SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

PROJECT REPORT

(TECHNICAL INTERNSHIP PROGRAM)

Organization: D-Sys Data Solutions Pvt. Ltd.

INDUSTRY MENTOR:

Dr. Setu Kumar Chaturvedi

FACULTY MENTOR:

Mr. Piyush Kumar Soni

A REPORT ON

CUSTOMER SEGMENTATION IN RETAIL INDUSTRY.

Prepared By:

Koustubh Sharma

Roll No. – I229 (70411118033)

Course/Branch – MBA Tech IT
(2018-2023)

An interim report submitted in partial fulfilment of the requirement of 5-year integrated MBA (Tech) program of SVKM's NMIMS Mukesh Patel School of Technology & Engineering.

TABLE OF CONTENT

Abstract

Customer segmentation is a marketing activity that involves breaking down your customers into several groups or clusters. There are a variety of ways to dive up your customer base, but the end goal is the same: to better understand your customer.

Understanding diverse groups of customers lets an organization focus on the target audience and improve efforts to increase productivity and profits. An organization can use all customer demographics, such as age, gender, and marital status, as targeting criteria in marketing campaigns on search platforms and social media.

In this project, efforts were made to perform customer segmentation on a retail dataset to understand the various nuances associated with the retail industry's customer base.

Introduction

Customer segmentation is the process of dividing your customer base into groups based on common characteristics. The customer segmentation process helps you understand who your target audience is, refine your customer experience and reduce churn.

There are numerous benefits to dividing up your customer base into individual segments. Let us look at two of the major benefits in depth.

First, segmenting the customer base enables an organization to better communicate with its customers. This is especially true if the business services a wide range of customers as it enables the organization to provide a segment specific, customized experience. If a business only targets a single group of customers, segmentation is less effective.

Second, customer segmentation can aid an organization in identifying new business opportunities. By better understanding what existing customers are, an organization will be able to find new problems to solve — in other words, new products, and services to offer — while leveraging the existing customer audience.

Customer segmentation has many benefits, most of which stem from the ability to better understand of the audience.

There are various methods of customer segmentation that need to be weighed against each other in terms of applicability and real benefits. They can also be combined. Few of these methods are: Needs/Value Segmentation, RFM Segmentation, Clustering, Predictive Models etc.

Problem Statement

In the retail industry, customer segmentation plays an important role. There are a lot of datasets available of the European and the American, and due to such availability of data, there exist many solutions for customer segmentations available highlighting the various nuances of the retail industry.

If we take the Asian region into consideration, due to the unavailability of proper data, there is no availability of the customer segmentation solutions and those which are available, are confined to small regions and limited data.

Proposed Solution

Understanding the problem, we planned to create a customer segmentation model specifically for the Asian region. This model development has been supported by the data which was be collected using web scrapping.

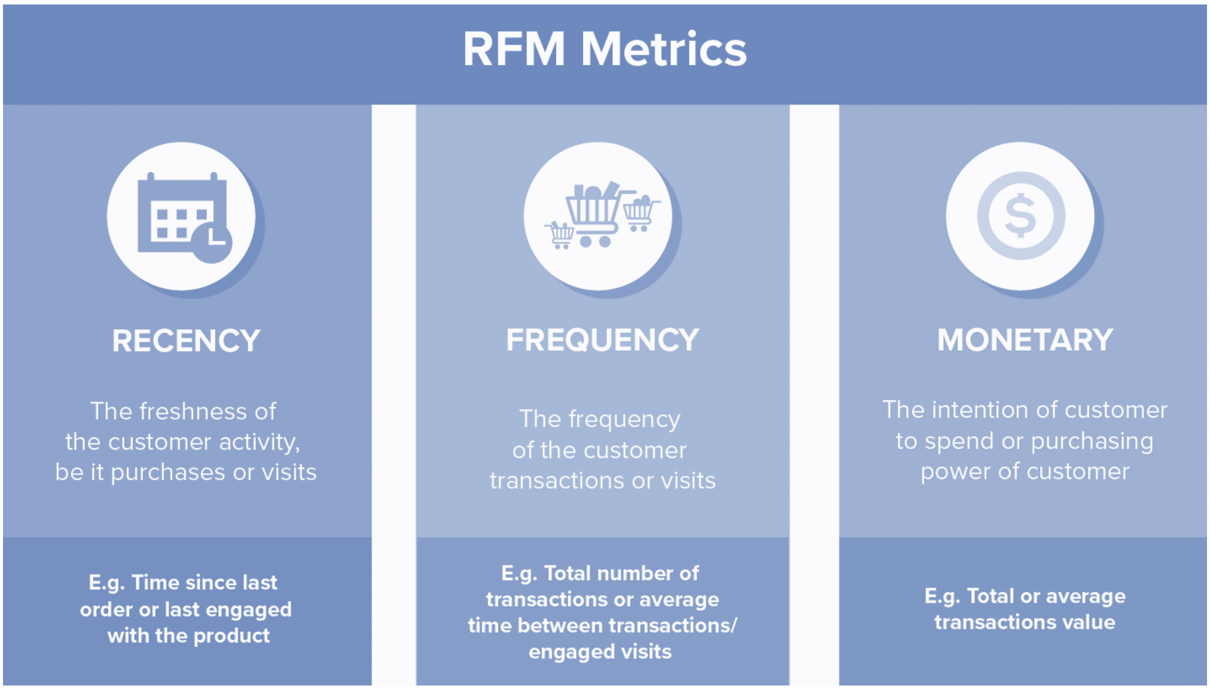
This model was be developed with RFM Segmentation method. This method, being the gold standard for segmentation, considers three core dimensions of a customer’s buying behaviour: Time since last purchase (Recency), purchase frequency (Frequency) and sales (Monetary Value).

For better understanding of the outcomes of the segmentation, front end visualization was performed using Tableau.

Research and Findings

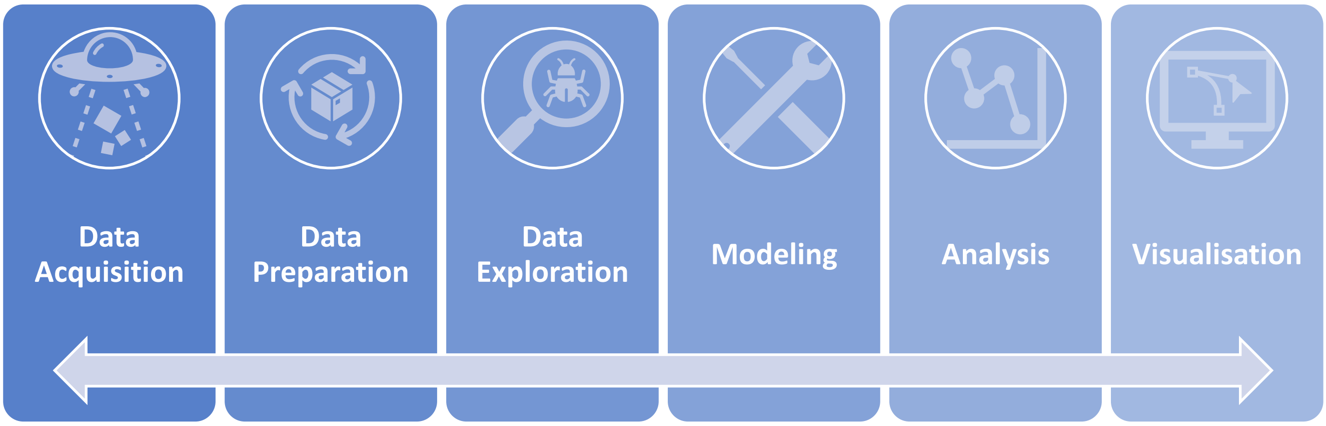
Paper	Method	Data	Advantage	Disadvantage
Magento (2014)	Magento	Demographic, Purchase History, Data Product, Data Media, Data Marketing, Server Log	Have clear variable customer segmentation	There is no data processing for each variable.
Baer (2012)	Business Rule	Demographic, Purchase history	Easy to apply, Use database query	Not focus on customer behaviour.
	Quantile membership	Purchase history	Can process small data, can be used with other data	Good result obtained when determining a good classification
	Supervised Clustering with decision tree	Demographic, Purchase history	Classify customers according to target	Use one variable to cluster
	Unsupervised Clustering	Purchase history	Use any number of customer attributes	Speed of computation depends on k values.
Colica (2011)	Customer Profiling	Demographic, Purchase history	use database query if data is small	Not focus on behaviour.
	Customer Likeness Clustering	Demographic, Purchase History, Data product	classify customers according to the target	Problem arises when there are different unit in record
	RFM Cell Classification Grouping	Purchase history	Efficient three - dimensional mapping	Good result obtained when determining a good classification
	Purchase Affinity Clustering	Purchase history, Data product	know the products most in demand	Specific to product segmentation.

After analysing and understanding these researches, it was concluded that for customer segmentation on the retail data, RFM method could be useful as RFM analysis allows you to segment customers by the frequency and value of purchases and identify those customers who spend the most money.



Customer Segmentation

- **Methodology**
  - Requirements:
    - Programming language: Python 3.7
    - IDE: Jupyter Notebook
    - Front end visualization: Tableau Desktop 2020.4
  - Development:



- **Development**

1. Data Acquisition
    - Data was scrapped from sales orders of 18 Asian countries, in 2 years. These countries are:
        - South Korea
        - Pakistan
        - Myanmar
        - Vietnam
        - India
        - Saudi Arabia
        - Philippines
        - Afghanistan
        - China
        - Bangladesh
        - Indonesia
        - Thailand
        - Iraq
        - Malaysia
        - Japan
        - Iran
        - Turkey
        - Uzbekistan
    - Features like country names, revenue, and customer id have been modified to maintain privacy.
    - Number of orders, dates and units have not been altered.
    - Data file format is csv file.

2. Data Preparation

- Reading the data

CODE

```
df1 = pd.read_csv('C:\\Users\\Asus\\Desktop\\customer_segmentation-main\\customer_segmentation-main\\data\\sales_asia.csv',
                  dtype={'week.year': str},
                  sep=';',
                  decimal=',')
```

- Data understanding and preprocessing

CODE

```
df1.head(),df1.tail()
```

OUTPUT

```
(  country       id week.year  revenue  units
 0      KR   702234   03.2019   808.08      1
 1      KR   702234   06.2019  1606.80      2
 2      KR  3618438   08.2019   803.40      1
 3      KR  3618438   09.2019   803.40      1
 4      KR  3618438   09.2019   803.40      1,
        country       id week.year  revenue  units
 235569      CN  2452476   27.2020  41160.0    200
 235570      CN  2452476   27.2020  50856.0    400
 235571      CN  2452476   27.2020  79920.0   1200
 235572      CN  4553904   27.2020   4788.0    100
 235573      CN  4553904   27.2020   4188.0    100)
```

CODE

```
df1.shape,df1.info()
```

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235574 entries, 0 to 235573
Data columns (total 5 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   country    235574 non-null  object
 1   id         235574 non-null  int64
 2   week.year  235574 non-null  object
 3   revenue    235574 non-null  float64
 4   units      235574 non-null  int64
dtypes: float64(1), int64(2), object(2)
memory usage: 9.0+ MB
```

CODE

Making 'week and 'year' columns from 'week.year'.

```
df1['week'] = df1['week.year'].astype(str).str.split('.').str[0]
df1['year'] = df1['week.year'].astype(str).str.split('.').str[1]
```

OUTPUT

	country	id	week.year	revenue	units	week	year
0	KR	702234	03.2019	808.08	1	03	2019
1	KR	702234	06.2019	1606.80	2	06	2019
2	KR	3618438	08.2019	803.40	1	08	2019
3	KR	3618438	09.2019	803.40	1	09	2019
4	KR	3618438	09.2019	803.40	1	09	2019

CODE

Making a column 'date' by converting year and week.

```
df1['date'] = pd.to_datetime(df1['year'].map(str) + df1['week'].map(str) + '-1', format='%Y%W-%w')
```

OUTPUT

	country	id	week.year	revenue	units	week	year	date
0	KR	702234	03.2019	808.08	1	03	2019	2019-01-21
1	KR	702234	06.2019	1606.80	2	06	2019	2019-02-11
2	KR	3618438	08.2019	803.40	1	08	2019	2019-02-25
3	KR	3618438	09.2019	803.40	1	09	2019	2019-03-04
4	KR	3618438	09.2019	803.40	1	09	2019	2019-03-04

CODE

Removing redundant information.

```
df2 = df1.drop(['week.year', 'week', 'year'], axis=1)
```

OUTPUT

	country	id	revenue	units	date
0	KR	702234	808.08	1	2019-01-21
1	KR	702234	1606.80	2	2019-02-11
2	KR	3618438	803.40	1	2019-02-25
3	KR	3618438	803.40	1	2019-03-04
4	KR	3618438	803.40	1	2019-03-04

CODE

Changing column names.

```
df2.rename({'revenue': 'monetary'}, axis="columns", inplace=True)
```

OUTPUT

	country	id	monetary	units	date
0	KR	702234	808.08	1	2019-01-21
1	KR	702234	1606.80	2	2019-02-11
2	KR	3618438	803.40	1	2019-02-25
3	KR	3618438	803.40	1	2019-03-04
4	KR	3618438	803.40	1	2019-03-04

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235574 entries, 0 to 235573
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   country   235574 non-null  object
 1   id        235574 non-null  int64
 2   monetary  235574 non-null  float64
 3   units     235574 non-null  int64
 4   date      235574 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 9.0+ MB
```

CODE

No. Of transactions: 2,35,574
Biggest transaction processed: 1,50,000 units
Biggest transaction returned: 1,50,000 units
Most expensive purchase: 24,10,000 lakh dollars.

```
df2.describe()
df2.isnull().sum()
```

OUTPUT

	id	monetary	units
count	2.355740e+05	2.355740e+05	235574.000000
mean	3.193118e+06	2.840211e+03	8.599642
std	7.371744e+06	2.247532e+04	602.939290
min	6.000180e+05	-1.061539e+05	-150000.000000
25%	2.214396e+06	3.994800e+02	1.000000
50%	3.140856e+06	1.150320e+03	1.000000
75%	3.892650e+06	2.216160e+03	2.000000
max	2.419308e+08	2.415857e+06	150000.000000

```
country     0
id          0
monetary    0
units       0
date        0
dtype: int64
```

CODE
Time period of the dataset `df2['date'].min(),df2['date'].max()`
OUTPUT
`(Timestamp('2019-01-07 00:00:00'), Timestamp('2020-11-30 00:00:00'))`
CODE
Replacing country codes with country names `clean_country(df2, "country")['country_clean'].unique()`
OUTPUT

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, layout=Layout(flex='2'), max=8.0), HTML(v
alue='')), la…

Country Cleaning Report:
        235574 values cleaned (100.0%)
Result contains 235574 (100.0%) values in the correct format and 0 null values (0.0%)

array(['South Korea', 'Pakistan', 'Myanmar', 'Vietnam', 'India',
       'Saudi Arabia', 'Philippines', 'Afghanistan', 'China',
       'Bangladesh', 'Indonesia', 'Thailand', 'Iraq', 'Malaysia', 'Japan',
       'Iran', 'Turkey', 'Uzbekistan'], dtype=object)
```

CODE

Making 'date' as index for plotting the time series graph

```
df2b = df2.set_index("date")
df2b.head()
```

OUTPUT

date	country	id	monetary	units
2019-01	KR	702234	808.08	1
2019-02	KR	702234	1606.80	2
2019-02	KR	3618438	803.40	1
2019-03	KR	3618438	803.40	1
2019-03	KR	3618438	803.40	1

CODE

Converting date to month.

```
plt.style.use('ggplot')
plt.title('Units sold per week')
plt.ylabel('units')
plt.xlabel('date');
df2b['units'].plot(figsize=(20,5), c='blue');
```

OUTPUT

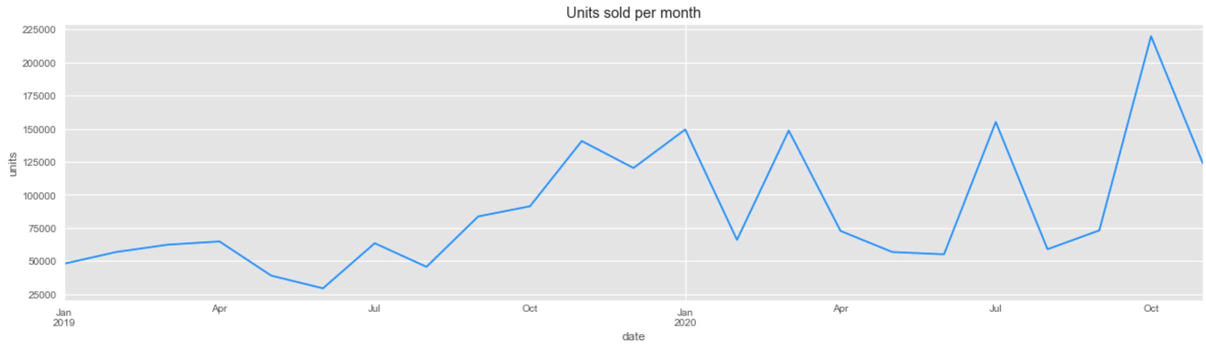
date	country	id	monetary	units
2019-01	KR	702234	808.08	1
2019-02	KR	702234	1606.80	2
2019-02	KR	3618438	803.40	1
2019-03	KR	3618438	803.40	1
2019-03	KR	3618438	803.40	1

CODE

Plotting the time series graph for units sold per month

```
plt.style.use('ggplot')
df2c['units'].groupby('date').agg(sum).plot(figsize=(20,5), c='dodgerblue')
plt.title('Units sold per month')
plt.ylabel('units')
plt.xlabel('date');
```

OUTPUT

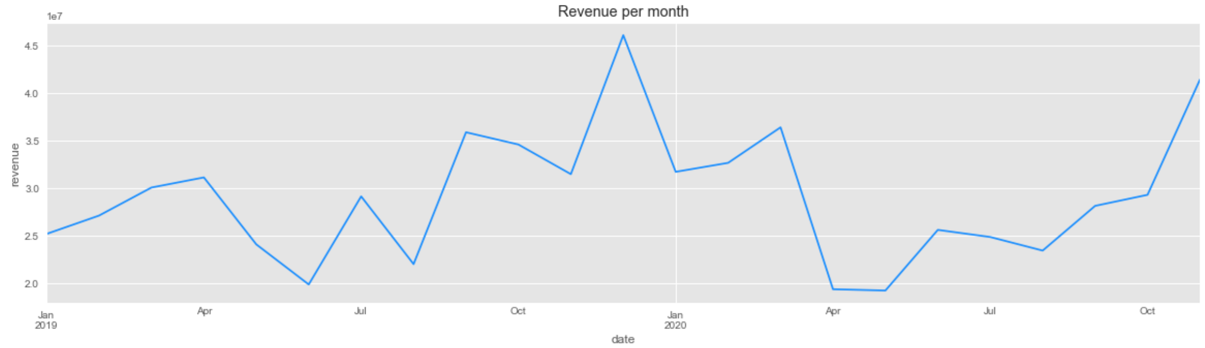


CODE

Plotting the time series graph for revenue sold per month

```
plt.style.use('ggplot')
df2c['monetary'].groupby('date').agg(sum).plot(figsize=(20,5), c='dodgerblue')
plt.title('Revenue per month')
plt.ylabel('revenue')
plt.xlabel('date');
```

OUTPUT



3. Data Exploration

CODE

Removing the data entries after all the entries of first 365 days and resetting the index.

```
period = 365
date_N_days_ago = df2['date'].max() - timedelta(days=period)
```

```
df2 = df2[df2['date']> date_N_days_ago]
```

```
df2.reset_index(drop=True, inplace=True)
```

OUTPUT

	country	id	monetary	units	date
0	KR	4375152	773.58	1	2019-12-16
1	KR	705462	337.26	1	2019-12-09
2	KR	705462	337.26	1	2019-12-23
3	KR	705462	421.56	2	2019-12-16
4	KR	706854	391.50	1	2019-12-09

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124640 entries, 0 to 124639
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   country   124640 non-null  object
 1   id        124640 non-null  int64
 2   monetary  124640 non-null  float64
 3   units     124640 non-null  int64
 4   date      124640 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 4.8+ MB
```

CODE

Combining country codes with customer identity to remove data redundancy.

```
df3 = df2.copy()

df3['id+'] = df3['country'].map(str) + df3['id'].map(str)
```

OUTPUT

	country	id	monetary	units	date	id+
0	KR	4375152	773.58	1	2019-12-16	KR4375152
1	KR	705462	337.26	1	2019-12-09	KR705462
2	KR	705462	337.26	1	2019-12-23	KR705462
3	KR	705462	421.56	2	2019-12-16	KR705462
4	KR	706854	391.50	1	2019-12-09	KR706854

CODE

Appending a new column to store duration from the last purchase and a day after last purchase

```
CURRENT = df3['date'].max() + timedelta(days=1)

df3['days_since_purchase'] = df3['date'].apply(lambda x:(CURRENT - x).days)
```

OUTPUT

	country	id	monetary	units	date	id+	days_since_purchase
0	KR	4375152	773.58	1	2019-12-16	KR4375152	351
1	KR	705462	337.26	1	2019-12-09	KR705462	358
2	KR	705462	337.26	1	2019-12-23	KR705462	344
3	KR	705462	421.56	2	2019-12-16	KR705462	351
4	KR	706854	391.50	1	2019-12-09	KR706854	358

CODE

```
df3[df3['id+']=='KR706854']
```

OUTPUT

	country	id	monetary	units	date	id+	days_since_purchase
4	KR	706854	391.50	1	2019-12-09	KR706854	358
5	KR	706854	388.68	1	2019-12-30	KR706854	337
14169	KR	706854	369.66	1	2020-04-06	KR706854	239
14192	KR	706854	374.76	1	2020-07-27	KR706854	127
14210	KR	706854	371.82	1	2020-11-09	KR706854	22

CODE

For RFM, minimum of 'days_since_purchase' will be recency and total of orders in the period of will be Frequency.

```
aggr = {
    'days_since_purchase': lambda x:x.min(),
    'date': lambda x: len([d for d in x if d >= CURRENT - timedelta(days=period)])
}
```

```
rfm = df3.groupby(['id', 'id+', 'country']).agg(aggr).reset_index()
rfm.rename(columns={'days_since_purchase': 'recency',
                   'date': 'frequency'},
          inplace=True)
```

OUTPUT

	id	id+	country	recency	frequency
0	600018	CN600018	CN	29	7
1	600060	CN600060	CN	155	1
2	600462	CN600462	CN	211	2
3	600888	CN600888	CN	8	3
4	601014	CN601014	CN	225	1
...	...	...	...	...	...
16564	241575552	IQ241575552	IQ	15	1
16565	241794972	IQ241794972	IQ	351	1
16566	241888554	IQ241888554	IQ	43	1
16567	241900254	IQ241900254	IQ	8	62
16568	241930824	IQ241930824	IQ	36	2

CODE

Revenue of every customer from past 365 days

```
df3[df3['date'] >= CURRENT - timedelta(days=period)]\
    .groupby('id+')['monetary'].sum()
```

OUTPUT

```
id+
AF186035892       277.86
AF1915092      250651.86
AF1915920        2238.60
AF1916280         612.78
AF1917144       29793.18
                  ...
VN991620         1093.86
VN993528         1018.86
VN993996         4037.28
VN995010          544.32
VN998130          384.84
Name: monetary, Length: 16569, dtype: float64
```

CODE

Appending rfm dataframe with revenue from df3 dataframe and dropping 'id+'

```
rfm['monetary'] = rfm['id+']\
    .apply(lambda x: df3[ (df3['id+'] == x) & (df3['date'] >= NOW - timedelta(days=period))]\
    .groupby(['id', 'country']).sum().iloc[0,0])
rfm.drop(['id+'], axis=1, inplace=True)
```

OUTPUT

	id	country	recency	frequency	monetary
0	600018	CN	29	7	21402.78
1	600060	CN	155	1	1201.14
2	600462	CN	211	2	2033.64
3	600888	CN	8	3	2335.80
4	601014	CN	225	1	230.52

4. Modelling

CODE

The parameters recency, monetary and frequency will be assigned a rating from 1 to 5. Also, we will divide every feature in groups that will hold 20 % of the whole data, using quintiles method.

```python
quintiles = rfm[['recency', 'frequency', 'monetary']].quantile([.2, .4, .6, .8]).to_dict()
quintiles
```

OUTPUT

```
{'recency': {0.2: 15.0, 0.4: 50.0, 0.6: 120.0, 0.8: 239.0},
 'frequency': {0.2: 1.0, 0.4: 2.0, 0.6: 4.0, 0.8: 9.0},
 'monetary': {0.2: 967.5,
  0.4: 2212.2,
  0.6: 4852.548000000001,
  0.8: 13957.500000000005}}
```

CODE

Assignment of rating to each of the customers.

```python
def r_score(x):
    if x <= quintiles['recency'][.2]:
        return 5
    elif x <= quintiles['recency'][.4]:
        return 4
    elif x <= quintiles['recency'][.6]:
        return 3
    elif x <= quintiles['recency'][.8]:
        return 2
    else:
        return 1
```

```python
def fm_score(x, c):
    if x <= quintiles[c][.2]:
        return 1
    elif x <= quintiles[c][.4]:
        return 2
    elif x <= quintiles[c][.6]:
        return 3
    elif x <= quintiles[c][.8]:
        return 4
    else:
        return 5
```

```python
rfm['r'] = rfm['recency'].apply(lambda x: r_score(x))
rfm['f'] = rfm['frequency'].apply(lambda x: fm_score(x, 'frequency'))
rfm['m'] = rfm['monetary'].apply(lambda x: fm_score(x, 'monetary'))
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m
0	600018	CN	29	7	21402.78	4	4	5
1	600060	CN	155	1	1201.14	2	1	2
2	600462	CN	211	2	2033.64	2	2	2
3	600888	CN	8	3	2335.80	5	3	3
4	601014	CN	225	1	230.52	2	1	1

CODE

Creating 'rfm_score' score by merging 'r', 'f' and 'm'.

```
rfm['rfm_score'] = rfm['r'].map(str) + rfm['f'].map(str) + rfm['m'].map(str)
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score
0	600018	CN	29	7	21402.78	4	4	5	445
1	600060	CN	155	1	1201.14	2	1	2	212
2	600462	CN	211	2	2033.64	2	2	2	222
3	600888	CN	8	3	2335.80	5	3	3	533
4	601014	CN	225	1	230.52	2	1	1	211

CODE

Currently, we have 125 different segmentations of customers which will make the analysis very complex. By combining 'f' and 'm' score (fm = (f + m) / 2), we would get 11 segmentations.

```
def truncate(x):
    return math.trunc(x)
```

```
rfm['fm'] = ((rfm['f'] + rfm['m'])/2).apply(lambda x: truncate(x))
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm
0	600018	CN	29	7	21402.78	4	4	5	445	4
1	600060	CN	155	1	1201.14	2	1	2	212	1
2	600462	CN	211	2	2033.64	2	2	2	222	2
3	600888	CN	8	3	2335.80	5	3	3	533	3
4	601014	CN	225	1	230.52	2	1	1	211	1

- Segment Description
  - **Champions:** Bought recently, buy often, and spend the most
  - **Loyal Customers:** Buy on a regular basis. Responsive to promotions.
  - **Potential Loyalists:** Recent customers with average frequency.
  - **Recent Customers:** Bought most recently, but not often.
  - **Promising:** Recent shoppers but have not spent much.

- **Customers Needing Attention:** Above average recency, frequency, and monetary values. May not have bought very recently though.
- **About to Sleep:** Below average recency and frequency. Will lose them if not reactivated.
- **At Risk:** Purchased often but a long time ago. Need to bring them back!
- **Cannot Lose Them:** Used to purchase frequently but haven't returned for a long time.
- **Hibernating:** Last purchase was long back and low number of orders.
- **Lost:** Purchased long time ago and never came back.

CODE
Creation of segment map of 11 segments with 'r' and 'fm'

```
segment_map = {
    r'22': 'hibernating',
    r'[1-2][1-2]': 'lost',
    r'15': 'can\'t lose',
    r'[1-2][3-5]': 'at risk',
    r'3[1-2]': 'about to sleep',
    r'33': 'need attention',
    r'55': 'champions',
    r'[3-5][4-5]': 'loyal customers',
    r'41': 'promising',
    r'51': 'new customers',
    r'[4-5][2-3]': 'potential loyalists'
}

rfm['segment'] = rfm['r'].map(str) + rfm['fm'].map(str)
rfm['segment'] = rfm['segment'].replace(segment_map, regex=True)
rfm.head()
rfm.isnull().sum()
```

OUTPUT

```
id           0
country      0
recency      0
frequency    0
monetary     0
r            0
f            0
m            0
rfm_score    0
fm           0
segment      0
dtype: int64
```

5. Analysis

CODE

```
rfm['segment'].unique()
```

OUTPUT

```
array(['loyal customers', 'lost', 'hibernating', 'potential loyalists',
       'new customers', 'need attention', 'at risk', 'champions',
       'about to sleep', 'promising', "can't lose"], dtype=object)
```

CODE

```
rfm[rfm['segment']=="can't lose"].sort_values(by='monetary', ascending=False)
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
13028	4096386	JP	260	105	220267.86	1	5	5	155	5	can't lose
3502	2443284	IN	246	10	102208.02	1	5	5	155	5	can't lose
14174	4262646	IN	316	10	91909.44	1	5	5	155	5	can't lose
2435	1803672	IN	267	12	70506.96	1	5	5	155	5	can't lose
13254	4132968	VN	253	26	42535.14	1	5	5	155	5	can't lose
11222	3815274	IN	267	11	37968.72	1	5	5	155	5	can't lose
1458	1031454	PH	267	23	31833.30	1	5	5	155	5	can't lose
5437	2809158	IN	274	12	27150.12	1	5	5	155	5	can't lose
14644	4326906	IN	337	11	22351.68	1	5	5	155	5	can't lose
259	668070	MM	267	11	21886.92	1	5	5	155	5	can't lose
15331	4418268	SA	302	10	14295.54	1	5	5	155	5	can't lose

CODE

```
rfm[rfm['segment']=="need attention"].sort_values(by='monetary', ascending=False).head(10)
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
8245	3242664	TR	64	1	73823.58	3	1	5	315	3	need attention
13065	4107798	JP	120	2	67257.48	3	2	5	325	3	need attention
9847	3561900	ID	120	1	59700.00	3	1	5	315	3	need attention
6626	2921070	ID	71	2	34730.22	3	2	5	325	3	need attention
10009	3587772	CN	92	1	29961.00	3	1	5	315	3	need attention
3087	2131194	JP	57	1	28543.74	3	1	5	315	3	need attention
13463	4160490	JP	99	1	24842.22	3	1	5	315	3	need attention
1251	993414	KR	71	2	22018.32	3	2	5	325	3	need attention
3936	2544588	BD	71	2	19043.82	3	2	5	325	3	need attention
3616	2468010	TH	85	2	18599.58	3	2	5	325	3	need attention

CODE

```
rfm[rfm['segment']=='loyal customers'].sort_values(by='monetary', ascending=False).head(10)
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
15420	4422780	TR	92	13	2315341.14	3	5	5	355	5	loyal customers
2882	2030526	JP	22	50	1519339.86	4	5	5	455	5	loyal customers
3220	2182446	JP	29	18	1492057.68	4	5	5	455	5	loyal customers
12660	4041366	PK	50	9	736626.96	4	4	5	445	4	loyal customers
5612	2853774	VN	8	6	712230.00	5	4	5	545	4	loyal customers
10343	3649728	PH	29	81	579167.52	4	5	5	455	5	loyal customers
8284	3248568	TR	64	3	573792.72	3	3	5	335	4	loyal customers
15450	4427148	IN	29	14	502843.32	4	5	5	455	5	loyal customers
14678	4332210	ID	43	21	474773.40	4	5	5	455	5	loyal customers
2802	1985592	IQ	78	4	460390.86	3	3	5	335	4	loyal customers

CODE

```
rfm[rfm['segment']=='champions'].sort_values(by='monetary', ascending=False).head(10)
```

OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
173	638544	CN	1	217	21482332.56	5	5	5	555	5	champions
15436	4424580	CN	1	104	16912322.46	5	5	5	555	5	champions
14754	4341960	TR	1	200	16550997.90	5	5	5	555	5	champions
11942	3929094	ID	1	470	8748884.64	5	5	5	555	5	champions
9626	3520734	JP	1	198	6207519.96	5	5	5	555	5	champions
15915	4494150	TR	1	57	4874668.14	5	5	5	555	5	champions
10168	3618438	KR	8	1020	4615660.08	5	5	5	555	5	champions
14027	4245048	PH	1	993	4358515.98	5	5	5	555	5	champions
3050	2111100	IN	1	876	4270717.80	5	5	5	555	5	champions
11742	3894492	PH	8	63	4106366.22	5	5	5	555	5	champions

CODE

```
rfm['monetary'].mean()
```

OUTPUT

```
21629.6111497373
```

CODE

Customer whose monetary value lies above mean

```
rfm[(rfm['monetary']>rfm['monetary'].mean()) & (rfm['segment']=='need attention')]\
    .sort_values(by='monetary', ascending=False)
```

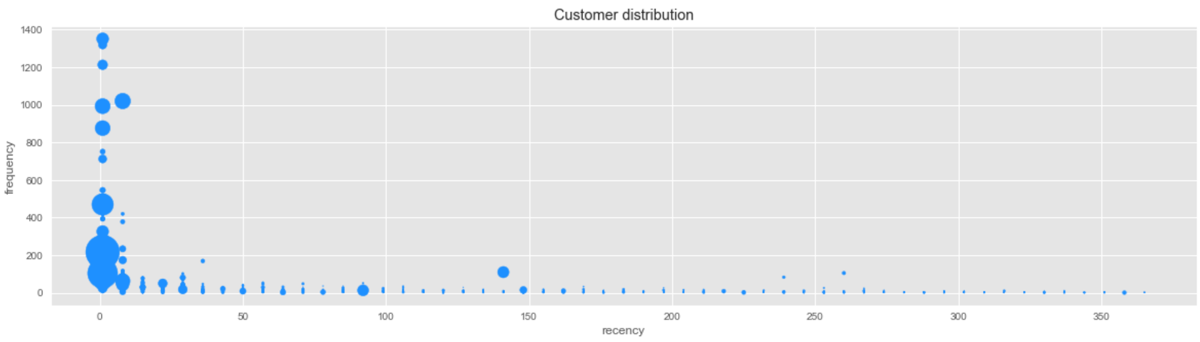
OUTPUT

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
8245	3242664	TR	64	1	73823.58	3	1	5	315	3	need attention
13065	4107798	JP	120	2	67257.48	3	2	5	325	3	need attention
9847	3561900	ID	120	1	59700.00	3	1	5	315	3	need attention
6626	2921070	ID	71	2	34730.22	3	2	5	325	3	need attention
10009	3587772	CN	92	1	29961.00	3	1	5	315	3	need attention
3087	2131194	JP	57	1	28543.74	3	1	5	315	3	need attention
13463	4160490	JP	99	1	24842.22	3	1	5	315	3	need attention
1251	993414	KR	71	2	22018.32	3	2	5	325	3	need attention

CODE
Using scatter plot to understand distribution of customers

```
plt.style.use('ggplot')
rfm.plot.scatter(x='recency', y='frequency', s=rfm['monetary']*5e-5, figsize=(20,5), c='dodgerblue')
plt.gca().set(xlabel='recency', ylabel='frequency', title='Customer distribution');
```

OUTPUT



CODE
Exporting dataframe for analysis and front-end visualization on Tableau.

```
rfm.to_csv('rfm_asia.csv', encoding='utf-8', index=False, float_format='%.2f')
```

6. Front End Visualisation

Front end visualisation has been performed on Tableau. Tableau is a visual analytics platform transforming the way we use data to solve problems—empowering people and organizations to make the most of their data.

As the market-leading choice for modern business intelligence, Tableau's analytics platform makes it easier for people to explore and manage data, and faster to discover and share insights that can change businesses and the world.

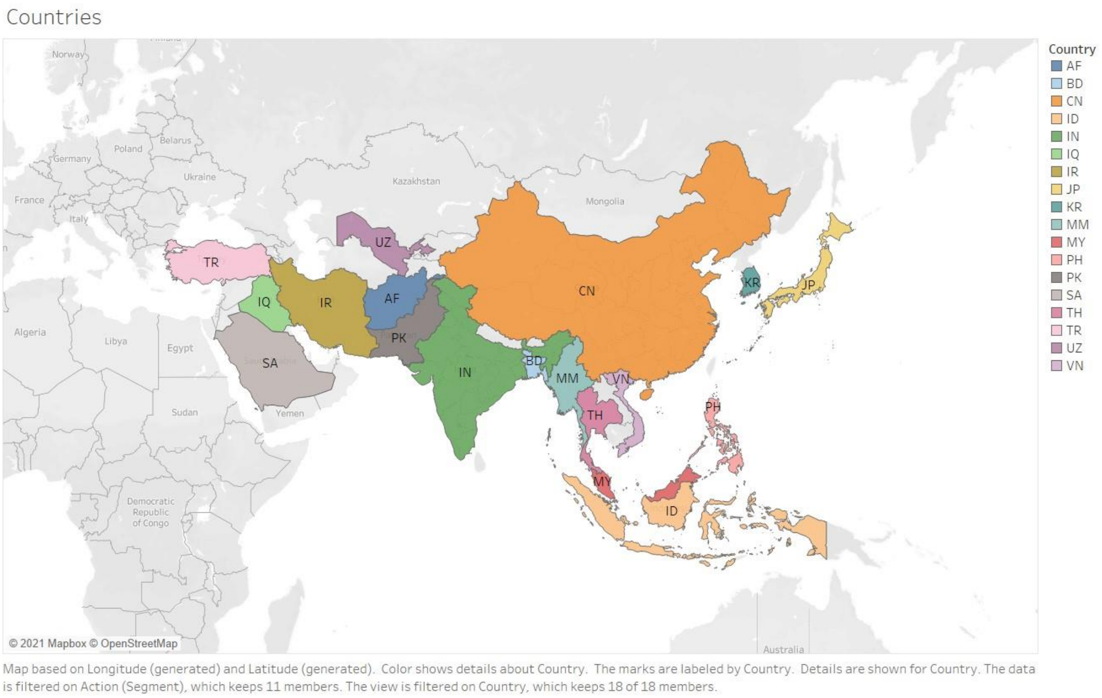
Tableau has more than one million active, diverse, and engaged members inspire and support one another through community forums, 500+ worldwide user groups, and unique events like the annual Tableau Conference.

Below we may observe the various visualisations, which were be used for the development of dashboards. These dashboards thus were compiled for creating a story. These visualisations have been published on Tableau Public server.

DISCRIPTION

The graph below shows various Asian countries present in the dataset

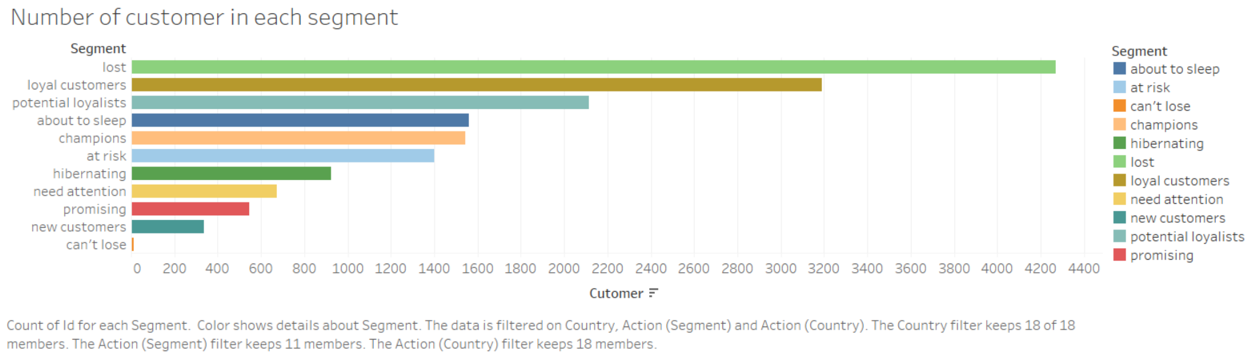
OUTPUT



DISCRIPTION

The graph below shows the number of customer present in each segment.

- *Lost: 4269 Customers*
- *Loyal Customers: 3189 Customers*
- *Potential Loyalists: 2115 Customers*
- *About to sleep: 1562 Customers*
- *Champions: 1543 Customers*
- *At risk: 1398 Customers*
- *Hibernating: 923 Customers*
- *Need Attention: 674 Customers*
- *Promising: 547 Customers*
- *New Customers: 338 Customers*
- *Can't Lose: 11 Customers*

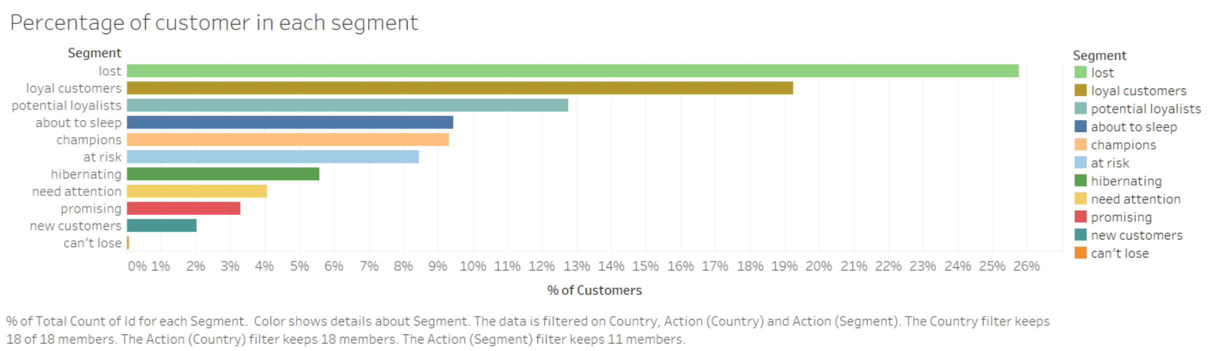
OUTPUT


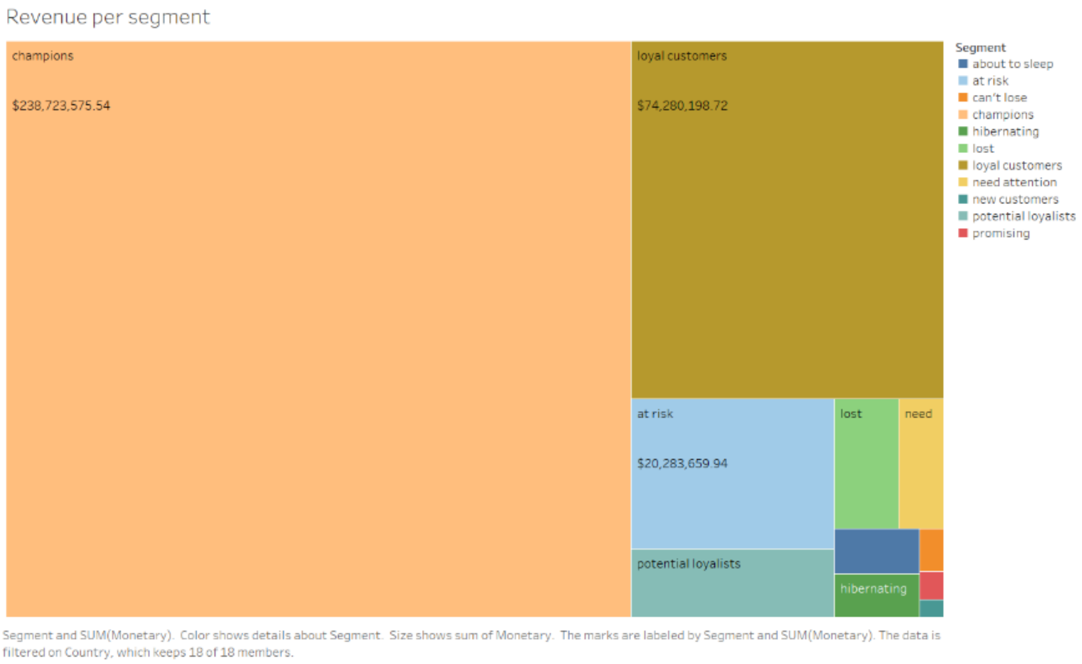
DISCRIPTION

The graph below shows the number of customer present in each segment.

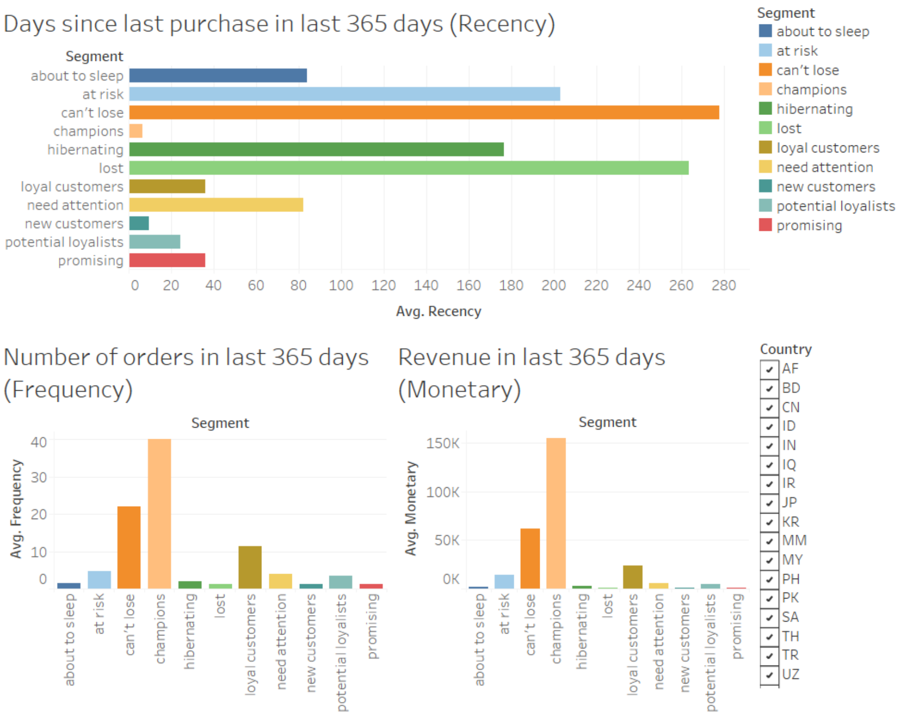
- *Lost: 25.76 % Customers*
- *Loyal Customers: 19.25 % Customers*
- *Potential Loyalists: 12.76 % Customers*
- *About to sleep: 9.43 % Customers*
- *Champions: 9.31 % Customers*
- *At risk: 8.44 % Customers*
- *Hibernating: 5.57 % Customers*
- *Need Attention: 4.07 % Customers*
- *Promising: 3.03 % Customers*
- *New Customers: 2.04 % Customers*
- *Can't Lose: 0.07 % Customers*

OUTPUT



DISCRIPTION
The graph below represents the revenue contributed by the customers of each segment. • *Lost: $ 5.5 Million* • *Loyal Customers: $ 74.2 Million* • *Potential Loyalists: $ 9.1 Million* • *About to sleep: $ 2.5 Million* • *Champions: $ 238.7 Million* • *At risk: $ 20.2 Million* • *Hibernating: $ 2.4 Million* • *Need Attention: $ 3.9 Million* • *Promising: $ 0.4 Million* • *New Customers: $ 0.2 Million* • *Can't Lose: $ 0.6 Million*
OUTPUT

DISCRIPTION
The graphs below give visual representation of the Recency, Frequency and Monetary data of each segment. • *Segment "can't lose" has the highest average Recency of 277.8.* • *Segment "champions" has the highest average Frequency of 40.05.* • *Segment "champions" has the highest average Monetary of $154.714.*

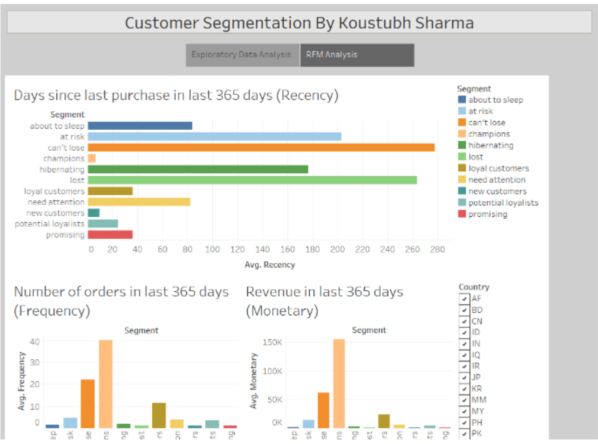
OUTPUT

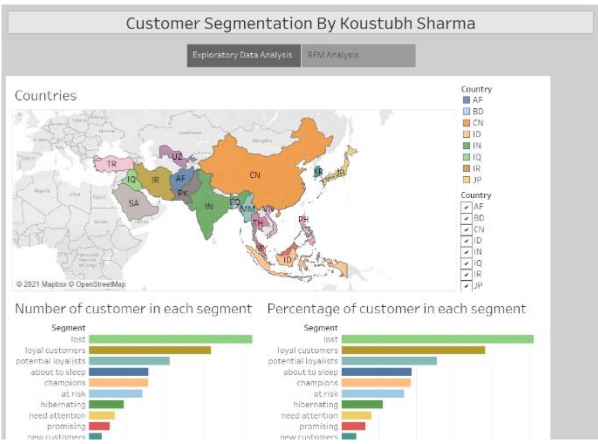


DISCRIPTION

Below are the dashboards created by the compilation of the various visuals discussed above. Each visualisation has been modified to provide the ability to users to view the data visualisation of each country specifically and of all the countries in general.

OUTPUT





Conclusion

Customer Segmentation is a better choice to know the customers better. It is not only bifurcating customers into good and bad but in the different criteria and segments via which we can pitch a message better. With customer segmentation, we can break down our customers into smaller, more detailed groups depending on their needs — and target them even more specifically.

The technical internship program has a very vital role to perform. It helps bridge the gap between the theoretical knowledge we gain in college and what is helpful in the industry.

While working as an intern as D-Sys Data Solutions, I understood the importance of data and analytical skills, performed the required study and research for the implementation of the project, collected and pre-processed the data, created the customer segmentations by the RFM analysis methodology and at last, front-end visualisation was created using Tableau.

My Industry mentor, Dr. Setu Kumar Chaturvedi, and faculty mentor, Mr. Piyush Kumar Soni, were a constant support throughout this internship program. As the technical internship program was the first opportunity to interact with the corporate environment and the practical utilisation of the knowledge, both the mentors provided me the right guidance and direction, and a chance to get maximum benefit from this opportunity while working as a student intern.

Future

The project developed had a lot of scope for further development. As the project has been created to provide a generalised model for customer segmentation in the retail industry, it could be modified and utilised in other industry. Also, with more data and parameters, its efficiency and scope could be increased.

Technologies Used in The Project

- **Jupyter Notebook**

  Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text, and multimedia resources in a single document

  For this project's development, Jupyter notebook was used for the development in the Python programming language.

- **Tableau**

  Tableau is a Business Intelligence tool for visually analysing the data. Users can create and distribute an interactive and shareable dashboard, which depict the trends, variations, and density of the data in the form of graphs and charts. Tableau can connect to files, relational and Big Data sources to acquire and process data.

  Tableau was used for the front-end visualisation of the data and the outcomes.

- **MS Excel**

  Microsoft Excel is a spreadsheet developed by Microsoft for Windows, macOS, Android and iOS. It features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications. It was used for basic data analysis and processing

References

- Magento. An Introduction to Customer Segmentation. 2014. info2.magento.com/.../
- Baer D. CSI: Customer Segmentation Intelligence for Increasing Profits. SAS Glob Forum. 2012:1-13. http://support.sas.com/resources/papers/proceedings12/103- 2012.pdf
- Colica R. Customer Segmentation and Clustering Using SAS Enterprise Minner Part I The Basics. 2011:1-14.
- What is Customer Segmentation: Quick Analysis and Models (pestleanalysis.com)
- Customer Segmentation: What, Why, and How | CrossEngage
- What Is Tableau? | Tableau
- RFM Analysis: A Complete Guide. Let's take a look at RFM analysis… | by Maryna Sharapa | GoBeyond.AI: E-commerce Magazine | Medium