# Data Cleaning, Preparation and Customer Analytics

Analyse the client's transaction dataset and identify customer's chip purchasing behaviours to generate insights and provide commercial recommendations.

1. Examine transaction data
2. Examine customer data
3. Data analysis and customer segments
4. Determine the customer segments to be targeted.

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

cdata = pd.read_excel('/content/QVI_transaction_data.xlsx')
cdata.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
```

```
cdata.shape
```

```
(264836, 8)
```

```
cdata.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
```

```
cdata.columns
```

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

```
cdata.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
```

```
cdata.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
```

```
cdata.nunique()
```

```
DATE                 364
STORE_NBR            272
LYLTY_CARD_NBR     72637
TXN_ID            263127
PROD_NBR             114
PROD_NAME            114
PROD_QTY               6
TOT_SALES            112
dtype: int64
```

```
cdata.count()
```

```
DATE                264836
STORE_NBR           264836
LYLTY_CARD_NBR      264836
TXN_ID              264836
PROD_NBR            264836
PROD_NAME           264836
PROD_QTY            264836
TOT_SALES           264836
dtype: int64
```

```
cdata[cdata.duplicated(['TXN_ID'])].head()
```

```
      DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
42   43605         55           55073   48887       113
377  43475          7            7364    7739        20
419  43391         12           12301   10982        93
476  43351         16           16427   14546        81
511  43315         19           19272   16683        31

                                    PROD_NAME  PROD_QTY  TOT_SALES
42                       Twisties Chicken270g         1        4.6
377           Doritos Cheese      Supreme 330g        2       11.4
419  Doritos Corn Chip Southern Chicken 150g         2        7.8
476            Pringles Original   Crisps 134g        1        3.7
511   Infzns Crn Crnchers Tangy Gcamole 110g         2        7.6
```

```
cdata.loc[cdata['TXN_ID']==48887, :]
```

```
     DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
41  43605         55           55073   48887         4
42  43605         55           55073   48887       113

                           PROD_NAME  PROD_QTY  TOT_SALES
41  Dorito Corn Chp     Supreme 380g         1       3.25
42              Twisties Chicken270g         1       4.60
```

So, two customers bought two different chips from the same store on the same day.

```
cdata.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
```

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

```
0    43390
1    43599
2    43605
3    43329
4    43330
Name: DATE, dtype: int64
```

```
import datetime
def digits_to_date(DigitsDate):
  excel_d = datetime.datetime(1900, 1, 1)
  if(DigitsDate<60):
    diff_days = datetime.timedelta(days = (DigitsDate-1))
  else:
    diff_days = datetime.timedelta(days = (DigitsDate-2))
  corrected_date = excel_d + diff_days
  return corrected_date
```

```
cdata['DATE'] = cdata['DATE'].apply(digits_to_date)
cdata.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
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
```

```
cdata['PROD_NAME'].head()
```

```
0      Natural Chip        Compny SeaSalt175g
1                    CCs Nacho Cheese    175g
2      Smiths Crinkle Cut  Chips Chicken 170g
```

```
3      Smiths Chip Thinly  S/Cream&Onion 175g
4    Kettle Tortilla ChpsHny&Jlpno Chili 150g
Name: PROD_NAME, dtype: object
```

```
cdata['PACK_SIZE'] = cdata.PROD_NAME.str.extract('(\d+)')
cdata.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
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
```

```
cdata.dtypes
```

```
DATE              datetime64[ns]
STORE_NBR                  int64
LYLTY_CARD_NBR             int64
TXN_ID                     int64
PROD_NBR                   int64
PROD_NAME                 object
PROD_QTY                   int64
TOT_SALES                float64
PACK_SIZE                 object
dtype: object
```

convert the object type of PACK_SIZE to numeric

```
cdata['PACK_SIZE'] = pd.to_numeric(cdata['PACK_SIZE'])
cdata['PACK_SIZE'].dtype
```

```
dtype('int64')
```

to remove / and & along with the numeric part at the end of PROD_NAME

```
import re
re.sub('[&/]',' ','Smiths Chip Thinly S/Cream&Onion 175g')

{"type":"string"}

re.sub('\d\w*',' ','Smiths Chip Thinly S/Cream&Onion 175g')

{"type":"string"}
```

Clean the PROD_NAME column:

```
def text_clean(text):
  text = re.sub('[&/]',' ',text)
  text = re.sub('\d\w*',' ',text)
  return text
cdata['PROD_NAME'] = cdata['PROD_NAME'].apply(text_clean)
cdata.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 SeaSalt          2        6.0
175
1                  CCs Nacho Cheese              3        6.3
175
2    Smiths Crinkle Cut  Chips Chicken           2        2.9
170
3    Smiths Chip Thinly  S Cream Onion           5       15.0
175
4  Kettle Tortilla ChpsHny Jlpno Chili           3       13.8
150

cdata['PROD_NAME'].str.partition().head()

         0  1                               2
0  Natural        Chip        Compny SeaSalt
1      CCs                  Nacho Cheese
2   Smiths       Crinkle Cut  Chips Chicken
3   Smiths       Chip Thinly  S Cream Onion
4   Kettle     Tortilla ChpsHny Jlpno Chili

cdata['PROD_NAME'].str.partition()[0].head()

0    Natural
1        CCs
```

```
2     Smiths
3     Smiths
4     Kettle
Name: 0, dtype: object
```

```
cdata['BRAND'] = cdata['PROD_NAME'].str.partition()[0]
cdata.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 SeaSalt          2        6.0
175
1                  CCs Nacho Cheese              3        6.3
175
2    Smiths Crinkle Cut  Chips Chicken           2        2.9
170
3    Smiths Chip Thinly  S Cream Onion           5       15.0
175
4  Kettle Tortilla ChpsHny Jlpno Chili           3       13.8
150

     BRAND
0  Natural
1      CCs
2   Smiths
3   Smiths
4   Kettle
```

```
cdata['BRAND'].unique()
```

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

```
cdata['BRAND'].replace('Ncc','Natural',inplace=True)
cdata['BRAND'].replace('Ccs','CCS',inplace=True)
cdata['BRAND'].replace('Smith','Smiths',inplace=True)
cdata['BRAND'].replace(['Grain','Grnwves'],'Grainwaves',inplace=True)
cdata['BRAND'].replace('Dorito','Doritos',inplace=True)
cdata['BRAND'].replace('Ww','Woolworths',inplace=True)
```

```
cdata['BRAND'].replace('Infzns','Infuzions',inplace=True)
cdata['BRAND'].replace(['Red','Rrd'],'Red Rock Deli',inplace=True)
cdata['BRAND'].replace('Snbts','Sunbites',inplace=True)
cdata['BRAND'].unique()
```

```
array(['Natural', 'CCs', 'Smiths', 'Kettle', 'Old', 'Grainwaves',
       'Doritos', 'Twisties', 'WW', 'Thins', 'Burger', 'NCC',
'Cheezels',
       'Infuzions', 'Red Rock Deli', 'Pringles', 'GrnWves',
'Tyrrells',
       'Cobs', 'Woolworths', 'French', 'RRD', 'Tostitos', 'Cheetos',
       'Sunbites'], dtype=object)
```

Removing outliers

```
cdata['PROD_QTY'].unique()
```

```
array([  2,   3,   5,   1,   4, 200])
```

```
cdata['PROD_QTY'].value_counts()
```

```
2      236039
1       27518
5         450
3         430
4         397
200         2
Name: PROD_QTY, dtype: int64
```

```
cdata.loc[cdata['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  PACK_SIZE
BRAND
69762  Dorito Corn Chp     Supreme       200      650.0        380
Doritos
69763  Dorito Corn Chp     Supreme       200      650.0        380
Doritos
```

```
cdata.loc[cdata['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  PACK_SIZE
BRAND
69762  Dorito Corn Chp     Supreme       200      650.0        380
```

```
Doritos
69763  Dorito Corn Chp      Supreme           200       650.0          380
Doritos
```

As this person has only made two transactions, he is not a retail customer. We can safely drop his records from the dataset to clear the 'outlier'

```
cdata.index[cdata['LYLTY_CARD_NBR'] == 226000]
```

```
Int64Index([69762, 69763], dtype='int64')
```

```
cdata.drop([69762, 69763], inplace=True)
cdata.index[cdata['LYLTY_CARD_NBR'] == 226000]
```

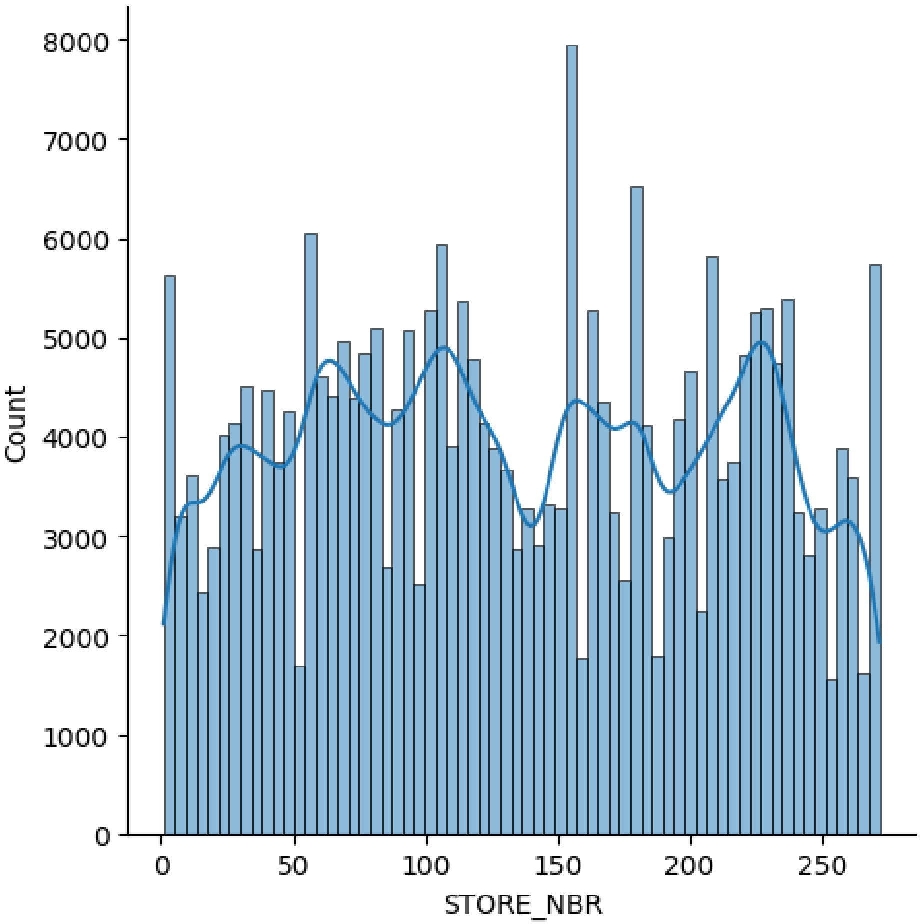
```
Int64Index([], dtype='int64')
```

```
cdata['STORE_NBR'].value_counts()
```

```
226    2020
88     1873
93     1832
165    1819
237    1785
       ...
11        2
252       2
206       2
92        1
76        1
Name: STORE_NBR, Length: 272, dtype: int64
```

```
sns.displot(cdata['STORE_NBR'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d9ce4f9efe0>
```

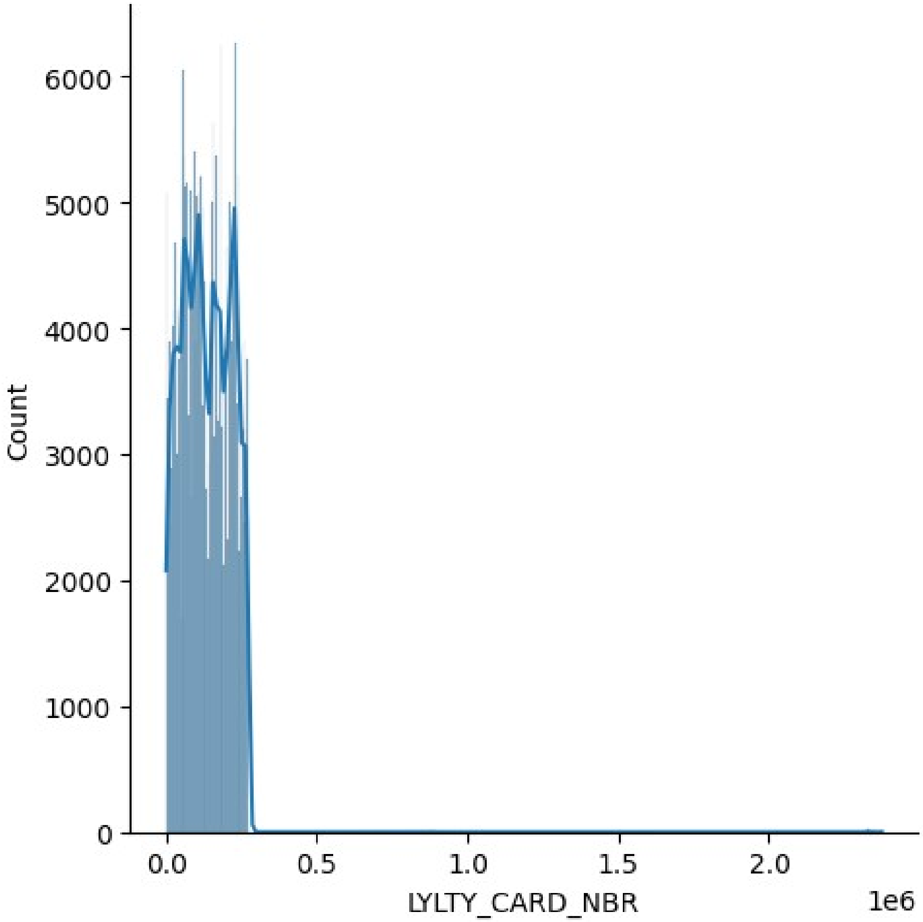


```
cdata['LYLTY_CARD_NBR'].value_counts()
```

```
172032    18
162039    18
13138     17
230078    17
128178    17
          ..
22190      1
22138      1
22099      1
22089      1
272380     1
Name: LYLTY_CARD_NBR, Length: 72636, dtype: int64
```

```
sns.displot(cdata['LYLTY_CARD_NBR'],kde=True)
```

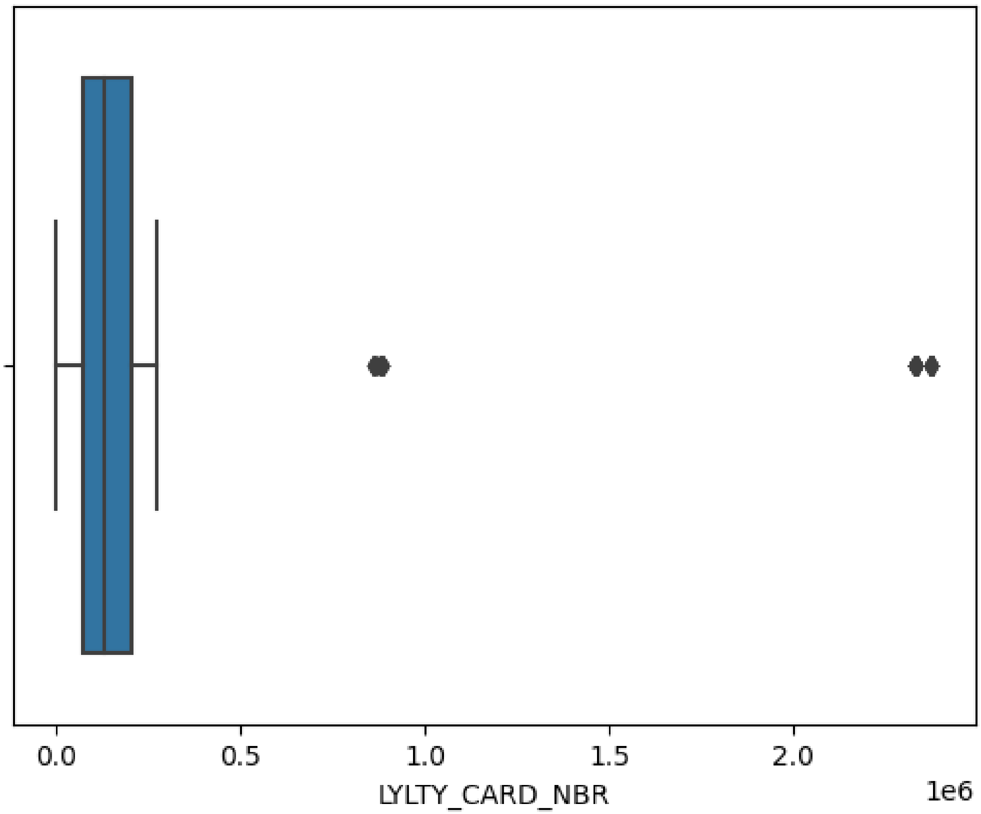
```
<seaborn.axisgrid.FacetGrid at 0x7d9ce4f9cc10>
```



skewed histogram. Customers are less loyal where they have not bought except a few times from the stores.

```
sns.boxplot(x = cdata['LYLTY_CARD_NBR'])
```

```
<Axes: xlabel='LYLTY_CARD_NBR'>
```

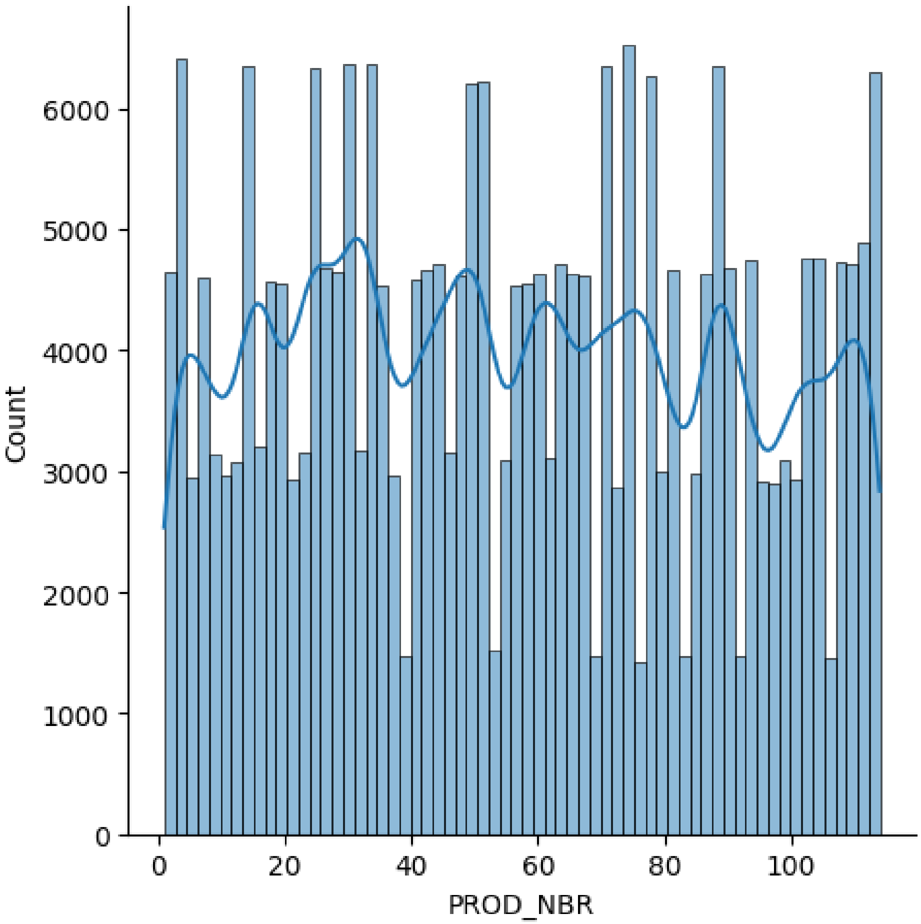


```
cdata['PROD_NBR'].value_counts()
```

```
102    3304
108    3296
33     3269
112    3268
75     3265
       ...
11     1431
76     1430
98     1419
29     1418
72     1410
Name: PROD_NBR, Length: 114, dtype: int64
```

```
sns.displot(cdata['PROD_NBR'], kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d9c97679b10>
```

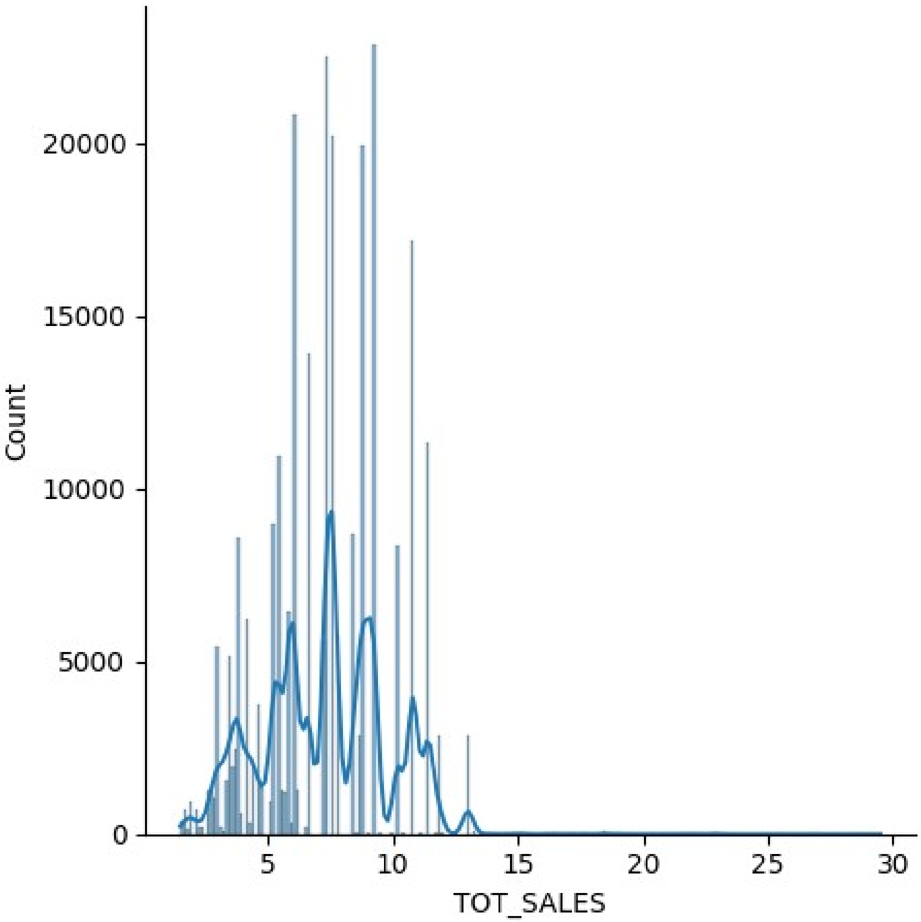


```
cdata['TOT_SALES'].value_counts()
```

```
9.2     22821
7.4     22513
6.0     20798
7.6     20212
8.8     19900
        ...  
15.5        3
9.3         3
6.9         3
12.4        2
11.2        2
Name: TOT_SALES, Length: 111, dtype: int64
```

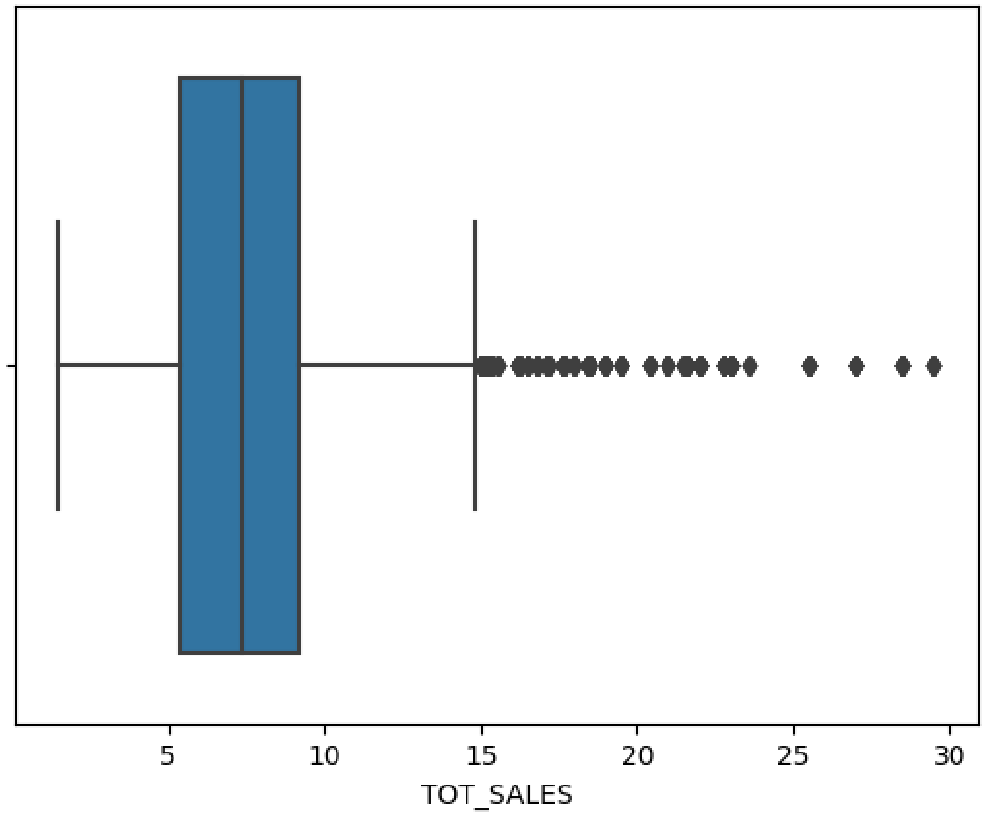
```
sns.displot(cdata['TOT_SALES'],kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x7d9c97337c70>
```



```
sns.boxplot(x = cdata['TOT_SALES'])
```

```
<Axes: xlabel='TOT_SALES'>
```

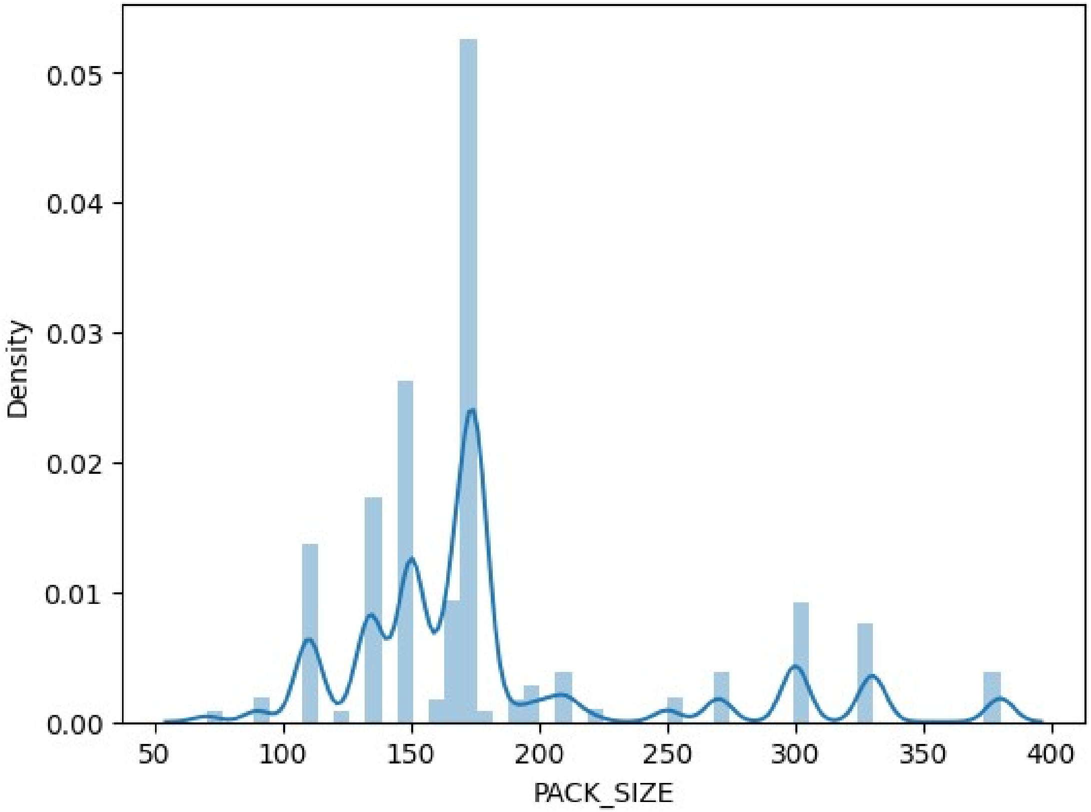


```
cdata['PACK_SIZE'].value_counts()
```

```
175    66390
150    43131
134    25102
110    22387
170    19983
165    15297
300    15166
330    12540
380     6416
270     6285
210     6272
200     4473
135     3257
250     3169
90      3008
190     2995
160     2970
220     1564
70      1507
180     1468
125     1454
Name: PACK_SIZE, dtype: int64
```

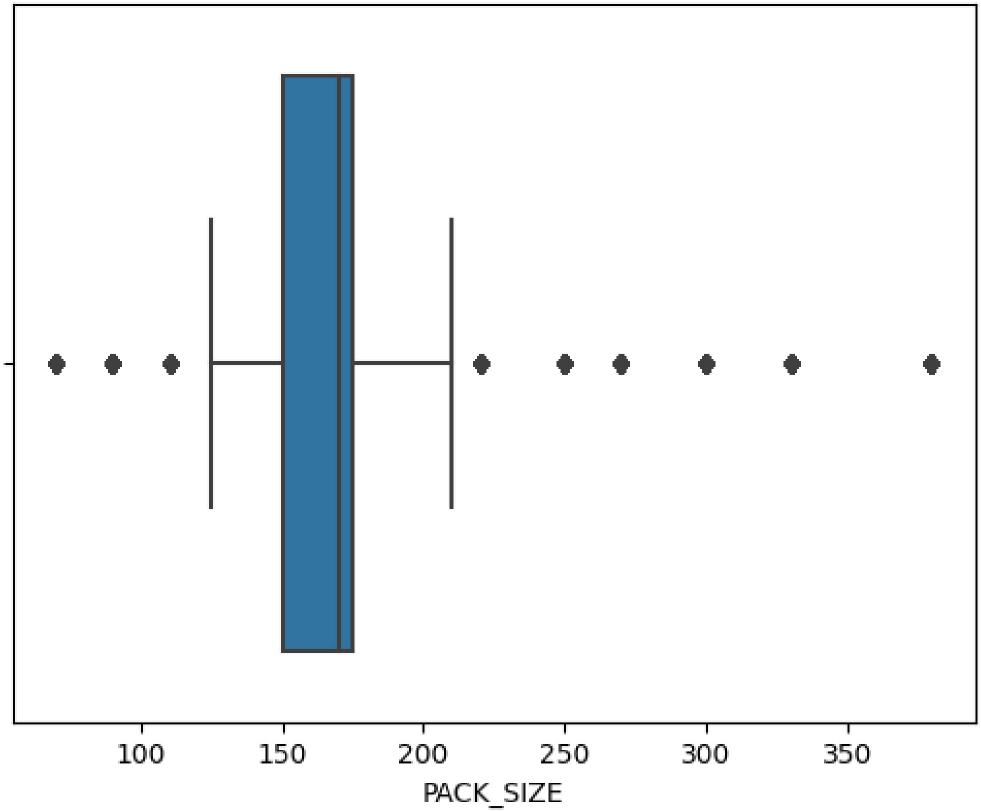
```
sns.distplot(cdata['PACK_SIZE'],kde=True)
```

```
<ipython-input-45-64805b1a6c05>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(cdata['PACK_SIZE'],kde=True)

<Axes: xlabel='PACK_SIZE', ylabel='Density'>
```



```
sns.boxplot(x=cdata['PACK_SIZE'])
```

```
<Axes: xlabel='PACK_SIZE'>
```



```
cdata['BRAND'].value_counts()
```

```
Kettle            41288
Smiths            31823
Doritos           28145
Pringles          25102
Infuzions         14201
Thins             14075
RRD               11894
WW                10320
Cobs               9693
Tostitos           9471
Twisties           9454
Old                9324
Tyrrells           6442
Grainwaves         6272
Natural            6050
Red Rock Deli      5885
Cheezels           4603
CCs                4551
Woolworths         4437
Sunbites           3008
Cheetos            2927
Burger             1564
```

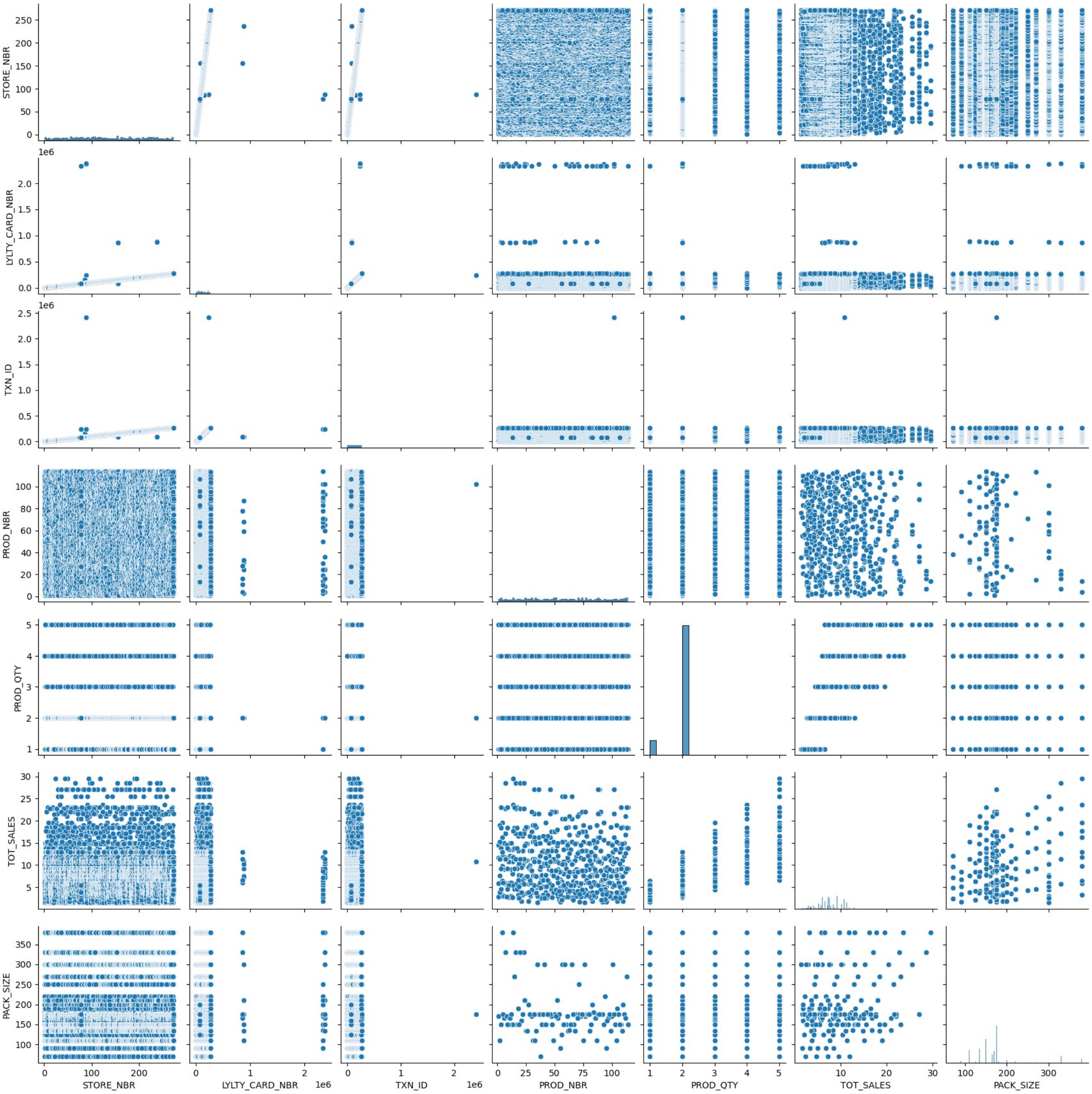
```
GrnWves            1468
NCC                1419
French             1418
Name: BRAND, dtype: int64
```

```
cdata['DATE'].nunique()
```

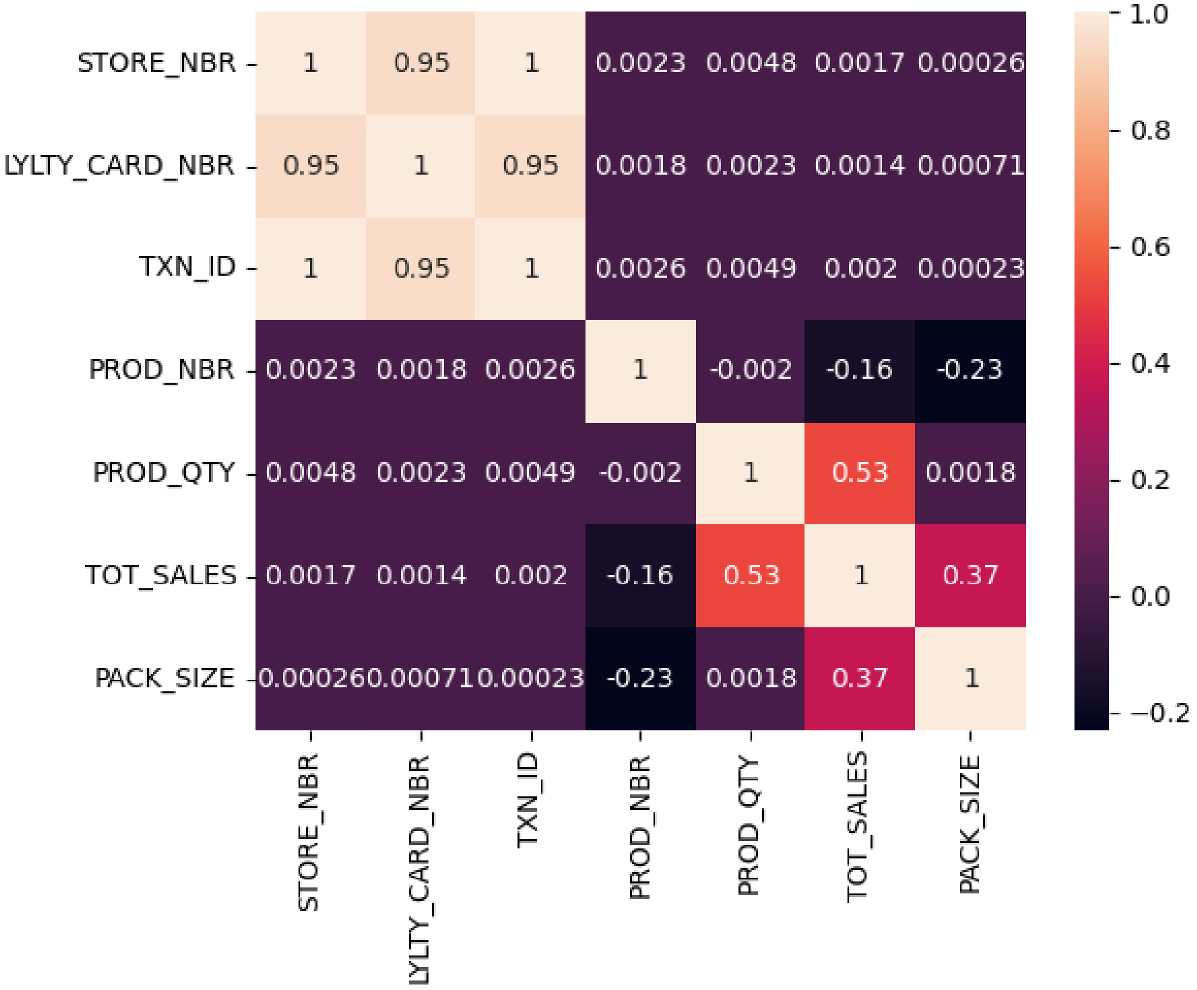
```
364
```

```
sns.pairplot(cdata)
```

```
<seaborn.axisgrid.PairGrid at 0x7d9c970239d0>
```



```
sns.heatmap(cdata.corr(),annot=True)

<ipython-input-50-2c870020c21c>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  sns.heatmap(cdata.corr(),annot=True)

<Axes: >
```



Total Sales by Date

```
pt =
pd.pivot_table(cdata,values='TOT_SALES',index='DATE',aggfunc='sum')
pt

            TOT_SALES
DATE
2018-07-01     5372.2
```

```
2018-07-02     5315.4
2018-07-03     5321.8
2018-07-04     5309.9
2018-07-05     5080.9
...               ...
2019-06-26     5305.0
2019-06-27     5202.8
2019-06-28     5299.6
2019-06-29     5497.6
2019-06-30     5423.4

[364 rows x 1 columns]
```

```
import plotly.express as px

fig = px.line(pt,y='TOT_SALES')
fig.show()
```

The total sales peak at the mid of Dec 2018, can be explained by year end purchases for Christmas and New Year parties, etc.

```
fig = px.scatter_3d(cdata,x='PROD_QTY',y='TOT_SALES',z='PACK_SIZE')
fig.show()
```

Total Sales by Brand

```
bt = cdata.groupby('BRAND').TOT_SALES.sum()
bt

BRAND
Burger              6831.0
CCs                18078.9
Cheetos            16884.5
Cheezels           40029.9
Cobs               70569.8
Doritos           240590.9
French              7929.0
Grainwaves         43048.8
GrnWves             8568.4
Infuzions          99047.6
Kettle            390239.8
NCC                 8046.0
Natural            34272.0
Old                90785.1
Pringles          177655.5
RRD                64954.5
Red Rock Deli      30091.5
Smiths            224660.2
Sunbites            9676.4
```

```
Thins                 88852.5
Tostitos              79789.6
Twisties              81522.1
Tyrrells              51647.4
WW                    35889.5
Woolworths            13454.1
Name: TOT_SALES, dtype: float64
```

```
type(bt)
```

```
pandas.core.series.Series
```

```
bt_df = bt.to_frame()
bt_df.reset_index(inplace=True)
```

```
bt_df.head()
```

```
      BRAND  TOT_SALES
0    Burger     6831.0
1       CCs    18078.9
2   Cheetos    16884.5
3  Cheezels    40029.9
4      Cobs    70569.8
```

```
fig = px.bar(bt_df,x='BRAND',y='TOT_SALES')
fig.show()
```

The highes total sales: Kettle, then Doritos and Smiths. Burger, French, NCC, Sunbites, Woolworths have the least total sales.

```
# Table 2
dfc = pd.read_csv('/content/QVI_purchase_behaviour.csv')
dfc.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
```

```
dfc.shape
```

```
(72637, 3)
```

```
dfc.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
```

```
dfc.dtypes
```

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

```
dfc.count()
```

```
LYLTY_CARD_NBR      72637
LIFESTAGE           72637
PREMIUM_CUSTOMER    72637
dtype: int64
```

```
dfc.isnull().sum()
```

```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

```
dfc.nunique()
```

```
LYLTY_CARD_NBR      72637
LIFESTAGE               7
PREMIUM_CUSTOMER        3
dtype: int64
```

```
len(dfc)==dfc['LYLTY_CARD_NBR'].nunique()
```

```
True
```

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

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

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

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

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

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

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

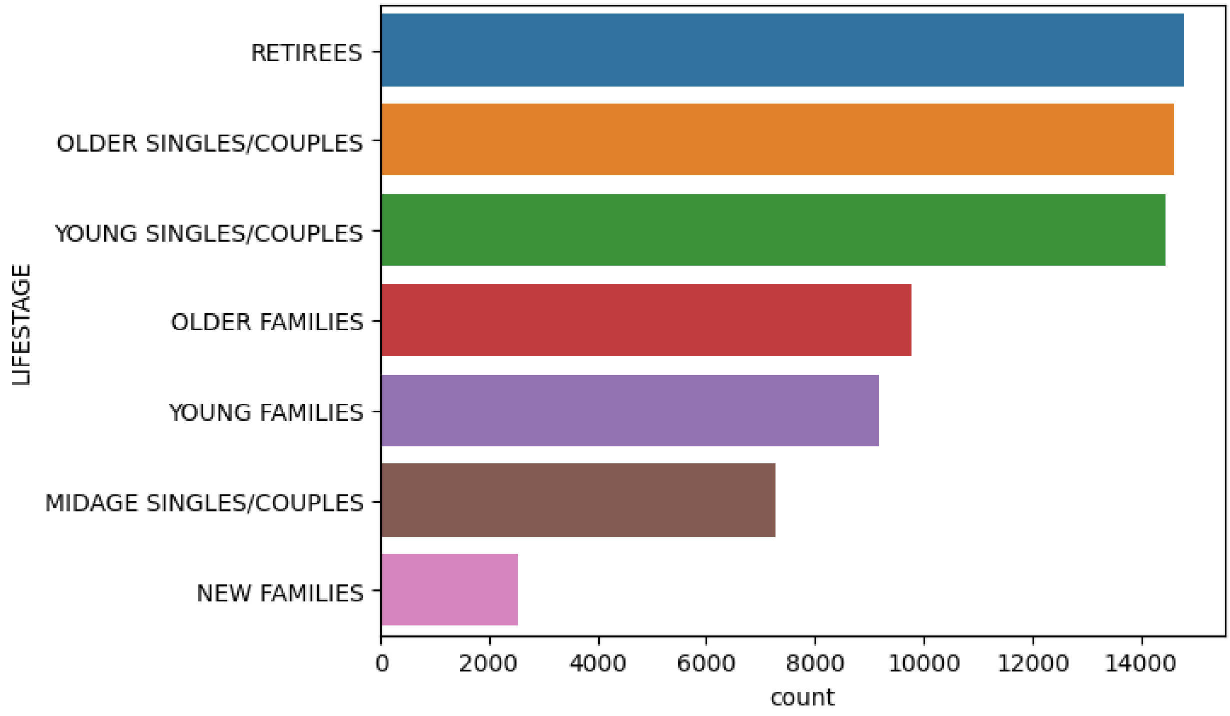
```
Mainstream    29245
Budget        24470
Premium       18922
Name: PREMIUM_CUSTOMER, dtype: int64
```

```
dfc['LIFESTAGE'].value_counts().index
```

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

```
sns.countplot(y =
dfc['LIFESTAGE'],order=dfc['LIFESTAGE'].value_counts().index)
```

```
<Axes: xlabel='count', ylabel='LIFESTAGE'>
```



```
dfc.describe(include='all')
```

```
        LYLTY_CARD_NBR LIFESTAGE PREMIUM_CUSTOMER
count     7.263700e+04     72637            72637
unique             NaN         7                3
top                NaN  RETIREES       Mainstream
freq               NaN     14805            29245
mean      1.361859e+05       NaN              NaN
std       8.989293e+04       NaN              NaN
min       1.000000e+03       NaN              NaN
25%       6.620200e+04       NaN              NaN
50%       1.340400e+05       NaN              NaN
75%       2.033750e+05       NaN              NaN
max       2.373711e+06       NaN              NaN
```

```
common_dfs = pd.merge(cdata,dfc)
common_dfs.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 SeaSalt          2        6.0        175   

1                     CCs Nacho Cheese         3        6.3        175   

2          WW Original Stacked Chips           2        3.8        160   

3                         CCs Original         2        2.1        175   

4  Smiths Crinkle Cut  Chips Chicken           2        2.9        170   


     BRAND              LIFESTAGE PREMIUM_CUSTOMER
0  Natural    YOUNG SINGLES/COUPLES          Premium
1      CCs  MIDAGE SINGLES/COUPLES           Budget
2       WW  MIDAGE SINGLES/COUPLES           Budget
3      CCs  MIDAGE SINGLES/COUPLES           Budget
4   Smiths  MIDAGE SINGLES/COUPLES           Budget
```

```
common_dfs.shape
```

```
(264834, 12)
```

```
cdata.shape, dfc.shape
```

```
((264834, 10), (72637, 3))
```

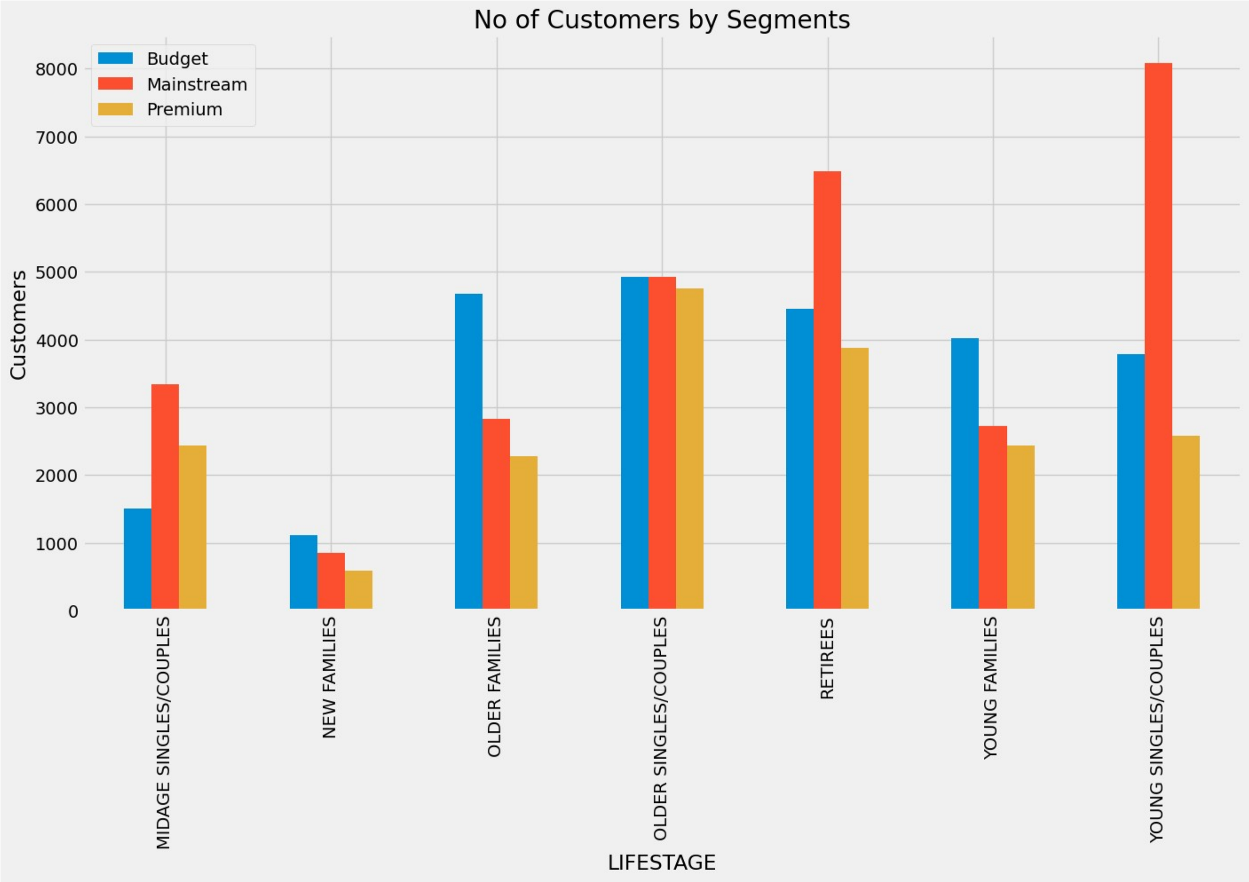
Customers in Each Segment

```
common_dfs['LYLTY_CARD_NBR'].nunique()

72636

common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).LYLTY_CARD_NBR.nu
nique()

PREMIUM_CUSTOMER  LIFESTAGE
Budget            MIDAGE SINGLES/COUPLES    1504
                  NEW FAMILIES              1112
                  OLDER FAMILIES            4675
                  OLDER SINGLES/COUPLES     4929
                  RETIREES                  4454
                  YOUNG FAMILIES            4017
                  YOUNG SINGLES/COUPLES     3779
Mainstream        MIDAGE SINGLES/COUPLES    3340
                  NEW FAMILIES               849
                  OLDER FAMILIES            2831
                  OLDER SINGLES/COUPLES     4930
                  RETIREES                  6479
                  YOUNG FAMILIES            2728
                  YOUNG SINGLES/COUPLES     8088
Premium           MIDAGE SINGLES/COUPLES    2431
                  NEW FAMILIES               588
                  OLDER FAMILIES            2273
                  OLDER SINGLES/COUPLES     4750
                  RETIREES                  3872
                  YOUNG FAMILIES            2433
                  YOUNG SINGLES/COUPLES     2574
Name: LYLTY_CARD_NBR, dtype: int64

cust =
pd.DataFrame(common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).LYLT
Y_CARD_NBR.nunique())
cust.rename(columns={'LYLTY_CARD_NBR':'Customers'},inplace=True)
cust.sort_values(by='Customers',ascending=False,inplace=True)
cust

                                         Customers
PREMIUM_CUSTOMER LIFESTAGE
Mainstream       YOUNG SINGLES/COUPLES        8088
                 RETIREES                     6479
                 OLDER SINGLES/COUPLES        4930
Budget           OLDER SINGLES/COUPLES        4929
Premium          OLDER SINGLES/COUPLES        4750
Budget           OLDER FAMILIES               4675
```

```
                 RETIREES                         4454
                 YOUNG FAMILIES                   4017
Premium          RETIREES                         3872
Budget           YOUNG SINGLES/COUPLES            3779
Mainstream       MIDAGE SINGLES/COUPLES           3340
                 OLDER FAMILIES                   2831
                 YOUNG FAMILIES                   2728
Premium          YOUNG SINGLES/COUPLES            2574
                 YOUNG FAMILIES                   2433
                 MIDAGE SINGLES/COUPLES           2431
                 OLDER FAMILIES                   2273
Budget           MIDAGE SINGLES/COUPLES           1504
                 NEW FAMILIES                     1112
Mainstream       NEW FAMILIES                      849
Premium          NEW FAMILIES                      588
```

```
cust =
pd.DataFrame(common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).LYLT
Y_CARD_NBR.nunique())
plt.style.use('fivethirtyeight')
cust.unstack().plot(kind='bar',figsize=(15,8),title='No of Customers
by Segments')
plt.ylabel('Customers')
plt.legend(['Budget','Mainstream','Premium'])
```

```
<matplotlib.legend.Legend at 0x7d9c8e5e95a0>
```



Young singles/couples and retirees are mainstream. This is not the case for older or new families.

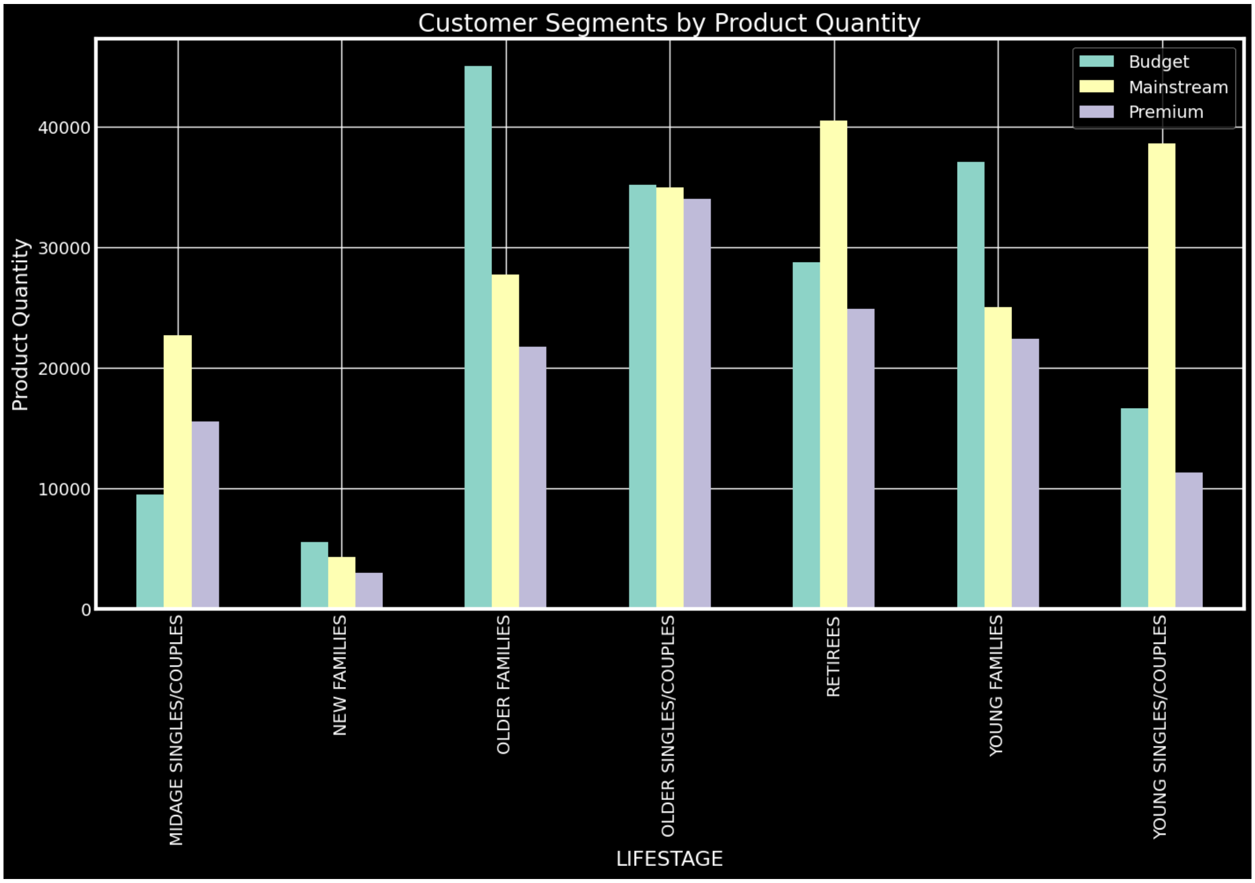
```
qty =
pd.DataFrame(common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).PROD
_QTY.sum())

qty.sort_values(by='PROD_QTY',ascending=False, inplace=True)
qty
```

```
                                        PROD_QTY
PREMIUM_CUSTOMER LIFESTAGE
Budget           OLDER FAMILIES            45065
Mainstream       RETIREES                  40518
                 YOUNG SINGLES/COUPLES     38632
Budget           YOUNG FAMILIES            37111
                 OLDER SINGLES/COUPLES     35220
Mainstream       OLDER SINGLES/COUPLES     34997
Premium          OLDER SINGLES/COUPLES     33986
Budget           RETIREES                  28764
Mainstream       OLDER FAMILIES            27756
                 YOUNG FAMILIES            25044
Premium          RETIREES                  24884
```

```
Mainstream        MIDAGE SINGLES/COUPLES     22699
Premium           YOUNG FAMILIES             22406
                  OLDER FAMILIES             21771
Budget            YOUNG SINGLES/COUPLES      16671
Premium           MIDAGE SINGLES/COUPLES     15526
                  YOUNG SINGLES/COUPLES      11331
Budget            MIDAGE SINGLES/COUPLES      9496
                  NEW FAMILIES                5571
Mainstream        NEW FAMILIES                4319
Premium           NEW FAMILIES                2957
```

```
product_plot =
pd.DataFrame(common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).PROD
_QTY.sum())
plt.style.use('dark_background')
product_plot.unstack().plot(kind='bar', figsize=(15,8),
title='Customer Segments by Product Quantity')
plt.ylabel("Product Quantity")
plt.legend(['Budget', 'Mainstream', 'Premium'])
```

```
<matplotlib.legend.Legend at 0x7d9c8e5a4fa0>
```



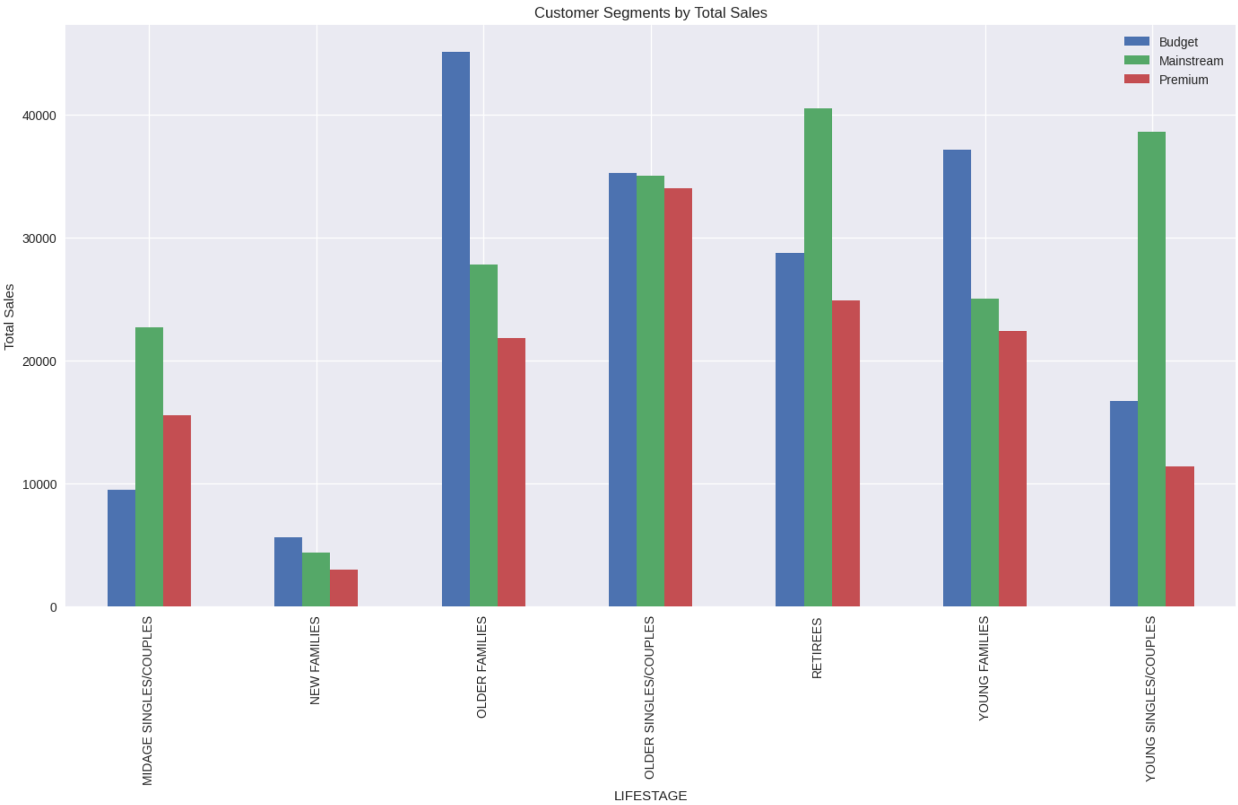
The budget older families bought the highest quantity of the product, despite their number being low. New families bought the least products. Overall, budget has the highest popularity, with 3 exceptions.

```
cust =
pd.DataFrame(common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).TOT_
SALES.sum())
cust.rename(columns={'TOT_SALES':'Total Sales'},inplace=True)
cust.sort_values(by='Total Sales',ascending=False, inplace=True)
cust
```

```
                                                Total Sales
PREMIUM_CUSTOMER LIFESTAGE
Budget           OLDER FAMILIES                   168363.25
Mainstream       YOUNG SINGLES/COUPLES            157621.60
                 RETIREES                         155677.05
Budget           YOUNG FAMILIES                   139345.85
                 OLDER SINGLES/COUPLES            136769.80
Mainstream       OLDER SINGLES/COUPLES            133393.80
Premium          OLDER SINGLES/COUPLES            132263.15
Budget           RETIREES                         113147.80
Mainstream       OLDER FAMILIES                   103445.55
Premium          RETIREES                          97646.05
Mainstream       YOUNG FAMILIES                    92788.75
                 MIDAGE SINGLES/COUPLES            90803.85
Premium          YOUNG FAMILIES                    84025.50
                 OLDER FAMILIES                    80658.40
Budget           YOUNG SINGLES/COUPLES             61141.60
Premium          MIDAGE SINGLES/COUPLES            58432.65
                 YOUNG SINGLES/COUPLES             41642.10
Budget           MIDAGE SINGLES/COUPLES            35514.80
                 NEW FAMILIES                      21928.45
Mainstream       NEW FAMILIES                      17013.90
Premium          NEW FAMILIES                      11491.10
```

```
total_plot =
pd.DataFrame(common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).TOT_
SALES.sum())
plt.style.use('seaborn-v0_8')
product_plot.unstack().plot(kind='bar', figsize=(15,8),
title='Customer Segments by Total Sales')
plt.ylabel("Total Sales")
plt.legend(['Budget', 'Mainstream', 'Premium'])
```

```
<matplotlib.legend.Legend at 0x7d9c8a3c1600>
```



Note that the total sales trends are similar to that of the product quantities.

```
# Customer Segments based on Average Product Quantities per Customer
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).PROD_QTY.sum()

LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget               9496
                        Mainstream          22699
                        Premium             15526
NEW FAMILIES            Budget               5571
                        Mainstream           4319
                        Premium              2957
OLDER FAMILIES          Budget              45065
                        Mainstream          27756
                        Premium             21771
OLDER SINGLES/COUPLES   Budget              35220
                        Mainstream          34997
                        Premium             33986
RETIREES                Budget              28764
                        Mainstream          40518
                        Premium             24884
YOUNG FAMILIES          Budget              37111
                        Mainstream          25044
                        Premium             22406
YOUNG SINGLES/COUPLES   Budget              16671
```

```
                        Mainstream          38632
                        Premium             11331
Name: PROD_QTY, dtype: int64
```

```
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).LYLTY_CARD_NBR.nunique()
```

```
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             2273
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
Name: LYLTY_CARD_NBR, dtype: int64
```

```
avg_units_per_customer =
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).PROD_QTY.sum()/
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).LYLTY_CARD_NBR.nunique()
avg_units_per_customer
```

```
LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget              6.313830
                        Mainstream          6.796108
                        Premium             6.386672
NEW FAMILIES            Budget              5.009892
                        Mainstream          5.087161
                        Premium             5.028912
OLDER FAMILIES          Budget              9.639572
                        Mainstream          9.804309
                        Premium             9.578091
OLDER SINGLES/COUPLES   Budget              7.145466
                        Mainstream          7.098783
                        Premium             7.154947
```

```
RETIREES                Budget              6.458015
                        Mainstream          6.253743
                        Premium             6.426653
YOUNG FAMILIES          Budget              9.238486
                        Mainstream          9.180352
                        Premium             9.209207
YOUNG SINGLES/COUPLES   Budget              4.411485
                        Mainstream          4.776459
                        Premium             4.402098
dtype: float64
```

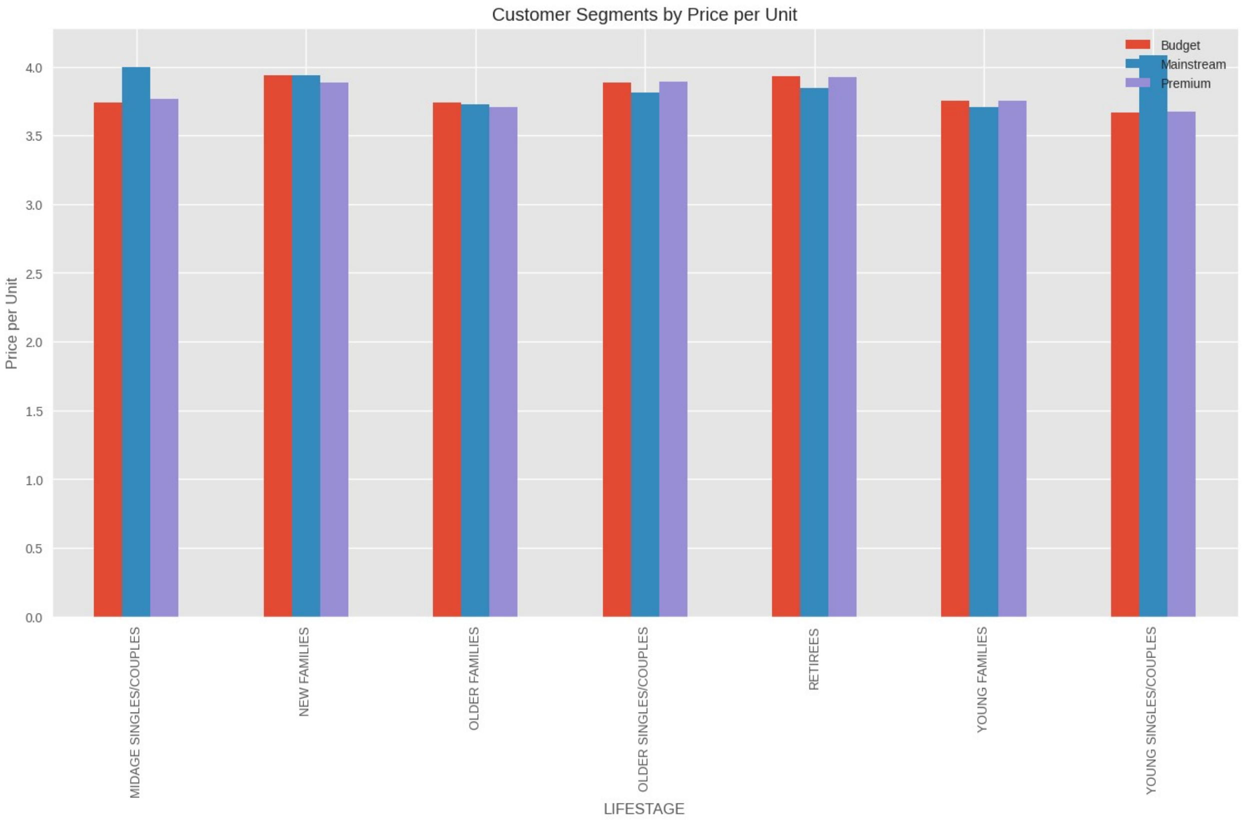
Hence, older and young families are contributing more when it comes to buying number of chips per customer.

```
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).TOT_SALES.sum()

LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget               35514.80
                        Mainstream           90803.85
                        Premium              58432.65
NEW FAMILIES            Budget               21928.45
                        Mainstream           17013.90
                        Premium              11491.10
OLDER FAMILIES          Budget              168363.25
                        Mainstream          103445.55
                        Premium              80658.40
OLDER SINGLES/COUPLES   Budget              136769.80
                        Mainstream          133393.80
                        Premium             132263.15
RETIREES                Budget              113147.80
                        Mainstream          155677.05
                        Premium              97646.05
YOUNG FAMILIES          Budget              139345.85
                        Mainstream           92788.75
                        Premium              84025.50
YOUNG SINGLES/COUPLES   Budget               61141.60
                        Mainstream          157621.60
                        Premium              41642.10
Name: TOT_SALES, dtype: float64

price_per_unit =
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).TOT_SALES.sum()/
common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).PROD_QTY.sum()
price_per_unit = pd.DataFrame(price_per_unit, columns=['Price per
Unit'])
price_per_unit.sort_values(by='Price per Unit',ascending=False,
inplace=True)
price_per_unit
```

```
                                           Price per Unit
LIFESTAGE              PREMIUM_CUSTOMER
YOUNG SINGLES/COUPLES  Mainstream                4.080079
MIDAGE SINGLES/COUPLES Mainstream                4.000346
NEW FAMILIES           Mainstream                3.939315
                       Budget                    3.936178
RETIREES               Budget                    3.933660
                       Premium                   3.924050
OLDER SINGLES/COUPLES  Premium                   3.891695
NEW FAMILIES           Premium                   3.886067
OLDER SINGLES/COUPLES  Budget                    3.883299
RETIREES               Mainstream                3.842170
OLDER SINGLES/COUPLES  Mainstream                3.811578
MIDAGE SINGLES/COUPLES Premium                   3.763535
YOUNG FAMILIES         Budget                    3.754840
                       Premium                   3.750134
MIDAGE SINGLES/COUPLES Budget                    3.739975
OLDER FAMILIES         Budget                    3.736009
                       Mainstream                3.726962
YOUNG FAMILIES         Mainstream                3.705029
OLDER FAMILIES         Premium                   3.704855
YOUNG SINGLES/COUPLES  Premium                   3.675060
                       Budget                    3.667542
```

```
price_per_unit.unstack().plot(kind='bar', figsize=(15,8),
title='Customer Segments by Price per Unit')
plt.style.use('bmh')
plt.ylabel("Price per Unit")
plt.legend(['Budget', 'Mainstream', 'Premium'])
```

```
<matplotlib.legend.Legend at 0x7d9c88bb3eb0>
```



Mainstream mid-age and young singles and couples are more willing to pay more per packet chips compared to their budget and premium counterparts, probably for entertainment purposes.

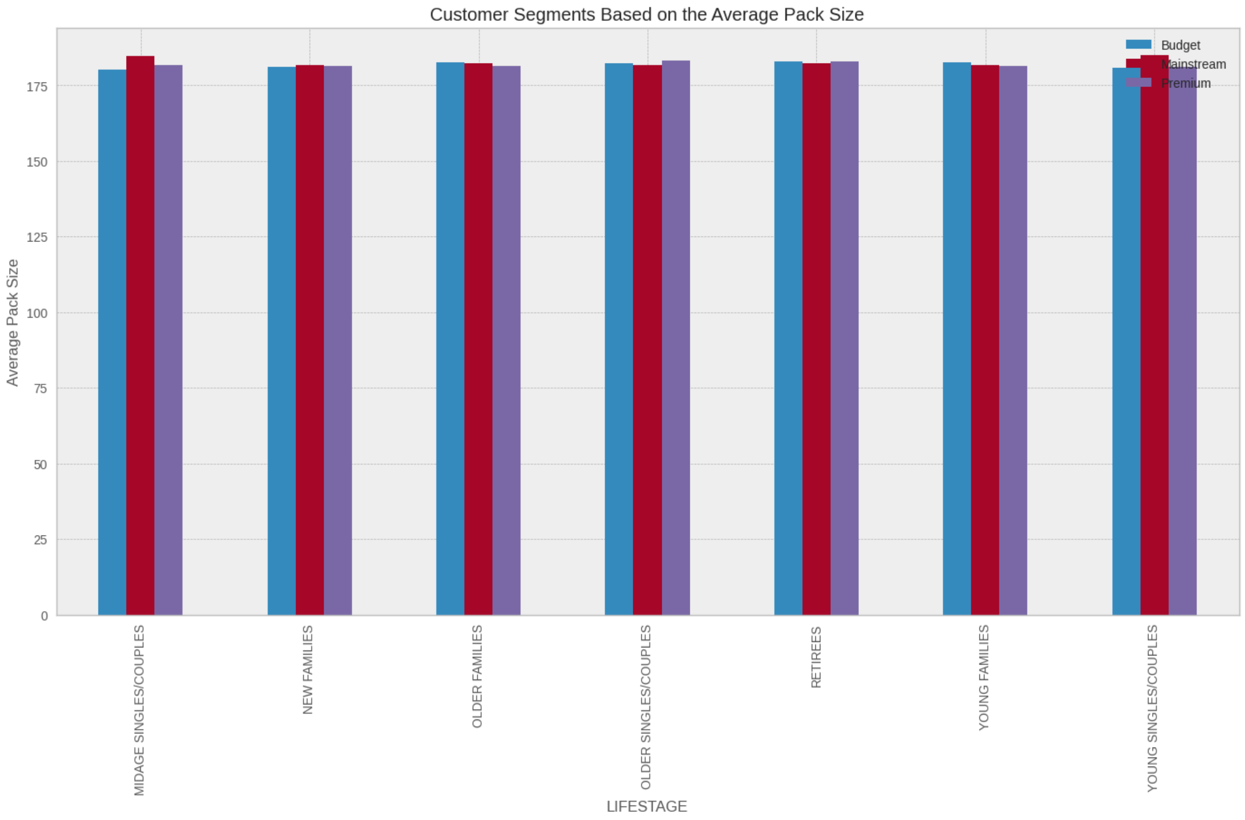
```
cust_ps =
pd.DataFrame(common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).PACK
_SIZE.mean())
cust_ps.rename(columns={'PACK_SIZE':'Avg Pack Size'},inplace=True)
cust_ps.sort_values(by='Avg Pack Size',ascending=False, inplace=True)
cust_ps
```

```
                                         Avg Pack Size
PREMIUM_CUSTOMER LIFESTAGE
Mainstream       YOUNG SINGLES/COUPLES      184.828330
                 MIDAGE SINGLES/COUPLES     184.582786
Premium          OLDER SINGLES/COUPLES      183.254534
                 RETIREES                   182.975260
Budget           RETIREES                   182.960200
                 YOUNG FAMILIES             182.490901
                 OLDER FAMILIES             182.487219
                 OLDER SINGLES/COUPLES      182.289183
Mainstream       RETIREES                   182.289062
                 OLDER FAMILIES             182.175021
                 NEW FAMILIES               181.699355
                 OLDER SINGLES/COUPLES      181.642101
```

```
Premium          MIDAGE SINGLES/COUPLES     181.577897
Mainstream       YOUNG FAMILIES             181.536531
Premium          OLDER FAMILIES             181.432618
                 YOUNG FAMILIES             181.351985
                 NEW FAMILIES               181.286973
Budget           NEW FAMILIES               181.161730
Premium          YOUNG SINGLES/COUPLES      181.056042
Budget           YOUNG SINGLES/COUPLES      180.694438
                 MIDAGE SINGLES/COUPLES     180.187450
```

```
ps_plot =
pd.DataFrame(common_dfs.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).PACK
_SIZE.mean())
ps_plot.unstack().plot(kind='bar', figsize=(15,8), title='Customer
Segments Based on the Average Pack Size')
plt.ylabel("Average Pack Size")
plt.legend(['Budget', 'Mainstream', 'Premium'])
```

```
<matplotlib.legend.Legend at 0x7d9c88a29870>
```



Mainstream mid-age and young singles and couples are buying the highest average sized pack of chips among all customer segments.

```
common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).BRAND.value_count
s()
```

```
PREMIUM_CUSTOMER  LIFESTAGE               BRAND
Budget            MIDAGE SINGLES/COUPLES  Kettle        713
                                          Smiths        633
                                          Doritos       533
                                          Pringles      449
                                          Infuzions     281
                                                       ...
Premium           YOUNG SINGLES/COUPLES   Cheetos        80
                                          Burger         57
                                          GrnWves        48
                                          French         45
                                          NCC            44
Name: BRAND, Length: 525, dtype: int64
```

```
cust_b =
pd.DataFrame(common_dfs.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).BRAN
D.value_counts())
cust_b
```

```
                                                   BRAND
PREMIUM_CUSTOMER LIFESTAGE              BRAND
Budget           MIDAGE SINGLES/COUPLES Kettle       713
                                        Smiths       633
                                        Doritos      533
                                        Pringles     449
                                        Infuzions    281
...                                                  ...
Premium          YOUNG SINGLES/COUPLES  Cheetos       80
                                        Burger        57
                                        GrnWves       48
                                        French        45
                                        NCC           44

[525 rows x 1 columns]
```

```
cust_b.columns
```

```
Index(['BRAND'], dtype='object')
```

```
cust_b.rename(columns={'BRAND':'Counts'}, inplace=True)
cust_b.columns
```

```
Index(['Counts'], dtype='object')
```

```
cust_b.index
```

```
MultiIndex([( 'Budget', 'MIDAGE SINGLES/COUPLES',        'Kettle'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',        'Smiths'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',       'Doritos'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',      'Pringles'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',     'Infuzions'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',         'Thins'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',           'RRD'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',            'WW'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',          'Cobs'),
            ( 'Budget', 'MIDAGE SINGLES/COUPLES',      'Twisties'),
            ...
            ('Premium',  'YOUNG SINGLES/COUPLES', 'Red Rock Deli'),
            ('Premium',  'YOUNG SINGLES/COUPLES',    'Grainwaves'),
            ('Premium',  'YOUNG SINGLES/COUPLES',      'Tyrrells'),
            ('Premium',  'YOUNG SINGLES/COUPLES',      'Cheezels'),
            ('Premium',  'YOUNG SINGLES/COUPLES',      'Sunbites'),
            ('Premium',  'YOUNG SINGLES/COUPLES',       'Cheetos'),
            ('Premium',  'YOUNG SINGLES/COUPLES',        'Burger'),
            ('Premium',  'YOUNG SINGLES/COUPLES',       'GrnWves'),
            ('Premium',  'YOUNG SINGLES/COUPLES',        'French'),
            ('Premium',  'YOUNG SINGLES/COUPLES',           'NCC')],
           names=['PREMIUM_CUSTOMER', 'LIFESTAGE', 'BRAND'],
length=525)
```

```python
len(common_dfs['PREMIUM_CUSTOMER'].unique()),len(common_dfs['LIFESTAGE'].unique()), len(common_dfs['BRAND'].unique()), len(cust_b)
```

```
(3, 7, 25, 525)
```

3 unique values in PREMIUM_CUSTOMER, 7 in LIFESTAGE, 25 in BRAND, and the length of the cust_b dataframe is 525. T.N: 525/25 = 7*3, i.e. every brand of chips in different combinations of PREMIUM_CUSTOMER and LIFESTAGE and at the top the most popular brand in that segment exist

```python
cust_b.head()
```

```
                                                   Counts
PREMIUM_CUSTOMER LIFESTAGE              BRAND            
Budget           MIDAGE SINGLES/COUPLES Kettle        713
                                        Smiths        633
                                        Doritos       533
                                        Pringles      449
                                        Infuzions     281
```

```python
cust_b.tail()
```

```
                                                Counts
PREMIUM_CUSTOMER LIFESTAGE             BRAND          
Premium          YOUNG SINGLES/COUPLES Cheetos      80
                                       Burger       57
                                       GrnWves      48
                                       French       45
                                       NCC          44
```

```python
cust_b.index[0]
```

```
('Budget', 'MIDAGE SINGLES/COUPLES', 'Kettle')
```

```
cust_b.index[25]
```

```
('Budget', 'NEW FAMILIES', 'Kettle')
```

```
cust_b.index[50]
```

```
('Budget', 'OLDER FAMILIES', 'Kettle')
```

```
l = []
for i in range(21):
  l.append(cust_b.index[25*i])
print(l)
```

```
[('Budget', 'MIDAGE SINGLES/COUPLES', 'Kettle'), ('Budget', 'NEW
FAMILIES', 'Kettle'), ('Budget', 'OLDER FAMILIES', 'Kettle'),
('Budget', 'OLDER SINGLES/COUPLES', 'Kettle'), ('Budget', 'RETIREES',
'Kettle'), ('Budget', 'YOUNG FAMILIES', 'Kettle'), ('Budget', 'YOUNG
SINGLES/COUPLES', 'Smiths'), ('Mainstream', 'MIDAGE SINGLES/COUPLES',
'Kettle'), ('Mainstream', 'NEW FAMILIES', 'Kettle'), ('Mainstream',
'OLDER FAMILIES', 'Kettle'), ('Mainstream', 'OLDER SINGLES/COUPLES',
'Kettle'), ('Mainstream', 'RETIREES', 'Kettle'), ('Mainstream', 'YOUNG
FAMILIES', 'Kettle'), ('Mainstream', 'YOUNG SINGLES/COUPLES',
'Kettle'), ('Premium', 'MIDAGE SINGLES/COUPLES', 'Kettle'),
('Premium', 'NEW FAMILIES', 'Kettle'), ('Premium', 'OLDER FAMILIES',
'Smiths'), ('Premium', 'OLDER SINGLES/COUPLES', 'Kettle'), ('Premium',
'RETIREES', 'Kettle'), ('Premium', 'YOUNG FAMILIES', 'Kettle'),
('Premium', 'YOUNG SINGLES/COUPLES', 'Kettle')]
```

```
len(l)
```

```
21
```

```
customers_best_brand = pd.DataFrame(l, columns=['PREMIUM_CUSTOMER',
'LIFESTAGE', 'Best Brand'])
customers_best_brand
```

```
   PREMIUM_CUSTOMER               LIFESTAGE Best Brand
0            Budget  MIDAGE SINGLES/COUPLES     Kettle
1            Budget            NEW FAMILIES     Kettle
2            Budget          OLDER FAMILIES     Kettle
3            Budget   OLDER SINGLES/COUPLES     Kettle
4            Budget                RETIREES     Kettle
5            Budget          YOUNG FAMILIES     Kettle
6            Budget   YOUNG SINGLES/COUPLES     Smiths
7        Mainstream  MIDAGE SINGLES/COUPLES     Kettle
8        Mainstream            NEW FAMILIES     Kettle
9        Mainstream          OLDER FAMILIES     Kettle
10       Mainstream   OLDER SINGLES/COUPLES     Kettle
11       Mainstream                RETIREES     Kettle
```

```
12       Mainstream          YOUNG FAMILIES     Kettle
13       Mainstream   YOUNG SINGLES/COUPLES     Kettle
14          Premium  MIDAGE SINGLES/COUPLES     Kettle
15          Premium            NEW FAMILIES     Kettle
16          Premium          OLDER FAMILIES     Smiths
17          Premium   OLDER SINGLES/COUPLES     Kettle
18          Premium                RETIREES     Kettle
19          Premium          YOUNG FAMILIES     Kettle
20          Premium   YOUNG SINGLES/COUPLES     Kettle
```