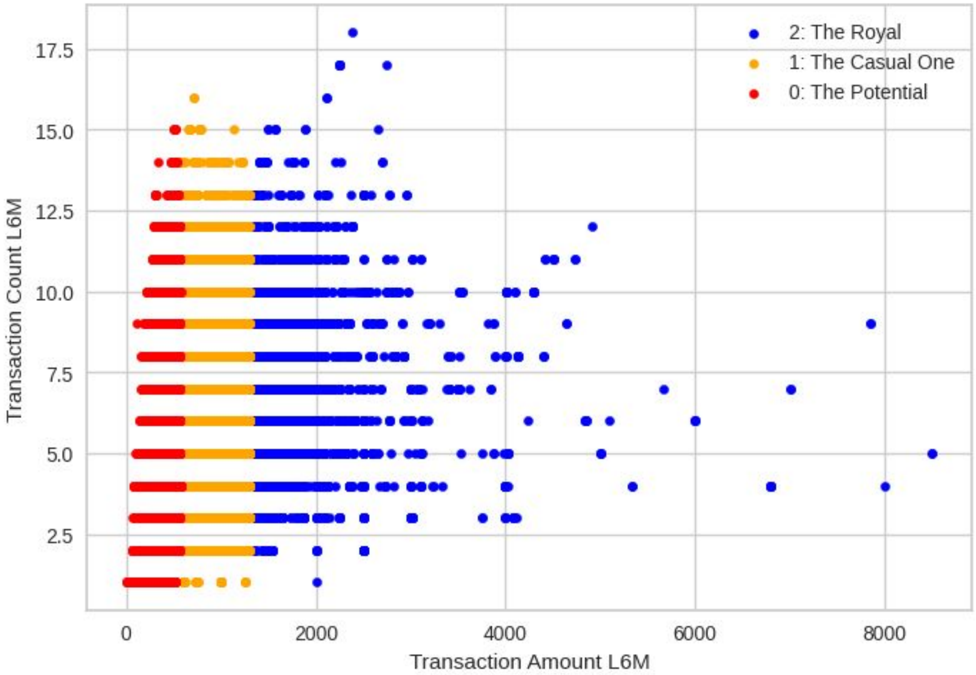
K-Mean Clustering

Kmean is implemented because it is an interactive method that is **easy to implement and is dynamic** on scattered data and most importantly, the **segmentation results are more accurate**. But this method has a drawback because the **k value must be determined first** to produce clustering. To overcome this, elbow and silhouette methods are needed which can help in finding the value of k, From **Elbow and Silhouette Scores** we get that there are **3 clusters.**

[Link to Syntax]

Histogram into data after Clustering

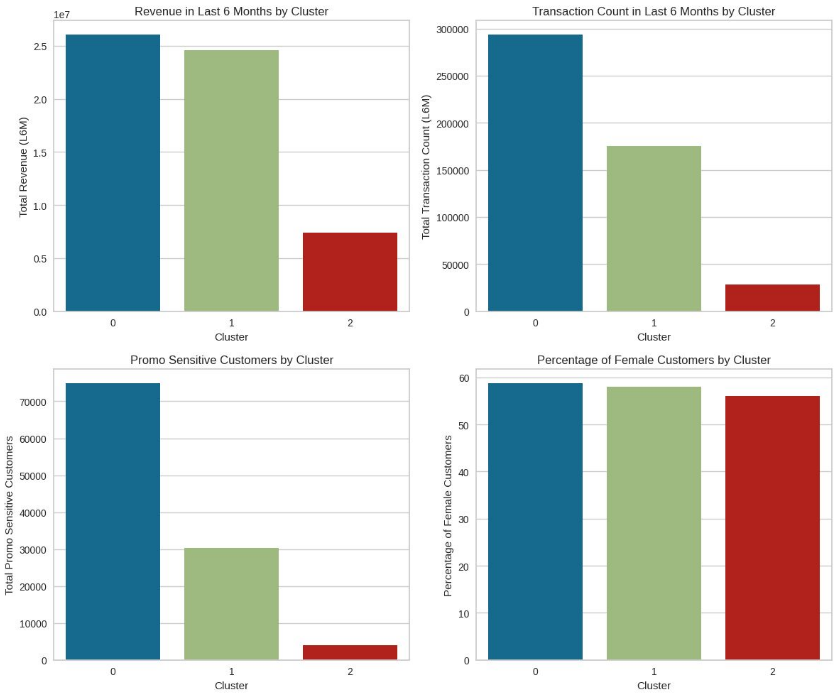




[Link to Syntax]

Interpreting cluster results

	Cluster	Revenue_L6M	Trans_Count_L6M	Promo_Sensitive	Average_AGE	Percentage_Female
0	0	26076393.1	293744.0	74928	40.507779	58.802309
1	1	24568291.9	175018.0	30374	40.508615	58.019924
2	2	7433163.9	28257.0	4004	40.617964	56.069506



Insight

Cluster 2: The Potential

This cluster **represents customers with low spending** These customers are not making significant transactions but are highly responsive to promotions. They might be occasional or **potential customers** who need extra incentives or promotions to increase their spending.

Cluster 1: The Casual One

This cluster represents customers with moderate spending behavior. These customers make **average transactions** without being too sensitive to promotions. They are relatively stable customers who make regular purchases without the need for heavy promotions.

Cluster 0: The Royal

This cluster represents customers with **high spending**. These customers make significant transactions and are less sensitive to promotions. They are the most valuable customers who consistently make **high-value purchases,** regardless of promotions.

05 Insight and Recommendation

Insight

We can derive several **insights** related to promo-sensitive customers at RevoShop:

Promo Sensitivity:
From the data analysis, we can identify customers who are sensitive to promotions. Customers are considered **promo-sensitive** if more than **50%** of their total sales come from promotional offers **provided by RevoShop**. The number of promo-sensitive customers can be calculated using the '**PROMO_SENSITIVE**' column, and we can determine the **percentage of total customers** falling into this category.

Customer Segmentation:
Based on **customer value, mobile app activity, transaction levels, and other factors**, we can perform customer segmentation into different groups. This allows **RevoShop** to identify the most **profitable potential customers** and develop tailored promotion strategies for each group.

Recommendation

Here are some **recommendations** to identify and attract promo-sensitive customers:

Cluster 2: The Potential
Focus on targeted promotions and offers to attract these potential customers and encourage them to make more purchases. Implement strategies to increase engagement with this segment and convert them into more regular and loyal customers.

Cluster 1: The Casual One
Offer occasional promotions or rewards to maintain the loyalty of this segment and keep them engaged. Identify opportunities to upsell or cross-sell products and services to increase their transaction amounts.

Cluster 0: The Royal
Provide personalized and exclusive offers to reward and retain these valuable customers. Strengthen customer loyalty programs to keep them engaged and satisfied with their purchases.

Thanks!

Any questions?

aanhisbullah7@gmail.com
+62 896 7217 6606