Projects Summary

1. KPMG Customer segmentation

Data Understanding

- The data was made available by KPMG as a part of their virtual internship program during the pandemic.
- The dataset has customer demographics/characterisitics and all customer transactions from January 2017 to December 2017 of a bicycle manufacturer and seller
- The Customer demographics data has the following features:
  - 'customer_id', 'first_name', 'last_name', 'gender','past_3_years_bike_related_purchases', 'job_title','job_industry_category', 'wealth_segment', 'deceased_indicator',’owns_car', 'tenure', 'Age'
- The Customer transactions data has the following features:
  - 'transaction_id', 'product_id', 'customer_id', 'transaction_date','online_order', 'order_status', 'brand', 'product_line','product_class', 'product_size', 'list_price', 'standard_cost', 'product_first_sold_date'

Data Preparation

- Identified null values in each column and based on the % of missing values imputed null values with the appropriate mean/median/mode of the column or dropped the entire column.
- During the data cleaning process, it was also discovered that some customers had their date of birth in early 1900 which made them over 100 years of age, such customers were dropped from the dataset
- Some of the operations performed to prepare the data for further analysis are given below
-

```
#Since this customer's DOB is 1835 which makes him 175 years old
df2 = df2[df2.customer_id != 34]
```

```
df2['Age'].fillna(df2['Age'].mean(),inplace=True)
```

```
df2.drop(columns='DOB',inplace=True)
```

```
df2['gender'].value_counts()
```

```
Female    2037
Male      1872
U           87
F            1
M            1
Femal        1
Name: gender, dtype: int64
```

```
df2['gender'] = df2['gender'].replace(['Femal', 'F'], 'Female')
df2['gender'] = df2['gender'].replace('M', 'Male')
```

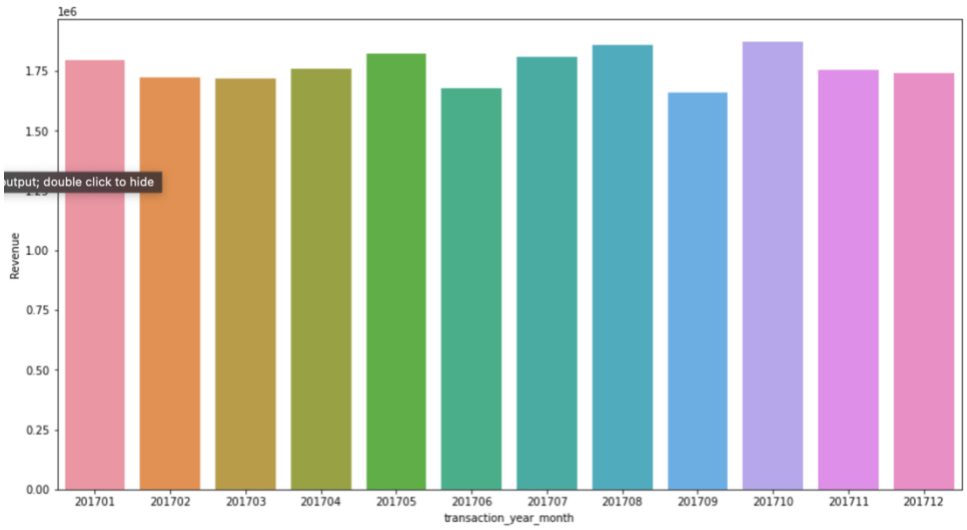
-

Exploratory Data Analysis

- Converted the transaction date to date time format and created a ‘year month’ column for ease of reporting and visualization

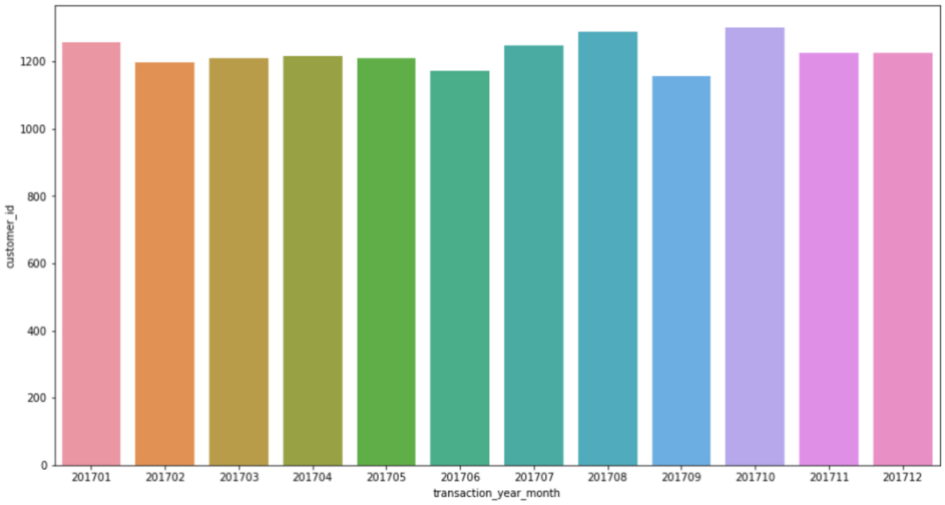
```
#converting the type of Invoice Date Field from string to datetime.
df_main['transaction_date'] = pd.to_datetime(df_main['transaction_date'])

#creating YearMonth field for the ease of reporting and visualization
df_main['transaction_year_month'] = df_main['transaction_date'].map(lambda date: 100*date.year + date.month)
```

The bar graph below shows the monthly revenue, which remains somewhat stable, but we can see slight increase and decrease over the few months



Monthly Active Customers

```
#creating monthly active customers dataframe by counting unique Customer IDs
df_monthly_active = df_main.groupby('transaction_year_month')['customer_id'].nunique().reset_index()
```

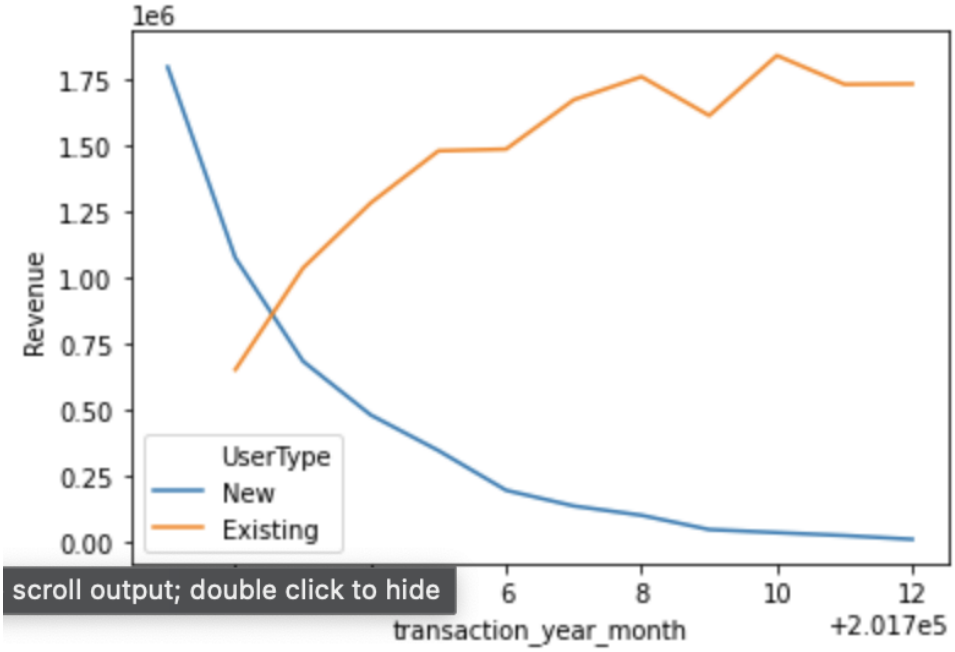


New Customer Ratio is a good indicator of whether the company is losing existing customers or unable to attract new ones.
First, it should be defined what is a new customer. In the dataset, we can assume a new customer is whoever did his/her first purchase in the time window defined.

Using .min() function first purchase date for each customer can be found and new customers can be defined based on that. The following operation was performed to get this result:

```
df_min_purchase =
df_main.groupby('customer_id').transaction_date.min().re
set_index()
df_min_purchase.columns =
['customer_id','MinPurchaseDate']
df_min_purchase['MinPurchaseYearMonth'] =
df_min_purchase['MinPurchaseDate'].map(lambda date:
100*date.year + date.month)

#Merging the dataframes on customer ID
df_main = pd.merge(df_main, df_min_purchase,
on='customer_id')

df_main['UserType'] =
np.where(df_main['transaction_year_month']>df_main['MinP
urchaseYearMonth'],'Existing','New')
```

The following line graph shows the revenue from Existing and new customers. It is quite evident that the revenue from new customers decreases from January to December



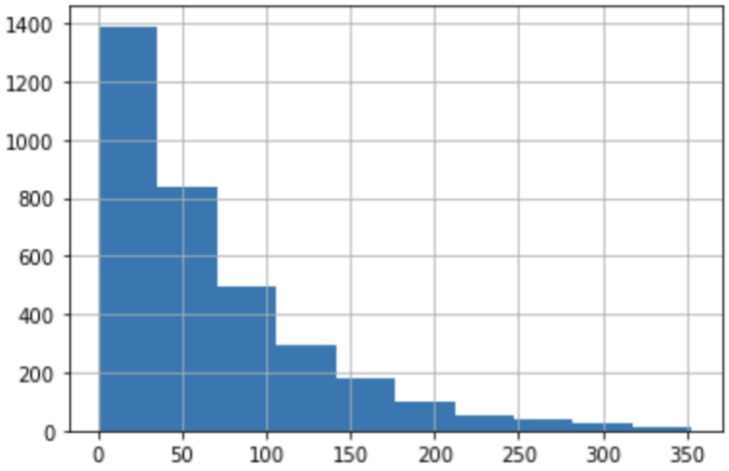
Segmentation

There are several ways to segment the customers depending on the purpose of segmentation. In this case the purpose of segmentation was to identify the most valued customers and their personas. The method used for in this case is Recency Frequency and Monetary Analysis also known as RFM Analysis.

Recency – Who are the most recent customers? Find the most recent date of purchase date of each customer and see the time they are inactive for. After finding number of inactive days for each customer, k-means clustering is used to assign customers a Recency score.

```
tx_user = pd.DataFrame(tx_data['customer_id'].unique())
tx_user.columns = ['customer_id']

#get the max purchase date for each customer and create a dataframe with it
tx_max_purchase = tx_data.groupby('customer_id').transaction_date.max().reset_index()
tx_max_purchase.columns = ['customer_id','MaxPurchaseDate']

#we take our observation point as the max invoice date in our dataset
tx_max_purchase['Recency'] = (tx_max_purchase['MaxPurchaseDate'].max() - tx_max_purchase['MaxPurchaseDate']).dt.days

#merge this dataframe to our new user dataframe
tx_user = pd.merge(tx_user, tx_max_purchase[['customer_id','Recency']], on='customer_id')

tx_user.head()
```

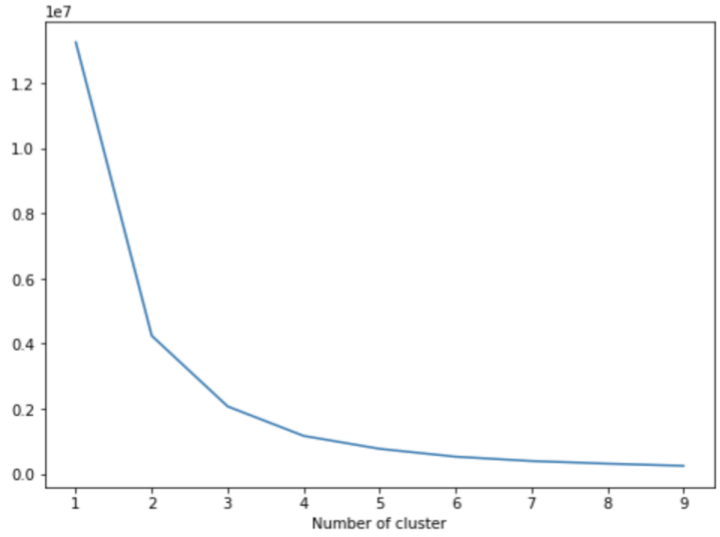
Although the mean recency is 65 days the median is 46 days which means the recency is right skewed. This is confirmed by the histogram of recency shown below:



	customer_id	Recency
count	3419.000000	3419.000000
mean	1752.774788	65.446329
std	1009.259013	62.308217
min	1.000000	0.000000
25%	879.500000	19.000000
50%	1752.000000	46.000000
75%	2626.500000	91.000000
max	3499.000000	353.000000

The next step involves applying K-means clustering to assign a recency score. In order to find the optimal number of clusters I have applied the elbow method using the following function:

```
from sklearn.cluster import KMeans

sse={}
tx_recency = tx_user[['Recency']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(tx_recency)
    tx_recency["clusters"] = kmeans.labels_
    sse[k] = kmeans.inertia_
plt.figure(figsize=(8,6))
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()
```



Here the optimal number of clusters looks to be 4. However, I have selected 3 clusters in each case.

```
#build 3 clusters for recency and add it to dataframe
kmeans = KMeans(n_clusters=3)
kmeans.fit(tx_user[['Recency']])
tx_user['RecencyCluster'] = kmeans.predict(tx_user[['Recency']])
```

Since K-means clustering assigns clusters as numbers but not in an ordered way, I have used the following function in order to arrange the clusters from best to worst.

```
#function for ordering cluster numbers
def order_cluster(cluster_field_name, target_field_name,df,ascending):
    new_cluster_field_name = 'new_' + cluster_field_name
    df_new = df.groupby(cluster_field_name)[target_field_name].mean().reset_index()
    df_new = df_new.sort_values(by=target_field_name,ascending=ascending).reset_index(drop=True)
    df_new['index'] = df_new.index
    df_final = pd.merge(df,df_new[[cluster_field_name,'index']], on=cluster_field_name)
    df_final = df_final.drop([cluster_field_name],axis=1)
    df_final = df_final.rename(columns={"index":cluster_field_name})
    return df_final
```

```
tx_user = order_cluster('RecencyCluster', 'Recency',tx_user,False)
```

```
tx_user.groupby('RecencyCluster')['Recency'].describe()
```

RecencyCluster	count	mean	std	min	25%	50%	75%	max
0	337.0	209.489614	47.685734	154.0	170.0	196.0	236.0	353.0
1	1030.0	96.873786	25.486131	62.0	75.0	91.0	115.0	153.0
2	2052.0	26.015107	17.627249	0.0	11.0	24.0	41.0	61.0

Thus, it is clear from the table above that Cluster 2 has the most recent customers as compared to Cluster 0 and Cluster 1.

Frequency – To create frequency clusters, I found the total number of orders for each customer.

```
tx_frequency = tx_data.groupby(['customer_id']).transaction_date.count().reset_index()
tx_frequency.columns = ['customer_id','Frequency']
```

```
tx_frequency
```

Applying the same logic for creating frequency clusters and assigning this to each customer.

	customer_id	Frequency
0	1	11
1	2	3
2	5	6
3	6	3
4	7	3
...	...	...
3414	3495	5
3415	3496	4
3416	3497	3
3417	3498	6
3418	3499	7

FrequencyCluster	count	mean	std	min	25%	50%	75%	max
0	1306.0	3.080398	0.929119	1.0	2.0	3.0	4.0	4.0
1	1530.0	5.875817	0.804014	5.0	5.0	6.0	7.0	7.0
2	583.0	9.048027	1.246752	8.0	8.0	9.0	10.0	14.0

Similar to the recency clusters, I have used the order function to arrange the clusters from worse to best. High frequency number indicates better customers and thus cluster 2 has more frequent customers as compared to the other clusters.

Revenue – The final clustering would be based on revenue. It is important to calculate the revenue generated by each customer before applying the same clustering method as above.

```
tx_revenue = tx_data.groupby(['customer_id']).Revenue.sum().reset_index()
```

RevenueCluster	count	mean	std	min	25%	50%	75%	max
0	1291.0	3355.254531	1163.917186	60.34	2546.060	3561.17	4315.54	5012.31
1	1477.0	6688.678937	1039.377717	5012.97	5810.690	6618.86	7545.15	8695.82
2	651.0	10733.385637	1725.083083	8697.36	9409.875	10296.12	11547.73	19071.32

Thus, Cluster 2 contains customers who have generated the maximum revenue in 2017 as compared to Cluster 1 and Cluster 0.

Scores for Recency, Frequency and Revenue/Monetary are now available. The next step is to calculate an overall score based on these 3 individual scores.

The overall score is calculated as follows:

```
#calculate overall score and use mean() to see details
tx_user['OverallScore'] = tx_user['RecencyCluster'] + tx_user['FrequencyCluster'] + tx_user['RevenueCluster']
tx_user.groupby('OverallScore')['Recency','Frequency','Revenue'].mean()
```

OverallScore	Recency	Frequency	Revenue
0	217.622407	2.477178	2783.901162
1	107.937656	2.960100	3334.776259
2	59.860856	3.642202	4100.429052
3	62.790769	5.276923	5845.697246
4	33.900645	6.086452	7094.875097
5	39.760355	7.721893	9237.718077
6	23.211111	9.347222	11355.056278

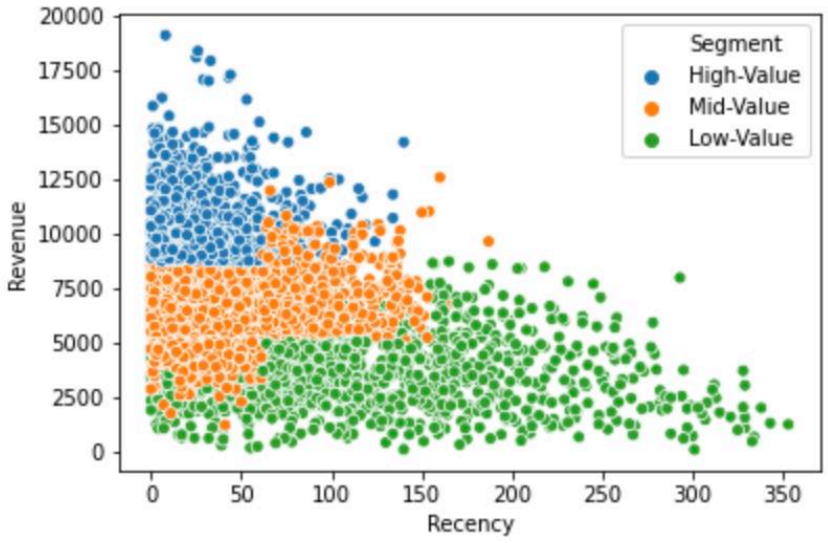
According to the overall scores we see that Customers with an overall score of 6 are our best customers (highly valued) whereas the Customers with an overall score of 0 are the worst customers (Lowest valued)

To further simplify things, I named these scores and formed 3 main segments as follows:

- Low Value Customers – Overall score 0 to 2
- Mid Value Customers – Overall score 2 to 4
- High Value Customers – Overall score 5 and above

```
tx_user['Segment'] = 'Low-Value'
tx_user.loc[tx_user['OverallScore']>2,'Segment'] = 'Mid-Value'
tx_user.loc[tx_user['OverallScore']>4,'Segment'] = 'High-Value'
```

The scatter plot on the right shows our customer segments based on Recency and Revenue:



The fig on the right shows us the distribution of customers throughout the 3 segments. We can see that there are about 700 High value customers which the company should take good care of, as these are the most recent, most frequent and the most revenue generating customers for the company. The way we can improve service to these customers is by knowing their personas so that the company can cater to similar audience in the future while targeting new customers as well as optimizing their experience by knowing more about them.

```
tx_user['Segment'].value_counts()
```

```
Mid-Value     1425
Low-Value     1296
High-Value     698
```

Conclusion/Insights

The dashboard below summarizes somcharacteristics of the high value customers of KPMG: