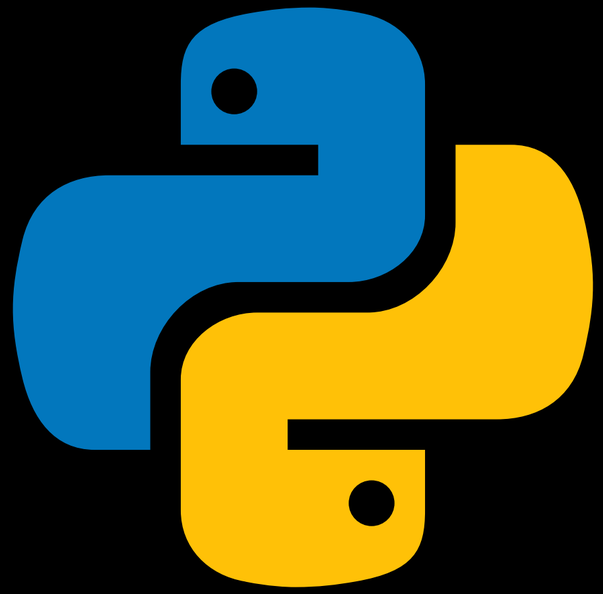
FULL STACK DATA ANALYTICS : WEEK 6-8

Python



Anang Hendro Wibowo - Section Barcelona

objectives

Gustavo just started an e-commerce startup based in Portugal that recently opened an online website to sell their product. Fortunately, Gustavo is launching their website when the Covid-19 hits and making them grow faster than ever.
However, Gustavo is still not using targeted marketing which hurts their marketing budget as only a fraction of their user comes back to their website. Gustavo needs your help to increase their marketing conversion rate by doing more targeted marketing user customer segmentation so that it will not hurt their budget.

tool

DATA CLEANING

- Import Datasets
- Remove Irrelevant Values
- Handling Missing Data
- Remove Duplicates
- Handling Outliers
- Join Datasets

INITIAL ANALYSIS

- Descriptive Analysis
- Exploratory Data Analysis

CLUSTER ANALYSIS

Customer segmentation

Exploratory Data Analysis and Descriptive Analysis

Descriptive Analysis

```
[ ] from scipy import stats


    def describe(df, statis):
      d = joined.describe()
      if 'mode' in statis:
        statis.remove('mode')
        d = d.append(joined.agg(statis))
        d = d.append(joined.agg(lambda x: stats.mode(x)[0]))
      else:
        d = d.append(joined.agg(statis))

      return d


    describe(joined, ['skew', 'mad','kurt','mode'])
```

```
    joined[['customer_city','customer_state','payment_type']].describe()
```

	payment_sequential	payment_installments	payment_value
count	93106.000000	93106.000000	93106.000000
mean	1.094827	2.629390	110.011894
std	0.677960	2.451304	72.682926
min	1.000000	0.000000	0.000000
25%	1.000000	1.000000	54.000000
50%	1.000000	1.000000	92.205000
75%	1.000000	3.000000	150.460000
max	26.000000	24.000000	344.340000
skew	14.557361	1.792364	0.998451
mad	0.180926	1.872394	58.130078
kurt	298.105063	3.073490	0.496030
0	1.000000	1.000000	50.000000

	customer_city	customer_state	payment_type
count	93106	93106	93106
unique	3965	27	4
top	sao paulo	SP	credit_card
freq	14741	39706	68325

Output :

From the descriptive analysis, we can get several insights :

- There are 93106 transactions on the dataset
- The average and mode of payment sequential are 1, which means that most customers usually only use one type of payment type.
- The mode of payment installments is 1 meaning that most customers usually only pay in single installments
- The average payment value is 110 with the smallest being 0 and the biggest being 344.
- All of the payment columns skewed positively (as seen on the histogram)
- The city with the most transactions is Sao Paulo with 14741 transactions.
- The state with the most transactions is SP with 39706 transactions
- The most preferred payment type is a credit card with 68325 transactions
- The datasets consist of transactions from 27 states and 3965 cities with 4 different payment types

Exploratory Data Analysis (1)

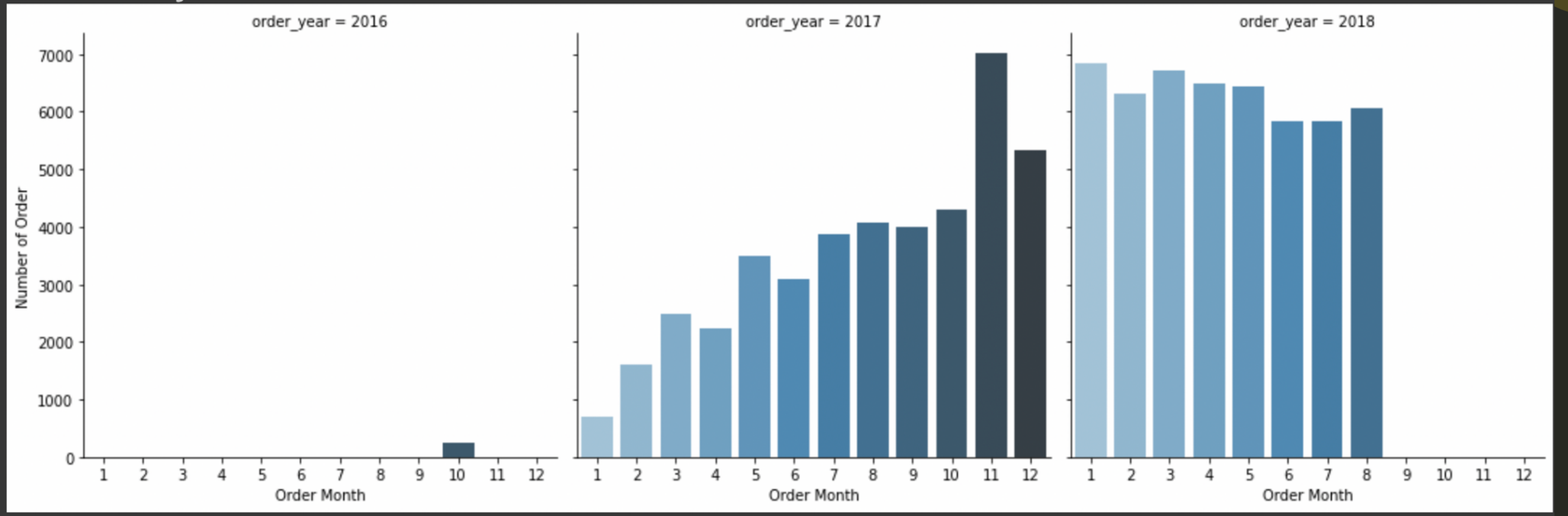
```
#ADD NEW COLUMN, CONVERT DATA TYPES AND GROUPING
eda1['order_year'] = pd.to_datetime(eda1['order_purchase_timestamp']).dt.year
eda1b = eda1.groupby(['order_year','order_month'],as_index=False)['order_purchase_timestamp'].count()
eda1b
```

Output :

	order_year	order_month	order_purchase_timestamp
0	2016	10	256
1	2016	12	1
2	2017	1	708
3	2017	2	1605
4	2017	3	2494
5	2017	4	2231
6	2017	5	3502
7	2017	6	3094
8	2017	7	3884
9	2017	8	4077
10	2017	9	4010
11	2017	10	4311
12	2017	11	7013
13	2017	12	5342
14	2018	1	6839
15	2018	2	6325
16	2018	3	6729
17	2018	4	6502
18	2018	5	6445
19	2018	6	5824
20	2018	7	5840
21	2018	8	6074

```
#VISUALIZE USING BAR CHART PER ORDER YEAR
ax = sns.catplot(x='order_month',y='order_purchase_timestamp',col='order_year',kind='bar',ci=None,data=eda1b,palette="Bl
ax.set(xlabel='Order Month', ylabel='Number of Order')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f1bc206b7d0>
```



From the chart, we can see the number of orders per month grouped by order year. In 2016 we don't see any order except for October. In 2017, the number of orders generally increase throughout the year, peaking in November. Number of orders jumping down in December 2017. In 2018, the number of orders is stable without extreme increment/decrement. The number of orders is noticably higher than the previous year with January's order being very similar with orders of November 2017 and keep performing well for the upcoming months.

Exploratory Data Analysis (2)

```
[ ]  #CONVERT DATA TYPE AND NEW COLUMN
     eda2 = joined.copy()
     eda2['order_day'] = pd.to_datetime(eda1['order_purchase_timestamp']).dt.day_name()
     eda2.head()
```

```
[ ]  #ORDER DAY IN ACTUAL DAILY ORDER
     from pandas.api.types import CategoricalDtype
     cats = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday']
     cat_type = CategoricalDtype(categories=cats, ordered=True)
     eda2['order_day'] = eda2['order_day'].astype(cat_type)
```

```
     #CALCULATE ORDER PER DAY
     eda2 = eda2.groupby(['order_day'],as_index=False)['order_purchase_timestamp'].count()
     eda2
```

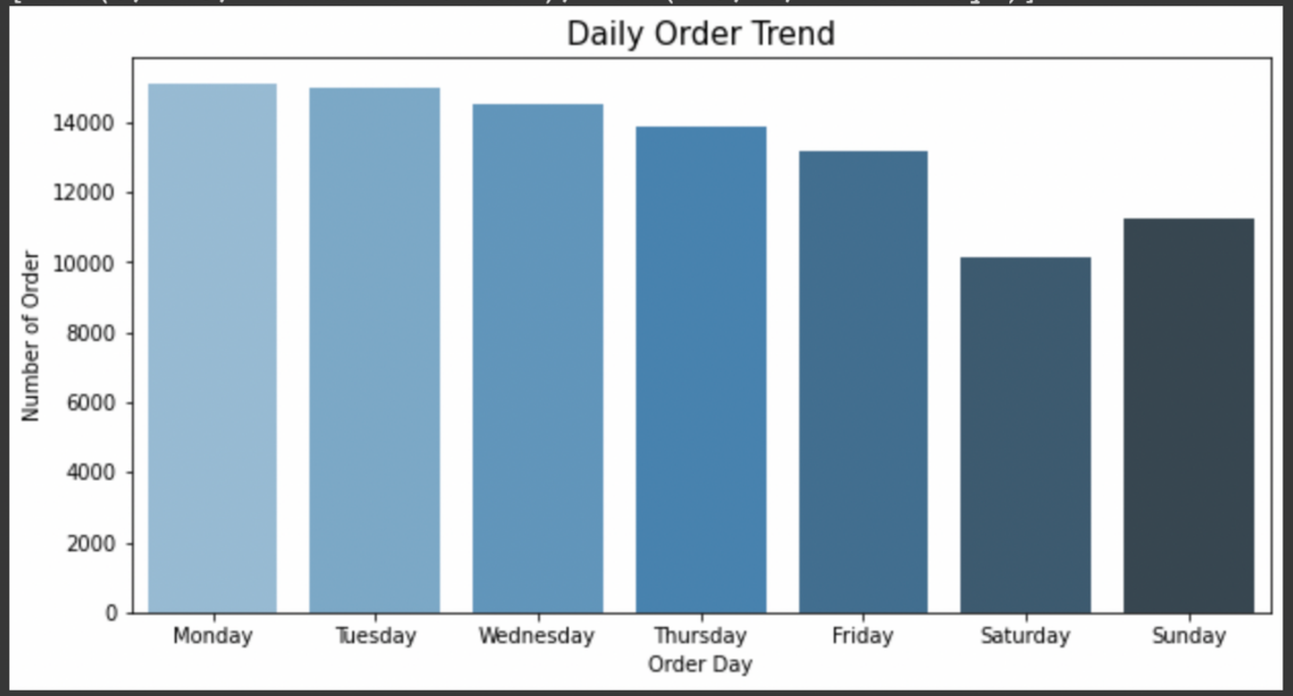
Output :

	order_day	order_purchase_timestamp
0	Monday	15102
1	Tuesday	15011
2	Wednesday	14542
3	Thursday	13851
4	Friday	13184
5	Saturday	10160
6	Sunday	11256

From the "Daily Order Trend" graph you could see that highest order occured on monday. Higher number of orders generally happened on weekdays rather than the weekend.The number of orders keep decreasing as days progressed. These trend could means that weekdays order generally happens because customers usually want to received the order in the same week or assume that delivery time is shorter because it is within the same week. Other possible reason that in the weekend the customers could be doing other activities other than browsing and ordering the products. By giving special promo or discount code could potentially increase the orders even more.

```python
#VISUALIZE IN BARCHART
fig, ax = plt.subplots(figsize=(10, 5))
ax = sns.barplot(x='order_day', y='order_purchase_timestamp', data=eda2, orient='v', ci=None,palette="Blues_d")
plt.title('Daily Order Trend',fontsize=15)
ax.set(xlabel='Order Day', ylabel='Number of Order')
```

```
[Text(0, 0.5, 'Number of Order'), Text(0.5, 0, 'Order Day')]
```



Exploratory Data Analysis (3)

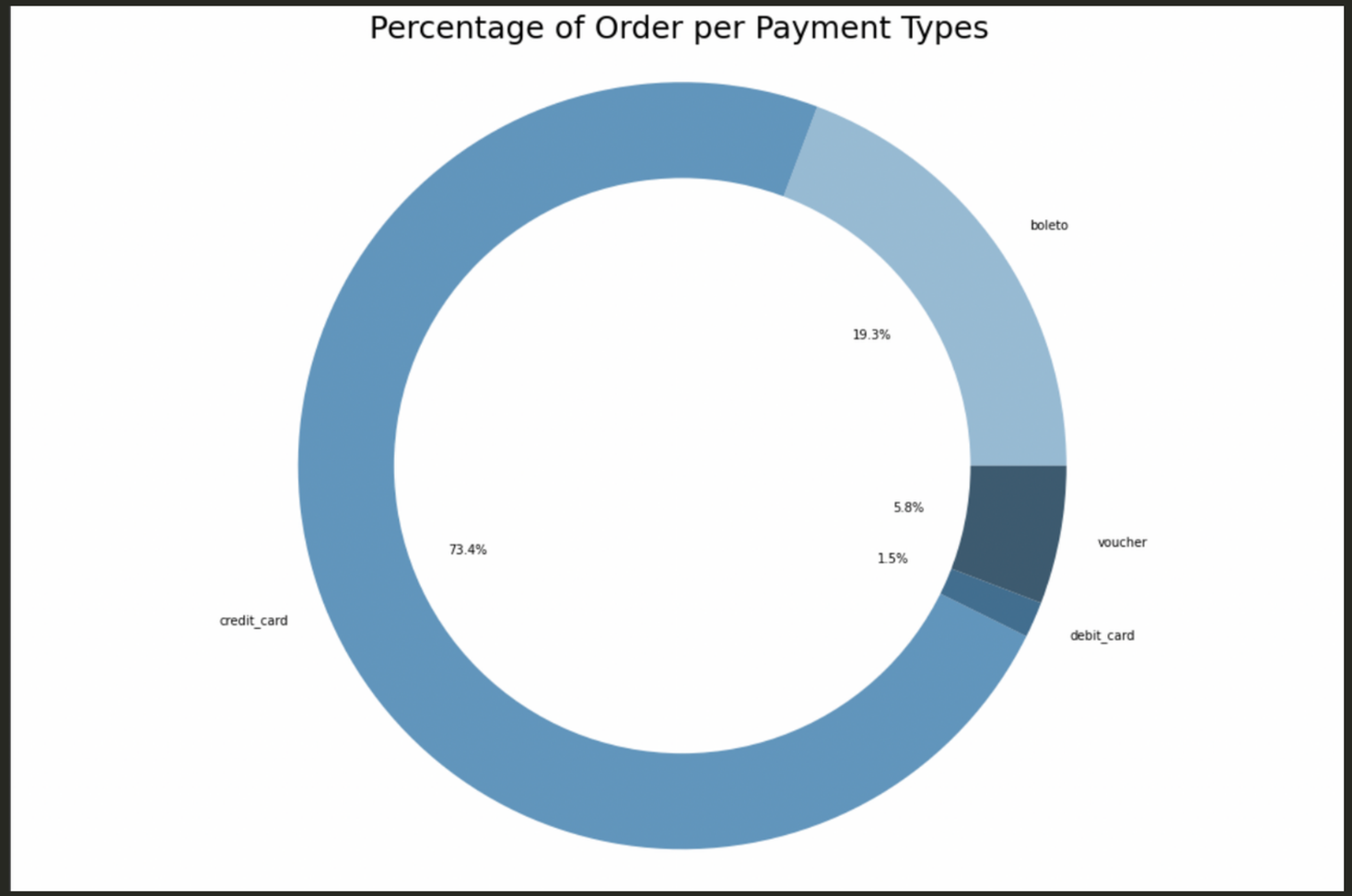
```
#CALCULATE NUMBER OF ORDER FOR EACH PAYMENT TYPES
eda3 = eda3.groupby(['payment_type'],as_index=False)['order_id'].count()
eda3
```

	payment_type	order_id
0	boleto	17953
1	credit_card	68325
2	debit_card	1406
3	voucher	5422

```
#SHOW PERCENTAGE NUMBER OF ORDERS FOR EACH PAYMENT TYPES
eda3['percentage'] = (eda3['order_id']/eda3['order_id'].sum())*100
eda3
```

	payment_type	order_id	percentage
0	boleto	17953	19.282323
1	credit_card	68325	73.384100
2	debit_card	1406	1.510107
3	voucher	5422	5.823470

```
#VISUALIZE WITH PIE CHART
plt.figure(figsize=(15, 10))
my_data = eda3['order_id']
my_labels = eda3['payment_type']
colors = ['#98bbd4','#6296bc','#436f90','#3d5b70']

plt.pie(my_data,colors=colors,labels=my_labels,autopct='%1.1f%%')
centre_circle = plt.Circle((0,0),0.75,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title('Percentage of Order per Payment Types', fontsize=25)
plt.axis('equal')
plt.tight_layout()
plt.show()
```



Percentages of order per payment types shows that highest payment type of choice is credit card with 73,4%, followed by boleto (19.3%), voucher (5.8%) and debit card (1.5%). With these finding, the company could made special promotion or discount for credit card payment to keep the usage of credit card payment in the higher percentage or even increase it.

Clustering Analysis : Customer Segmentation

Preparation

```
#IMPORT LIBRARY
from sklearn.datasets import make_blobs
from sklearn import cluster
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
```

```
raw = joined.copy()
raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 93106 entries, 0 to 93105
Data columns (total 14 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   order_id                       93106 non-null  object
 1   customer_id                    93106 non-null  object
 2   order_status                   93106 non-null  object
 3   order_purchase_timestamp       93106 non-null  object
 4   order_approved_at              93106 non-null  object
 5   order_delivered_carrier_date   93106 non-null  object
 6   order_delivered_customer_date  93106 non-null  object
 7   customer_unique_id             93106 non-null  object
 8   customer_city                  93106 non-null  object
 9   customer_state                 93106 non-null  object
 10  payment_sequential             93106 non-null  int64
 11  payment_type                   93106 non-null  object
 12  payment_installments           93106 non-null  int64
 13  payment_value                  93106 non-null  float64
dtypes: float64(1), int64(2), object(11)
memory usage: 10.7+ MB
```

Transform Categorical Columns

```
#MAKE DUMMY DATASET AND MERGE THE DATASET
dum_df = pd.get_dummies(df_cluster, columns=["payment_type"], prefix=["payment_type"])
df_cluster = df_cluster.merge(dum_df)
df_cluster
```

payment_type	payment_installments	payment_value	payment_type_boleto	payment_type_credit_card	payment_type_debit_card	payment_type_voucher
credit_card	1	18.12	0	1	0	0
voucher	1	2.00	0	0	0	1
voucher	1	18.59	0	0	0	1
boleto	1	141.46	1	0	0	0
credit_card	3	179.12	0	1	0	0
...	...	...	...	...	...	...
credit_card	3	155.99	0	1	0	0
credit_card	3	85.08	0	1	0	0
credit_card	3	195.00	0	1	0	0
credit_card	5	271.01	0	1	0	0
debit_card	1	86.86	0	0	1	0

Drop Irrelevant Columns

```
[ ]  #REMOVE IRRELEVANT COLUMNS
     df_cluster = df_cluster.drop(['order_id','customer_id','order_status','order_purchase_timestamp',
                                   'order_approved_at','order_delivered_carrier_date','order_delivered_customer_date',
                                   'customer_unique_id','customer_city','customer_state','payment_type'],axis=1)
```

Scale the Numerical Columns

```
     #STANDARDIZE THE NUMERICAL VALUE
     from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     df_cluster[num_column] = scaler.fit_transform(df_cluster[num_column])
     df_cluster
```

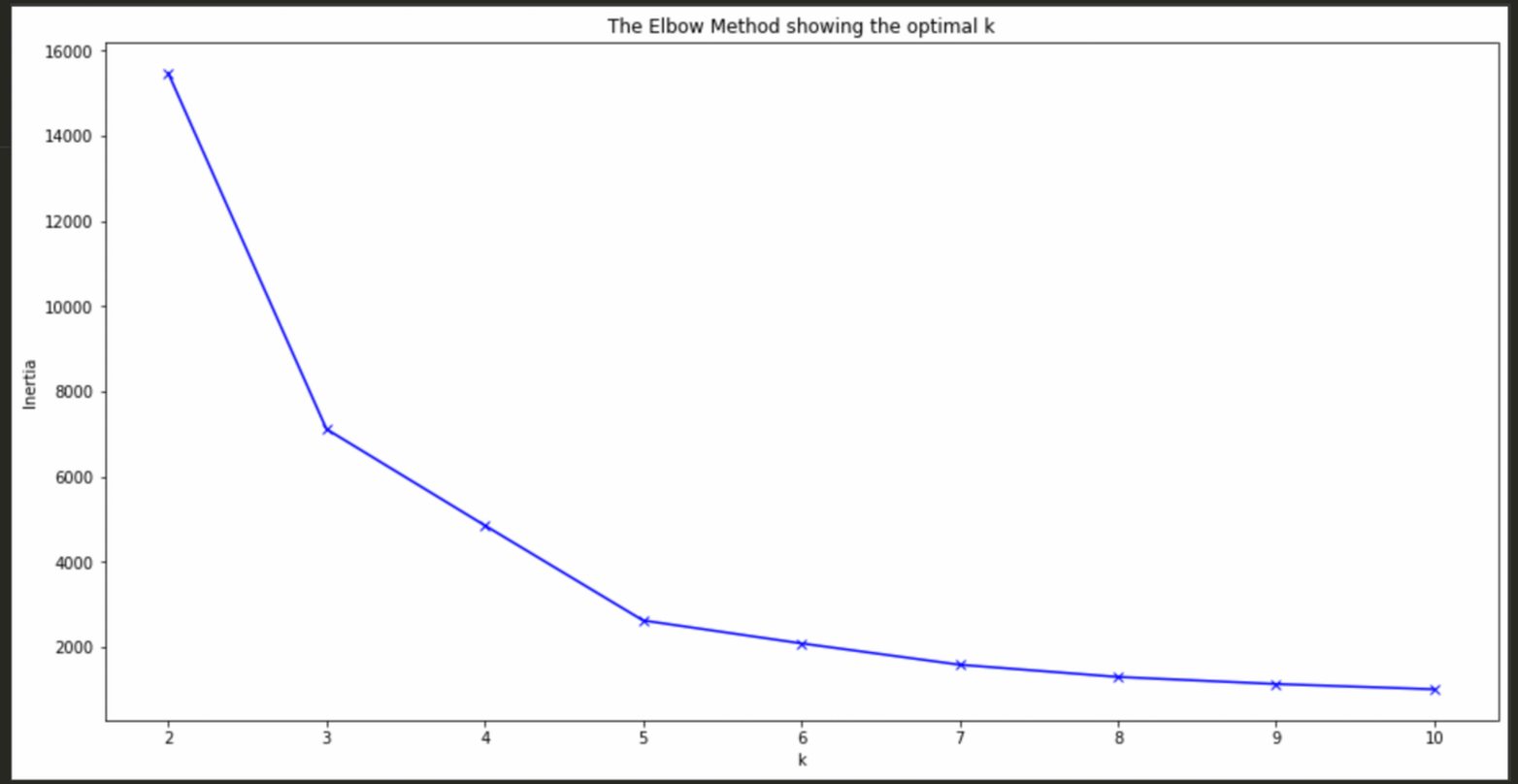
Output :

	payment_sequential	payment_installments	payment_value	payment_type_boleto	payment_type_credit_card	payment_type_debit_card	payment_type_voucher
0	0.00	0.041667	0.052622	0	1	0	0
1	0.08	0.041667	0.005808	0	0	0	1
2	0.04	0.041667	0.053987	0	0	0	1
3	0.00	0.041667	0.410815	1	0	0	0
4	0.00	0.125000	0.520184	0	1	0	0
...	...	...	...	...	...	...	...
93101	0.00	0.125000	0.453012	0	1	0	0
93102	0.00	0.125000	0.247081	0	1	0	0
93103	0.00	0.125000	0.566301	0	1	0	0
93104	0.00	0.208333	0.787042	0	1	0	0
93105	0.00	0.041667	0.252251	0	0	1	0

93106 rows × 7 columns

Find Optimal K

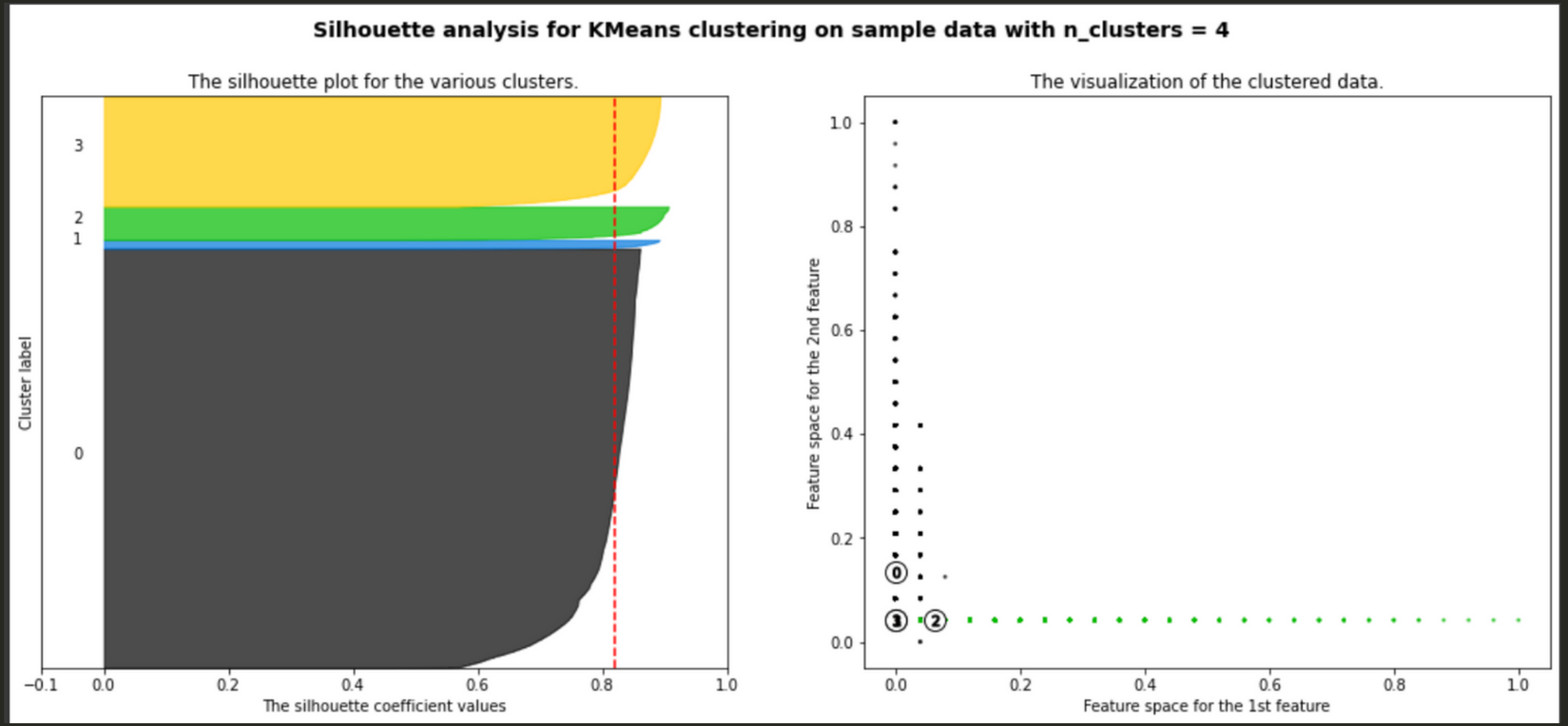
```
[ ]  #FIND OPTIMAL K
     distortions = []
     K = range(2,11)
     for k in K:
         kmeanModel = cluster.KMeans(n_clusters=k)
         kmeanModel.fit(df_cluster)
         distortions.append(kmeanModel.inertia_)

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



Calculate Silhouette Score

```
[ ]  silhoutte_analysis(df_cluster,list(range(2,11)))
```

```
For n_clusters = 2 The average silhouette_score is : 0.7243029492624402
For n_clusters = 3 The average silhouette_score is : 0.7990098651384397
For n_clusters = 4 The average silhouette_score is : 0.8194417362296466
For n_clusters = 5 The average silhouette_score is : 0.6122814745768793
For n_clusters = 6 The average silhouette_score is : 0.5491277993683532
For n_clusters = 7 The average silhouette_score is : 0.5086057744266824
For n_clusters = 8 The average silhouette_score is : 0.5286603677602545
For n_clusters = 9 The average silhouette_score is : 0.4878080059738101
For n_clusters = 10 The average silhouette_score is : 0.4984470481729299
```



Clustering

```
[ ] #CLUSTERING
    cluster_model = cluster.KMeans(n_clusters=4)
    cluster_model.fit(df_cluster)
    cluster_label = cluster_model.labels_
    raw['cluster'] = cluster_label
    raw
```

red_customer_date	customer_unique_id	customer_city	customer_state	payment_sequential	payment_type	payment_installments	payment_value	cluster
2017-10-10 21:25:13	7c396fd4830fd04220f754e42b4e5bff	sao paulo	SP	1	credit_card	1	18.12	1
2017-10-10 21:25:13	7c396fd4830fd04220f754e42b4e5bff	sao paulo	SP	3	voucher	1	2.00	2
2017-10-10 21:25:13	7c396fd4830fd04220f754e42b4e5bff	sao paulo	SP	2	voucher	1	18.59	2
2018-08-07 15:27:45	af07308b275d755c9edb36a90c618231	barreiras	BA	1	boleto	1	141.46	0
2018-08-17 18:06:29	3a653a41f6f9fc3d2a113cf8398680e8	vianopolis	GO	1	credit_card	3	179.12	1
...	...	...	...	...	...	...	...	...
2017-03-06 11:08:08	831ce3f1bacbd424fc4e38fbd4d66d29	sao paulo	SP	1	credit_card	3	155.99	1
2017-03-17 15:08:01	6359f309b166b0196dbf7ad2ac62bb5a	sao jose dos campos	SP	1	credit_card	3	85.08	1
2018-02-28 17:37:56	da62f9e57a76d978d02ab5362c509660	praia grande	SP	1	credit_card	3	195.00	1
2017-09-21 11:24:17	737520a9aad80b3fbbdad19b66b37b30	nova vicosa	BA	1	credit_card	5	271.01	1
2018-03-16 13:08:30	60350aa974b26ff12caad89e55993bd6	lapa	PR	1	debit_card	1	86.86	3

Analyze the Clusters (1)

```
[ ] raw.groupby(['cluster'])['payment_value'].agg(['sum','count','mean','median','max','min'])
```

cluster	sum	count	mean	median	max	min
0	4372900.46	21358	204.742975	191.58	344.34	112.91
1	1917751.94	17953	106.820695	87.38	344.33	11.62
2	446585.44	6828	65.405015	47.70	341.02	0.00
3	3505529.56	46967	74.638141	71.53	149.25	0.01

```
[ ] raw.groupby(['cluster'])['payment_installments'].agg(['count','mean','median','max','min'])
```

cluster	count	mean	median	max	min
0	21358	4.805085	4.0	24	1
1	17953	1.000000	1.0	1	1
2	6828	1.000000	1.0	1	1
3	46967	2.499713	2.0	11	0

Analyze the Clusters (2)

```
[ ] raw.groupby(['cluster'])['payment_sequential'].agg(['count','mean','median','max','min'])
```

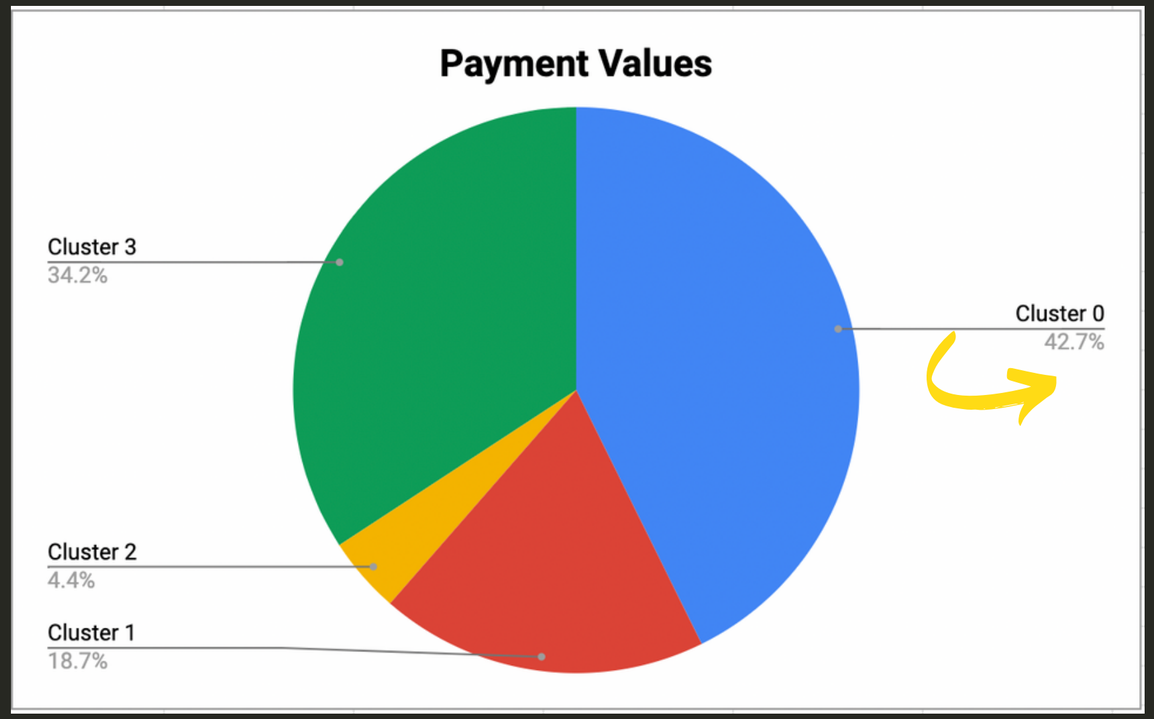
cluster	count	mean	median	max	min
0	21358	1.002997	1.0	2	1
1	17953	1.000056	1.0	2	1
2	6828	2.252343	2.0	26	1
3	46967	1.004535	1.0	3	1

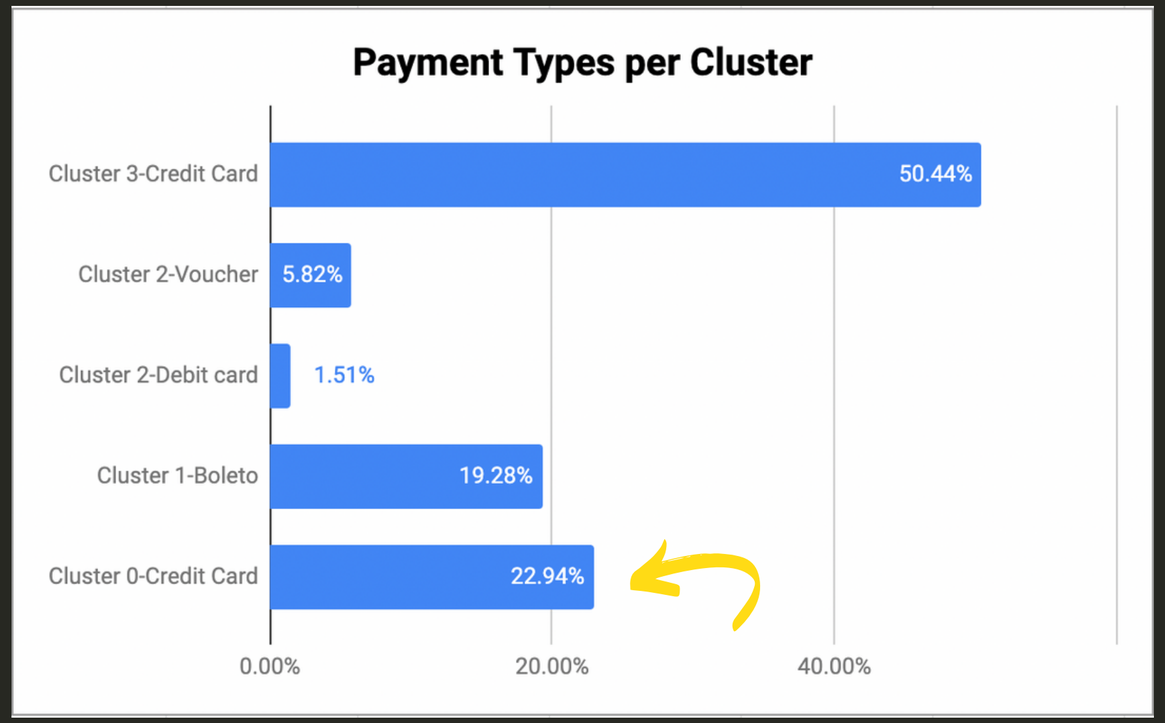
```
[ ] raw.groupby(['cluster','payment_type'])['payment_sequential'].agg(['count','mean','median','max','min'])
```

cluster	payment_type	count	mean	median	max	min
0	credit_card	21358	1.002997	1.0	2	1
1	boleto	17953	1.000056	1.0	2	1
2	debit_card	1406	1.034851	1.0	3	1
	voucher	5422	2.568056	2.0	26	1
3	credit_card	46967	1.004535	1.0	3	1

Cluster 0 : High Spending Trendy Group

- Consists of customers with high payment values but prefer longer installments
- We could assume that people in this cluster tend to buy expensive things such as the newest gadgets, trendy clothes or viral skincares, therefore they choose to have longer installments as it is hard to pay such big money in a single time.
- The cluster 0 prefer single sequential payment using credit cards as is it possible to have long-term installments plans.
- We could say that this specific cluster consists of young people or people in their early career who need to be updated with the trends (need another dataset to confirm).



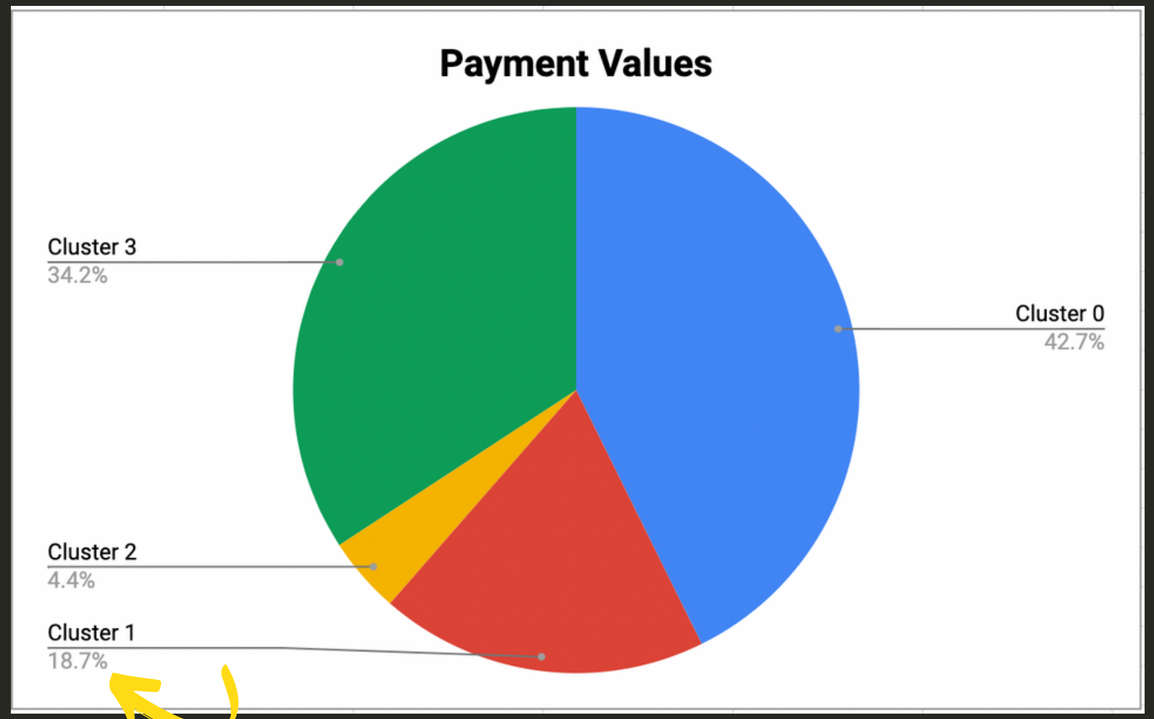


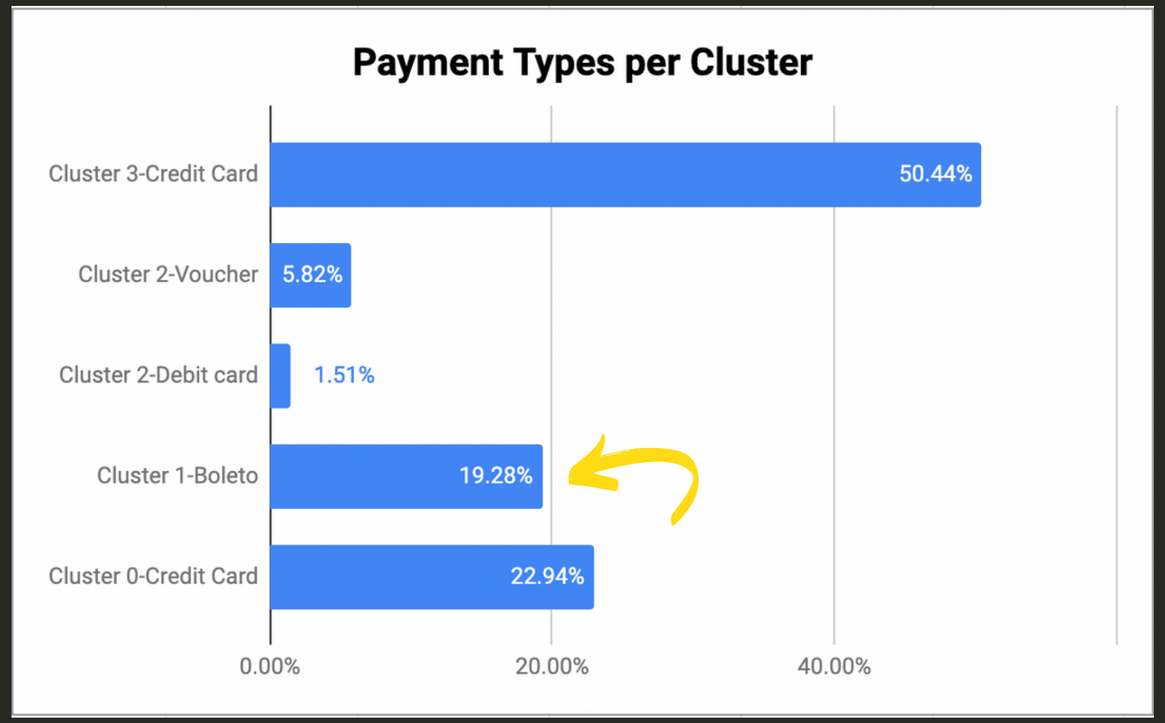
Recommendation :

- For 'High Spending Trendy Group' the company could push more trendy items such as newest gadgets, viral skincare or fashion items to them specifically and offer various installment payments as they usually spend in higher values.
- The company could give special promo or discount using credit cards for this group
- The company could propose 'paylater' method as alternative payment as they work in quite similar way with credit cards with various number of installments

Cluster 1 : Medium Spending Established Group

- Consists of customers with medium payment values who prefer single installment payment.
- We could assume that people in this cluster are adults or older adults who had established jobs/careers because they could afford to spend quite high payments in a single payment. They could be purchasing stuff for their own space e.g furniture, home decor, etc, or even for their hobbies (need another dataset to confirm).
- They pay with a single sequential using boleto. Boleto is commonly used by older people who feel that Boleto is safer and more convenient than banking payments such as credit/debit cards.



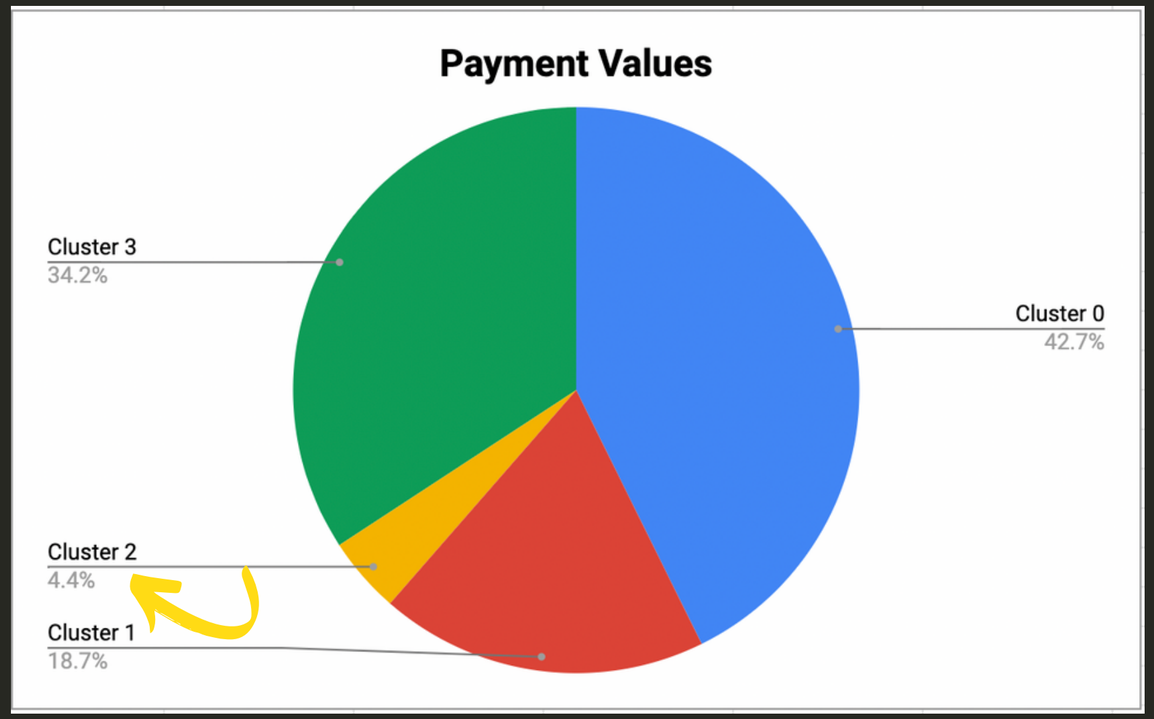


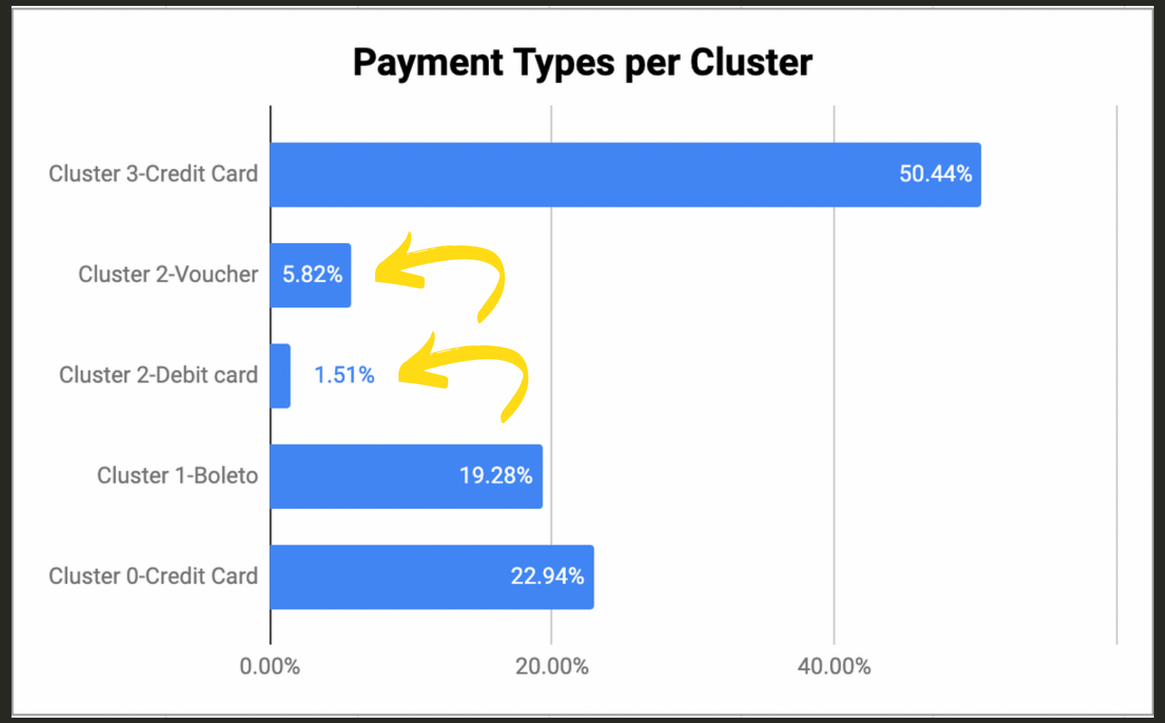
Recommendation :

- For 'Medium Spending Established Group', the company could offer a special discount/promotion code using boleto method as all of the people in this cluster only use boleto

Cluster 2 : Low Spending Frugal Group

- Consists of customers with low payment values who prefer single installment payments.
- This cluster specifically tends to pay in double sequential payments using debit cards combine with vouchers.
- We could assume that people in this cluster only purchase cheap items or items that they specifically need to buy e.g groceries, food, etc, considering their payment choices.
- This cluster could be consist of people with lower incomes or people with frugal life as they preferred to split the payments to different methods in order to portion their spending or get cheaper prices.



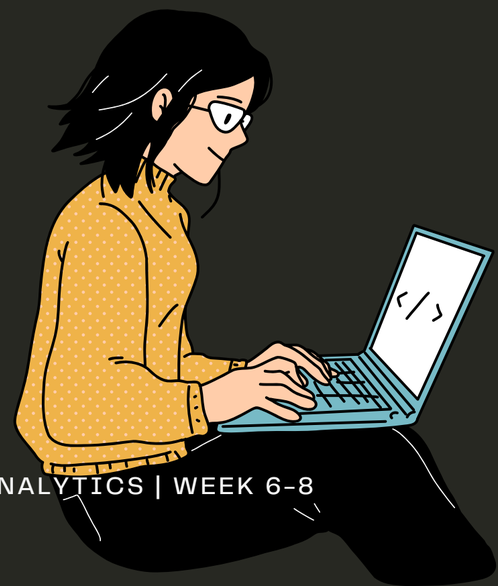


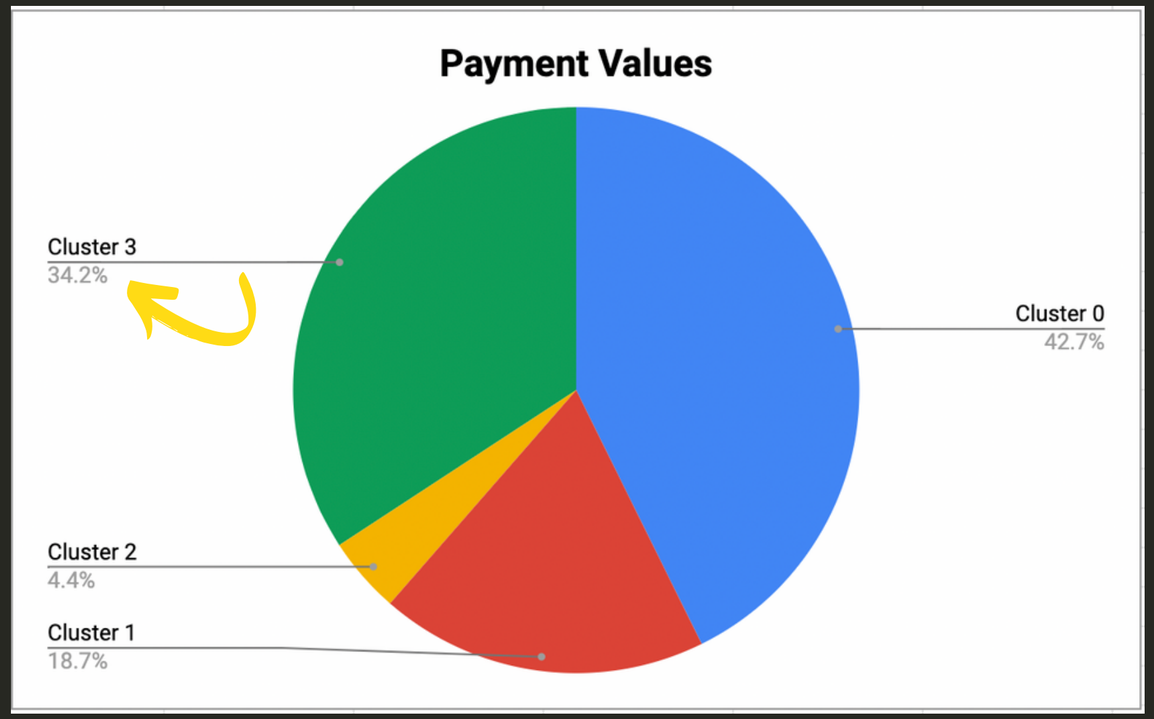
Recommendation :

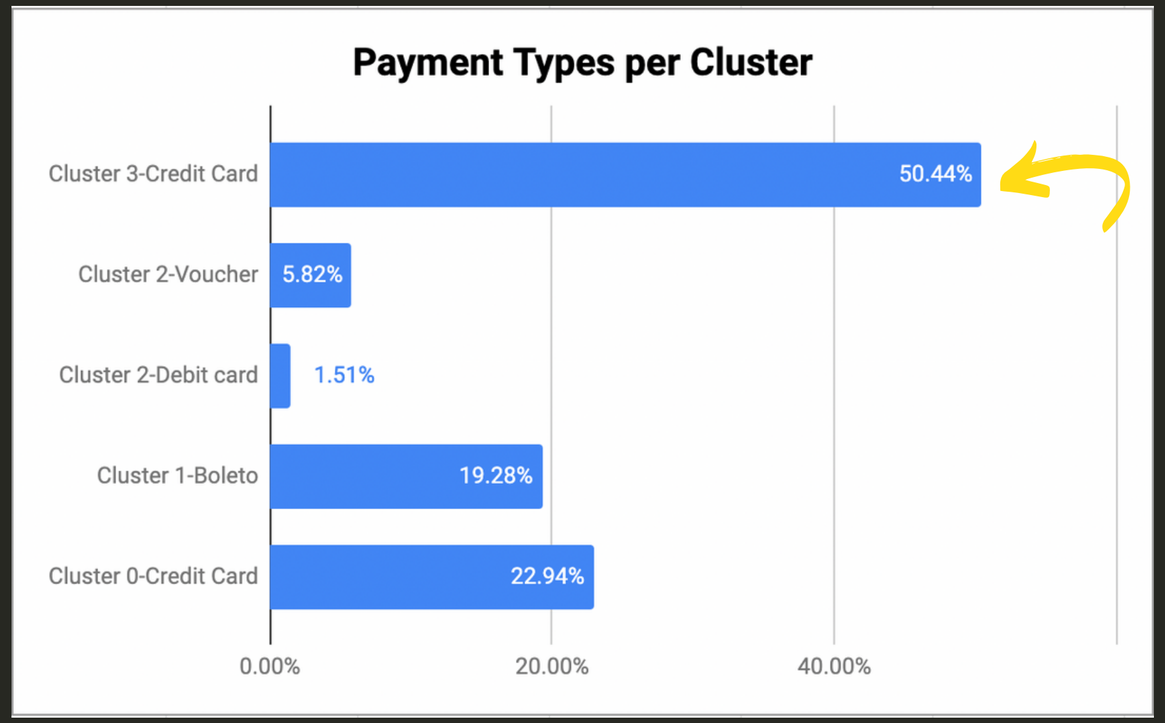
- For 'Low Spending Frugal Group', the company could offer special prices or discounts on groceries or daily products.
- The company could also hold special event/promo where people could buy products in bundles with special prices and give promos within their chosen payment types (debit cards and vouchers)

Cluster 3 : High Spending Mindful Group

- Consists of customers with high payment values with double installments payment with preferred single payment type using a credit card.
- This cluster shows similarity with Cluster 0 with the only difference being shorter payment installments.
- With that being said, we could assume that they had similar customer types with a tendency to purchase expensive products.
- However, shorter payment installment could mean that financially they're doing better than people in Cluster 0. They still want to keep up with the trends but also want to portion their spending by splitting the payment.







Recommendation :

- For 'High Spending Mindful Group' the company could push more trendy items such as newest gadgets, viral skincare or fashion items to them specifically and offer various installment payments as they usually spend in higher values.
- The company could give special promo or discount using credit cards for this group
- The company could propose 'paylater' method as alternative payment as they work in quite similar way with credit cards with various number of installments

Assign Cluster Names

```
[ ]  #ASSIGN THE NAME OF EACH CLUSTER
     raw['cluster'] = raw['cluster'].map({
         0: 'High Spending Trendy Group',
         1: 'Medium Spending Established Group',
         2: 'Low Spending Frugal Group',
         3: 'High Spending Mindful Group',
     })

     raw
```

ed_customer_date	customer_unique_id	customer_city	customer_state	payment_sequential	payment_type	payment_installments	payment_value	cluster
2017-10-10 21:25:13	7c396fd4830fd04220f754e42b4e5bff	sao paulo	SP	1	credit_card	1	18.12	High Spending Trendy Group
2017-10-10 21:25:13	7c396fd4830fd04220f754e42b4e5bff	sao paulo	SP	3	voucher	1	2.00	Low Spending Frugal Group
2017-10-10 21:25:13	7c396fd4830fd04220f754e42b4e5bff	sao paulo	SP	2	voucher	1	18.59	Low Spending Frugal Group
2018-08-07 15:27:45	af07308b275d755c9edb36a90c618231	barreiras	BA	1	boleto	1	141.46	Medium Spending Established Group

Conclusion

The current analysis could be useful to a certain extent. However, several issues could be addressed in addition to another dataset so that it can help define the problems more clearly.

Google Collab File

Thank You
Thank You
Thank You