Quantum data preparation and customer analytics-

Round 2

Describing business lines, starting data first and looking at what is inside.
p.s I was not able to convert the DATE in python so I did it in Excel and I convert DIM_transaction_date from .csv file to .xlsx in order to read it.

```
!pip install numpy==1.19.5
!pip install xlrd
import numpy as np
print("Numpy version: ", np.__version__)
```

```
Requirement already satisfied: numpy==1.19.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: xlrd in /usr/local/lib/python3.7/dist-packages (1.1.0)
Numpy version:  1.19.5
```

We will check for null values across

```
import pandas as pd
DIM_TRANSACTION_DATA = pd.read_excel("/content/drive/MyDrive/Quantum/DIM_TRANSACTION_DATA.xlsx")
DIM_TRANSACTION_DATA.isnull().sum()
```

```
TRANSACTION_DATE    0
DAY_OF_WEEK         0
MONTH               0
YEAR                0
dtype: int64
```

```
DIM_TRANSACTION_DATA.sample(3)
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
1	2018-01-02	2	1	2018
2	2018-01-03	3	1	2018
3	2018-01-04	4	1	2018

```
DIM_TRANSACTION_DATA.shape
```

```
(1095, 4)
```

```
DIM_TRANSACTION_DATA.info()
```

```
<class'pandas.core.frame.DataFrame'>
RangeIndex: 1095 entries, 0 to 1094
Data columns (total 4 columns):
 TRANSACTION_DATE  1095 non-null  datetime64[ns]
 DAY_OF_WEEK       1095 non-null  int64
 MONTH             1095 non-null  int64
 YEAR              1095 non-null  int64
dtype: datetime64[ns], int64(3)
memory usage: 34.3 KB
```

```
DIM_TRANSACTION_DATA.describe()
```

	DAY_OF_WEEK	MONTH	YEAR
count	1095.000000	1095.000000	1095.000000
mean	4.000000	6.526027	2018.499543
std	2.000457	3.447968	0.500228
min	1.000000	1.000000	2018.000000
25%	2.000000	3.500000	2018.000000
50%	4.000000	6.526027	2018.499543
75%	6.000000	9.000000	2019.000000
max	7.000000	12.000000	2019.000000

```
DIM_TRANSACTION_DATA.head(3)
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
0	2018-01-01	1	1	2018
1	2018-01-02	2	1	2018
2	2018-01-03	3	1	2018

Now that we know that there is no null value let we proceed

Data Cleaning and Exploration

The cleaning involves where is the number of unique values in each column and I utilized dropna to see the total number of rows and column of the data

```
DIM_TRANSACTION_DATA.dropna().nunique(axis=0)
```

```
TRANSACTION_DATE    1095
DAY_OF_WEEK            7
MONTH                 12
YEAR                   2
dtype: int64
```

```
DIM_TRANSACTION_DATA.dropna().shape
```

```
(1095, 4)
```

```
DIM_TRANSACTION_DATA.dtypes
```

```
TRANSACTION_DATE    datetime64[ns]
DAY_OF_WEEK                  int64
MONTH                        int64
YEAR                         int64
dtype: datetime64[ns], int64(3)
```

```
DIM_TRANSACTION_DATA.dropna().dtypes
```

```
TRANSACTION_DATE    datetime64[ns]
DAY_OF_WEEK                  int64
MONTH                        int64
YEAR                         int64
dtype: datetime64[ns], int64(3)
```

```
DIM_TRANSACTION_DATA.dropna().describe()
```

	DAY_OF_WEEK	MONTH	YEAR
count	1095.000000	1095.000000	1095.000000
mean	4.000000	6.526027	2018.499543
std	2.000457	3.447968	0.500228
min	1.000000	1.000000	2018.000000
25%	2.000000	3.500000	2018.000000
50%	4.000000	6.526027	2018.499543
75%	6.000000	9.000000	2019.000000
max	7.000000	12.000000	2019.000000

```
DIM_TRANSACTION_DATA.head(3)
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
0	2018-01-01	1	1	2018
1	2018-01-02	2	1	2018
2	2018-01-03	3	1	2018

We will check for null values across

```
DIM_CUSTOMER_DATA.isnull().sum()
```

```
CUSTOMER_NAME    0
CUSTOMER_ID      0
CITY             0
COUNTRY          0
CONTINENT        0
dtype: int64
```

```
DIM_CUSTOMER_DATA.nunique(axis=0)
```

```
CUSTOMER_NAME    1000
CUSTOMER_ID      1000
CITY              100
COUNTRY            10
CONTINENT           4
dtype: int64
```

```
DIM_CUSTOMER_DATA.dropna().nunique(axis=0)
```

```
CUSTOMER_NAME    1000
CUSTOMER_ID      1000
CITY              100
COUNTRY            10
CONTINENT           4
dtype: int64
```

```
DIM_CUSTOMER_DATA.dropna().shape
```

```
(1000, 5)
```

```
DIM_CUSTOMER_DATA.dtypes
```

```
CUSTOMER_NAME    object
CUSTOMER_ID      object
CITY             object
COUNTRY          object
CONTINENT        object
dtype: object
```

```
DIM_CUSTOMER_DATA.dropna().dtypes
```

```
CUSTOMER_NAME    object
CUSTOMER_ID      object
CITY             object
COUNTRY          object
CONTINENT        object
dtype: object
```

```
DIM_CUSTOMER_DATA.dropna().describe()
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
count	1000	1000	1000	1000	1000
unique	1000	1000	100	10	4
top	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
freq	1	1	15	150	500

```
DIM_CUSTOMER_DATA.head(3)
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
0	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas
1	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas
2	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas

Now that we know that there is no null value as can proceed

Data Cleaning and Exploration

```
DIM_CUSTOMER_DATA.sample(3)
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
1000	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
1000	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
1000	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas

There seems to be a number of the missing values
Ms. Yvonne Hernandez with CUS-1000 is probably the same product. Not judging from the job since nobody seem to come drop it buying 3 times.

```
DIM_CUSTOMER_DATA[DIM_CUSTOMER_DATA['CUSTOMER_NAME'] == 'Ms. Yvonne Hernandez']
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
999	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
1000	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
1001	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas

```
DIM_CUSTOMER_DATA[DIM_CUSTOMER_DATA['CUSTOMER_ID'] == 'CUS-1000']
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
999	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
1000	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
1001	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas

```
DIM_CUSTOMER_DATA.shape
```

```
(1000, 5)
```

```
DIM_CUSTOMER_DATA.info()
```

```
<class'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 CUSTOMER_NAME  1000 non-null  object
 CUSTOMER_ID    1000 non-null  object
 CITY           1000 non-null  object
 COUNTRY        1000 non-null  object
 CONTINENT      1000 non-null  object
dtype: object
memory usage: 39.1 KB
```

```
DIM_CUSTOMER_DATA.describe()
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
count	1000	1000	1000	1000	1000
unique	1000	1000	100	10	4
top	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
freq	1	1	15	150	500

```
DIM_CUSTOMER_DATA.head(3)
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
0	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas
1	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas
2	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas

Looks like order shows up clients and City brands and Round Rock Texas seem to be a line service brands with slightly different names. And I have called the data to the Excel.

```
DIM_PRODUCTION_BRAND_NAMES.sample(3)
```

	BRAND_NAME	PRODUCTION_BRANDS
1000	Round Rock Texas	Round Rock Texas
1000	Round Rock Texas	Round Rock Texas
1000	Round Rock Texas	Round Rock Texas

After Cleaning brand names and finding length of product, I would like to look for Products that are not ships, which I will use the key word Flight length to find "Not" and I found two values this 2 seem as for the Sample Solution. Did not even know what else was. = =

```
DIM_PRODUCT_DATA[DIM_PRODUCT_DATA['PRODUCTION_BRANDS'].sample(3)]
```

	PRODUCT_NAME	PRODUCT_ID	PRODUCT_LENGTH	PRODUCT_WIDTH	PRODUCT_HEIGHT	PRODUCT_WEIGHT	PRODUCT_COLOR	PRODUCT_MANUFACTURER	PRODUCTION_BRANDS
1000	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Ultra Ship Steel
1000	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Ultra Ship Steel
1000	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Ultra Ship Steel

```
DIM_PRODUCT_DATA[DIM_PRODUCT_DATA['PRODUCTION_BRANDS'] == 'Not']
```

	PRODUCT_NAME	PRODUCT_ID	PRODUCT_LENGTH	PRODUCT_WIDTH	PRODUCT_HEIGHT	PRODUCT_WEIGHT	PRODUCT_COLOR	PRODUCT_MANUFACTURER	PRODUCTION_BRANDS
1000	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Not
1000	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Not

```
DIM_PRODUCT_DATA.shape
```

```
(1000, 9)
```

```
DIM_PRODUCT_DATA.info()
```

```
<class'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 PRODUCT_NAME        1000 non-null  object
 PRODUCT_ID          1000 non-null  object
 PRODUCT_LENGTH      1000 non-null  float64
 PRODUCT_WIDTH       1000 non-null  float64
 PRODUCT_HEIGHT      1000 non-null  float64
 PRODUCT_WEIGHT      1000 non-null  float64
 PRODUCT_COLOR       1000 non-null  object
 PRODUCT_MANUFACTURER  1000 non-null  object
 PRODUCTION_BRANDS   1000 non-null  object
dtypes: object(5), float64(4)
memory usage: 71.3 KB
```

```
DIM_PRODUCT_DATA.dtypes
```

```
PRODUCT_NAME        object
PRODUCT_ID          object
PRODUCT_LENGTH     float64
PRODUCT_WIDTH      float64
PRODUCT_HEIGHT     float64
PRODUCT_WEIGHT     float64
PRODUCT_COLOR       object
PRODUCT_MANUFACTURER  object
PRODUCTION_BRANDS   object
dtype: object
```

```
DIM_PRODUCT_DATA.describe()
```

	PRODUCT_LENGTH	PRODUCT_WIDTH	PRODUCT_HEIGHT	PRODUCT_WEIGHT
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	15.000000	10.000000	5.000000	2.000000
std	0.000000	0.000000	0.000000	0.000000
min	15.000000	10.000000	5.000000	2.000000
25%	15.000000	10.000000	5.000000	2.000000
50%	15.000000	10.000000	5.000000	2.000000
75%	15.000000	10.000000	5.000000	2.000000
max	15.000000	10.000000	5.000000	2.000000

```
DIM_PRODUCT_DATA.head(3)
```

	PRODUCT_NAME	PRODUCT_ID	PRODUCT_LENGTH	PRODUCT_WIDTH	PRODUCT_HEIGHT	PRODUCT_WEIGHT	PRODUCT_COLOR	PRODUCT_MANUFACTURER	PRODUCTION_BRANDS
0	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Ultra Ship Steel
1	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Ultra Ship Steel
2	Ultra Ship Steel Hammer	PRD-1000	15.00	10.00	5.00	2.00	Black	Ultra Ship Steel	Ultra Ship Steel

Find above values that are in Missing values and the range seems to be accurate.

Data Analysis-

```
DIM_TRANSACTION_DATA.isnull().sum()
```

```
TRANSACTION_DATE    0
DAY_OF_WEEK         0
MONTH               0
YEAR                0
dtype: int64
```

```
DIM_TRANSACTION_DATA.nunique(axis=0)
```

```
TRANSACTION_DATE    1095
DAY_OF_WEEK            7
MONTH                 12
YEAR                   2
dtype: int64
```

```
DIM_TRANSACTION_DATA.dropna().nunique(axis=0)
```

```
TRANSACTION_DATE    1095
DAY_OF_WEEK            7
MONTH                 12
YEAR                   2
dtype: int64
```

```
DIM_TRANSACTION_DATA.dropna().shape
```

```
(1095, 4)
```

```
DIM_TRANSACTION_DATA.dtypes
```

```
TRANSACTION_DATE    datetime64[ns]
DAY_OF_WEEK                  int64
MONTH                        int64
YEAR                         int64
dtype: datetime64[ns], int64(3)
```

```
DIM_TRANSACTION_DATA.dropna().dtypes
```

```
TRANSACTION_DATE    datetime64[ns]
DAY_OF_WEEK                  int64
MONTH                        int64
YEAR                         int64
dtype: datetime64[ns], int64(3)
```

```
DIM_TRANSACTION_DATA.dropna().describe()
```

	DAY_OF_WEEK	MONTH	YEAR
count	1095.000000	1095.000000	1095.000000
mean	4.000000	6.526027	2018.499543
std	2.000457	3.447968	0.500228
min	1.000000	1.000000	2018.000000
25%	2.000000	3.500000	2018.000000
50%	4.000000	6.526027	2018.499543
75%	6.000000	9.000000	2019.000000
max	7.000000	12.000000	2019.000000

```
DIM_TRANSACTION_DATA.head(3)
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
0	2018-01-01	1	1	2018
1	2018-01-02	2	1	2018
2	2018-01-03	3	1	2018

```
DIM_TRANSACTION_DATA.sample(3)
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
1095	2019-12-31	6	12	2019
1095	2019-12-31	6	12	2019
1095	2019-12-31	6	12	2019

```
DIM_TRANSACTION_DATA[DIM_TRANSACTION_DATA['TRANSACTION_DATE'] == '2019-12-31']
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
1095	2019-12-31	6	12	2019
1095	2019-12-31	6	12	2019
1095	2019-12-31	6	12	2019

```
DIM_TRANSACTION_DATA.shape
```

```
(1095, 4)
```

```
DIM_TRANSACTION_DATA.info()
```

```
<class'pandas.core.frame.DataFrame'>
RangeIndex: 1095 entries, 0 to 1094
Data columns (total 4 columns):
 TRANSACTION_DATE  1095 non-null  datetime64[ns]
 DAY_OF_WEEK       1095 non-null  int64
 MONTH             1095 non-null  int64
 YEAR              1095 non-null  int64
dtype: datetime64[ns], int64(3)
memory usage: 34.3 KB
```

```
DIM_TRANSACTION_DATA.describe()
```

	DAY_OF_WEEK	MONTH	YEAR
count	1095.000000	1095.000000	1095.000000
mean	4.000000	6.526027	2018.499543
std	2.000457	3.447968	0.500228
min	1.000000	1.000000	2018.000000
25%	2.000000	3.500000	2018.000000
50%	4.000000	6.526027	2018.499543
75%	6.000000	9.000000	2019.000000
max	7.000000	12.000000	2019.000000

```
DIM_TRANSACTION_DATA.head(3)
```

	TRANSACTION_DATE	DAY_OF_WEEK	MONTH	YEAR
0	2018-01-01	1	1	2018
1	2018-01-02	2	1	2018
2	2018-01-03	3	1	2018

Find above values that are in Missing values and the range seems to be accurate.

Data Analysis-

```
DIM_CUSTOMER_DATA.isnull().sum()
```

```
CUSTOMER_NAME    0
CUSTOMER_ID      0
CITY             0
COUNTRY          0
CONTINENT        0
dtype: int64
```

```
DIM_CUSTOMER_DATA.nunique(axis=0)
```

```
CUSTOMER_NAME    1000
CUSTOMER_ID      1000
CITY              100
COUNTRY            10
CONTINENT           4
dtype: int64
```

```
DIM_CUSTOMER_DATA.dropna().nunique(axis=0)
```

```
CUSTOMER_NAME    1000
CUSTOMER_ID      1000
CITY              100
COUNTRY            10
CONTINENT           4
dtype: int64
```

```
DIM_CUSTOMER_DATA.dropna().shape
```

```
(1000, 5)
```

```
DIM_CUSTOMER_DATA.dtypes
```

```
CUSTOMER_NAME    object
CUSTOMER_ID      object
CITY             object
COUNTRY          object
CONTINENT        object
dtype: object
```

```
DIM_CUSTOMER_DATA.dropna().dtypes
```

```
CUSTOMER_NAME    object
CUSTOMER_ID      object
CITY             object
COUNTRY          object
CONTINENT        object
dtype: object
```

```
DIM_CUSTOMER_DATA.dropna().describe()
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
count	1000	1000	1000	1000	1000
unique	1000	1000	100	10	4
top	Ms. Yvonne Hernandez	CUS-1000	Kingston	Jamaica	Americas
freq	1	1	15	150	500

```
DIM_CUSTOMER_DATA.head(3)
```

	CUSTOMER_NAME	CUSTOMER_ID	CITY	COUNTRY	CONTINENT
0	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas
1	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas
2	Mr. Ryan Ortiz	CUS-1000	Kingston	Jamaica	Americas

Customer Segmentation

Find the Number of the amount of customer in each CONTINENT and PROVINCIAL CUSTOMERS
Which group brands do they and bought the most. And what the average sales per customer segments.

```
!pip install -U kaleido
import sys, os, glob, shutil, zipfile, urllib.request, gzip
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import kaleido
import sklearn
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_segment import LinearSegmentationRegressor, LinearSegmentationRegressorCV
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR, LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score, silhouette_samples
from scipy import stats
import warnings
warnings.filterwhere('ignore')
np.random.seed(42)
np.set_printoptions(precision=20)
pd.set_option("display.max_columns", 100)
pd.set_option("display.max_rows", 100)
pd.set_option("display.width", 1000)
pd.set_option("display.precision", 2)
pd.set_option("display.float_format", lambda x: '%.2f' % x)
```

```
Collecting kaleido
  Downloading kaleido-0.2.1.b2-py2.py3-none-manylinux_2_17_x86_64.whl (4.1 MB)
     |▏                               | 10 kB 17.8 MB/s eta 0:00:01
     |▏                               | 20 kB 8.9 MB/s eta 0:00:01
     |▏                               | 30 kB 7.1 MB/s eta 0:00:01
     |▏                               | 40 kB 6.4 MB/s eta 0:00:01
     |▏                               | 50 kB 3.6 MB/s eta 0:00:01
     |▏                               | 60 kB 4.2 MB/s eta 0:00:01
     |▏                               | 70 kB 4.4 MB/s eta 0:00:01
     |▏                               | 80 kB 4.5 MB/s eta 0:00:01
     |▏                               | 90 kB 4.7 MB/s eta 0:00:01
     |▏                               | 100 kB 4.7 MB/s eta 0:00:01
     |▏                               | 110 kB 4.7 MB/s eta 0:00:01
     |▏                               | 120 kB 4.7 MB/s eta 0:00:01
     |▏                               | 130 kB 4.7 MB/s eta 0:00:01
     |▏                               | 140 kB 4.7 MB/s eta 0:00:01
     |▏                               | 150 kB 4.7 MB/s eta 0:00:01
     |▏                               | 160 kB 4.7 MB/s eta 0:00:01
     |▏                               | 170 kB 4.7 MB/s eta 0:00:01
     |▏                               | 180 kB 4.7 MB/s eta 0:00:01
     |▏                               | 190 kB 4.7 MB/s eta 0:00:01
     |▏                               | 200 kB 4.7 MB/s eta 0:00:01
     |▏                               | 210 kB 4.7 MB/s eta 0:00:01
     |▏                               | 220 kB 4.7 MB/s eta 0:00:01
     |▏                               | 230 kB 4.7 MB/s eta 0:00:01
     |▏                               | 240 kB 4.7 MB/s eta 0:00:01
     |▏                               | 250 kB 4.7 MB/s eta 0:00:01
     |▏                               | 260 kB 4.7 MB/s eta 0:00:01
     |▏                               | 270 kB 4.7 MB/s eta 0:00:01
     |▏                               | 280 kB 4.7 MB/s eta 0:00:01
     |▏                               | 290 kB 4.7 MB/s eta 0:00:01
     |▏                               | 300 kB 4.7 MB/s eta 0:00:01
     |▏                               | 310 kB 4.7 MB/s eta 0:00:01
     |▏                               | 320 kB 4.7 MB/s eta 0:00:01
     |▏                               | 330 kB 4.7 MB/s eta 0:00:01
     |▏                               | 340 kB 4.7 MB/s eta 0:00:01
     |▏                               | 350 kB 4.7 MB/s eta 0:00:01
     |▏                               | 360 kB 4.7 MB/s eta 0:00:01
     |▏                               | 370 kB 4.7 MB/s eta 0:00:01
     |▏                               | 380 kB 4.7 MB/s eta 0:00:01
     |▏                               | 390 kB 4.7 MB/s eta 0:00:01
     |▏                               | 400 kB 4.7 MB/s eta 0:00:01
     |▏                               | 410 kB 4.7 MB/s eta 0:00:01
     |▏                               | 420 kB 4.7 MB/s eta 0:00:01
     |▏                               | 430 kB 4.7 MB/s eta 0:00:01
     |▏                               | 440 kB 4.7 MB/s eta 0:00:01
     |▏                               | 450 kB 4.7 MB/s eta 0:00:01
     |▏                               | 460 kB 4.7 MB/s eta 0:00:01
     |▏                               | 470 kB 4.7 MB/s eta 0:00:01
     |▏                               | 480 kB 4.7 MB/s eta 0:00:01
     |▏                               | 490 kB 4.7 MB/s eta 0:00:01
     |▏                               | 500 kB 4.7 MB/s eta 0:00:01
     |▏                               | 510 kB 4.7 MB/s eta 0:00:01
     |▏                               | 520 kB 4.7 MB/s eta 0:00:01
     |▏                               | 530 kB 4.7 MB/s eta 0:00:01
     |▏                               | 540 kB 4.7 MB/s eta 0:00:01
     |▏                               | 550 kB 4.7 MB/s eta 0:00:01
     |▏                               | 560 kB 4.7 MB/s eta 0:00:01
     |▏                               | 570 kB 4.7 MB/s eta 0:00:01
     |▏                               | 580 kB 4.7 MB/s eta 0:00:01
     |▏                               | 590 kB 4.7 MB/s eta 0:00:01
     |▏                               | 600 kB 4.7 MB/s eta 0:00:01
     |▏                               | 610 kB 4.7 MB/s eta 0:00:01
     |▏                               | 620 kB 4.7 MB/s eta 0:00:01
     |▏                               | 630 kB 4.7 MB/s eta 0:00:01
     |▏                               | 640 kB 4.7 MB/s eta 0:00:01
     |▏                               | 650 kB 4.7 MB/s eta 0:00:01
     |▏                               | 660 kB 4.7 MB/s eta 0:00:01
     |▏                               | 670 kB 4.7 MB/s eta 0:00:01
     |▏                               | 680 kB 4.7 MB/s eta 0:00:01
     |▏                               | 690 kB 4.7 MB/s eta 0:00:01
     |▏                               | 700 kB 4.7 MB/s eta 0:00:01
     |▏                               | 710 kB 4.7 MB/s eta 0:00:01
     |▏                               | 720 kB 4.7 MB/s eta 0:00:01
     |▏                               | 730 kB 4.7 MB/s eta 0:00:01
     |▏                               | 740 kB 4.7 MB/s eta 0:00:01
     |▏                               | 750 kB 4.7 MB/s eta 0:00:01
     |▏                               | 760 kB 4.7 MB/s eta 0:00:01
     |▏                               | 770 kB 4.7 MB/s eta 0:00:01
     |▏                               | 780 kB 4.7 MB/s eta 0:00:01
     |▏                               | 790 kB 4.7 MB/s eta 0:00:01
     |▏                               | 800 kB 4.7 MB/s eta 0:00:01
     |▏                               | 810 kB 4.7 MB/s eta 0:00:01
     |▏                               | 820 kB 4.7 MB/s eta 0:00:01
     |▏                               | 830 kB 4.7 MB/s eta 0:00:01
     |▏                               | 840 kB 4.7 MB/s eta 0:00:01
     |▏                               | 850 kB 4.7 MB/s eta 0:00:01
     |▏                               | 860 kB 4.7 MB/s eta 0:00:01
     |▏                               | 870 kB 4.7 MB/s eta 0:00:01
     |▏                               | 880 kB 4.7 MB/s eta 0:00:01
     |▏                               | 890 kB 4.7 MB/s eta 0:00:01
     |▏                               | 900 kB 4.7 MB/s eta 0:00:01
     |▏                               | 910 kB 4.7 MB/s eta 0:00:01
     |▏                               | 920 kB 4.7 MB/s eta 0:00:01
     |▏                               | 930 kB 4.7 MB/s eta 0:00:01
     |▏                               | 940 kB 4.7 MB/s eta 0:00:01
     |▏                               | 950 kB 4.7 MB/s eta 0:00:01
     |▏                               | 960 kB 4.7 MB/s eta 0:00:01
     |▏                               | 970 kB 4.7 MB/s eta 0:00:01
     |▏                               | 980 kB 4.7 MB/s eta 0:00:01
     |▏                               | 990 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1000 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1010 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1020 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1030 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1040 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1050 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1060 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1070 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1080 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1090 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1100 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1110 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1120 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1130 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1140 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1150 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1160 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1170 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1180 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1190 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1200 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1210 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1220 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1230 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1240 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1250 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1260 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1270 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1280 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1290 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1300 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1310 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1320 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1330 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1340 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1350 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1360 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1370 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1380 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1390 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1400 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1410 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1420 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1430 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1440 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1450 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1460 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1470 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1480 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1490 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1500 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1510 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1520 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1530 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1540 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1550 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1560 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1570 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1580 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1590 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1600 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1610 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1620 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1630 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1640 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1650 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1660 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1670 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1680 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1690 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1700 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1710 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1720 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1730 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1740 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1750 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1760 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1770 kB 4.7 MB/s eta 0:00:01
     |▏                               | 1780 k
```

```
In [10]: Top = df[ df[ 'Customer_Segment'].isin(['Platinum'])].sort_values('Total_Sales', ascending=False)
Top.head(3)[['Customer_ID', 'Customer_Segment', 'Total_Sales', 'Total_Orders', 'Avg_Order_Value', 'Recency', 'Frequency', 'Monetary']]
Top.groupby('Customer_Segment')[['Total_Sales', 'Total_Orders', 'Avg_Order_Value', 'Recency', 'Frequency', 'Monetary']].mean().round(2)
```

From above do we see that Bonita stays same the for the only thing mend it have a pretty even customer segments.
Budget shoppers equal Bloomsbury and TOLSON PARK LLP were one near TILSON WINGLEDON GROUP LP

CONCLUSION :

1. Bonita Bangalore Bonita seem to produce better order sales.
2. Chips sales has 3 spice before Christmas and peak at Christmas Eve . Stocking up at the beginning of December and Discount on Christmas Eve could increase sales revenue.
3. TILSON PARK LLP and TOLSON PARK LLP were Bloomsbury customer. Which is where most sales are made.
4. TILSON PARK LLP and TOLSON PARK LLP have same Budget customer. And TILSON WINGLEDON GROUP has a pretty even in all
5. Bonita Bangalore Bonita seem to produce better order sales.
6. Bonita Bonita has the most sales amount of one among the Top 3 Acts accounts