Descriptive Analysis For Retail Data

In [1]:
```
#DESCRIPTION

#Problem Statement

#It is a critical requirement for business to understand the value deri
ved from a customer. RFM is a method used for analyzing customer value.
#Customer segmentation is the practice of segregating the customer base
into groups of individuals based on some common characteristics such as
age, gender, interests, and spending habits
#Perform customer segmentation using RFM analysis. The resulting segmen
ts can be ordered from most valuable (highest recency, frequency, and v
alue) to least valuable (lowest recency, frequency, and value).
#Dataset Description

#This is a transnational data set which contains all the transactions t
hat occurred between 01/12/2010 and 09/12/2011 for a UK-based and regis
tered non-store online retail. The company mainly sells unique and all-
occasion gifts.

#Variables      Description
#InvoiceNo      Invoice number. Nominal, a six digit integral number un
iquely assigned to each transaction. If this code starts with letter
 'c', it indicates a cancellation
#StockCode      Product (item) code. Nominal, a five digit integral num
ber uniquely assigned to each distinct product
#Description    Product (item) name. Nominal
#Quantity       The quantities of each product (item) per transaction.
 Numeric
#InvoiceDate    Invoice Date and time. Numeric, the day and time when e
ach transaction was generated
#UnitPrice      Unit price. Numeric, product price per unit in sterling
#CustomerID     Customer number. Nominal, a six digit integral number u
```

```
niquely assigned to each customer
#Country        Country name. Nominal, the name of the country where ea
ch customer resides
#Project Task: Week 1

#Data Cleaning:

#1. Perform a preliminary data inspection and data cleaning.

#a. Check for missing data and formulate an apt strategy to treat them.

#b. Remove duplicate data records.

#c. Perform descriptive analytics on the given data.

#Data Transformation:

#2. Perform cohort analysis (a cohort is a group of subjects that share
a defining characteristic). Observe how a cohort behaves across time an
d compare it to other cohorts.

#a. Create month cohorts and analyze active customers for each cohort.

#b. Analyze the retention rate of customers.

#Project Task: Week 2

#Data Modeling :

#1. Build a RFM (Recency Frequency Monetary) model. Recency means the n
umber of days since a customer made the last purchase. Frequency is the
number of purchase in a given period. It could be 3 months, 6 months or
1 year. Monetary is the total amount of money a customer spent in that
 given period. Therefore, big spenders will be differentiated among oth
er customers such as MVP (Minimum Viable Product) or VIP.

#2. Calculate RFM metrics.

#3. Build RFM Segments. Give recency, frequency, and monetary scores in
dividually by dividing them into quartiles.
```

```
#b1. Combine three ratings to get a RFM segment (as strings).

#b2. Get the RFM score by adding up the three ratings.

#b3. Analyze the RFM segments by summarizing them and comment on the fi
ndings.

#Note: Rate “recency" for customer who has been active more recently hi
gher than the less recent customer, because each company wants its cust
omers to be recent.

#Note: Rate “frequency" and “monetary" higher, because the company want
s the customer to visit more often and spend more money

#Project Task: Week 3

#Data Modeling :

#1. Create clusters using k-means clustering algorithm.

#a. Prepare the data for the algorithm. If the data is asymmetrically d
istributed, manage the skewness with appropriate transformation. Standa
rdize the data.

#b. Decide the optimum number of clusters to be formed.

#c. Analyze these clusters and comment on the results.

#Project Task: Week 4

#Data Reporting:

#1. Create a dashboard in tableau by choosing appropriate chart types a
nd metrics useful for the business. The dashboard must entail the follo
wing:

#a. Country-wise analysis to demonstrate average spend. Use a bar chart
to show the monthly figures
```

```
#b. Bar graph of top 15 products which are mostly ordered by the users
 to show the number of products sold

#c. Bar graph to show the count of orders vs. hours throughout the day

#d. Plot the distribution of RFM values using histogram and frequency c
harts

#e. Plot error (cost) vs. number of clusters selected

#f. Visualize to compare the RFM values of the clusters using heatmap
```

Data Cleaning(Week 1)

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
df=pd.read_csv('Online-Retail.csv')
df.head()
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

In [5]:
```
pip install pandas-profiling
```

```
Requirement already satisfied: pandas-profiling in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (2.10.0)
Requirement already satisfied: visions[type_image_path]==0.6.0 in /Volu
```

```
mes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from pan
das-profiling) (0.6.0)
Requirement already satisfied: phik>=0.10.0 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from pandas-profiling) (0.1
0.0)
Requirement already satisfied: confuse>=1.0.0 in /Volumes/Samsung_T5/An
aconda/anaconda3/lib/python3.7/site-packages (from pandas-profiling)
(1.4.0)
Requirement already satisfied: tangled-up-in-unicode>=0.0.6 in /Volume
s/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from panda
s-profiling) (0.0.6)
Requirement already satisfied: numpy>=1.16.0 in /Volumes/Samsung_T5/Ana
conda/anaconda3/lib/python3.7/site-packages (from pandas-profiling) (1.
18.1)
Requirement already satisfied: jinja2>=2.11.1 in /Volumes/Samsung_T5/An
aconda/anaconda3/lib/python3.7/site-packages (from pandas-profiling)
(2.11.1)
Requirement already satisfied: seaborn>=0.10.1 in /Volumes/Samsung_T5/A
naconda/anaconda3/lib/python3.7/site-packages (from pandas-profiling)
(0.11.1)
Requirement already satisfied: scipy>=1.4.1 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from pandas-profiling) (1.
4.1)
Requirement already satisfied: tqdm>=4.48.2 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from pandas-profiling) (4.5
5.2)
Requirement already satisfied: ipywidgets>=7.5.1 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from pandas-profilin
g) (7.5.1)
Requirement already satisfied: matplotlib>=3.2.0 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from pandas-profilin
g) (3.3.3)
Requirement already satisfied: missingno>=0.4.2 in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from pandas-profiling)
(0.4.2)
Requirement already satisfied: pandas!=1.0.0,!=1.0.1,!=1.0.2,!=1.1.0,>=
0.25.3 in /Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-pac
kages (from pandas-profiling) (1.2.0)
Requirement already satisfied: requests>=2.24.0 in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from pandas-profiling)
```

```
(2.25.1)
Requirement already satisfied: htmlmin>=0.1.12 in /Volumes/Samsung_T5/A
naconda/anaconda3/lib/python3.7/site-packages (from pandas-profiling)
(0.1.12)
Requirement already satisfied: attrs>=19.3.0 in /Volumes/Samsung_T5/Ana
conda/anaconda3/lib/python3.7/site-packages (from pandas-profiling) (1
9.3.0)
Requirement already satisfied: joblib in /Volumes/Samsung_T5/Anaconda/a
naconda3/lib/python3.7/site-packages (from pandas-profiling) (0.14.1)
Requirement already satisfied: networkx>=2.4 in /Volumes/Samsung_T5/Ana
conda/anaconda3/lib/python3.7/site-packages (from visions[type_image_pa
th]==0.6.0->pandas-profiling) (2.4)
Requirement already satisfied: Pillow; extra == "type_image_path" in /V
olumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from
visions[type_image_path]==0.6.0->pandas-profiling) (7.0.0)
Requirement already satisfied: imagehash; extra == "type_image_path" in
/Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (fro
m visions[type_image_path]==0.6.0->pandas-profiling) (4.2.0)
Requirement already satisfied: numba>=0.38.1 in /Volumes/Samsung_T5/Ana
conda/anaconda3/lib/python3.7/site-packages (from phik>=0.10.0->pandas-
profiling) (0.48.0)
Requirement already satisfied: pyyaml in /Volumes/Samsung_T5/Anaconda/a
naconda3/lib/python3.7/site-packages (from confuse>=1.0.0->pandas-profi
ling) (5.3)
Requirement already satisfied: MarkupSafe>=0.23 in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from jinja2>=2.11.1->pa
ndas-profiling) (1.1.1)
Requirement already satisfied: widgetsnbextension~=3.5.0 in /Volumes/Sa
msung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from ipywidget
s>=7.5.1->pandas-profiling) (3.5.1)
Requirement already satisfied: nbformat>=4.2.0 in /Volumes/Samsung_T5/A
naconda/anaconda3/lib/python3.7/site-packages (from ipywidgets>=7.5.1->
pandas-profiling) (5.0.4)
Requirement already satisfied: ipykernel>=4.5.1 in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from ipywidgets>=7.5.1-
>pandas-profiling) (5.1.4)
Requirement already satisfied: ipython>=4.0.0; python_version >= "3.3"
in /Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages
(from ipywidgets>=7.5.1->pandas-profiling) (7.12.0)
Requirement already satisfied: traitlets>=4.3.1 in /Volumes/Samsung_T5/
```

```
Anaconda/anaconda3/lib/python3.7/site-packages (from ipywidgets>=7.5.1-
>pandas-profiling) (4.3.3)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3
in /Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages
(from matplotlib>=3.2.0->pandas-profiling) (2.4.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from matplotlib>=3.2.
0->pandas-profiling) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from matplotlib>=3.2.0->pan
das-profiling) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /Volumes/Samsung
_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from matplotlib>=3.
2.0->pandas-profiling) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from pandas!=1.0.0,!=1.0.
1,!=1.0.2,!=1.1.0,>=0.25.3->pandas-profiling) (2019.3)
Requirement already satisfied: certifi>=2017.4.17 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from requests>=2.24.0
->pandas-profiling) (2019.11.28)
Requirement already satisfied: idna<3,>=2.5 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from requests>=2.24.0->pand
as-profiling) (2.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Volumes/Samsun
g_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from requests>=2.2
4.0->pandas-profiling) (1.25.8)
Requirement already satisfied: chardet<5,>=3.0.2 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from requests>=2.24.0
->pandas-profiling) (3.0.4)
Requirement already satisfied: decorator>=4.3.0 in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from networkx>=2.4->vis
ions[type_image_path]==0.6.0->pandas-profiling) (4.4.1)
Requirement already satisfied: six in /Volumes/Samsung_T5/Anaconda/anac
onda3/lib/python3.7/site-packages (from imagehash; extra == "type_image
_path"->visions[type_image_path]==0.6.0->pandas-profiling) (1.14.0)
Requirement already satisfied: PyWavelets in /Volumes/Samsung_T5/Anacon
da/anaconda3/lib/python3.7/site-packages (from imagehash; extra == "typ
e_image_path"->visions[type_image_path]==0.6.0->pandas-profiling) (1.1.
1)
Requirement already satisfied: llvmlite<0.32.0,>=0.31.0dev0 in /Volume
```

```
s/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from numba
>=0.38.1->phik>=0.10.0->pandas-profiling) (0.31.0)
Requirement already satisfied: setuptools in /Volumes/Samsung_T5/Anacon
da/anaconda3/lib/python3.7/site-packages (from numba>=0.38.1->phik>=0.1
0.0->pandas-profiling) (46.0.0.post20200309)
Requirement already satisfied: notebook>=4.4.1 in /Volumes/Samsung_T5/A
naconda/anaconda3/lib/python3.7/site-packages (from widgetsnbextension~
=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (6.0.3)
Requirement already satisfied: jupyter-core in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from nbformat>=4.2.0->ipywi
dgets>=7.5.1->pandas-profiling) (4.6.1)
Requirement already satisfied: ipython-genutils in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from nbformat>=4.2.0->i
pywidgets>=7.5.1->pandas-profiling) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /Volumes/Sams
ung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from nbformat>=
4.2.0->ipywidgets>=7.5.1->pandas-profiling) (3.2.0)
Requirement already satisfied: appnope; platform_system == "Darwin" in
/Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (fro
m ipykernel>=4.5.1->ipywidgets>=7.5.1->pandas-profiling) (0.1.0)
Requirement already satisfied: jupyter-client in /Volumes/Samsung_T5/An
aconda/anaconda3/lib/python3.7/site-packages (from ipykernel>=4.5.1->ip
ywidgets>=7.5.1->pandas-profiling) (5.3.4)
Requirement already satisfied: tornado>=4.2 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipyw
idgets>=7.5.1->pandas-profiling) (6.0.3)
Requirement already satisfied: jedi>=0.10 in /Volumes/Samsung_T5/Anacon
da/anaconda3/lib/python3.7/site-packages (from ipython>=4.0.0; python_v
ersion >= "3.3"->ipywidgets>=7.5.1->pandas-profiling) (0.14.1)
Requirement already satisfied: pygments in /Volumes/Samsung_T5/Anaconda
a/anaconda3/lib/python3.7/site-packages (from ipython>=4.0.0; python_ve
rsion >= "3.3"->ipywidgets>=7.5.1->pandas-profiling) (2.5.2)
Requirement already satisfied: pickleshare in /Volumes/Samsung_T5/Anaco
nda/anaconda3/lib/python3.7/site-packages (from ipython>=4.0.0; python_
version >= "3.3"->ipywidgets>=7.5.1->pandas-profiling) (0.7.5)
Requirement already satisfied: pexpect; sys_platform != "win32" in /Vol
umes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from ip
ython>=4.0.0; python_version >= "3.3"->ipywidgets>=7.5.1->pandas-profil
ing) (4.8.0)
Requirement already satisfied: backcall in /Volumes/Samsung_T5/Anacond
```

```
a/anaconda3/lib/python3.7/site-packages (from ipython>=4.0.0; python_ve
rsion >= "3.3"->ipywidgets>=7.5.1->pandas-profiling) (0.1.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=
2.0.0 in /Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-pack
ages (from ipython>=4.0.0; python_version >= "3.3"->ipywidgets>=7.5.1->
pandas-profiling) (3.0.3)
Requirement already satisfied: Send2Trash in /Volumes/Samsung_T5/Anacon
da/anaconda3/lib/python3.7/site-packages (from notebook>=4.4.1->widgets
nbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (1.5.0)
Requirement already satisfied: prometheus-client in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from notebook>=4.4.1-
>widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (0.7.
1)
Requirement already satisfied: nbconvert in /Volumes/Samsung_T5/Anacond
a/anaconda3/lib/python3.7/site-packages (from notebook>=4.4.1->widgetsn
bextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (5.6.1)
Requirement already satisfied: terminado>=0.8.1 in /Volumes/Samsung_T5/
Anaconda/anaconda3/lib/python3.7/site-packages (from notebook>=4.4.1->w
idgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (0.8.3)
Requirement already satisfied: pyzmq>=17 in /Volumes/Samsung_T5/Anacond
a/anaconda3/lib/python3.7/site-packages (from notebook>=4.4.1->widgetsn
bextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (18.1.1)
Requirement already satisfied: importlib-metadata; python_version < "3.
8" in /Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-package
s (from jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets>=7.5.1->pa
ndas-profiling) (1.5.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from jsonschema!=2.5.
0,>=2.4->nbformat>=4.2.0->ipywidgets>=7.5.1->pandas-profiling) (0.15.7)
Requirement already satisfied: parso>=0.5.0 in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from jedi>=0.10->ipython>=
4.0.0; python_version >= "3.3"->ipywidgets>=7.5.1->pandas-profiling)
(0.5.2)
Requirement already satisfied: ptyprocess>=0.5 in /Volumes/Samsung_T5/A
naconda/anaconda3/lib/python3.7/site-packages (from pexpect; sys_platfo
rm != "win32"->ipython>=4.0.0; python_version >= "3.3"->ipywidgets>=7.
5.1->pandas-profiling) (0.6.0)
Requirement already satisfied: wcwidth in /Volumes/Samsung_T5/Anaconda/
anaconda3/lib/python3.7/site-packages (from prompt-toolkit!=3.0.0,!=3.
0.1,<3.1.0,>=2.0.0->ipython>=4.0.0; python_version >= "3.3"->ipywidgets
```

```
>=7.5.1->pandas-profiling) (0.1.8)
Requirement already satisfied: testpath in /Volumes/Samsung_T5/Anacond
a/anaconda3/lib/python3.7/site-packages (from nbconvert->notebook>=4.4.
1->widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (0.
4.4)
Requirement already satisfied: defusedxml in /Volumes/Samsung_T5/Anacon
da/anaconda3/lib/python3.7/site-packages (from nbconvert->notebook>=4.
4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling)
(0.6.0)
Requirement already satisfied: entrypoints>=0.2.2 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from nbconvert->noteb
ook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profil
ing) (0.3)
Requirement already satisfied: bleach in /Volumes/Samsung_T5/Anaconda/a
naconda3/lib/python3.7/site-packages (from nbconvert->notebook>=4.4.1->
widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profiling) (3.1.0)
Requirement already satisfied: mistune<2,>=0.8.1 in /Volumes/Samsung_T
5/Anaconda/anaconda3/lib/python3.7/site-packages (from nbconvert->noteb
ook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-profil
ing) (0.8.4)
Requirement already satisfied: pandocfilters>=1.4.1 in /Volumes/Samsung
_T5/Anaconda/anaconda3/lib/python3.7/site-packages (from nbconvert->not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-prof
iling) (1.4.2)
Requirement already satisfied: zipp>=0.5 in /Volumes/Samsung_T5/Anacond
a/anaconda3/lib/python3.7/site-packages (from importlib-metadata; pytho
n_version < "3.8"->jsonschema!=2.5.0,>=2.4->nbformat>=4.2.0->ipywidgets
>=7.5.1->pandas-profiling) (2.2.0)
Requirement already satisfied: webencodings in /Volumes/Samsung_T5/Anac
onda/anaconda3/lib/python3.7/site-packages (from bleach->nbconvert->not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.5.1->pandas-prof
iling) (0.5.1)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
from pandas_profiling import ProfileReport
profile = ProfileReport(df, title="Retail Profiling Report")
```

In [7]:
```
profile.to_file("Retail_Profiling.html")
```

In [8]:
```
profile.to_notebook_iframe()
```

Overview

Overview | Warnings 7 | Reproduction

Dataset statistics

Number of variables	8
Number of observations	541909
Missing cells	136534
Missing cells (%)	3.1%
Duplicate rows	5268
Duplicate rows (%)	1.0%
Total size in memory	33.1 MiB
Average record size in memory	64.0 B

Variable types

Categorical	5
Numeric	3

Variables

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	m
Quantity	541909.0	9.552250	218.081158	-80995.00	1.00	3.00	10.00	80995
UnitPrice	541909.0	4.611114	96.759853	-11062.06	1.25	2.08	4.13	3897(
CustomerID	406829.0	15287.690570	1713.600303	12346.00	13953.00	15152.00	16791.00	18287

In [10]:
```
df.isnull().any()
```

Out[10]:
```
InvoiceNo      False
StockCode      False
Description     True
Quantity       False
InvoiceDate    False
UnitPrice      False
CustomerID      True
Country        False
dtype: bool
```

In [11]:
```
df.Description.value_counts()
```

Out[11]:
```
WHITE HANGING HEART T-LIGHT HOLDER    2369
```

```
REGENCY CAKESTAND 3 TIER              2200
JUMBO BAG RED RETROSPOT               2159
PARTY BUNTING                         1727
LUNCH BAG RED RETROSPOT               1638
                                      ...
water damaged                            1
re-adjustment                            1
damages/credits from ASOS.               1
WALL ART,ONLY ONE PERSON                 1
dotcom adjust                            1
Name: Description, Length: 4223, dtype: int64
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [13]:
```
df = df.dropna()
df.shape
```

Out[13]:
```
(406829, 8)
```

In [14]:
```
df.duplicated().sum()
```

Out[14]:
```
5225
```

In [15]:
```
df.drop_duplicates(keep='first', inplace=True)
df.shape
```

```
/Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[15]: (401604, 8)

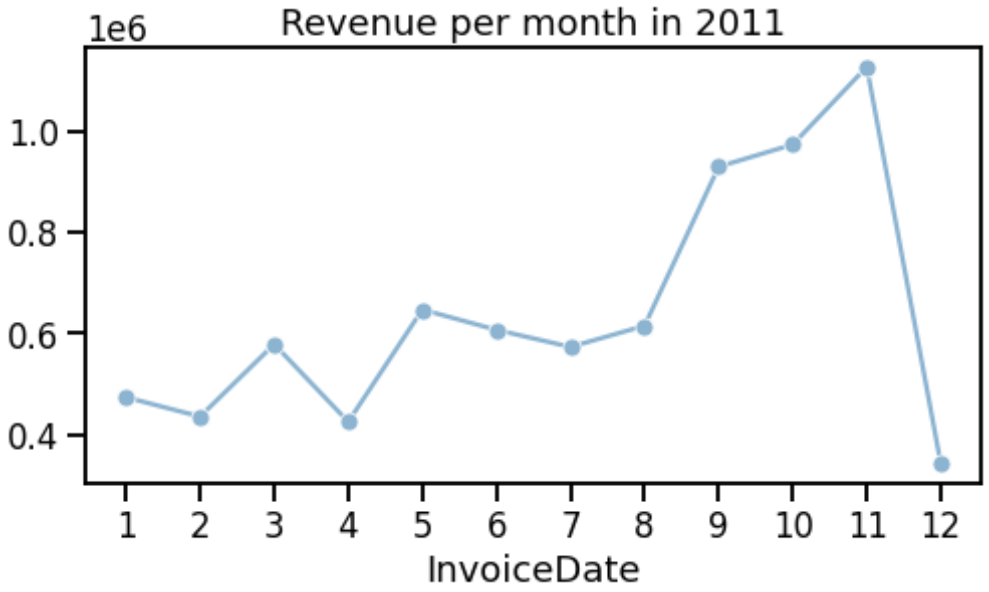
In [16]:
```
retail = df.copy()
```

Exploration of the data

In [17]:
```
retail['Revenue'] = retail['Quantity'] * retail['UnitPrice']
```

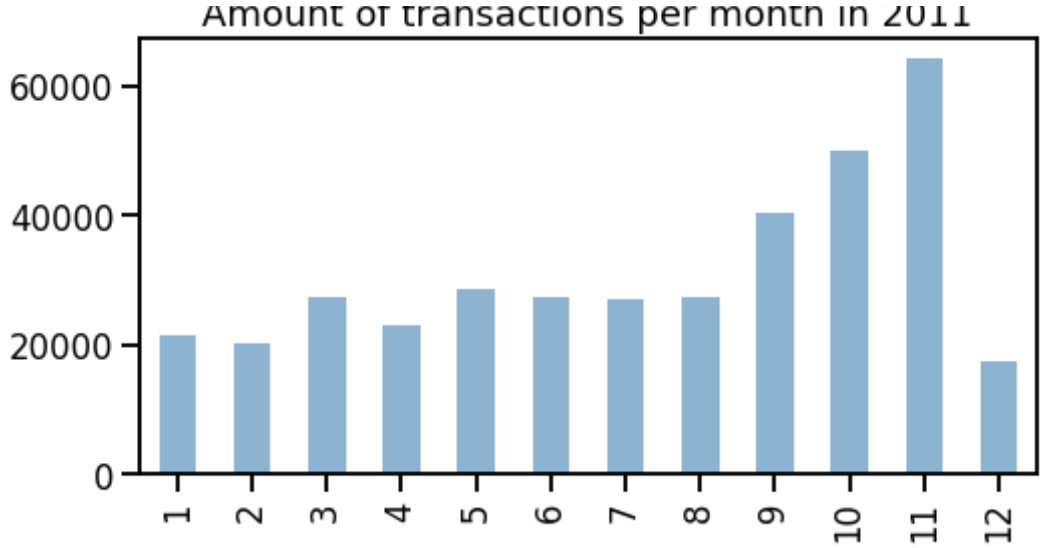
In [18]:
```
retail.InvoiceDate = pd.to_datetime(retail['InvoiceDate'], format='%m/%d/%Y %H:%M')
```

In [19]:
```
# Let's visualize the top grossing months
retail_month = retail[retail.InvoiceDate.dt.year==2011]
monthly_gross = retail_month.groupby(retail_month.InvoiceDate.dt.month).Revenue.sum()

plt.figure(figsize=(8,4))
sns.set_context("talk")
sns.set_palette("PuBuGn_d")
sns.lineplot(y=monthly_gross.values,x=monthly_gross.index, marker='o')
plt.xticks(range(1,13))
plt.title("Revenue per month in 2011")
plt.show()
```



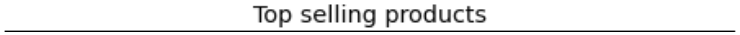
Here we observe that maximum revenue is coming in between october and december

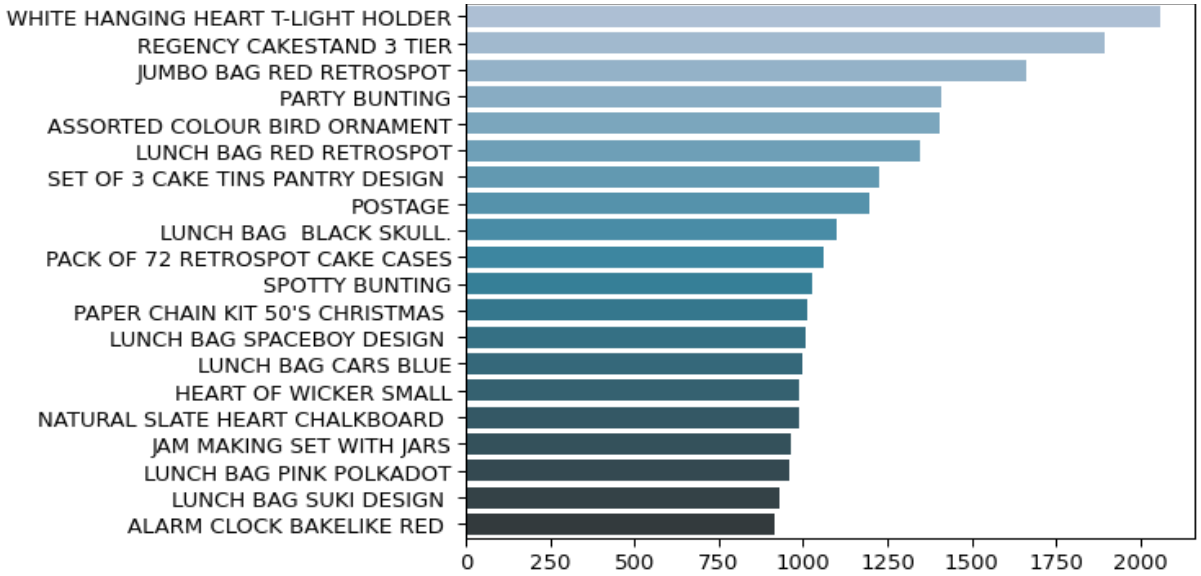
In [20]:
```
#amount of transactions per month
plt.figure(figsize=(8,4))
retail[retail.InvoiceDate.dt.year==2011].InvoiceDate.dt.month.value_cou
nts(sort=False).plot(kind='bar')
plt.title("Amount of transactions per month in 2011")
plt.show()
```



Maximum number of transaction is coming in between october and december

In [21]:
```
# Let's visualize some top products from the whole range
top_products = retail['Description'].value_counts()[:20]
plt.figure(figsize=(8,6))
sns.set_context("paper", font_scale=1.5)
sns.barplot(y = top_products.index,
            x = top_products.values,
           palette='PuBuGn_d')
plt.title("Top selling products")
plt.show()
plt.savefig('top_products.png')
```






```
<Figure size 432x288 with 0 Axes>
```

Cohort Analysis

In [22]:
```
#Assign acquisition month cohort to each customer
#creating invoice month column to see first month when customer purchas
ed
import datetime as dt
retail['InvoiceMonth'] = retail['InvoiceDate'].apply(lambda x: dt.datet
ime(x.year, x.month, 1))
```

In [23]:
```
grouping = retail.groupby('CustomerID')['InvoiceMonth']
#assign smallest invoice value to each customer
retail['CohortMonth'] = grouping.transform('min')
retail.head()
```

Out[23]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Re

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Re
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	

In [24]:
```
#function to extract year, month, day as integers
def get_date_int(df, column):
    year = df[column].dt.year
    month = df[column].dt.month
    day = df[column].dt.day
    return year, month, day
```

In [25]:
```
#extract month
invoice_year, invoice_month, _ = get_date_int(retail, 'InvoiceMonth')
cohort_year, cohort_month, _ = get_date_int(retail, 'CohortMonth')
```

In [26]:
```
years_diff = invoice_year - cohort_year
```

```
months_diff = invoice_month - cohort_month
```

In [27]:
```
# Extract the difference in days from all previous values
retail['CohortIndex'] = years_diff * 12 + months_diff + 1
retail.head()
```

Out[27]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Re
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	

In [28]:
```
#count monthly active customers from each cohort
grouping = retail.groupby(['CohortMonth', 'CohortIndex'])
cohort_data = grouping['CustomerID'].apply(pd.Series.nunique)
cohort_data = cohort_data.reset_index()
cohort_counts = cohort_data.pivot(index='CohortMonth', columns = 'Cohor
tIndex', values='CustomerID')
```

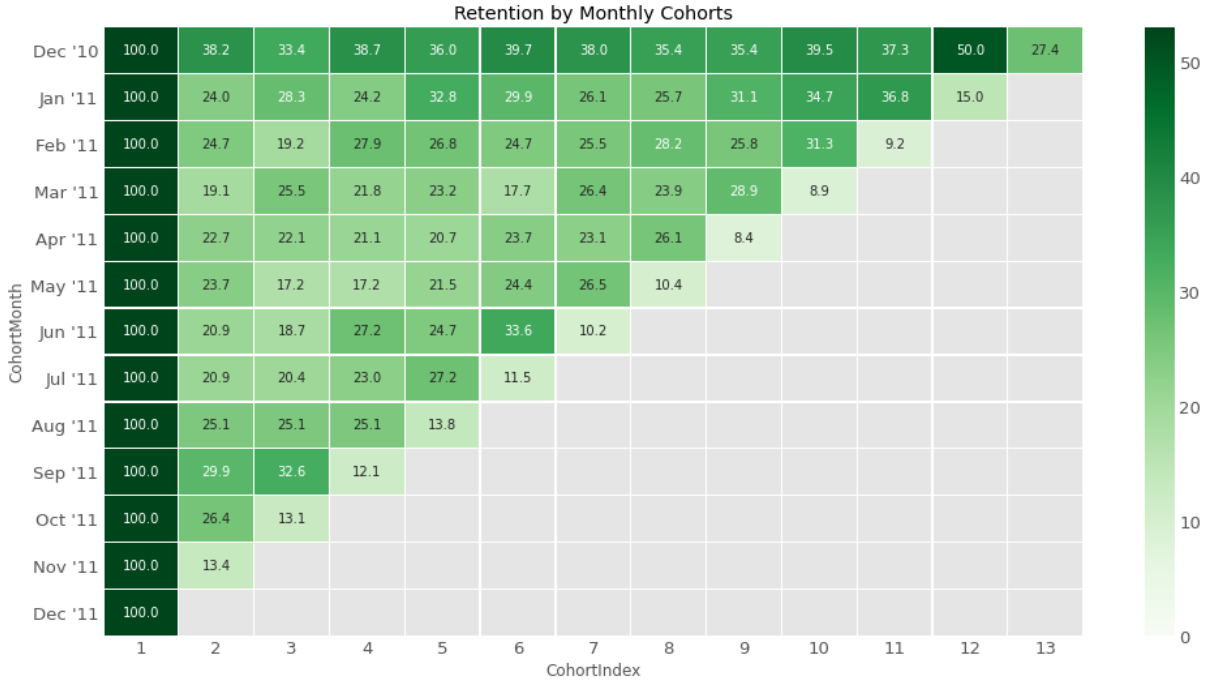
In [29]:
```
#Customer retention
cohort_sizes = cohort_counts.iloc[:,0]
retention = cohort_counts.divide(cohort_sizes, axis=0)
retention = retention.round(3) * 100
retention.head(20)
```

Out[29]:

CohortIndex	1	2	3	4	5	6	7	8	9	10	11	12	13
CohortMonth													
2010-12-01	100.0	38.2	33.4	38.7	36.0	39.7	38.0	35.4	35.4	39.5	37.3	50.0	27.4
2011-01-01	100.0	24.0	28.3	24.2	32.8	29.9	26.1	25.7	31.1	34.7	36.8	15.0	NaN
2011-02-01	100.0	24.7	19.2	27.9	26.8	24.7	25.5	28.2	25.8	31.3	9.2	NaN	NaN
2011-03-01	100.0	19.1	25.5	21.8	23.2	17.7	26.4	23.9	28.9	8.9	NaN	NaN	NaN
2011-04-01	100.0	22.7	22.1	21.1	20.7	23.7	23.1	26.1	8.4	NaN	NaN	NaN	NaN
2011-05-01	100.0	23.7	17.2	17.2	21.5	24.4	26.5	10.4	NaN	NaN	NaN	NaN	NaN
2011-06-01	100.0	20.9	18.7	27.2	24.7	33.6	10.2	NaN	NaN	NaN	NaN	NaN	NaN
2011-07-01	100.0	20.9	20.4	23.0	27.2	11.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-08-01	100.0	25.1	25.1	25.1	13.8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-09-01	100.0	29.9	32.6	12.1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-10-01	100.0	26.4	13.1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-11-01	100.0	13.4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-12-01	100.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [67]:
```
month_list = ["Dec '10", "Jan '11", "Feb '11", "Mar '11", "Apr '11",\
              "May '11", "Jun '11", "Jul '11", "Aug '11", "Sep '11", \
              "Oct '11", "Nov '11", "Dec '11"]

plt.figure(figsize=(15,8))
plt.title('Retention by Monthly Cohorts')
sns.heatmap(data=retention,
            annot = True,
```

```
            cmap = "Greens",
            vmin = 0.0,
            vmax = list(retention.max().sort_values(ascending = False))
[1]+3,
            fmt = '.1f',
            linewidth = 0.3,
            yticklabels=month_list)

plt.show()
```



RFM Analysis(Week 2)

In [31]:
```
#which customers are the best ones by examining how recently a customer
has purchased (recency), how often
#they purchase (frequency), and how much the customer spends (monetary)
```

In [32]:
```
#12 months of data
print('Min:{}; Max:{}'.format(min(retail.InvoiceDate), max(retail.Invoi
ceDate)))
```

```
Min:2010-12-01 08:26:00; Max:2011-12-09 12:50:00
```

In [33]:
```
#calculate revenue per row and add new column
retail['MonetaryValue'] = retail['Quantity'] * retail['UnitPrice']
```

In [34]:
```
#let's look at amount spend per customer (revenue contributed) M-Moneta
ry
retail_mv = retail.groupby(['CustomerID']).agg({'MonetaryValue': sum}).
reset_index()
retail_mv.head()
```

Out[34]:

	CustomerID	MonetaryValue
0	12346.0	0.00
1	12347.0	4310.00
2	12348.0	1797.24
3	12349.0	1757.55
4	12350.0	334.40

In [35]:
```
#F-frequency (how many purchases each customer made)
retail_f = retail.groupby('CustomerID')['InvoiceNo'].count()
retail_f = retail_f.reset_index()
retail_f.head()
```

Out[35]:

	CustomerID	InvoiceNo
0	12346.0	2
1	12347.0	182

	CustomerID	InvoiceNo
2	12348.0	31
3	12349.0	73
4	12350.0	17

In [36]:
```
#merge previous dataframes together (mv+f)
retail_mv_f = pd.merge(retail_mv, retail_f, on='CustomerID', how='inne
r')
retail_mv_f.head()
```

Out[36]:

	CustomerID	MonetaryValue	InvoiceNo
0	12346.0	0.00	2
1	12347.0	4310.00	182
2	12348.0	1797.24	31
3	12349.0	1757.55	73
4	12350.0	334.40	17

In [37]:
```
#R-recency
#last transaction date

retail['InvoiceDate'] = pd.to_datetime(retail['InvoiceDate'],format='%d
-%m-%Y %H:%M')
max_date = max(retail['InvoiceDate'])

#difference between last date and transaction date
retail['Diff'] = max_date - retail['InvoiceDate']
retail.head()
```

Out[37]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Re
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Re
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	

In [38]:
```
#recency per customer (last transaction date)
retail_r = retail.groupby('CustomerID')['Diff'].min()
retail_r = retail_r.reset_index()

# Extract number of days only
retail_r['Diff'] = retail_r['Diff'].dt.days
```

In [39]:
```
#merge R dataframe with FM

retail_rfm = pd.merge(retail_mv_f, retail_r, on='CustomerID', how='inner')
retail_rfm.columns = ['CustomerID', 'MonetaryValue', 'Frequency', 'Recency']
retail_rfm.head()
```

Out[39]:

	CustomerID	MonetaryValue	Frequency	Recency
0	12346.0	0.00	2	325

	CustomerID	MonetaryValue	Frequency	Recency
1	12347.0	4310.00	182	1
2	12348.0	1797.24	31	74
3	12349.0	1757.55	73	18
4	12350.0	334.40	17	309

In [40]:
```
cols = retail_rfm.columns.tolist()
cols
```

Out[40]:
```
['CustomerID', 'MonetaryValue', 'Frequency', 'Recency']
```

In [41]:
```
#changed columns order
cols = ['CustomerID', 'Recency', 'Frequency', 'MonetaryValue']
retail_rfm = retail_rfm[cols]
retail_rfm.head()
```

Out[41]:

	CustomerID	Recency	Frequency	MonetaryValue
0	12346.0	325	2	0.00
1	12347.0	1	182	4310.00
2	12348.0	74	31	1797.24
3	12349.0	18	73	1757.55
4	12350.0	309	17	334.40

In [42]:
```
# create labels and assign them to tree percentile groups
r_labels = range(4, 0, -1)
r_groups = pd.qcut(retail_rfm.Recency, q = 4, labels = r_labels)
f_labels = range(1, 5)
f_groups = pd.qcut(retail_rfm.Frequency, q = 4, labels = f_labels)
m_labels = range(1, 5)
m_groups = pd.qcut(retail_rfm.MonetaryValue, q = 4, labels = m_labels)
```

In [43]:
```
# make a new column for group labels
retail_rfm['R'] = r_groups.values
retail_rfm['F'] = f_groups.values
retail_rfm['M'] = m_groups.values
# sum up the three columns
retail_rfm['RFM_Segment'] = retail_rfm.apply(lambda x: str(x['R']) + st
r(x['F']) + str(x['M']), axis = 1)
retail_rfm['RFM_Score'] = retail_rfm[['R', 'F', 'M']].sum(axis = 1)
retail_rfm.head()
```

Out[43]:

	CustomerID	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score
0	12346.0	325	2	0.00	1	1	1	1.01.01.0	3
1	12347.0	1	182	4310.00	4	4	4	4.04.04.0	12
2	12348.0	74	31	1797.24	2	2	4	2.02.04.0	8
3	12349.0	18	73	1757.55	3	3	4	3.03.04.0	10
4	12350.0	309	17	334.40	1	1	2	1.01.02.0	4

In [44]:
```
# assign labels from total score
score_labels = ['Green', 'Bronze', 'Silver', 'Gold']
score_groups = pd.qcut(retail_rfm.RFM_Score, q = 4, labels = score_labe
ls)
retail_rfm['RFM_Level'] = score_groups.values
retail_rfm.sort_values(by='RFM_Score', ascending=False)
retail_rfm.head(10)
```

Out[44]:

	CustomerID	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score	RFM_
0	12346.0	325	2	0.00	1	1	1	1.01.01.0	3	
1	12347.0	1	182	4310.00	4	4	4	4.04.04.0	12	
2	12348.0	74	31	1797.24	2	2	4	2.02.04.0	8	
3	12349.0	18	73	1757.55	3	3	4	3.03.04.0	10	
4	12350.0	309	17	334.40	1	1	2	1.01.02.0	4	

	CustomerID	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score	RFM_
5	12352.0	35	95	1545.41	3	3	3	3.03.03.0	9	
6	12353.0	203	4	89.00	1	1	1	1.01.01.0	3	
7	12354.0	231	58	1079.40	1	3	3	1.03.03.0	7	B
8	12355.0	213	13	459.40	1	1	2	1.01.02.0	4	
9	12356.0	22	59	2811.43	3	3	4	3.03.04.0	10	

In [45]:
```
retail_rfm_levels = retail_rfm.groupby('RFM_Level')['CustomerID'].count
().reset_index(name='counts')
retail_rfm_levels.head()
```

Out[45]:

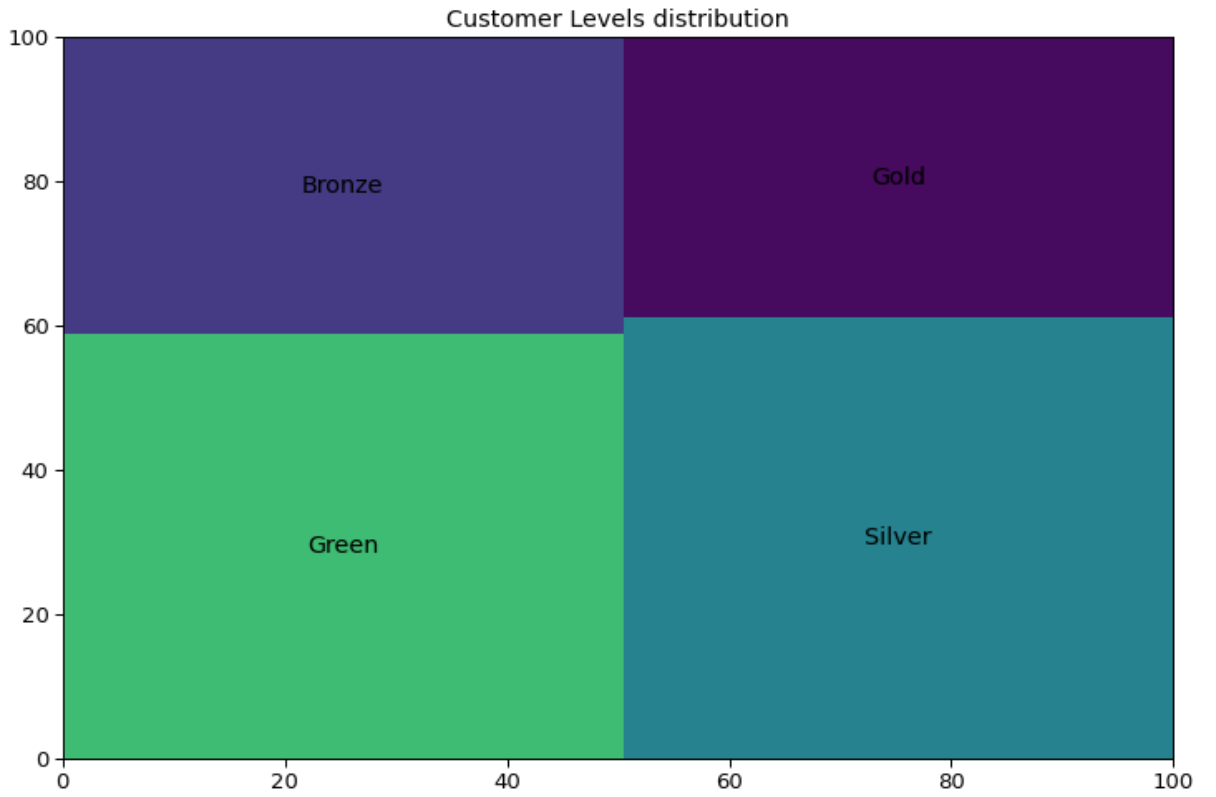
	RFM_Level	counts
0	Green	1298
1	Bronze	908
2	Silver	1322
3	Gold	844

In [46]:
```
pip install squarify
```

```
Requirement already satisfied: squarify in /Volumes/Samsung_T5/Anacond
a/anaconda3/lib/python3.7/site-packages (0.4.3)
Note: you may need to restart the kernel to use updated packages.
```

In [47]:
```
#let's exclude others segment for visualization
import squarify
levels = list(retail_rfm_levels.RFM_Level)
score = list(retail_rfm_levels.counts)
plt.figure(figsize=(12,8))
plt.title('Customer Levels distribution')
squarify.plot(sizes=score, label=levels)
```

```
plt.show()
```

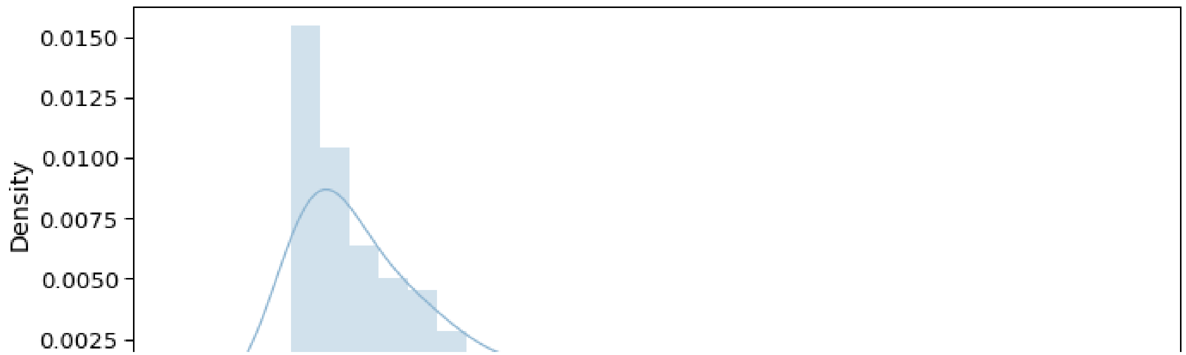


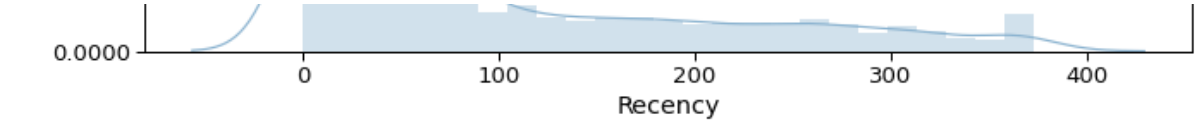
In [48]:
```
rfm_rfm = retail_rfm[['Recency','Frequency','MonetaryValue']]
print(rfm_rfm.describe())
```

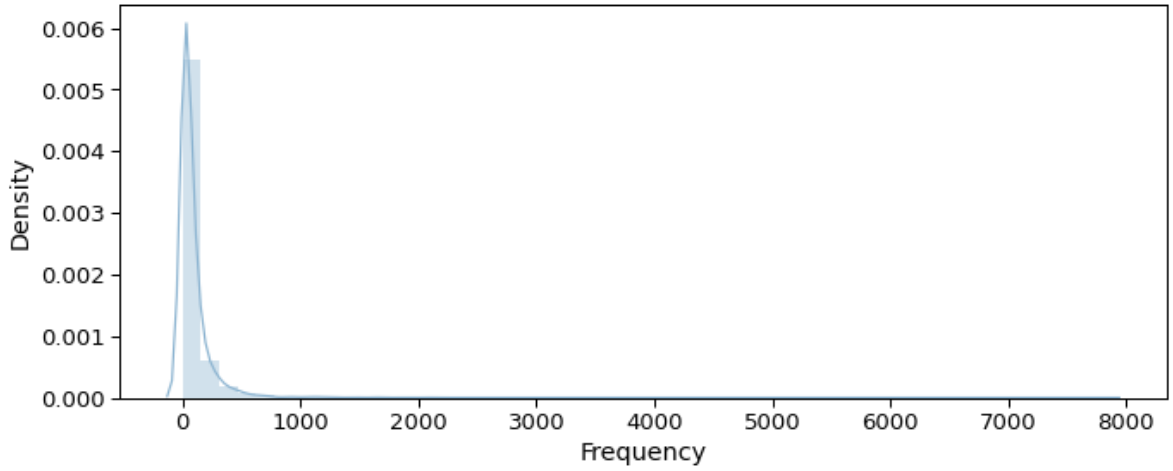
```
           Recency    Frequency  MonetaryValue
count  4372.000000  4372.000000    4372.000000
mean     91.047118    91.858188    1893.531433
std     100.765435   229.223566    8218.696204
min       0.000000     1.000000   -4287.630000
25%      16.000000    17.000000     291.795000
50%      49.000000    41.000000     644.070000
75%     142.000000    99.250000    1608.335000
max     373.000000  7812.000000  279489.020000
```

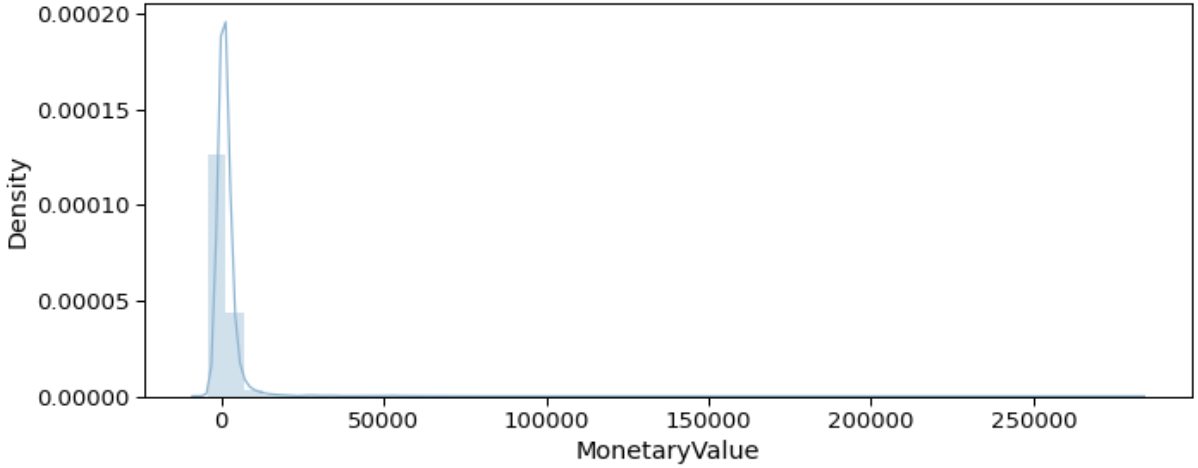
In [49]:
```
f,ax = plt.subplots(figsize=(10, 12))
plt.subplot(3, 1, 1); sns.distplot(rfm_rfm.Recency, label = 'Recency')
plt.subplot(3, 1, 2); sns.distplot(rfm_rfm.Frequency, label = 'Frequenc
y')
plt.subplot(3, 1, 3); sns.distplot(rfm_rfm.MonetaryValue, label = 'Mone
tary Value')
plt.style.use('fivethirtyeight')
plt.tight_layout()
plt.show()
```

```
/Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages/seab
orn/distributions.py:2557: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages/seab
orn/distributions.py:2557: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Volumes/Samsung_T5/Anaconda/anaconda3/lib/python3.7/site-packages/seab
orn/distributions.py:2557: FutureWarning: `distplot` is a deprecated fu
nction and will be removed in a future version. Please adapt your code
to use either `displot` (a figure-level function with similar flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```









In [50]:
```
#Also, there is another Problem: UnSymmetric distribution of variables
 (data skewed)

#Soluation:Logarithmic transformation (positive values only) will manag
e skewness
```

```
#We use these Sequence of structuring pre-processing steps: 1. Unskew t
he data - log transformation

#2. Standardize to the same average values

#3. Scale to the same standard deviation

#4. Store as a separate array to be used for clustering

#Why the sequence matters?

#Log transformation only works with positive data
#Normalization forces data to have negative values and log will not wor
k
```

In [51]:
```
retail_rfm.isnull().sum()
```

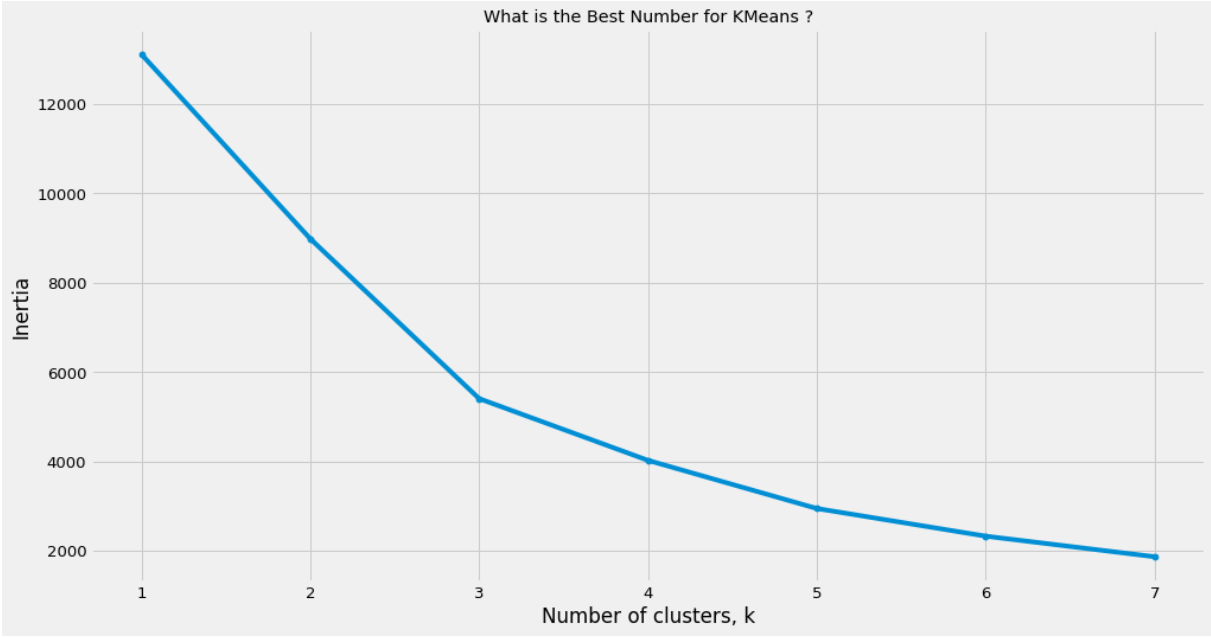
Out[51]:
```
CustomerID       0
Recency          0
Frequency        0
MonetaryValue    0
R                0
F                0
M                0
RFM_Segment      0
RFM_Score        0
RFM_Level        0
dtype: int64
```

Implementation of K-Means Clustering(Week 3)

In [52]:
```
#Key steps

#Data pre-processing
#Choosing a number of clusters
#Running k-means clustering on pre-processed data
#Analyzing average RFM values of each cluster
```

In [53]:
```
#Normalize the variables with StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(rfm_rfm)
#Store it separately for clustering
rfm_normalized= scaler.transform(rfm_rfm)
```

In [54]:
```
from sklearn.cluster import KMeans

#First : Get the Best KMeans
ks = range(1,8)
inertias=[]
for k in ks :
    # Create a KMeans clusters
    kc = KMeans(n_clusters=k,random_state=1)
    kc.fit(rfm_normalized)
    inertias.append(kc.inertia_)

# Plot ks vs inertias
f, ax = plt.subplots(figsize=(15, 8))
plt.plot(ks, inertias, '-o')
plt.xlabel('Number of clusters, k')
plt.ylabel('Inertia')
plt.xticks(ks)
plt.style.use('ggplot')
plt.title('What is the Best Number for KMeans ?')
plt.show()
```



In [68]:
```
range_n_clusters = list (range(2,10))
```

In [71]:
```
from sklearn.metrics import silhouette_score
for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict(rfm_rfm)
    centers = clusterer.cluster_centers_

    score = silhouette_score(rfm_rfm, preds)
    print("For n_clusters = {}, silhouette score is {})".format(n_clust
ers, score))
```

```
For n_clusters = 2, silhouette score is 0.98630357275616)
For n_clusters = 3, silhouette score is 0.9630745414576583)
For n_clusters = 4, silhouette score is 0.8826378276586778)
For n_clusters = 5, silhouette score is 0.8148946297109425)
For n_clusters = 6, silhouette score is 0.7768985939280555)
For n_clusters = 7, silhouette score is 0.7722712273216985)
For n_clusters = 8, silhouette score is 0.719408365268113)
For n_clusters = 9, silhouette score is 0.6713235099155885)
```

In [55]:
```
kc = KMeans(n_clusters= 3, random_state=1)
kc.fit(rfm_normalized)

#Create a cluster label column in the original DataFrame
cluster_labels = kc.labels_

#Calculate average RFM values and size for each cluster:
rfm_rfm_k3 = rfm_rfm.assign(K_Cluster = cluster_labels)

#Calculate average RFM values and sizes for each cluster:
rfm_rfm_k3.groupby('K_Cluster').agg({'Recency': 'mean','Frequency': 'mean',
                                     'MonetaryValue': ['mean', 'count'],}).round(0)
```

Out[55]:

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
K_Cluster				
0	246.0	27.0	459.0	1096
1	39.0	104.0	1946.0	3264
2	4.0	2814.0	118565.0	12

In [58]:
```
rfm_normalized = pd.DataFrame(rfm_normalized,index=rfm_rfm.index,columns=rfm_rfm.columns)
rfm_normalized['K_Cluster'] = kc.labels_
rfm_normalized['General_Segment'] = retail_rfm['RFM_Segment']
rfm_normalized['CustomerID'] = retail_rfm['CustomerID']
rfm_normalized.reset_index(inplace = True)

#Melt the data into a long format so RFM values and metric names are stored in 1 column each
rfm_melt = pd.melt(rfm_normalized,id_vars=['CustomerID','General_Segment','K_Cluster'],value_vars=['Recency', 'Frequency', 'MonetaryValue'],
```

```
var_name='Metric',value_name='Value')
rfm_melt.head()
```

Out[58]:

	CustomerID	General_Segment	K_Cluster	Metric	Value
0	12346.0	1.01.01.0	0	Recency	2.322023
1	12347.0	4.04.04.0	1	Recency	-0.893733
2	12348.0	2.02.04.0	1	Recency	-0.169196
3	12349.0	3.03.04.0	1	Recency	-0.725005
4	12350.0	1.01.02.0	0	Recency	2.163220

In [60]:
```
#Relative importance of segment attributes

#Useful technique to identify relative importance of each segment's att
ribute
#Calculate average values of each cluster
#Calculate average values of population
#Let’s try again with a heat map. Heat maps are a graphical representat
ion of data where larger values
#were colored in darker scales and smaller values in lighter scales. We
can compare the variance between
#the groups quite intuitively by colors.
```

In [61]:
```
# The further a ratio is from 0, the more important that attribute is f
or a segment relative to the total population
cluster_avg = rfm_rfm_k3.groupby(['K_Cluster']).mean()
population_avg = rfm_rfm.mean()
relative_imp = cluster_avg / population_avg - 1
relative_imp.round(2)
```

Out[61]:

K_Cluster	Recency	Frequency	MonetaryValue
0	1.70	-0.70	-0.76

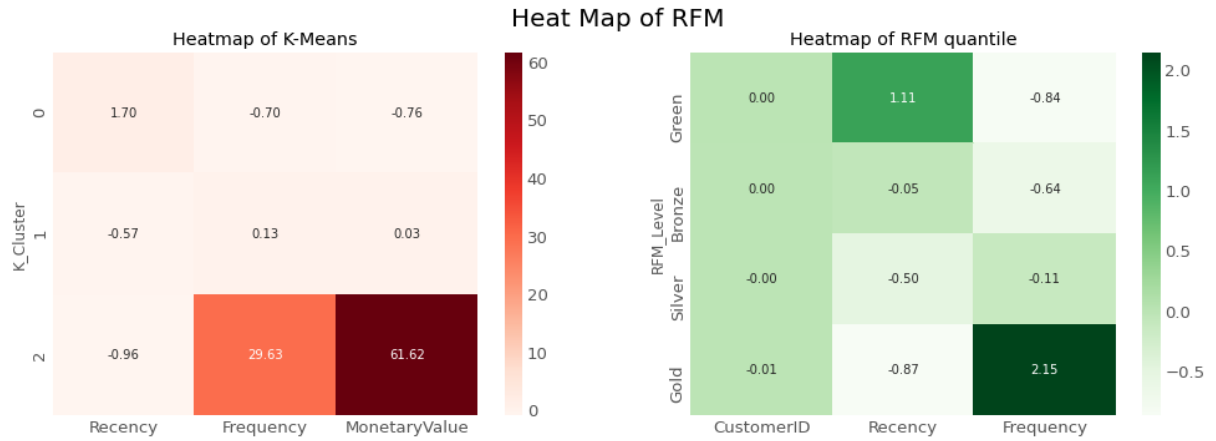
K_Cluster	Recency	Frequency	MonetaryValue
1	-0.57	0.13	0.03
2	-0.96	29.63	61.62

In [62]:
```
total_avg = retail_rfm.iloc[:, 0:3].mean()
# calculate the proportional gap with total mean
cluster_avg = retail_rfm.groupby('RFM_Level').mean().iloc[:, 0:3]
prop_rfm = cluster_avg/total_avg - 1
prop_rfm.round(2)
```

Out[62]:

RFM_Level	CustomerID	Recency	Frequency
Green	0.00	1.11	-0.84
Bronze	0.00	-0.05	-0.64
Silver	-0.00	-0.50	-0.11
Gold	-0.01	-0.87	2.15

In [66]:
```
# heatmap with RFM
f, (ax1, ax2) = plt.subplots(1,2, figsize=(15, 5))
sns.heatmap(data=relative_imp, annot=True, fmt='.2f', cmap='Reds',ax=ax
1)
ax1.set(title = "Heatmap of K-Means")

# a snake plot with K-Means
sns.heatmap(prop_rfm, cmap= 'Greens', fmt= '.2f', annot = True,ax=ax2)
ax2.set(title = "Heatmap of RFM quantile")

plt.suptitle("Heat Map of RFM",fontsize=20) #make title fontsize subtit
le
plt.show()
```



Creating Tableau Dashboard(Week 4)

https://public.tableau.com/profile/sayar.samanta#!/vizhome/CapstoneRetailProject_161219406126(
publish=yes