```
#VIRTUAL INTERNSHIP (QUANTIUM)

AUTHOR :PRINCE ADJORKEY


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


plt.style.use('fivethirtyeight')


#Load dataset
df_trans= pd.read_csv(r"C:\Users\USER\OneDrive\Documents\pandasApp\
assignment\QVI_transaction_data.csv")
print(df_trans)
```

```
         DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0       43390          1            1000       1         5
1       43599          1            1307     348        66
2       43605          1            1343     383        61
3       43329          2            2373     974        69
4       43330          2            2426    1038       108
...       ...        ...             ...     ...       ...
264831  43533        272          272319  270088        89
264832  43325        272          272358  270154        74
264833  43410        272          272379  270187        51
264834  43461        272          272379  270188        42
264835  43365        272          272380  270189        74

                                       PROD_NAME  PROD_QTY  TOT_SALES

0         Natural Chip        Compny SeaSalt175g         2        6.0

1                       CCs Nacho Cheese    175g         3        6.3

2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9

3         Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0

4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8

...                                          ...       ...        ...

264831   Kettle Sweet Chilli And Sour Cream 175g         2       10.8

264832             Tostitos Splash Of  Lime 175g         1        4.4
```

```
264833                  Doritos Mexicana    170g         2        8.8

264834   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8

264835             Tostitos Splash Of  Lime 175g         2        8.8


[264836 rows x 8 columns]
```

```
#load dataset 2
df_behv=pd.read_csv(r"C:\Users\USER\OneDrive\Documents\pandasApp\
assignment\QVI_purchase_behaviour.csv")
print (df_behv)
```

```
       LYLTY_CARD_NBR               LIFESTAGE PREMIUM_CUSTOMER
0                1000   YOUNG SINGLES/COUPLES          Premium
1                1002   YOUNG SINGLES/COUPLES       Mainstream
2                1003          YOUNG FAMILIES           Budget
3                1004   OLDER SINGLES/COUPLES       Mainstream
4                1005  MIDAGE SINGLES/COUPLES       Mainstream
...               ...                     ...              ...
72632         2370651  MIDAGE SINGLES/COUPLES       Mainstream
72633         2370701          YOUNG FAMILIES       Mainstream
72634         2370751          YOUNG FAMILIES          Premium
72635         2370961           OLDER FAMILIES           Budget
72636         2373711   YOUNG SINGLES/COUPLES       Mainstream

[72637 rows x 3 columns]
```

```
#Data Cleaning
#Transaction data set
#find missing values
df_trans.isnull().sum()
```

```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
dtype: int64
```

```
#Look at data types
df_trans.dtypes
```

```
DATE                int64
STORE_NBR           int64
LYLTY_CARD_NBR      int64
TXN_ID              int64
```

```
PROD_NBR              int64
PROD_NAME            object
PROD_QTY              int64
TOT_SALES           float64
dtype: object
```

```python
#Change "DATE" column to datetime and "PROD_NAME" to string

df_trans['DATE']= pd.to_datetime(df_trans['DATE'])
df_trans['PROD_NAME']=df_trans['PROD_NAME'].astype('string')

#look at distribution and extreme values
df_trans.describe()
```

```
                                DATE      STORE_NBR  LYLTY_CARD_NBR  \
count                         264833  264833.000000    2.648330e+05
mean   1970-01-01 00:00:00.000043464     135.079529    1.355489e+05
min    1970-01-01 00:00:00.000043282       1.000000    1.000000e+03
25%    1970-01-01 00:00:00.000043373      70.000000    7.002100e+04
50%    1970-01-01 00:00:00.000043464     130.000000    1.303570e+05
75%    1970-01-01 00:00:00.000043555     203.000000    2.030940e+05
max    1970-01-01 00:00:00.000043646     272.000000    2.373711e+06
std                              NaN      76.784189    8.058003e+04

             TXN_ID       PROD_NBR       PROD_QTY      TOT_SALES  \
count  2.648330e+05  264833.000000  264833.000000  264833.000000
mean   1.351577e+05      56.583598       1.905812       7.299351
min    1.000000e+00       1.000000       1.000000       1.500000
25%    6.760000e+04      28.000000       2.000000       5.400000
50%    1.351370e+05      56.000000       2.000000       7.400000
75%    2.027000e+05      85.000000       2.000000       9.200000
max    2.415841e+06     114.000000       5.000000      29.500000
std    7.813305e+04      32.826498       0.343437       2.527244

         Packet Size
count  264833.000000
mean      182.425540
min        70.000000
25%       150.000000
50%       170.000000
75%       175.000000
max       380.000000
std        64.325268
```

```python
#lets investigate the highest TOT_sales value
df_trans[df_trans['TOT_SALES']==650]
```

```
Empty DataFrame
Columns: [DATE, STORE_NBR, LYLTY_CARD_NBR, TXN_ID, PROD_NBR,
PROD_NAME, PROD_QTY, TOT_SALES, Packet Size, year_month, brand_name]
Index: []
```

```python
#since the second highest TOT_SALES value is 29.5,this is clearly an
outlier so i will drop this value
df_trans.drop([69762, 69763], axis=0, inplace=True)


#check for duplicate values
df_trans[df_trans.duplicated()]

#There seems to be a single duplicated row. I will drop this.

df_trans.drop(124845, axis = 0, inplace = True)


#clean up PROD_NAME column by

#fixing spaces
for i in range(25):
    df_trans['PROD_NAME'] = df_trans['PROD_NAME'].replace(i*' ', ' ')

# removing whitespace

df_trans['PROD_NAME'] = df_trans['PROD_NAME'].str.strip()

 # correcting misspelling of Doritos brand

df_trans['PROD_NAME'] = df_trans['PROD_NAME'].str.replace('Dorito',
'Doritos')
df_trans['PROD_NAME'] = df_trans['PROD_NAME'].str.replace('Doritoss',
'Doritos')

df_trans['PROD_NAME'].value_counts().to_frame()
```

```
                                          count
PROD_NAME
Kettle Mozzarella   Basil & Pesto 175g     3304
Kettle Tortilla ChpsHny&Jlpno Chili 150g   3296
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g   3269
Tyrrells Crisps     Ched & Chives 165g     3268
Cobs Popd Sea Salt  Chips 110g             3265
...                                         ...
RRD Pc Sea Salt     165g                   1431
Woolworths Medium   Salsa 300g             1430
NCC Sour Cream &    Garden Chives 175g     1419
French Fries Potato Chips 175g             1418
WW Crinkle Cut      Original 175g          1410

[114 rows x 1 columns]
```

```python
#BEHAVOURIAL DATASET
# look at data types

df_behv.dtypes
```

```
LYLTY_CARD_NBR         int64
LIFESTAGE             object
PREMIUM_CUSTOMER      object
dtype: object
```

```
# change "LIFESTAGE" and "PREMIUM_CUSTOMER" variables to 'string' type

df_behv['LIFESTAGE'] = df_behv['LIFESTAGE'].astype('string')
df_behv['PREMIUM_CUSTOMER'] =
df_behv['PREMIUM_CUSTOMER'].astype('string')

# check for duplicates

df_behv[df_behv.duplicated()]
```

```
Empty DataFrame
Columns: [LYLTY_CARD_NBR, LIFESTAGE, PREMIUM_CUSTOMER]
Index: []
```

```
#There are no duplicate

#DATE ENGINEERING
# create feature to capture 'packet size'

df_trans['Packet Size'] =
df_trans['PROD_NAME'].astype('str').str.extractall(r'(\
d+)').unstack().fillna('').sum(axis=1).astype(int)

# create feature to capture year and month

df_trans['year_month'] = df_trans['DATE'].dt.to_period('M')

# create brand name feature

df_trans['brand_name'] = df_trans['PROD_NAME'].str.split(' ').str[0]

merged_df = pd.merge(df_trans, df_behv, how = 'inner', on =
'LYLTY_CARD_NBR')
merged_df.head()
```

```
                           DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID
PROD_NBR  \
0 1970-01-01 00:00:00.000043390          1            1000       1
5
1 1970-01-01 00:00:00.000043599          1            1307     348
66
2 1970-01-01 00:00:00.000043605          1            1343     383
61
3 1970-01-01 00:00:00.000043329          2            2373     974
69
4 1970-01-01 00:00:00.000043330          2            2426    1038
108
```

```
                                  PROD_NAME  PROD_QTY  TOT_SALES
Packet Size  \
0    Natural Chip        Compny SeaSalt175g         2        6.0
175
1                   CCs Nacho Cheese    175g         3        6.3
175
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
170
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
175
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
150

  year_month brand_name               LIFESTAGE PREMIUM_CUSTOMER
0    1970-01    Natural   YOUNG SINGLES/COUPLES          Premium
1    1970-01        CCs  MIDAGE SINGLES/COUPLES           Budget
2    1970-01     Smiths  MIDAGE SINGLES/COUPLES           Budget
3    1970-01     Smiths  MIDAGE SINGLES/COUPLES           Budget
4    1970-01     Kettle  MIDAGE SINGLES/COUPLES           Budget

#EDA
# Univariate Analysis - Purchase Frequency
#We can find purchase frequency by looking at the number of
transactions segmented by different variables


import matplotlib.pyplot as plt

def uni_plot(feature, color, suptitle, title):
        a = merged_df[feature].value_counts().to_frame()
        a.reset_index(inplace = True)
        a.rename({'index':str(feature), str(feature):'Frequency'},
                   axis = 1, inplace = True)
        a.sort_values(by = 'Frequency', ascending = True, inplace =
True)

        if len(a) > 10:
            a = a.iloc[len(a)-10:len(a)]

        if  merged_df[feature].dtype == int:
            merged_df[feature] = merged_df[feature].astype('string')


        plt.figure(figsize = (10, 6))
        plt.barh(a[feature], a['Frequency'], color = color, alpha =
0.7)
        plt.xlabel('Number of Transactions', size = 11)
        plt.xticks(size = 9)
        plt.ylabel(str(feature), size = 11)
```

```
        plt.yticks(size = 9)
        plt.title(title, size = 14)
        plt.suptitle(suptitle)

        plt.show()



uni_plot('LIFESTAGE', '#CF9893', 'Older Customers Purchase Chips Most
Frequently',
         'Distribution of Transactions by Customers\' Lifestage')




# let's also quickly find out how many customers are in each stage

merged_df.LYLTY_CARD_NBR.nunique() # yields 72636
```

```
72636
```

```
#And what proportion of the total customers do customers from the
above segment make up?


# and what proportion of the total customers do customers from the
above segments

    from pywaffle import Waffle


    cust_count = []
    lifestages = list(set(merged_df.LIFESTAGE))

    for i in lifestages:
        cust_count.append(merged_df[merged_df['LIFESTAGE'] == i]
['LYLTY_CARD_NBR'].nunique())

    cust_per_lifestage = pd.DataFrame({'Lifestage': lifestages,
                                       'num_customers':cust_count})




    value = {'New Families':2549, 'Young Families':9178,
             'Midage Singles/Couples':7275, 'Retirees':14805,
             'Older Families':9779, 'Young Singles/Couples':14441,
             'Older Singles/Couples':14609}

    value = dict(sorted(value.items(), key=lambda x:x[1]))
```

```
    # Waffle chart
    plt.figure(figsize = (12, 8),
        FigureClass = Waffle,
        rows = 10,
        columns = 10,
        values = value,
        vertical = False,
        facecolor = 'white',
        icons = 'person-dress',
        icon_legend = 'True',
        #title = {'label':''}
        legend = {
                  'loc':'upper left',
                  'bbox_to_anchor':(1, 1)})

    plt.suptitle('Young Singles/Couples, Retirees and Older 
Singles/Couples Account for 3 out of every 5 Customers', size = 16)
    plt.title('Distribution of each segment of customers if there were 
only 100 customers', size = 12)




#PRODUCT NAME
uni_plot('brand_name', '#BC7C9C', '"Kettle" and "Smiths" Brand Chips 
are Purchased with the Greatest Frequency',
         'Top 10 Most Frequently Purchased Products (Number of 
Transactions)')

#Takeaways:
#Older customers, whether they are married, single, retired or with 
children purchase chips with the greatest frequency. This suggests 
these are valuable segments to our client. However, we must note that 
this is not the same as saying “older customers buy the greatest 
amount of chip products”. We will explore this particular question 
later.

#The “Kettle” and “Smiths” brands saw the greatest number of purchases 
by transaction, not by number of chip products sold. “Kettle” chips 
are a clear outlier because there is a substantial gap between it and 
2nd-placed “Smiths” chips. “Pringles” and “Doritos” round out the top 
4 with the rest of the pack traili

#CUSTOMER CATEGORY
uni_plot('PREMIUM_CUSTOMER', '#7A5980',
             'The Greatest Proportion of Transactions Involve the 
"Mainstream" Customer Category',
```

```
              'Number of Transactions by Customer Category')



#EVOLUTION OF NUMBER OFTRANSACTION THROUGH TIME

transactions_per_month =
merged_df.year_month.value_counts().to_frame()

    transactions_per_month.reset_index(inplace = True)

    transactions_per_month.sort_values(by = 'index', inplace = True)

    transactions_per_month['index'] =
transactions_per_month['index'].astype('string')

    transactions_per_month.plot(x = 'index', y = 'year_month',
                                kind = 'line', figsize = (10, 6),
                                color = '#A96DA3', alpha = 0.7)

    plt.xlabel('Month', size = 11)
    plt.xticks(size = 9)
    plt.ylabel('Number of Transactions', size = 11)
    plt.yticks(size = 9)
    plt.title('Number of Transactions by Month', size = 14)
    plt.suptitle('Vast Majority of Transactions Occur Between 07-2018
and 06-2019')

    plt.legend().remove()



#TOTAL QUANTITIES OF PRODUCT SOLS(BEST SELLING PRODUCT
def aggregator(x, y, color, suptitle, title):
        df = merged_df.groupby(x)
[y].sum().to_frame().reset_index().sort_values(by = y, ascending =
True)

        if len(df) > 10:
            df = df.iloc[len(df) - 10:len(df)]

        plt.figure(figsize = (10, 6))
        plt.barh(df[x], df[y], color = color, alpha = 0.7)
        plt.xlabel(y, size = 11)
        plt.xticks(size = 9)
        plt.ylabel(x, size = 11)
        plt.yticks(size = 9)
        plt.title(title, size = 14)
        plt.suptitle(suptitle)
```

```
        plt.show()

#TOTAL PRODUCT SOLD BY BRAND

aggregator('brand_name', 'TOT_SALES', '#3B3B58',
               '"Kettle" chips dominate the competition and 
significantly outsold competitors',
               'Top 10 Products By Total Units Sold ')
 #The product with the greatest number of units sold belongs to the 
“Kettle” and “Doritos” brands. With our previous findings, we can 
conclude that not only are “Kettle” and “Doritos” brands chips bought 
most often, they are also bought at the greatest quantities (number of 
units) overall

#For example, “Kettle” chips were bought in over 1 million 
transactions with almost 4 million units bought. So we can also say 
that the average number of “Kettle” chips bought per transaction was 
about 4

#As before, “Smiths” and “Pringles” round out the top 4. 
Interestingly, however, we see that while “Doritos” chips outsold 
“Smiths” chips in terms of total quantity, Smiths chips were sold more 
often (by a slight amount). So, on average, customers bought “Smiths” 
chips more often but in smaller quantities compared to “Doritos” chips


 #TOTAL PRODUCT SOLD BY LIFESTAGE
 aggregator('LIFESTAGE', 'TOT_SALES', '#CF9893',
               'Older Customers are the Biggest Purchasers of Chips',
               'Total Product Sales by Customer Lifestage')
#We see that not only are Older Singles/Couples, Retirees and Older 
Families the customer segments that purchase chip products with the 
greatest frequency, they also purchase the largest quantities of 
chips.

#These findings point to the idea that these segments are very 
valuable to our client and that the client should continue to target 
and promote these segments in their supermarket’s strategic plan



#TOTAL PRODUCT SALES BY CUSTOMER CATEGORY

aggregator('PREMIUM_CUSTOMER', 'TOT_SALES', '#BC7C9C',
               'Mainstream Category Customers are the Biggest 
Purchasers of Chips',
               'Total Product Sales by Customer Category')

#We see a similar distribution here as in the “Customer Category vs 
Number of Transactions” visualisation.
```

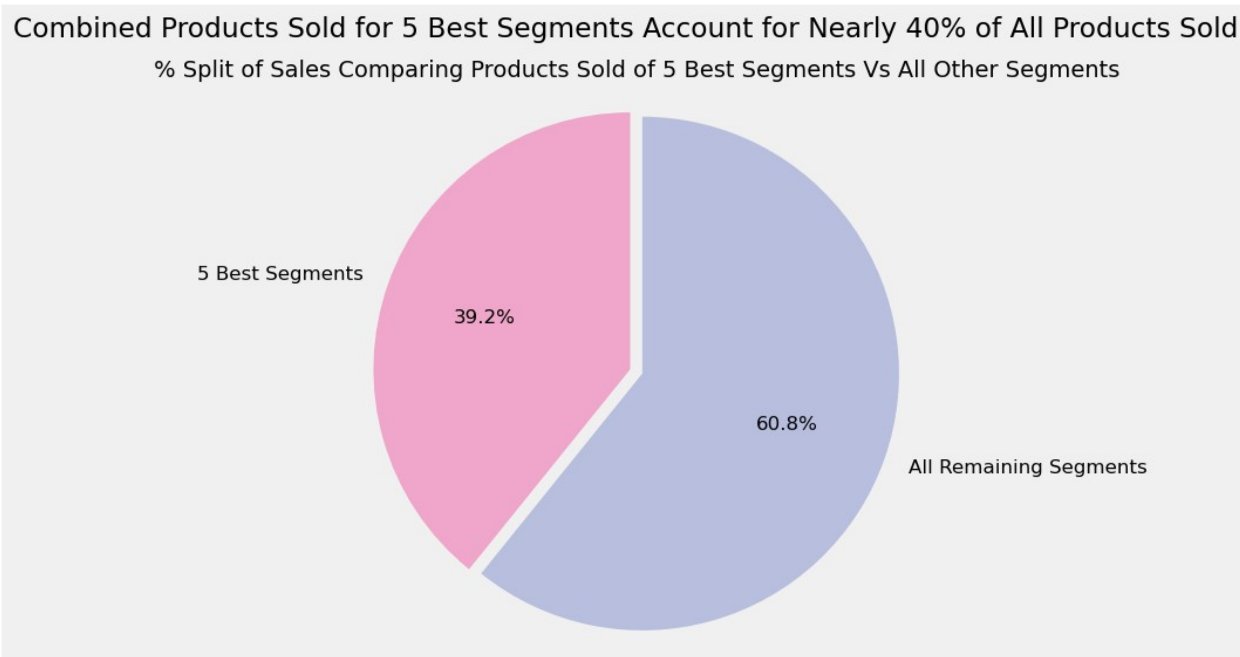
```
#At first glance, we may think that Mainstream customers are the most
valuable to our client because they buy the most products and do so
with the greatest frequency. However, this would be an erroneous
inference. This is because we have no data on the prices of the
products that these customers buy, so we do not know what proportion
of total revenue they’re responsible for.

#In other words, it is possible (and even likely) that the Premium
customers are responsible for the greatest share of revenue for our
client. Premium customers have that category name assigned to them for
a reason; they likely buy more expensive chips. And finally, it is
important to recall that one of the most basic laws of economics
states that price is negatively correlated with quantity demanded, so
the visualisation above is quite in line with expectations




#Total Product Sales by Packet Size



merged_df.groupby('Packet Size')
['TOT_SALES'].sum().to_frame().reset_index().plot(kind = 'bar',

x = 'Packet Size',

y = 'TOT_SALES',

figsize = (10, 6),

color = '#7A5980',

alpha = 0.7)
    plt.xlabel('Packet Size (Grams)', size = 11)
    plt.ylabel("Total Number of Chip Products Sold", size = 11)
    plt.xticks(size = 9)
    plt.yticks(size = 9)
    plt.title('Total Chip Products Sold By Packet Size in grams', size
= 14)
    plt.suptitle('Customers Overwhelmingly Prefer 175g and 150g Packet
Sizes')
    plt.legend().remove()

175g and 150g, packets that can be considered “medium-sized”, proved
to be the most popular with customers.

Interestingly, we see a segment of customers that prefer larger-sized
packets as well, ranging from 270g to 380g.
```

```
Below we see that for the subset of customers who buy these larger
packets, the best-selling brand is “Smiths”, not “Kettle”. Recall that
“Kettle” was the best-selling brand overall by a tremendous margin
overall, but it is nowhere to be found here. We can conclude that
either “Kettle” does not offer large packet sizes or if they do,
customers do not prefer larger sizes for this brand

merged_df[merged_df['Packet Size'].isin([270, 300, 330,
380])].groupby('brand_name')['TOT_SALES'].sum().sort_values(ascending
= False)

# the "Old" value refers to "Old El Palso Salsa Dip". It is not a
chips brand so we exclude it

merged_df[merged_df['brand_name'].str.contains('Old')]

                                    DATE  STORE_NBR  LYLTY_CARD_NBR
TXN_ID  \
5       1970-01-01 00:00:00.000043604          4            4074
2982
25      1970-01-01 00:00:00.000043600         39           39144
35506
71      1970-01-01 00:00:00.000043327         94           94233
93956
87      1970-01-01 00:00:00.000043327        116          116184
120270
114     1970-01-01 00:00:00.000043328        157          157185
159562
...                                  ...        ...             ...      ..
.
264591 1970-01-01 00:00:00.000043539        261          261323
261068
264604 1970-01-01 00:00:00.000043411        262          262061
261665
264612 1970-01-01 00:00:00.000043335        262          262084
261793
264625 1970-01-01 00:00:00.000043466        264          264165
262926
264672 1970-01-01 00:00:00.000043575        265          265103
263419

        PROD_NBR                                  PROD_NAME
PROD_QTY  \
5             57  Old El Paso Salsa   Dip Tomato Mild 300g         1

25            57  Old El Paso Salsa   Dip Tomato Mild 300g         1

71            65  Old El Paso Salsa   Dip Chnky Tom Ht300g         1

87            59   Old El Paso Salsa   Dip Tomato Med 300g         1
```

```
114           59   Old El Paso Salsa   Dip Tomato Med 300g         2

...          ...                                     ...       ...

264591        65  Old El Paso Salsa   Dip Chnky Tom Ht300g         2

264604        65  Old El Paso Salsa   Dip Chnky Tom Ht300g         2

264612        57  Old El Paso Salsa   Dip Tomato Mild 300g         2

264625        65  Old El Paso Salsa   Dip Chnky Tom Ht300g         2

264672        59   Old El Paso Salsa   Dip Tomato Med 300g         1


        TOT_SALES  Packet Size year_month brand_name
LIFESTAGE  \
5             5.1          300    1970-01        Old  MIDAGE
SINGLES/COUPLES
25            5.1          300    1970-01        Old  MIDAGE
SINGLES/COUPLES
71            5.1          300    1970-01        Old  MIDAGE
SINGLES/COUPLES
87            5.1          300    1970-01        Old  MIDAGE
SINGLES/COUPLES
114          10.2          300    1970-01        Old  MIDAGE
SINGLES/COUPLES
...           ...          ...        ...        ...
...
264591       10.2          300    1970-01        Old   YOUNG
SINGLES/COUPLES
264604       10.2          300    1970-01        Old   YOUNG
SINGLES/COUPLES
264612       10.2          300    1970-01        Old   YOUNG
SINGLES/COUPLES
264625       10.2          300    1970-01        Old   YOUNG
SINGLES/COUPLES
264672        5.1          300    1970-01        Old   YOUNG
SINGLES/COUPLES

       PREMIUM_CUSTOMER
5                Budget
25               Budget
71               Budget
87               Budget
114              Budget
...                 ...
264591          Premium
264604          Premium
264612          Premium
```

```
264625          Premium
264672          Premium

[9324 rows x 13 columns]

# Most Valuable Segments by Total Products Sold
sales_best_segments = (
    merged_df
    .groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES']
    .sum()
    .to_frame()
    .sort_values(by='TOT_SALES', ascending=False)
)

sales_best_segments.head()


                                          TOT_SALES
LIFESTAGE              PREMIUM_CUSTOMER
OLDER FAMILIES         Budget             168363.25
YOUNG SINGLES/COUPLES Mainstream          157621.60
RETIREES               Mainstream         155677.05
YOUNG FAMILIES         Budget             139345.85
OLDER SINGLES/COUPLES Budget              136769.80

#MOST VALUABLE SEGMENT SHARE OF PRODUCT SALES
sum_sales_best_segments = sales_best_segments.iloc[:5].sum().sum()

labels = ['5 Best Segments', 'All Remaining Segments']
sizes = [sum_sales_best_segments, sales_best_segments.TOT_SALES.sum()
- sum_sales_best_segments]
explode = (0.05, 0)
colors = ['#f0a6ca', '#b8bedd']

fig1, ax1 = plt.subplots(figsize=(6, 6))

ax1.pie(
    sizes,
    explode=explode,
    labels=labels,
    autopct='%1.1f%%',
    shadow=False,
    startangle=90,
    textprops={'fontsize': '12'},
    colors=colors
)

ax1.axis('equal')
plt.title('% Split of Sales Comparing Products Sold of 5 Best Segments
Vs All Other Segments', size=14)
plt.suptitle('Combined Products Sold for 5 Best Segments Account for
```

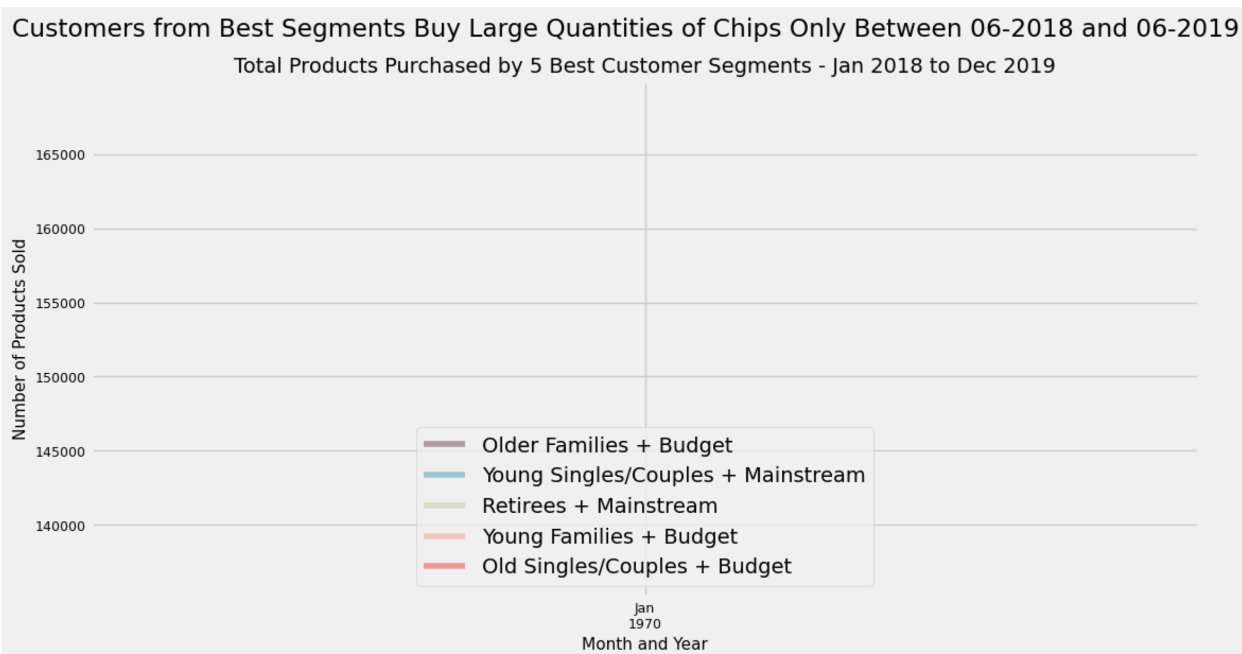
```
Nearly 40% of All Products Sold')
# plt.tight_layout()
plt.show()
```




```
#EVOLUTION SPENDING PATTERN OVERTIME OF 5BEST SEGMENT

def segment_sales(lifestage, category, title):
    df = merged_df.loc[
        (merged_df['LIFESTAGE'] == lifestage) &
        (merged_df['PREMIUM_CUSTOMER'] == category)
    ]

    df = df.groupby('year_month')['TOT_SALES'].sum().reset_index()
    df.rename(columns={'TOT_SALES': title}, inplace=True)
    return df

# Get data for each segment
older_fam_budget_sales = segment_sales('OLDER FAMILIES', 'Budget',
'old_fam_budget_sales')
young_single_mainstream_sales = segment_sales('YOUNG SINGLES/COUPLES',
'Mainstream', 'young_single_mainstream_sales')
```

```
retirees_mainstream_sales = segment_sales('RETIREES', 'Mainstream',
'retirees_mainstream_sales')
young_families_budget_sales = segment_sales('YOUNG FAMILIES',
'Budget', 'youngFam_budget_sales')
older_couples_budget_sales = segment_sales('OLDER SINGLES/COUPLES',
'Budget', 'old_sin_couple_sales')

# Merge them all on year_month
best_segments_sales =
older_fam_budget_sales.merge(young_single_mainstream_sales,
on='year_month') \
    .merge(retirees_mainstream_sales, on='year_month') \
    .merge(young_families_budget_sales, on='year_month') \
    .merge(older_couples_budget_sales, on='year_month')

# Plotting
best_segments_sales.plot(
    kind='line',
    x='year_month',
    y=[
        'old_fam_budget_sales',
        'young_single_mainstream_sales',
        'retirees_mainstream_sales',
        'youngFam_budget_sales',
        'old_sin_couple_sales'
    ],
    alpha=0.7,
    figsize=(12, 6),
    color=['#987284', '#75B9BE', '#D0D6B5', '#F9B5AC', '#EE7674']
)

plt.legend(
    loc='lower center',
    labels=[
        'Older Families + Budget',
        'Young Singles/Couples + Mainstream',
        'Retirees + Mainstream',
        'Young Families + Budget',
        'Old Singles/Couples + Budget'
    ]
)
plt.xlabel('Month and Year', size=11)
plt.ylabel('Number of Products Sold', size=11)
plt.xticks(size=9)
plt.yticks(size=9)
plt.suptitle('Customers from Best Segments Buy Large Quantities of
Chips Only Between 06-2018 and 06-2019')
plt.title('Total Products Purchased by 5 Best Customer Segments - Jan
2018 to Dec 2019', size=14)
plt.show()
```

```
C:\Users\USER\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\
core.py:1561: UserWarning: Attempting to set identical low and high
xlims makes transformation singular; automatically expanding.
  ax.set_xlim(left, right)
C:\Users\USER\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\
core.py:1561: UserWarning: Attempting to set identical low and high
xlims makes transformation singular; automatically expanding.
  ax.set_xlim(left, right)
C:\Users\USER\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\
core.py:1561: UserWarning: Attempting to set identical low and high
xlims makes transformation singular; automatically expanding.
  ax.set_xlim(left, right)
C:\Users\USER\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\
core.py:1561: UserWarning: Attempting to set identical low and high
xlims makes transformation singular; automatically expanding.
  ax.set_xlim(left, right)
C:\Users\USER\anaconda3\Lib\site-packages\pandas\plotting\_matplotlib\
core.py:1561: UserWarning: Attempting to set identical low and high
xlims makes transformation singular; automatically expanding.
  ax.set_xlim(left, right)
```




```
#Average Frequency of Chip Purchases for Most Valuable Segments
from datetime import timedelta

# Function to calculate average days between purchases for a customer
segment
def freq_purchase(lifestage, category):
    df = merged_df[(merged_df['LIFESTAGE'] == lifestage) &
                   (merged_df['PREMIUM_CUSTOMER'] == category)].copy()
```

```python
    df['DATE'] = pd.to_datetime(df['DATE'])  # Ensure it's datetime
    df = df.sort_values(by=['LYLTY_CARD_NBR', 'DATE'])

    # Calculate time between purchases
    df['prev_date'] = df.groupby('LYLTY_CARD_NBR')['DATE'].shift()
    df['days_diff'] = (df['DATE'] - df['prev_date']).dt.days

    # Average days per customer (excluding first purchase = NaN)
    avg_days = df.groupby('LYLTY_CARD_NBR')['days_diff'].mean().mean()

    return avg_days

# Get average days for each customer segment
avg_days_cust_segment = [
    freq_purchase('OLDER FAMILIES', 'Budget'),
    freq_purchase('YOUNG SINGLES/COUPLES', 'Mainstream'),
    freq_purchase('RETIREES', 'Mainstream'),
    freq_purchase('YOUNG FAMILIES', 'Budget'),
    freq_purchase('OLDER SINGLES/COUPLES', 'Budget')
]

# Build a DataFrame to display or plot
avg_days_df = pd.DataFrame({
    'Segments': [
        'Older Families + Budget',
        'Younger Singles/Couples + Mainstream',
        'Retirees + Mainstream',
        'Young Families + Budget',
        'Older Singles/Couples + Budget'
    ],
    'Average Number of Days Between Purchases': avg_days_cust_segment
})

   # Optional: Display
print(avg_days_df)
```

```
                               Segments  \
0               Older Families + Budget
1  Younger Singles/Couples + Mainstream
2                 Retirees + Mainstream
3               Young Families + Budget
4        Older Singles/Couples + Budget

   Average Number of Days Between Purchases
0                                       0.0
1                                       0.0
2                                       0.0
```

```
3                                                       0.0
4                                                       0.0
```

```
#Conclusions

#The “best” segments for our client in the context of their chip-
purchasing behaviour when it comes to both frequency and total
quantity bought are older customers. In particular, Older
Singles/Couples + Budget, Retirees + Mainstream and Older Families +
Budget, all of whom also belong to the “Budget” category, all rank
near the top of these measures.

#However, if we look at the average quantity bought by customers in
each segment, then the unifying factor behind the best segments is
families and the budget category. This is because Older Families +
Budget and Young Families + Budget rank 1st and 2nd in both the
frequency of purchasing and in the average qty of chips bought per
customer.

#Older And Young Families + Budget make a trip to the supermarket to
buy chips approximately every 2.5 months, which is by far the greatest
frequency. They buy 36 and 34.7 chip products per customer,
respectively, over the timeframe of this dataset. Again, this is
ranked 1st and 2nd in this measure.

#Theoretically, this makes sense. Chips products are usually more
popular with children and teens. And of course, if a customer has a
family, then they are likely to buy a greater quantity of chips
because they may be buying it not only for themselves, but for their
family members too.

#Along with Young Singles/Couples + Mainstream customers, the
aforementioned segments account for 2 out of every 5 units of chips
sold.

#If we pivot and look at the brand and product-centric inferences, we
find that Kettle and Smiths chips dominate the competition by both
frequency of purchase and number of units sold. Kettle in particular
is a clear outlier here with significant leads over 2nd-placed Smiths
in both measures.

#Interestingly, when it comes to packet sizes, Kettle does not seem to
offer larger packet sizes (those including and above 270g), so Smiths
is the most popular brand here. Generally, mid-sized packets (150g,
170g and 175g) are the most popular with customers by far. However,
there is a set of customers that does prefer the aforementioned larger
packets.

#Lastly, there is something peculiar occurring with the purchasing
behaviour with respect to time. Something caused total purchases to
```

```
increase tremendously in June 2018. This stayed at a high level until
July 2019, when purchases crashed and returned to pre-June 2019. I am
unsure regarding why this occurred. Maybe new supermarkets were opened
which increased sales, but that would not explain why sales crashed
again in July 2019 (unless these supermarkets were closed in that
month). So this warrants further investigation.

#Overall, my recommendation to our client is to focus on the segments
where customers have families and are in the budget category. Brand-
wise, it would be a good idea to promote Kettle, Doritos and Smiths
chips overall, and Smiths and Doritos chips in the larger packet sizes
```