"Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plot
import seaborn as sns

#load the data for purchase
df_purchase =
pd.read_csv('/Users/punarva/Desktop/Projects/Quantium/QVI_purchase_beh
aviour.csv')

df_purchase.head()
```

```
   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
```

```
df_purchase.tail()
```

```
       LYLTY_CARD_NBR               LIFESTAGE PREMIUM_CUSTOMER
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
```

```
df_purchase.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72637 entries, 0 to 72636
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   LYLTY_CARD_NBR    72637 non-null  int64
 1   LIFESTAGE         72637 non-null  object
 2   PREMIUM_CUSTOMER  72637 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.7+ MB
```

```
#There are no null values in the table

df_purchase['LIFESTAGE'].unique()
```

```
array(['YOUNG SINGLES/COUPLES', 'YOUNG FAMILIES', 'OLDER
SINGLES/COUPLES',
```

```
       'MIDAGE SINGLES/COUPLES', 'NEW FAMILIES', 'OLDER FAMILIES',
       'RETIREES'], dtype=object)
```

```
df_purchase['LIFESTAGE'].value_counts()
```

```
LIFESTAGE
RETIREES                14805
OLDER SINGLES/COUPLES   14609
YOUNG SINGLES/COUPLES   14441
OLDER FAMILIES           9780
YOUNG FAMILIES           9178
MIDAGE SINGLES/COUPLES   7275
NEW FAMILIES             2549
Name: count, dtype: int64
```

```
df_purchase['PREMIUM_CUSTOMER'].unique()
```

```
array(['Premium', 'Mainstream', 'Budget'], dtype=object)
```

```
df_purchase['PREMIUM_CUSTOMER'].value_counts()
```

```
PREMIUM_CUSTOMER
Mainstream    29245
Budget        24470
Premium       18922
Name: count, dtype: int64
```

```
#7 Unique values in lifestage and 3 in premium customer
```

```
#checkig for duplicates
df_purchase.duplicated().any()
```

```
False
```

```
#loading transaction data set
df_transaction =
pd.read_excel('/Users/punarva/Desktop/Projects/Quantium/QVI_transactio
n_data.xlsx')
```

```
df_transaction.head()
```

```
    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

                                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
```

```
df_transaction.tail()
```

```
         DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
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
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
```

```
df_transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            264836 non-null  int64
 1   STORE_NBR       264836 non-null  int64
 2   LYLTY_CARD_NBR  264836 non-null  int64
 3   TXN_ID          264836 non-null  int64
 4   PROD_NBR        264836 non-null  int64
 5   PROD_NAME       264836 non-null  object
 6   PROD_QTY        264836 non-null  int64
 7   TOT_SALES       264836 non-null  float64
dtypes: float64(1), int64(6), object(1)
memory usage: 16.2+ MB
```

```
df_transaction.columns
```

```
Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES'],
      dtype='object')
```

```
df_transaction.shape
```

```
(264836, 8)
```

```
df_transaction.describe()
```

```
                DATE     STORE_NBR  LYLTY_CARD_NBR        TXN_ID  \
count  264836.000000  264836.00000    2.648360e+05  2.648360e+05
mean    43464.036260     135.08011    1.355495e+05  1.351583e+05
```

```
std       105.389282      76.78418    8.057998e+04  7.813303e+04
min     43282.000000       1.00000    1.000000e+03  1.000000e+00
25%     43373.000000      70.00000    7.002100e+04  6.760150e+04
50%     43464.000000     130.00000    1.303575e+05  1.351375e+05
75%     43555.000000     203.00000    2.030942e+05  2.027012e+05
max     43646.000000     272.00000    2.373711e+06  2.415841e+06
```

```
            PROD_NBR       PROD_QTY      TOT_SALES
count  264836.000000  264836.000000  264836.000000
mean       56.583157       1.907309       7.304200
std        32.826638       0.643654       3.083226
min         1.000000       1.000000       1.500000
25%        28.000000       2.000000       5.400000
50%        56.000000       2.000000       7.400000
75%        85.000000       2.000000       9.200000
max       114.000000     200.000000     650.000000
```

```
df_transaction['PROD_NAME'].unique()
```

```
array(['Natural Chip        Compny SeaSalt175g',
       'CCs Nacho Cheese    175g',
       'Smiths Crinkle Cut  Chips Chicken 170g',
       'Smiths Chip Thinly  S/Cream&Onion 175g',
       'Kettle Tortilla ChpsHny&Jlpno Chili 150g',
       'Old El Paso Salsa   Dip Tomato Mild 300g',
       'Smiths Crinkle Chips Salt & Vinegar 330g',
       'Grain Waves         Sweet Chilli 210g',
       'Doritos Corn Chip Mexican Jalapeno 150g',
       'Grain Waves Sour    Cream&Chives 210G',
       'Kettle Sensations   Siracha Lime 150g',
       'Twisties Cheese     270g', 'WW Crinkle Cut      Chicken 175g',
       'Thins Chips Light&  Tangy 175g', 'CCs Original 175g',
       'Burger Rings 220g', 'NCC Sour Cream &    Garden Chives 175g',
       'Doritos Corn Chip Southern Chicken 150g',
       'Cheezels Cheese Box 125g', 'Smiths Crinkle      Original
330g',
       'Infzns Crn Crnchers Tangy Gcamole 110g',
       'Kettle Sea Salt     And Vinegar 175g',
       'Smiths Chip Thinly  Cut Original 175g', 'Kettle Original
175g',
       'Red Rock Deli Thai  Chilli&Lime 150g',
       'Pringles Sthrn FriedChicken 134g', 'Pringles Sweet&Spcy BBQ
134g',
       'Red Rock Deli SR    Salsa & Mzzrlla 150g',
       'Thins Chips         Originl saltd 175g',
       'Red Rock Deli Sp    Salt & Truffle 150G',
       'Smiths Thinly       Swt Chli&S/Cream175G', 'Kettle Chilli
175g',
       'Doritos Mexicana    170g',
       'Smiths Crinkle Cut  French OnionDip 150g',
```

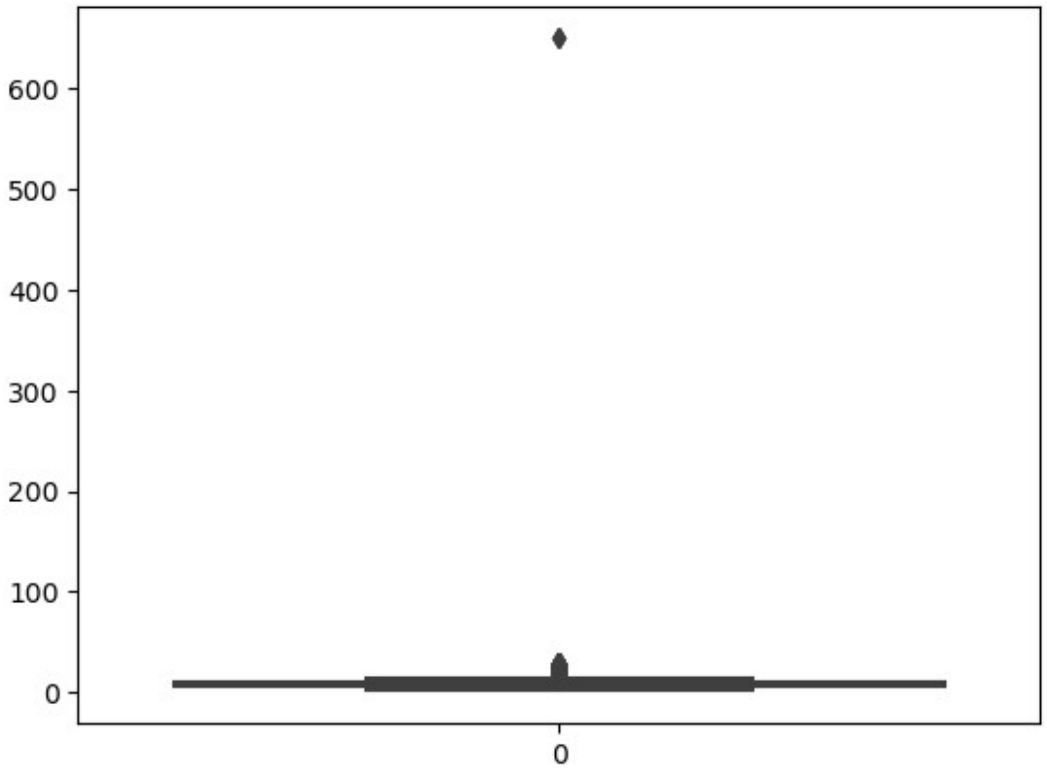
```
       'Natural ChipCo      Hony Soy Chckn175g',
       'Dorito Corn Chp     Supreme 380g', 'Twisties Chicken270g',
       'Smiths Thinly Cut   Roast Chicken 175g',
       'Smiths Crinkle Cut  Tomato Salsa 150g',
       'Kettle Mozzarella   Basil & Pesto 175g',
       'Infuzions Thai SweetChili PotatoMix 110g',
       'Kettle Sensations   Camembert & Fig 150g',
       'Smith Crinkle Cut   Mac N Cheese 150g',
       'Kettle Honey Soy    Chicken 175g',
       'Thins Chips Seasonedchicken 175g',
       'Smiths Crinkle Cut  Salt & Vinegar 170g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'Doritos Salsa       Medium 300g', 'Kettle 135g Swt Pot Sea
Salt',
       'Pringles SourCream  Onion 134g',
       'Doritos Corn Chips  Original 170g',
       'Twisties Cheese     Burger 250g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
       'Woolworths Mild     Salsa 300g',
       'Natural Chip Co     Tmato Hrb&Spce 175g',
       'Smiths Crinkle Cut  Chips Original 170g',
       'Cobs Popd Sea Salt  Chips 110g',
       'Smiths Crinkle Cut  Chips Chs&Onion170g',
       'French Fries Potato Chips 175g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
       'Doritos Corn Chips  Cheese Supreme 170g',
       'Pringles Original   Crisps 134g',
       'RRD Chilli&         Coconut 150g',
       'WW Original Corn    Chips 200g',
       'Thins Potato Chips  Hot & Spicy 175g',
       'Cobs Popd Sour Crm  &Chives Chips 110g',
       'Smiths Crnkle Chip  Orgnl Big Bag 380g',
       'Doritos Corn Chips  Nacho Cheese 170g',
       'Kettle Sensations   BBQ&Maple 150g',
       'WW D/Style Chip     Sea Salt 200g',
       'Pringles Chicken    Salt Crips 134g',
       'WW Original Stacked Chips 160g',
       'Smiths Chip Thinly  CutSalt/Vinegr175g', 'Cheezels Cheese
330g',
       'Tostitos Lightly    Salted 175g',
       'Thins Chips Salt &  Vinegar 175g',
       'Smiths Crinkle Cut  Chips Barbecue 170g', 'Cheetos Puffs
165g',
       'RRD Sweet Chilli &  Sour Cream 165g',
       'WW Crinkle Cut      Original 175g',
```

```
       'Tostitos Splash Of  Lime 175g', 'Woolworths Medium   Salsa
300g',
       'Kettle Tortilla ChpsBtroot&Ricotta 150g',
       'CCs Tasty Cheese    175g', 'Woolworths Cheese   Rings 190g',
       'Tostitos Smoked     Chipotle 175g', 'Pringles Barbeque
134g',
       'WW Supreme Cheese   Corn Chips 200g',
       'Pringles Mystery    Flavour 134g',
       'Tyrrells Crisps     Ched & Chives 165g',
       'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
       'Cheetos Chs & Bacon Balls 190g', 'Pringles Slt Vingar 134g',
       'Infuzions SourCream&Herbs Veg Strws 110g',
       'Kettle Tortilla ChpsFeta&Garlic 150g',
       'Infuzions Mango     Chutny Papadums 70g',
       'RRD Steak &         Chimuchurri 150g',
       'RRD Honey Soy       Chicken 165g',
       'Sunbites Whlegrn    Crisps Frch/Onin 90g',
       'RRD Salt & Vinegar  165g', 'Doritos Cheese      Supreme 330g',
       'Smiths Crinkle Cut  Snag&Sauce 150g',
       'WW Sour Cream &OnionStacked Chips 160g',
       'RRD Lime & Pepper   165g',
       'Natural ChipCo Sea  Salt & Vinegr 175g',
       'Red Rock Deli Chikn&Garlic Aioli 150g',
       'RRD SR Slow Rst     Pork Belly 150g', 'RRD Pc Sea Salt
165g',
       'Smith Crinkle Cut   Bolognese 150g', 'Doritos Salsa Mild
300g'],
      dtype=object)
```

Find Outliers

```
sns.boxplot(df_transaction.TOT_SALES)
```

```
<Axes: >
```




```
df_transaction.duplicated().value_counts()

False    264835
True          1
Name: count, dtype: int64

df_transaction.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
```

There seems to be no nullvalue

```
#convert date column to date format
df_transaction['DATE']=
pd.to_datetime(df_transaction['DATE'],unit='D', origin='1899-12-30')

df_transaction['DATE'].head()
```

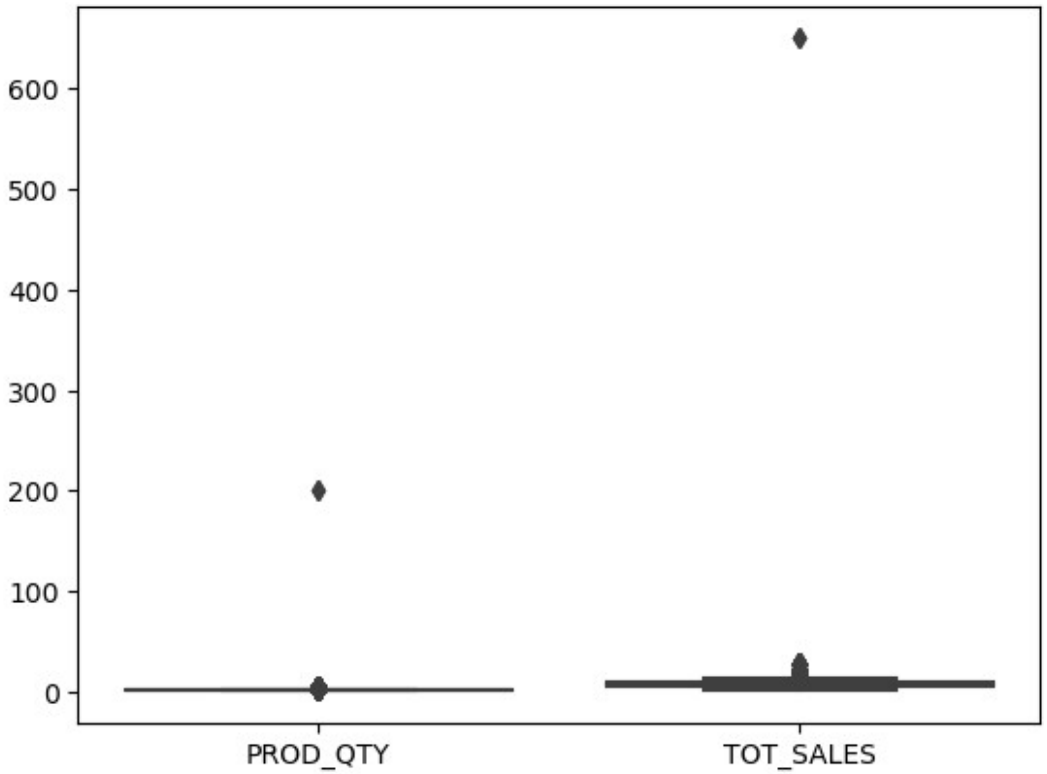
```
0   2018-10-17
1   2019-05-14
2   2019-05-20
3   2018-08-17
4   2018-08-18
Name: DATE, dtype: datetime64[ns]
```

```
#  Split product names as well as remove all digits and special
characters such as '&'
import re
PROD_word = df_transaction['PROD_NAME'].str.replace('([0-9]+
[gG])','').str.replace('[^\w]', ' ').str.split()

# Calculate the frequency grouped by words and sort them
PROD_freq = pd.value_counts([word for name in PROD_word
                                  for word in
name]).sort_values(ascending=False)
PROD_freq.head()
```

```
175g      60561
Chips     49770
150g      41633
Kettle    41288
&         35565
Name: count, dtype: int64
```

```
# Remove word salsa
df_transaction =
df_transaction[~df_transaction["PROD_NAME"].str.contains("[Ss]alsa")]

# Boxplot
sns.boxplot(data=df_transaction.loc[:,["PROD_QTY","TOT_SALES"]])
```

```
<Axes: >
```



There seems to be an outlier in product quantity. Lets investigate further which sales have 200 potato chips

```
#
df_transaction.loc[df_transaction['PROD_QTY']== 200]
```

```
            DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
69762 2018-08-19        226          226000  226201         4
69763 2019-05-20        226          226000  226210         4

                            PROD_NAME  PROD_QTY  TOT_SALES
69762  Dorito Corn Chp     Supreme 380g       200      650.0
69763  Dorito Corn Chp     Supreme 380g       200      650.0
```

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer.

```
#find other purchases made by the customer
df_transaction.loc[df_transaction['LYLTY_CARD_NBR']==226000]
```

```
            DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
69762 2018-08-19        226          226000  226201         4
69763 2019-05-20        226          226000  226210         4

                            PROD_NAME  PROD_QTY  TOT_SALES
```

```
69762  Dorito Corn Chp     Supreme 380g       200      650.0
69763  Dorito Corn Chp     Supreme 380g       200      650.0
```

It looks like this customer has only had the two transactions over the year and is not an ordinary retail customer. The customer might be buying chips for commercial purposes instead. We'll remove this loyalty card number from further analysis.

```
# Filter out the customer based on the loyalty card number
df_transaction = df_transaction.drop(index=[69762,69763])

# Re-examine transaction data
df_transaction.loc[df_transaction["LYLTY_CARD_NBR"]==226000]
```

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

```
# Counting the number of transactions by date
transactions_by_date = df_transaction.groupby(['DATE']).count()
transactions_by_date.head()
```

```
            STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  PROD_NAME
PROD_QTY  \
DATE

2018-07-01        663             663     663       663        663
663
2018-07-02        650             650     650       650        650
650
2018-07-03        674             674     674       674        674
674
2018-07-04        669             669     669       669        669
669
2018-07-05        660             660     660       660        660
660

            TOT_SALES
DATE
2018-07-01        663
2018-07-02        650
2018-07-03        674
2018-07-04        669
2018-07-05        660
```

```
transactions_by_date.describe()
```

```
        STORE_NBR  LYLTY_CARD_NBR      TXN_ID    PROD_NBR   PROD_NAME
\
count  364.000000      364.000000  364.000000  364.000000  364.000000
```

```
mean   677.857143      677.857143  677.857143  677.857143  677.857143
std     33.687536       33.687536   33.687536   33.687536   33.687536
min    607.000000      607.000000  607.000000  607.000000  607.000000
25%    658.000000      658.000000  658.000000  658.000000  658.000000
50%    674.000000      674.000000  674.000000  674.000000  674.000000
75%    694.250000      694.250000  694.250000  694.250000  694.250000
max    865.000000      865.000000  865.000000  865.000000  865.000000


          PROD_QTY   TOT_SALES
count  364.000000  364.000000
mean   677.857143  677.857143
std     33.687536   33.687536
min    607.000000  607.000000
25%    658.000000  658.000000
50%    674.000000  674.000000
75%    694.250000  694.250000
max    865.000000  865.000000
```

There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

```
# Create a column of dates that includes every day from 1 Jul 2018
to30 Jun 2019
dates_seq = pd.date_range("2018-07-01","2019-06-30")

# Join it onto the data to fill in the missing day.
fill_dates = transactions_by_date.reindex(dates_seq)
fill_dates
```

```
            STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  PROD_NAME
PROD_QTY  \
2018-07-01      663.0           663.0   663.0     663.0      663.0
663.0
2018-07-02      650.0           650.0   650.0     650.0      650.0
650.0
2018-07-03      674.0           674.0   674.0     674.0      674.0
674.0
2018-07-04      669.0           669.0   669.0     669.0      669.0
669.0
2018-07-05      660.0           660.0   660.0     660.0      660.0
660.0
```

```
...                ...             ...     ...       ...         ...
...
2019-06-26      657.0           657.0   657.0     657.0       657.0
657.0
2019-06-27      669.0           669.0   669.0     669.0       669.0
669.0
2019-06-28      673.0           673.0   673.0     673.0       673.0
673.0
2019-06-29      703.0           703.0   703.0     703.0       703.0
703.0
2019-06-30      704.0           704.0   704.0     704.0       704.0
704.0

             TOT_SALES
2018-07-01       663.0
2018-07-02       650.0
2018-07-03       674.0
2018-07-04       669.0
2018-07-05       660.0
...                ...
2019-06-26       657.0
2019-06-27       669.0
2019-06-28       673.0
2019-06-29       703.0
2019-06-30       704.0

[365 rows x 7 columns]
```

```
# Find out the missing day
fill_dates.index.difference(df_transaction["DATE"])
```

```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

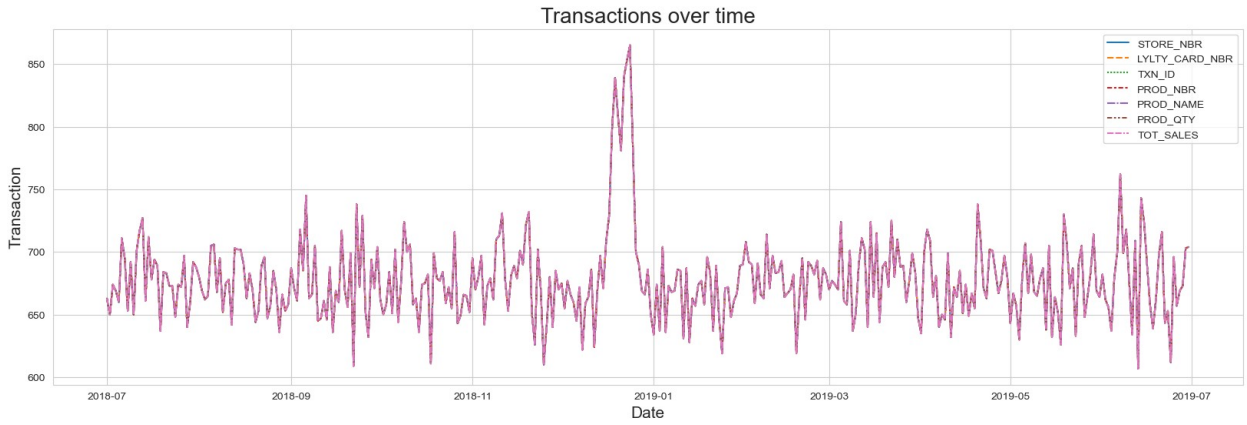
```
# It seems like 2018-12-25 is the missing day
fill_dates.loc['2018-12-25',:]
```

```
STORE_NBR        NaN
LYLTY_CARD_NBR   NaN
TXN_ID           NaN
PROD_NBR         NaN
PROD_NAME        NaN
PROD_QTY         NaN
TOT_SALES        NaN
Name: 2018-12-25 00:00:00, dtype: float64
```

```
# Plot transactions over time
sns.set_style("whitegrid")
plot.figure(figsize=(20,6))
plot.title("Transactions over time",fontsize=20)
sns.lineplot(data=fill_dates)
```

```
plot.xlabel("Date",fontsize=15)
plot.ylabel("Transaction",fontsize=15)
```

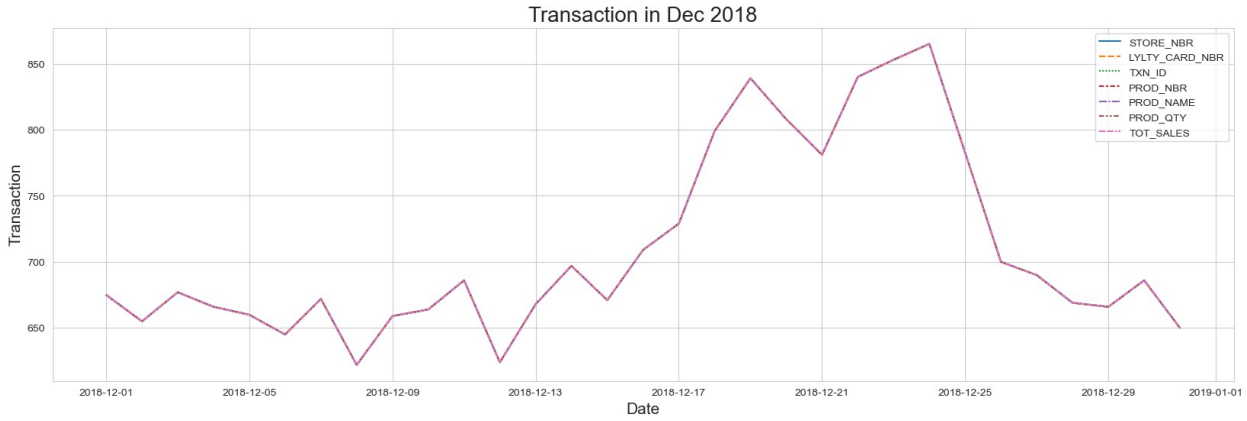
```
Text(0, 0.5, 'Transaction')
```



We can see that there is an increase in purchases in December and a break in late December.Lets look more into it

```
# Month December and look at individual days
plot.figure(figsize=(20,6))
plot.title("Transaction in Dec 2018",fontsize=20)
sns.lineplot(data=fill_dates.loc['2018-12-01':'2018-12-31',:])
plot.xlabel("Date",fontsize=15)
plot.ylabel("Transaction",fontsize=15)
```

```
Text(0, 0.5, 'Transaction')
```



We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day. Thus, we will not treat this point as an outlier. Now we can move on to create other features such as brand of chips or pack size from PROD_NAME.

CREATING FEATURES

```
# Pack size
# Exttract the package size from product name
df_transaction['PACK_SIZE'] =
df_transaction['PROD_NAME'].str.extract("([0-9]+)").astype(float)

# Check result
df_transaction['PACK_SIZE']
```

```
0         175.0
1         175.0
2         170.0
3         175.0
4         150.0
          ...
264831    175.0
264832    175.0
264833    170.0
264834    150.0
264835    175.0
Name: PACK_SIZE, Length: 246740, dtype: float64
```

```
df_transaction['PACK_SIZE'].describe()
```

```
count    246740.000000
mean        175.583521
std          59.432118
min          70.000000
25%         150.000000
50%         170.000000
75%         175.000000
max         380.000000
Name: PACK_SIZE, dtype: float64
```

```
# Product Brand
df_transaction['BRAND_NAME'] =
df_transaction['PROD_NAME'].str.split().str[0]
df_transaction['BRAND_NAME']
```

```
0          Natural
1              CCs
2           Smiths
3           Smiths
4           Kettle
            ...
264831      Kettle
264832    Tostitos
264833     Doritos
264834     Doritos
264835    Tostitos
Name: BRAND_NAME, Length: 246740, dtype: object
```

```
# Unique brand names
df_transaction['BRAND_NAME'].unique()
```

```
array(['Natural', 'CCs', 'Smiths', 'Kettle', 'Grain', 'Doritos',
       'Twisties', 'WW', 'Thins', 'Burger', 'NCC', 'Cheezels',
'Infzns',
       'Red', 'Pringles', 'Dorito', 'Infuzions', 'Smith', 'GrnWves',
       'Tyrrells', 'Cobs', 'French', 'RRD', 'Tostitos', 'Cheetos',
       'Woolworths', 'Snbts', 'Sunbites'], dtype=object)
```

```
df_transaction['BRAND_NAME'].value_counts()
```

```
BRAND_NAME
Kettle        41288
Smiths        27390
Pringles      25102
Doritos       22041
Thins         14075
RRD           11894
Infuzions     11057
WW            10320
Cobs           9693
Tostitos       9471
Twisties       9454
Tyrrells       6442
Grain          6272
Natural        6050
Cheezels       4603
CCs            4551
Red            4427
Dorito         3183
Infzns         3144
Smith          2963
Cheetos        2927
Snbts          1576
Burger         1564
Woolworths     1516
GrnWves        1468
Sunbites       1432
NCC            1419
French         1418
Name: count, dtype: int64
```

Some of the brand names look like they are of the same brands

Dorito and Doritos Grain and GrnWves (not sure) Infuzions and Infzns NCC and Natural (not sure) Red and RRD (not sure) Smith and Smiths Snbts and Sunbites WW and Woolworths Let's combine these together.

```
# Check brands which are not sure
df_transaction["PROD_NAME"].loc[df_transaction["BRAND_NAME"]=="Grain"]
.head()
```

```
7      Grain Waves         Sweet Chilli 210g
9      Grain Waves Sour    Cream&Chives 210G
85     Grain Waves         Sweet Chilli 210g
181    Grain Waves         Sweet Chilli 210g
225    Grain Waves         Sweet Chilli 210g
Name: PROD_NAME, dtype: object
```

```
df_transaction["PROD_NAME"].loc[df_transaction["BRAND_NAME"]=="GrnWves
"].head()
```

```
56     GrnWves Plus Btroot & Chilli Jam 180g
298    GrnWves Plus Btroot & Chilli Jam 180g
301    GrnWves Plus Btroot & Chilli Jam 180g
387    GrnWves Plus Btroot & Chilli Jam 180g
578    GrnWves Plus Btroot & Chilli Jam 180g
Name: PROD_NAME, dtype: object
```

```
df_transaction["PROD_NAME"].loc[df_transaction["BRAND_NAME"]=="NCC"].h
ead()
```

```
17     NCC Sour Cream &    Garden Chives 175g
21     NCC Sour Cream &    Garden Chives 175g
437    NCC Sour Cream &    Garden Chives 175g
535    NCC Sour Cream &    Garden Chives 175g
828    NCC Sour Cream &    Garden Chives 175g
Name: PROD_NAME, dtype: object
```

```
df_transaction["PROD_NAME"].loc[df_transaction["BRAND_NAME"]=="Natural
"].head()
```

```
0       Natural Chip        Compny SeaSalt175g
40      Natural ChipCo      Hony Soy Chckn175g
75     Natural Chip Co     Tmato Hrb&Spce 175g
214    Natural Chip Co     Tmato Hrb&Spce 175g
234     Natural ChipCo      Hony Soy Chckn175g
Name: PROD_NAME, dtype: object
```

```
df_transaction["PROD_NAME"].loc[df_transaction["BRAND_NAME"]=="Red"].h
ead()
```

```
28        Red Rock Deli Thai  Chilli&Lime 150g
34     Red Rock Deli Sp    Salt & Truffle 150G
212    Red Rock Deli Sp    Salt & Truffle 150G
297       Red Rock Deli Thai  Chilli&Lime 150g
331    Red Rock Deli Sp    Salt & Truffle 150G
Name: PROD_NAME, dtype: object
```

```
df_transaction["PROD_NAME"].loc[df_transaction["BRAND_NAME"]=="RRD"].h
ead()
```

```
92        RRD Chilli&         Coconut 150g
118       RRD Chilli&         Coconut 150g
140    RRD Sweet Chilli &  Sour Cream 165g
294    RRD Sweet Chilli &  Sour Cream 165g
302       RRD Chilli&         Coconut 150g
Name: PROD_NAME, dtype: object
```

```
# Clean brand names
def rename_brand(new,old):

df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old] =
new

rename_brand("Doritos","Dorito")
rename_brand("Grain Waves","Grain")
rename_brand("Grain Waves","GrnWves")
rename_brand("Infuzions","Infzns")
rename_brand("Natural Chip Co","NCC")
rename_brand("Natural Chip Co","Natural")
rename_brand("Red Rock Deli","Red")
rename_brand("Red Rock Deli","RRD")
rename_brand("Smiths","Smith")
rename_brand("Sunbites","Snbts")
rename_brand("Woolworths","WW")
```

```
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/
ipykernel_1924/3987069382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
```

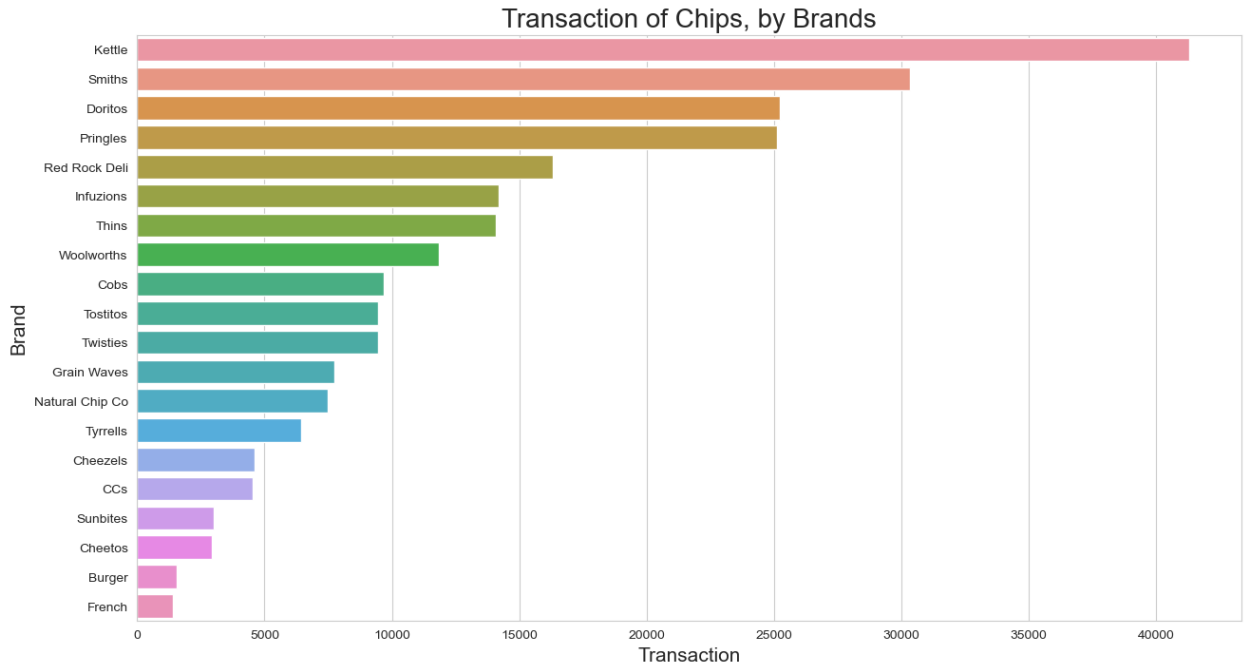
```
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/398706
9382.py:3: SettingWithCopyWarning:
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
  df_transaction["BRAND_NAME"].loc[df_transaction["BRAND_NAME"]==old]
= new
```

```
df_transaction['BRAND_NAME'].unique()
```

```
array(['Natural Chip Co', 'CCs', 'Smiths', 'Kettle', 'Grain Waves',
       'Doritos', 'Twisties', 'Woolworths', 'Thins', 'Burger',
'Cheezels',
       'Infuzions', 'Red Rock Deli', 'Pringles', 'Tyrrells', 'Cobs',
       'French', 'Tostitos', 'Cheetos', 'Sunbites'], dtype=object)
```

```
# Plot a barchart to show the total sales of each brand
plot.figure(figsize=(15,8))
sns.barplot(x =
df_transaction["BRAND_NAME"].value_counts(),y=df_transaction["BRAND_NA
ME"].value_counts().index)
plot.title("Transaction of Chips, by Brands",fontsize=20)
plot.ylabel("Brand",fontsize=15)
plot.xlabel("Transaction",fontsize=15)
```

```
Text(0.5, 0, 'Transaction')
```



The bar chart clearly indicates that Kettle products the most popular chips.

```
# transaction table
df_transaction.head()
```

```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2018-10-17          1            1000       1         5
1 2019-05-14          1            1307     348        66
2 2019-05-20          1            1343     383        61
3 2018-08-17          2            2373     974        69
4 2018-08-18          2            2426    1038       108

                                PROD_NAME  PROD_QTY  TOT_SALES
PACK_SIZE  \
0    Natural Chip        Compny SeaSalt175g         2        6.0
175.0
1                  CCs Nacho Cheese    175g         3        6.3
175.0
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
170.0
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
175.0
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
150.0

        BRAND_NAME
0  Natural Chip Co
1              CCs
2           Smiths
```

```
3           Smiths
4           Kettle
```

MERGE DATASET

```
# Merge transaction data to purchase data
merge_data = pd.merge(df_transaction,df_purchase,on="LYLTY_CARD_NBR")
merge_data.head()
```

```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2018-10-17          1            1000       1         5
1 2019-05-14          1            1307     348        66
2 2018-11-10          1            1307     346        96
3 2019-03-09          1            1307     347        54
4 2019-05-20          1            1343     383        61

                                PROD_NAME  PROD_QTY  TOT_SALES
PACK_SIZE  \
0  Natural Chip        Compny SeaSalt175g         2        6.0
175.0
1                CCs Nacho Cheese    175g         3        6.3
175.0
2          WW Original Stacked Chips 160g         2        3.8
160.0
3                       CCs Original 175g         1        2.1
175.0
4  Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
170.0

        BRAND_NAME               LIFESTAGE PREMIUM_CUSTOMER
0  Natural Chip Co   YOUNG SINGLES/COUPLES          Premium
1              CCs  MIDAGE SINGLES/COUPLES           Budget
2       Woolworths  MIDAGE SINGLES/COUPLES           Budget
3              CCs  MIDAGE SINGLES/COUPLES           Budget
4           Smiths  MIDAGE SINGLES/COUPLES           Budget
```

```
merge_data.count()
```

```
DATE                246740
STORE_NBR           246740
LYLTY_CARD_NBR      246740
TXN_ID              246740
PROD_NBR            246740
PROD_NAME           246740
PROD_QTY            246740
TOT_SALES           246740
PACK_SIZE           246740
BRAND_NAME          246740
LIFESTAGE           246740
```

```
PREMIUM_CUSTOMER    246740
dtype: int64
```

```
merge_data.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
PACK_SIZE           0
BRAND_NAME          0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

Data analysis on customer segments Now that the data is ready for analysis, we can define some metrics of interest to the client:

Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is How many customers are in each segment How many chips are bought per customer by segment What's the average chip price by customer segment

```
# Total Sales
#  Calculate the total sales by those dimensions
totsales_l_p =
pd.DataFrame(merge_data.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"])
["TOT_SALES"].sum())
totsales_l_p
```

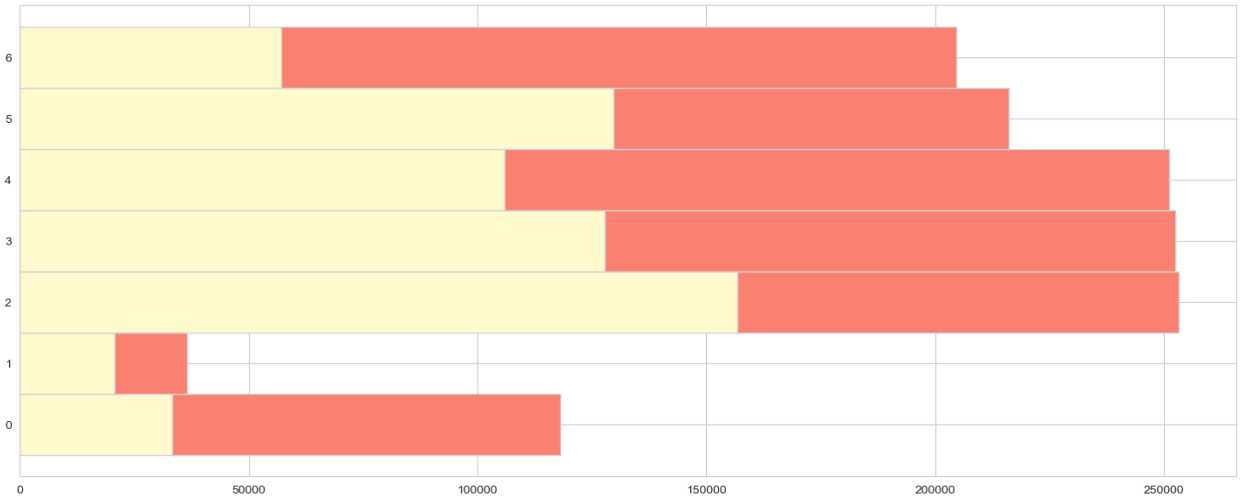
		TOT_SALES
LIFESTAGE	PREMIUM_CUSTOMER	
MIDAGE SINGLES/COUPLES	Budget	33345.70
	Mainstream	84734.25
	Premium	54443.85
NEW FAMILIES	Budget	20607.45
	Mainstream	15979.70
	Premium	10760.80
OLDER FAMILIES	Budget	156863.75
	Mainstream	96413.55
	Premium	75242.60
OLDER SINGLES/COUPLES	Budget	127833.60
	Mainstream	124648.50
	Premium	123537.55
RETIREES	Budget	105916.30
	Mainstream	145168.95
	Premium	91296.65

```
YOUNG FAMILIES          Budget             129717.95
                        Mainstream          86338.25
                        Premium             78571.70
YOUNG SINGLES/COUPLES   Budget              57122.10
                        Mainstream         147582.20
                        Premium             39052.30
```

```
bars1 =
totsales_l_p[totsales_l_p.index.get_level_values("PREMIUM_CUSTOMER")
== "Budget"]["TOT_SALES"]
bars2 =
totsales_l_p[totsales_l_p.index.get_level_values("PREMIUM_CUSTOMER")
== "Mainstream"]["TOT_SALES"]
bars3 =
totsales_l_p[totsales_l_p.index.get_level_values("PREMIUM_CUSTOMER")
== "Premium"]["TOT_SALES"]

bars1_text = (bars1 /
sum(totsales_l_p["TOT_SALES"])).apply("{:.1%}".format)
bars2_text = (bars2 /
sum(totsales_l_p["TOT_SALES"])).apply("{:.1%}".format)
bars3_text = (bars3 /
sum(totsales_l_p["TOT_SALES"])).apply("{:.1%}".format)

# Names of group and bar width
names = totsales_l_p.index.get_level_values("LIFESTAGE").unique()

# The position of the bars on the x-axis
r = np.arange(len(names))

plot.figure(figsize=(18,7))

colors = ['salmon', 'lemonchiffon', 'lightskyblue']
# Create yellow bars
budget_bar = plot.barh(r, bars1, color=colors[1],
edgecolor="lightgrey", height=1, label="Budget")
# Create red bars (middle)
mains_bar = plot.barh(r, bars2, left=bars1, color=colors[0],
edgecolor="lightgrey", height=1, label="Mainstream")
# Create blue bars (top)
prem_bar = plot.barh(r, bars3,
left=bars1.append(bars2).groupby(["LIFESTAGE"]).sum(),
color=colors[2], edgecolor="lightgrey", height=1, label="Premium")

for i in range(7):
    budget_width = budget_bar[i].get_width()
    budget_main_width = budget_width + mains_bar[i].get_width()
    plot.text(budget_width/2, i, bars1_text[i], va='center',
ha='center', size=13, color='midnightblue')
    plot.text(budget_width + mains_bar[i].get_width()/2, i,
```

```
bars2_text[i], va='center', ha='center', size=13,
color='midnightblue')
    plot.text(budget_main_width + prem_bar[i].get_width()/2, i,
bars3_text[i], va='center', ha='center', size=13,
color='midnightblue')

# Custom X axis
plot.yticks(r, names)
plot.ylabel("Lifestage",fontsize=15)
plot.xlabel("Total Sales",fontsize=15)
plot.legend(loc='center left', bbox_to_anchor=(1.0, 0.5),fontsize=15)

plot.title("Total Sales, by Lifestage and Customer Type",fontsize=20)

# Show graphic
plot.show()
```

```
---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
last)
/var/folders/6z/lj2vnr8x6_n0z3x4xh4xj0ww0000gn/T/ipykernel_1924/562192
020.py in ?()
     19 budget_bar = plot.barh(r, bars1, color=colors[1],
edgecolor="lightgrey", height=1, label="Budget")
     20 # Create red bars (middle)
     21 mains_bar = plot.barh(r, bars2, left=bars1, color=colors[0],
edgecolor="lightgrey", height=1, label="Mainstream")
     22 # Create blue bars (top)
---> 23 prem_bar = plot.barh(r, bars3,
left=bars1.append(bars2).groupby(["LIFESTAGE"]).sum(),
color=colors[2], edgecolor="lightgrey", height=1, label="Premium")
     24
     25 for i in range(7):
     26     budget_width = budget_bar[i].get_width()

~/anaconda3/lib/python3.11/site-packages/pandas/core/generic.py in ?
(self, name)
   5985             and name not in self._accessors
   5986             and
self._info_axis._can_hold_identifiers_and_holds_name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'Series' object has no attribute 'append'
```



Sales are coming mainly from

Budget - Older Families Mainstream - Young Singles/Couples Mainstream - Retirees

```
#  Calculate the customer count in each segment
count_l_p =
pd.DataFrame(df_purchase.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"]).cou
nt())
count_l_p.columns=["CUSTOMER_COUNTS"]
count_l_p
```

		CUSTOMER_COUNTS
LIFESTAGE	PREMIUM_CUSTOMER	
MIDAGE SINGLES/COUPLES	Budget	1504
	Mainstream	3340
	Premium	2431
NEW FAMILIES	Budget	1112
	Mainstream	849
	Premium	588
OLDER FAMILIES	Budget	4675
	Mainstream	2831
	Premium	2274
OLDER SINGLES/COUPLES	Budget	4929
	Mainstream	4930
	Premium	4750
RETIREES	Budget	4454
	Mainstream	6479
	Premium	3872
YOUNG FAMILIES	Budget	4017
	Mainstream	2728
	Premium	2433
YOUNG SINGLES/COUPLES	Budget	3779
	Mainstream	8088
	Premium	2574

```
# Average Sales quantity
# Calculate the average number of units per customer by those two
dimensions
merge_l_p =
pd.merge(totsales_l_p,count_l_p,on=["LIFESTAGE","PREMIUM_CUSTOMER"])
merge_l_p['SALES_QTY'] =
merge_data.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"])["PROD_QTY"].sum()
merge_l_p['AVG_SALES_QTY'] =
(merge_l_p["SALES_QTY"]/merge_l_p["CUSTOMER_COUNTS"])
merge_l_p
```

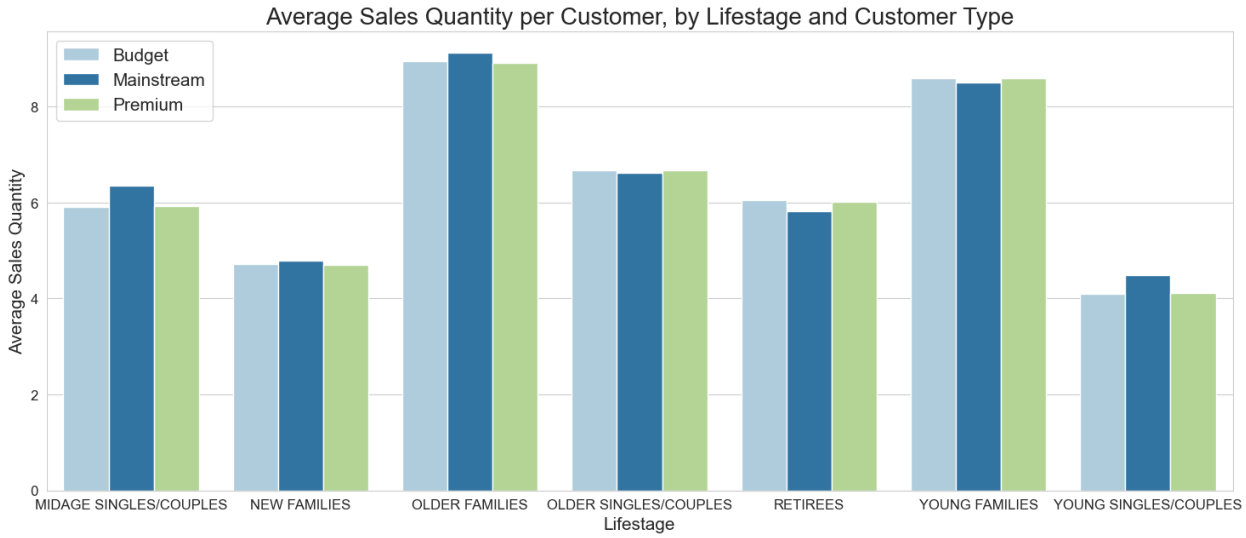
		TOT_SALES	CUSTOMER_COUNTS	\
LIFESTAGE	PREMIUM_CUSTOMER			
MIDAGE SINGLES/COUPLES	Budget	33345.70	1504	
	Mainstream	84734.25	3340	
	Premium	54443.85	2431	
NEW FAMILIES	Budget	20607.45	1112	
	Mainstream	15979.70	849	
	Premium	10760.80	588	
OLDER FAMILIES	Budget	156863.75	4675	
	Mainstream	96413.55	2831	
	Premium	75242.60	2274	
OLDER SINGLES/COUPLES	Budget	127833.60	4929	
	Mainstream	124648.50	4930	
	Premium	123537.55	4750	
RETIREES	Budget	105916.30	4454	
	Mainstream	145168.95	6479	
	Premium	91296.65	3872	
YOUNG FAMILIES	Budget	129717.95	4017	
	Mainstream	86338.25	2728	
	Premium	78571.70	2433	
YOUNG SINGLES/COUPLES	Budget	57122.10	3779	
	Mainstream	147582.20	8088	
	Premium	39052.30	2574	

		SALES_QTY	AVG_SALES_QTY
LIFESTAGE	PREMIUM_CUSTOMER		
MIDAGE SINGLES/COUPLES	Budget	8883	5.906250
	Mainstream	21213	6.351198
	Premium	14400	5.923488
NEW FAMILIES	Budget	5241	4.713129
	Mainstream	4060	4.782097
	Premium	2769	4.709184
OLDER FAMILIES	Budget	41853	8.952513
	Mainstream	25804	9.114800
	Premium	20239	8.900176
OLDER SINGLES/COUPLES	Budget	32883	6.671333
	Mainstream	32607	6.613996
	Premium	31695	6.672632
RETIREES	Budget	26932	6.046700

```
                          Mainstream              37677       5.815249
                          Premium                 23266       6.008781
YOUNG FAMILIES            Budget                  34482       8.584018
                          Mainstream              23194       8.502199
                          Premium                 20901       8.590629
YOUNG SINGLES/COUPLES     Budget                  15500       4.101614
                          Mainstream              36225       4.478858
                          Premium                 10575       4.108392
```

```
# Plot the average number of units per customer by those two
dimensions
plot.figure(figsize=(18,7))
plot.tick_params(labelsize=12)
sns.barplot(x=merge_l_p.reset_index()["LIFESTAGE"],
y=merge_l_p.reset_index()["AVG_SALES_QTY"],
            hue=merge_l_p.reset_index()["PREMIUM_CUSTOMER"],
            palette=sns.color_palette("Paired", 3))
plot.title("Average Sales Quantity per Customer, by Lifestage and
Customer Type",fontsize=20)
plot.legend(fontsize=15, bbox_to_anchor=(0.14,0.98),borderaxespad =
0.)
plot.xlabel("Lifestage",fontsize=15)
plot.ylabel("Average Sales Quantity",fontsize=15)
```

```
Text(0, 0.5, 'Average Sales Quantity')
```



Older families and young families in general buy more chips per customer.

Average sales price

```
# Calculate the average price per unit sold (average sale price) by
those two customer dimensions
merge_l_p['AVG_SALES_PRICE'] =
```

```
(merge_l_p["TOT_SALES"]/merge_l_p["SALES_QTY"])
merge_l_p
```

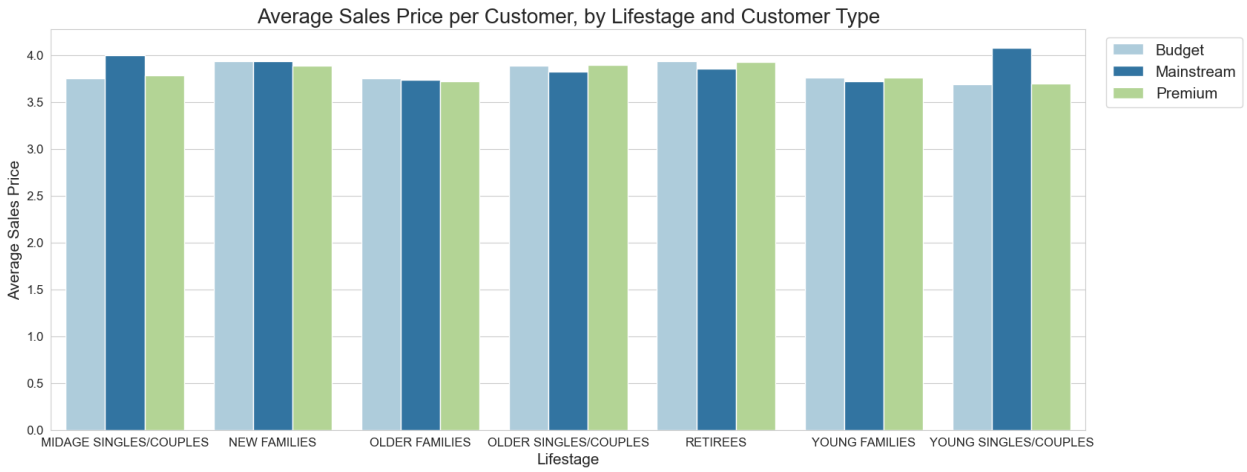
LIFESTAGE	PREMIUM_CUSTOMER	TOT_SALES	CUSTOMER_COUNTS	\
MIDAGE SINGLES/COUPLES	Budget	33345.70	1504	
	Mainstream	84734.25	3340	
	Premium	54443.85	2431	
NEW FAMILIES	Budget	20607.45	1112	
	Mainstream	15979.70	849	
	Premium	10760.80	588	
OLDER FAMILIES	Budget	156863.75	4675	
	Mainstream	96413.55	2831	
	Premium	75242.60	2274	
OLDER SINGLES/COUPLES	Budget	127833.60	4929	
	Mainstream	124648.50	4930	
	Premium	123537.55	4750	
RETIREES	Budget	105916.30	4454	
	Mainstream	145168.95	6479	
	Premium	91296.65	3872	
YOUNG FAMILIES	Budget	129717.95	4017	
	Mainstream	86338.25	2728	
	Premium	78571.70	2433	
YOUNG SINGLES/COUPLES	Budget	57122.10	3779	
	Mainstream	147582.20	8088	
	Premium	39052.30	2574	

LIFESTAGE	PREMIUM_CUSTOMER	SALES_QTY	AVG_SALES_QTY	\
MIDAGE SINGLES/COUPLES	Budget	8883	5.906250	
	Mainstream	21213	6.351198	
	Premium	14400	5.923488	
NEW FAMILIES	Budget	5241	4.713129	
	Mainstream	4060	4.782097	
	Premium	2769	4.709184	
OLDER FAMILIES	Budget	41853	8.952513	
	Mainstream	25804	9.114800	
	Premium	20239	8.900176	
OLDER SINGLES/COUPLES	Budget	32883	6.671333	
	Mainstream	32607	6.613996	
	Premium	31695	6.672632	
RETIREES	Budget	26932	6.046700	
	Mainstream	37677	5.815249	
	Premium	23266	6.008781	
YOUNG FAMILIES	Budget	34482	8.584018	
	Mainstream	23194	8.502199	
	Premium	20901	8.590629	
YOUNG SINGLES/COUPLES	Budget	15500	4.101614	
	Mainstream	36225	4.478858	
	Premium	10575	4.108392	

```
                                            AVG_SALES_PRICE
LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget                     3.753878
                        Mainstream                 3.994449
                        Premium                    3.780823
NEW FAMILIES            Budget                     3.931969
                        Mainstream                 3.935887
                        Premium                    3.886168
OLDER FAMILIES          Budget                     3.747969
                        Mainstream                 3.736380
                        Premium                    3.717703
OLDER SINGLES/COUPLES   Budget                     3.887529
                        Mainstream                 3.822753
                        Premium                    3.897698
RETIREES                Budget                     3.932731
                        Mainstream                 3.852986
                        Premium                    3.924037
YOUNG FAMILIES          Budget                     3.761903
                        Mainstream                 3.722439
                        Premium                    3.759232
YOUNG SINGLES/COUPLES   Budget                     3.685297
                        Mainstream                 4.074043
                        Premium                    3.692889
```

```
# Plot the average price per unit sold (average sale price) by those
two customer dimensions.
plot.figure(figsize=(18,7))
plot.tick_params(labelsize=12)
sns.barplot(x=merge_l_p.reset_index()["LIFESTAGE"],
y=merge_l_p.reset_index()["AVG_SALES_PRICE"],
            hue=merge_l_p.reset_index()["PREMIUM_CUSTOMER"],
            palette=sns.color_palette("Paired", 3))
plot.title("Average Sales Price per Customer, by Lifestage and
Customer Type",fontsize=20)
plot.legend(fontsize=15, bbox_to_anchor=(1.02,0.98),borderaxespad =
0.)
plot.xlabel("Lifestage",fontsize=15)
plot.ylabel("Average Sales Price",fontsize=15)
```

```
Text(0, 0.5, 'Average Sales Price')
```



Mainstream midage and young singles/couples are more willing to pay more per packet of chips compared to their budget and premium counterparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consumption. This is also supported by there being fewer premium midage and young singles and couples buying chips compared to their mainstream counterparts.