```
#importing libraries

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


customerData = pd.read_csv('/content/QVI_purchase_behaviour.csv')



transactionData = pd.read_csv('/content/QVI_transaction_data.csv')


customerData.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

```
transactionData.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips	2	

```
transactionData.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
```

```
transactionData['DATE'] = pd.to_datetime(transactionData['DATE'], origin="1899-12-30", unit = 'D')


transactionData['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]
```

```
transactionData['PROD_NAME'].describe()
```

```
count                                 264836
unique                                   114
top       Kettle Mozzarella   Basil & Pesto 175g
```

```
freq                                3304
Name: PROD_NAME, dtype: object
```

Converting the PROD_NAME column to column with just the name of the product.

```
Prod_Name_Only = transactionData['PROD_NAME'].str.replace('[0-9]', ' ').str.replace('[gG]', ' ').str.replace('[^\w]',
```

```
<ipython-input-12-0ebc3398f8fe>:1: FutureWarning: The default value of regex will change from True to False in a f
  Prod_Name_Only = transactionData['PROD_NAME'].str.replace('[0-9]', ' ').str.replace('[gG]', ' ').str.replace('[^
```

```
Prod_Name_Only.head()
```

```
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]
Name: PROD_NAME, dtype: object
```

```
Prod_Total = pd.value_counts([word for name in Prod_Name_Only
                              for word in name])
```

```
Prod_Total.head()
```

```
Chips     49770
Kettle    41288
Smiths    28860
Salt      27976
Cheese    27890
dtype: int64
```

```
Prod_Total = Prod_Total.sort_values(ascending = False)
```

It looks like the products also include the Salsa products but as we are only looking for at the Chips data we will remove the Salsa Products.

```
print(Prod_Total['Salsa'])
```

```
18094
```

```
SalsaRows = transactionData['PROD_NAME'].str.contains('[Ss]alsa')
```

```
SalsaRows.head(10)
```

```
0    False
1    False
2    False
3    False
4    False
5     True
6    False
7    False
8    False
9    False
Name: PROD_NAME, dtype: bool
```

```
transactionData = transactionData[~SalsaRows]
```

```
transactionData.head(10)
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno	3	

Now the data only has the information about chips.

```
transactionData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 246742 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype         
---  ------          --------------   -----         
 0   DATE            246742 non-null  datetime64[ns]
 1   STORE_NBR       246742 non-null  int64         
 2   LYLTY_CARD_NBR  246742 non-null  int64         
 3   TXN_ID          246742 non-null  int64         
 4   PROD_NBR        246742 non-null  int64         
 5   PROD_NAME       246742 non-null  object        
 6   PROD_QTY        246742 non-null  int64         
 7   TOT_SALES       246742 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 16.9+ MB
```

```
transactionData.describe()
```

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	
count	246742.000000	2.467420e+05	2.467420e+05	246742.000000	246742.000000	2467
mean	135.051098	1.355310e+05	1.351311e+05	56.351789	1.908062	
std	76.787096	8.071528e+04	7.814772e+04	33.695428	0.659831	
min	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	
25%	70.000000	7.001500e+04	6.756925e+04	26.000000	2.000000	
50%	130.000000	1.303670e+05	1.351830e+05	53.000000	2.000000	
75%	203.000000	2.030840e+05	2.026538e+05	87.000000	2.000000	
max	272.000000	2.373711e+06	2.415841e+06	114.000000	200.000000	6

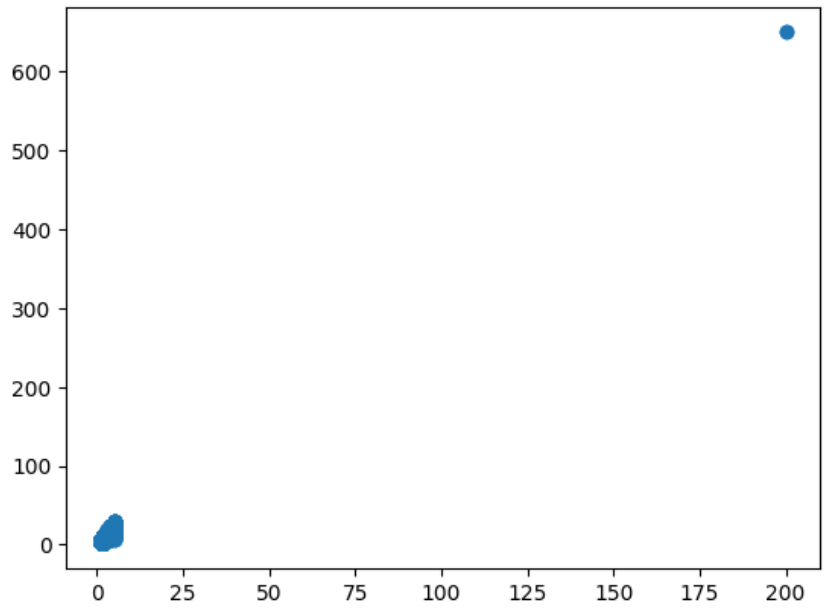
```
transactionData.isnull().sum()
```

```
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 is no null values in any rows. But the PROD_QTY and TOT_SALES is showing an outlier.

```
plt.scatter(transactionData['PROD_QTY'], transactionData['TOT_SALES'])
```

```
<matplotlib.collections.PathCollection at 0x7892de6a70d0>
```



There is definitely an outlier in PROD_QTY so let's look in it further.

```
transactionData.loc[transactionData['PROD_QTY'] == 200]
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme	200	6

There are 2 transaction by same customer so we are going to remove it.

```
PROD_QTY_OUT = transactionData.loc[transactionData['PROD_QTY'] == 200 ]
PROD_QTY_OUT.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme	200	6

```
transactionData =transactionData.drop(index =[69762, 69763])
```

```
transactionData.loc[transactionData['PROD_QTY'] == 200]
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES

Now that looks better.

```
transaction_date = transactionData.groupby(['DATE']).count()
```

```
transaction_date.describe()
```

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT
count	364.000000	364.000000	364.000000	364.000000	364.000000	364.000000	364.
mean	677.857143	677.857143	677.857143	677.857143	677.857143	677.857143	677.
std	33.687536	33.687536	33.687536	33.687536	33.687536	33.687536	33.
min	607.000000	607.000000	607.000000	607.000000	607.000000	607.000000	607.
25%	658.000000	658.000000	658.000000	658.000000	658.000000	658.000000	658.
50%	674.000000	674.000000	674.000000	674.000000	674.000000	674.000000	674.
75%	694.250000	694.250000	694.250000	694.250000	694.250000	694.250000	694.
max	865.000000	865.000000	865.000000	865.000000	865.000000	865.000000	865.

The data is for a whole year but total rows are 364 that means there is a missing value. So let's look for it.

```
transactionData.head(10)
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno	3	

First we will the find the missing date then add it to our data.

```
date_pattern = pd.date_range(start = '2018-07-01', end = '2019-06-30')
print(date_pattern)
```

```
DatetimeIndex(['2018-07-01', '2018-07-02', '2018-07-03', '2018-07-04',
               '2018-07-05', '2018-07-06', '2018-07-07', '2018-07-08',
               '2018-07-09', '2018-07-10',
               ...
               '2019-06-21', '2019-06-22', '2019-06-23', '2019-06-24',
               '2019-06-25', '2019-06-26', '2019-06-27', '2019-06-28',
               '2019-06-29', '2019-06-30'],
              dtype='datetime64[ns]', length=365, freq='D')
```

```
new_dates = transaction_date.reindex(date_pattern)
```

```
new_dates
```

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
2018-07-01	663.0	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	650.0
2018-07-03	674.0	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	669.0
2018-07-05	660.0	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	657.0
2019-06-27	669.0	669.0	669.0	669.0	669.0	669.0	669.0

```
new_dates.index.difference(transactionData['DATE'])
```

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

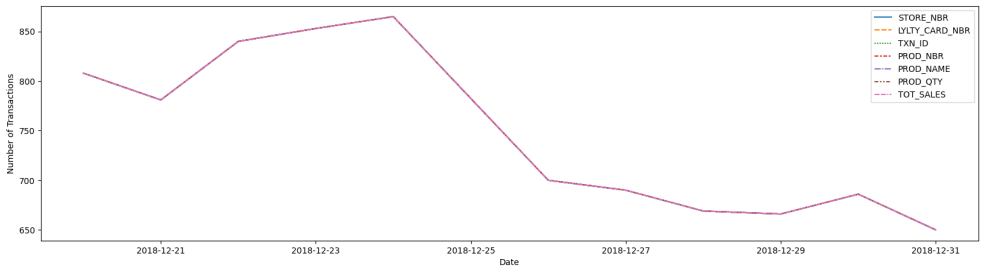
```
new_dates.loc['2018-12-25', :]
```

```
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
```

As the difference is in december so we will look only at december dates.

```
fig = plt.subplots(figsize=(20,5))
sns.lineplot(data= new_dates.loc['2018-12-20':'2018-12-31', :])
plt.xlabel("Date")
plt.ylabel("Number of Transactions")
```

```
Text(0, 0.5, 'Number of Transactions')
```



The reason for this missing is that it is Christmas holiday so it is not treated as outlier.

Now looking at pack sizes

```
pack_size = transactionData['PROD_NAME'].str.extract('([0-9]+)').astype('float')
```

```
pack_size.head()
```

	0
0	175.0
1	175.0
2	170.0
3	175.0
4	150.0

```
pack_size.describe()
```

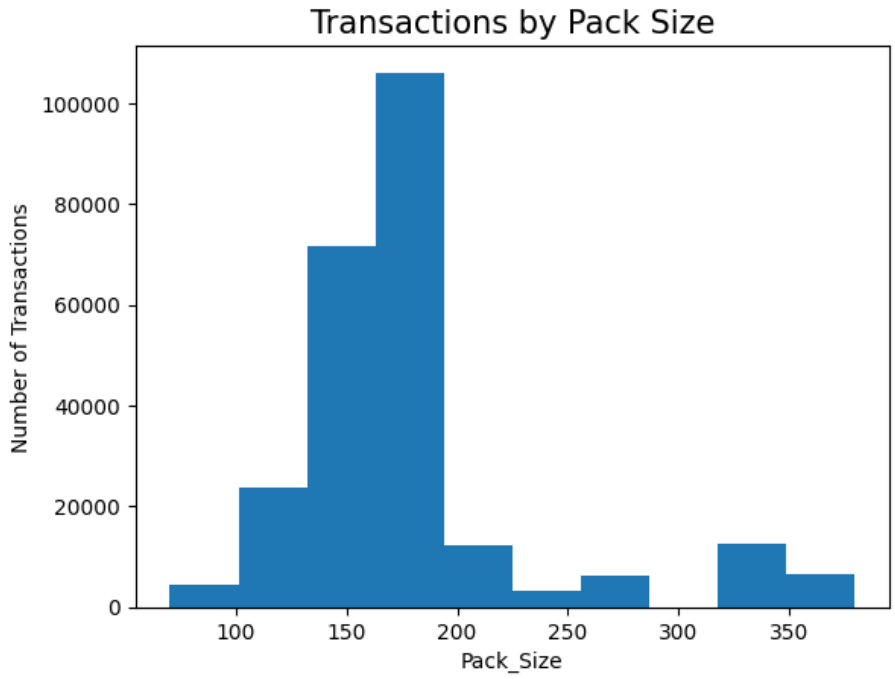
	0
count	246740.000000
mean	175.583521
std	59.432118
min	70.000000
25%	150.000000
50%	170.000000
75%	175.000000
max	380.000000

```
transactionData['pack_size'] = pack_size
```

Looks Sensible

```
plt.hist(x= transactionData['pack_size'],bins = 10)
plt.title("Transactions by Pack Size",fontsize=15)
plt.xlabel("Pack_Size",fontsize=10)
plt.ylabel("Number of Transactions",fontsize=10)
```

```
Text(0, 0.5, 'Number of Transactions')
```




```
transactionData["brand_name"] = transactionData["PROD_NAME"].str.split().str[0]
```

```
transactionData['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
```

```
transactionData['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)
```

If we look at it more closely, we see that some brand names are same and are written differently. If we don't change it products of same company will be divided into parts and analysis regarding the brands will be wrong. So we will change the names of them.

```
brand_new = transactionData['brand_name'].replace({"Dorito":'Doritos', 'Red':'Red Rock Deli', 'RRD': 'Red Rock Deli','
```

```
brand_new.unique()
```

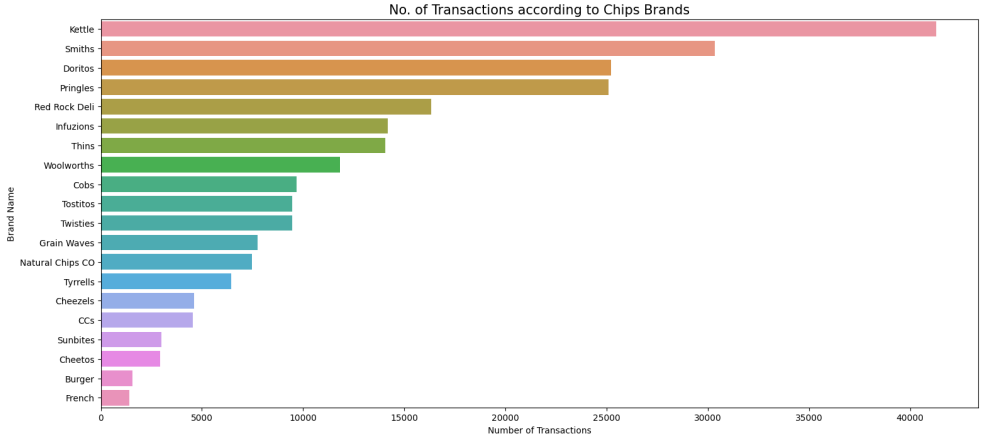
```
array(['Natural Chips CO', 'CCs', 'Smiths', 'Kettle', 'Grain Waves',
       'Doritos', 'Twisties', 'Woolworths', 'Thins', 'Burger', 'Cheezels',
       'Infuzions', 'Red Rock Deli', 'Pringles', 'Tyrrells', 'Cobs',
       'French', 'Tostitos', 'Cheetos', 'Sunbites'], dtype=object)
```

```
transactionData['brand_new'] = brand_new
transactionData.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	

```
fig3 = plt.figure(figsize=(18,8))
sns.barplot(x = transactionData['brand_new'].value_counts(), y=transactionData["brand_new"].value_counts().index,)
plt.title('No. of Transactions according to Chips Brands', fontsize =15)
plt.xlabel("Number of Transactions")
plt.ylabel("Brand Name")
```

```
Text(0, 0.5, 'Brand Name')
```



Now let's explore the customer data

```
customerData.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

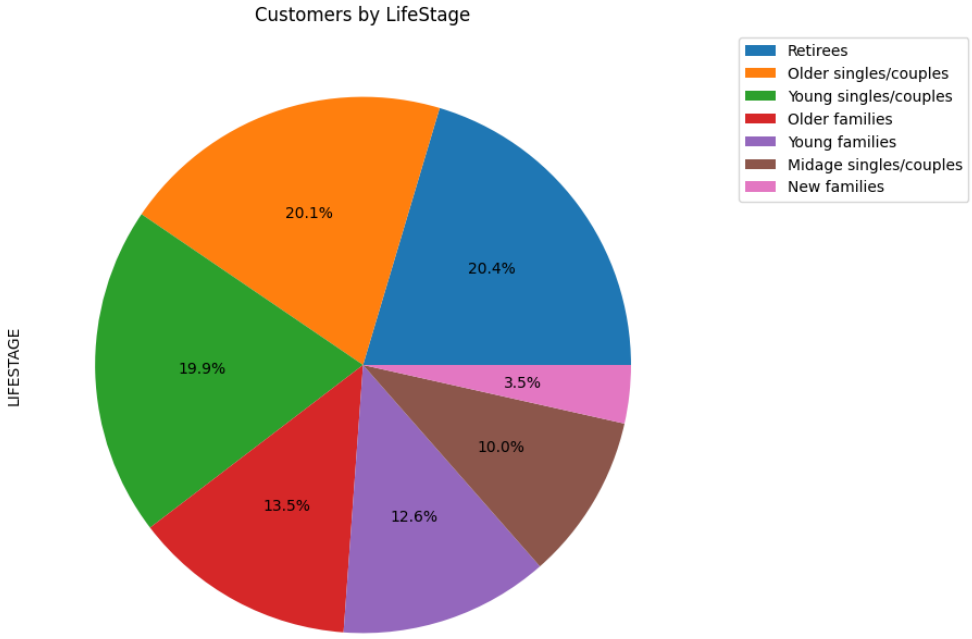
```
customerData.info()
```

```
<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
```

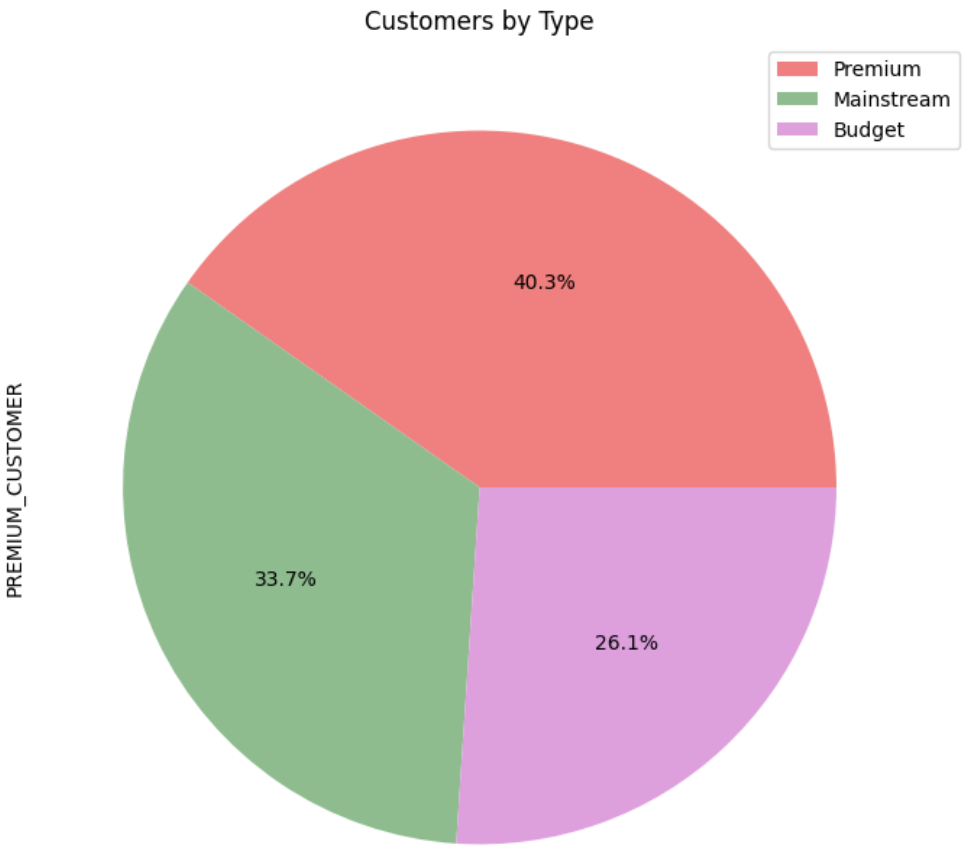
```
customerData.describe()
```

	LYLTY_CARD_NBR
count	7.263700e+04
mean	1.361859e+05
std	8.989293e+04
min	1.000000e+03
25%	6.620200e+04
50%	1.340400e+05
75%	2.033750e+05
max	2.373711e+06

```
plt.figure(figsize=(15,8))
customerData['LIFESTAGE'].value_counts().plot(kind='pie',labels = None, autopct='%.1f%%')
plt.legend(customerData['LIFESTAGE'].value_counts().index.str.capitalize(), loc = 'best', bbox_to_anchor = (1.05,1))
plt.title('Customers by LifeStage')
plt.show()
```




```
plt.figure(figsize=(15,8))
customerData['PREMIUM_CUSTOMER'].value_counts().plot(kind='pie',colors = ['lightcoral', 'darkseagreen', 'plum'], labels
plt.legend(customerData['PREMIUM_CUSTOMER'], loc = 'best', bbox_to_anchor = (1.05,1))
plt.title('Customers by Type')
plt.show()
```



Merging the two datasets for further analysis

```
data_merged = pd.merge(customerData, transactionData, how = 'outer', on = 'LYLTY_CARD_NBR')
data_merged.head()
```

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID	PRC
0	1000	YOUNG SINGLES/COUPLES	Premium	2018-10-17	1.0	1.0	
1	1002	YOUNG SINGLES/COUPLES	Mainstream	2018-09-16	1.0	2.0	
2	1003	YOUNG FAMILIES	Budget	2019-03-07	1.0	3.0	
3	1003	YOUNG FAMILIES	Budget	2019-03-08	1.0	4.0	
4	1004	OLDER SINGLES/COUPLES	Mainstream	2018-11-02	1.0	5.0	

Saving our data as csv file

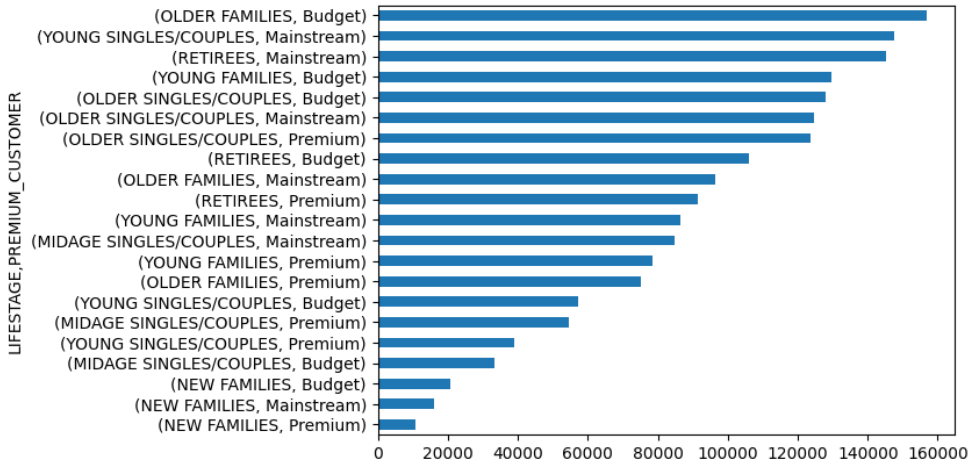
```
data_merged = data_merged.to_csv('data_merged.csv')
```

```
data_merged = pd.read_csv('/content/data_merged.csv')
```

```
total_sales = data_merged.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES'].sum()
total_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
Name: TOT_SALES, dtype: float64
```

```
total_sales.sort_values().plot(kind = 'barh')
plt.show()
```

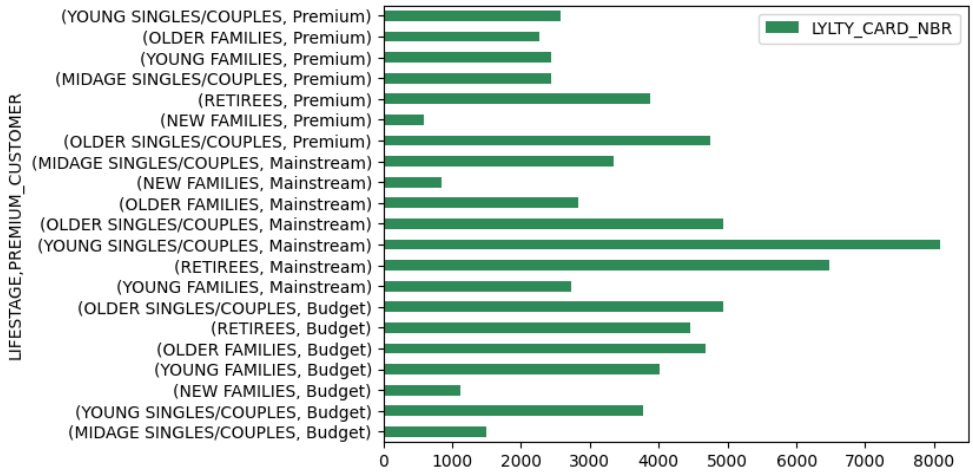



```
tot_customer = customerData.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).count()
tot_customer
```

		LYLTY_CARD_NBR
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

```
tot_customer.sort_values(by = 'PREMIUM_CUSTOMER').plot(kind = 'barh', color = ['seagreen'])
```

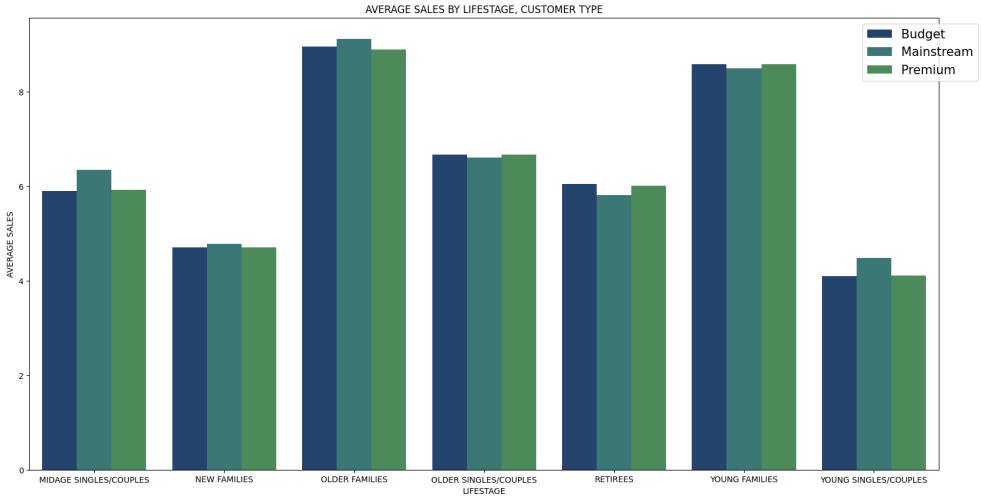
```
<Axes: ylabel='LIFESTAGE,PREMIUM_CUSTOMER'>
```



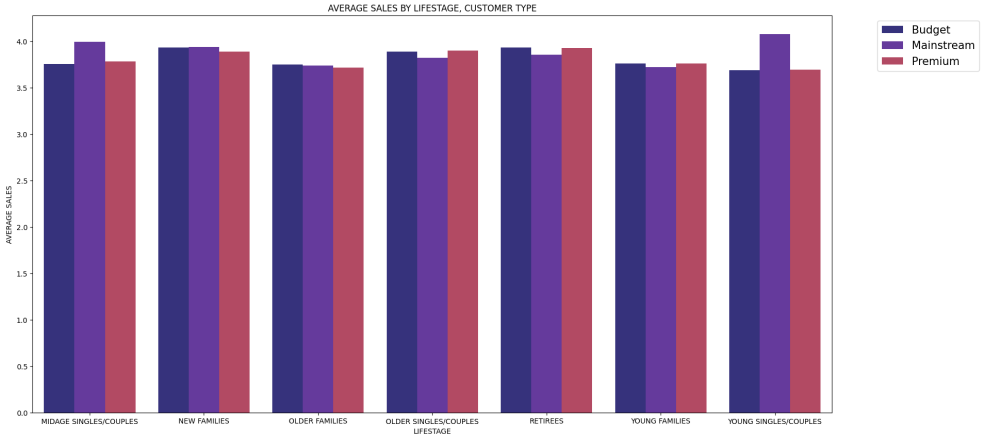
```
merge_lp = pd.concat([total_sales, tot_customer],axis=1)
merge_lp = pd.DataFrame(merge_lp)
merge_lp.rename(columns = {'LYLTY_CARD_NBR': 'CUSTOMER_CNT'}, inplace = True)
merge_lp['TOT_QTY'] = data_merged.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['PROD_QTY'].sum()
merge_lp['AVG_QTY'] = merge_lp['TOT_QTY'] / merge_lp['CUSTOMER_CNT']
merge_lp['AVG_PR'] = merge_lp['TOT_SALES'] / merge_lp['TOT_QTY']
merge_lp
```

		TOT_SALES	CUSTOMER_CNT	TOT_QTY	AVG_QTY	AV
LIFESTAGE	PREMIUM_CUSTOMER					
MIDAGE SINGLES/COUPLES	Budget	33345.70	1504	8883.0	5.906250	3.75
	Mainstream	84734.25	3340	21213.0	6.351198	3.99
	Premium	54443.85	2431	14400.0	5.923488	3.78
NEW FAMILIES	Budget	20607.45	1112	5241.0	4.713129	3.93
	Mainstream	15979.70	849	4060.0	4.782097	3.93
	Premium	10760.80	588	2769.0	4.709184	3.88
OLDER FAMILIES	Budget	156863.75	4675	41853.0	8.952513	3.74
	Mainstream	96413.55	2831	25804.0	9.114800	3.73
	Premium	75242.60	2274	20239.0	8.900176	3.71
OLDER SINGLES/COUPLES	Budget	127833.60	4929	32883.0	6.671333	3.88
	Mainstream	124648.50	4930	32607.0	6.613996	3.82
	Premium	123537.55	4750	31695.0	6.672632	3.89
RETIREES	Budget	105916.30	4454	26932.0	6.046700	3.93
	Mainstream	145168.95	6479	37677.0	5.815249	3.85
	Premium	91296.65	3872	23266.0	6.008781	3.92
YOUNG FAMILIES	Budget	129717.95	4017	34482.0	8.584018	3.76
	Mainstream	86338.25	2728	23194.0	8.502199	3.72
	Premium	78571.70	2433	20901.0	8.590629	3.75
YOUNG SINGLES/COUPLES	Budget	57122.10	3779	15500.0	4.101614	3.68
	Mainstream	147582.20	8088	36225.0	4.478858	4.07
	Premium	39052.30	2574	10575.0	4.108392	3.69

```
plt.figure(figsize=(20,10))
sns.barplot(x=merge_lp.reset_index()['LIFESTAGE'], y=merge_lp.reset_index()['AVG_QTY'], hue=merge_lp.reset_index()['PR
plt.xlabel('LIFESTAGE')
plt.ylabel('AVERAGE SALES')
plt.title('AVERAGE SALES BY LIFESTAGE, CUSTOMER TYPE')
plt.legend( loc='best', bbox_to_anchor=(1.05, 1), fontsize =15)
plt.show()
```




```
plt.figure(figsize=(20,10))
sns.barplot(x=merge_lp.reset_index()['LIFESTAGE'], y=merge_lp.reset_index()['AVG_PR'], hue=merge_lp.reset_index()['PRE
plt.xlabel('LIFESTAGE')
plt.ylabel('AVERAGE SALES')
plt.title('AVERAGE SALES BY LIFESTAGE, CUSTOMER TYPE')
plt.legend( loc='best', bbox_to_anchor=(1.05, 1), fontsize =15)
plt.show()
```




```
from scipy import stats
```

```
merge_lp.shape
```

```
(21, 5)
```

```
merge_lp = merge_lp.reset_index()
```

```
MS = merge_lp.loc[merge_lp['PREMIUM_CUSTOMER'] =='Mainstream']
Pr_Bu = merge_lp.loc[(merge_lp['PREMIUM_CUSTOMER'] == 'Premium') | (merge_lp['PREMIUM_CUSTOMER'] =='Budget')]
MS
```

```
Pr_Bu
```

	LIFESTAGE	PREMIUM_CUSTOMER	TOT_SALES	CUSTOMER_CNT	TOT_QTY	AVG_QTY
0	MIDAGE SINGLES/COUPLES	Budget	33345.70	1504	8883.0	5.906250
2	MIDAGE SINGLES/COUPLES	Premium	54443.85	2431	14400.0	5.923488
3	NEW FAMILIES	Budget	20607.45	1112	5241.0	4.713129
5	NEW FAMILIES	Premium	10760.80	588	2769.0	4.709184
6	OLDER FAMILIES	Budget	156863.75	4675	41853.0	8.952513
8	OLDER FAMILIES	Premium	75242.60	2274	20239.0	8.900176
9	OLDER SINGLES/COUPLES	Budget	127833.60	4929	32883.0	6.671333
11	OLDER SINGLES/COUPLES	Premium	123537.55	4750	31695.0	6.672632
12	RETIREES	Budget	105916.30	4454	26932.0	6.046700
14	RETIREES	Premium	91296.65	3872	23266.0	6.008781
15	YOUNG FAMILIES	Budget	129717.95	4017	34482.0	8.584018
17	YOUNG FAMILIES	Premium	78571.70	2433	20901.0	8.590629

```
MS_mid_y = MS["AVG_PR"][(merge_lp["LIFESTAGE"]=="MIDAGE SINGLES/COUPLES")|(merge_lp["LIFESTAGE"]=="YOUNG SINGLES/COUPL
PB_mid_y = Pr_Bu["AVG_PR"][(merge_lp["LIFESTAGE"]=="MIDAGE SINGLES/COUPLES")|(merge_lp["LIFESTAGE"]=="YOUNG SINGLES/CO
```

```
MS_mid_y
```

```
1     3.994449
19    4.074043
Name: AVG_PR, dtype: float64
```

```
stats.ttest_ind(MS_mid_y, PB_mid_y)
```

```
TtestResult(statistic=7.181830997406295, pvalue=0.0019909612272297904, df=4.0)
```

```
MS_mysc = data_merged[(data_merged['PREMIUM_CUSTOMER'] == 'Mainstream') & (data_merged['LIFESTAGE'] == 'YOUNG SINGLES/
MS_mysc
```

MIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
Mainstream	2018-09-16	1.0	2.0	58.0	Red Rock Deli Chikn&Garlic Aioli 150g	1.0	2.7
Mainstream	2018-09-09	1.0	10.0	51.0	Doritos Mexicana 170g	2.0	8.8
Mainstream	2018-09-03	1.0	22.0	3.0	Kettle Sensations Camembert & Fig 150g	1.0	4.6
Mainstream	2018-11-28	1.0	23.0	97.0	RRD Salt & Vinegar 165g	1.0	3.0
Mainstream	2019-06-20	1.0	24.0	38.0	Infuzions Mango Chutny Papadums 70g	1.0	2.4
...	...	...	...	...	...	...	...
Mainstream	2018-12-07	272.0	270205.0	63.0	Kettle 135g Swt Pot Sea Salt	2.0	8.4
Mainstream	2018-09-23	77.0	236718.0	24.0	Grain Waves Sweet Chilli 210g	2.0	7.2
Mainstream	2018-07-30	77.0	236756.0	71.0	Twisties Cheese Burger 250g	2.0	8.6
Mainstream	2018-08-02	88.0	240146.0	36.0	Kettle Chilli 175g	2.0	10.8
Mainstream	2018-12-14	88.0	241815.0	16.0	Smiths Crinkle Chips Salt & Vinegar 330g	2.0	11.4

```
MS_mysc['brand_new'].value_counts().head()
```

```
Kettle       3844
Doritos      2379
Pringles     2315
Smiths       1921
Infuzions    1250
Name: brand_new, dtype: int64
```

```
MS_mysc['pack_size'].value_counts().head()
```

```
175.0    4997
150.0    3080
134.0    2315
110.0    2051
170.0    1575
Name: pack_size, dtype: int64
```

```
brand_count = MS_mysc.groupby(['LYLTY_CARD_NBR', 'brand_new'])['PROD_QTY'].sum().unstack()
basket_brand = brand_count.applymap(lambda x:1 if x>0 else 0)
basket_brand
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarnin
  and should_run_async(code)
```

brand_new	Burger	CCs	Cheetos	Cheezels	Cobs	Doritos	French	Grain Waves	Infuzio
LYLTY_CARD_NBR									
1000	0	0	0	0	0	0	0	0	
1002	0	0	0	0	0	0	0	0	
1003	0	0	0	0	0	0	0	1	
1004	0	0	0	0	0	0	0	0	
1005	0	0	1	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
2370651	0	0	0	0	0	1	0	0	
2370701	0	0	0	0	0	0	0	1	
2370751	0	0	0	0	0	0	0	0	
2370961	0	0	0	0	0	0	0	0	
2373711	0	0	0	0	0	0	0	0	

71287 rows × 20 columns

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

freq_itemset1 = apriori(basket_brand, min_support = 0.04, use_colnames = True).sort_values(by='support', ascending = Fa

freq_itemset1
```

51	0.060712	(Smiths, Thins)
33	0.060474	(Infuzions, Smiths)
0	0.059562	(CCs)
21	0.059450	(Doritos, Infuzions)
39	0.058748	(Kettle, Twisties)
26	0.058482	(Doritos, Thins)
32	0.058440	(Pringles, Infuzions)
38	0.058328	(Kettle, Tostitos)
41	0.057865	(Woolworths, Kettle)
45	0.057430	(Pringles, Thins)
24	0.052450	(Doritos, Red Rock Deli)
50	0.051580	(Woolworths, Red Rock Deli)
43	0.051566	(Pringles, Red Rock Deli)
54	0.050177	(Doritos, Pringles, Kettle)
56	0.047723	(Pringles, Kettle, Smiths)
55	0.047624	(Doritos, Smiths, Kettle)
42	0.047526	(Smiths, Natural Chips CO)
30	0.047274	(Grain Waves, Kettle)
57	0.042378	(Smiths, Kettle, Red Rock Deli)
17	0.042154	(Doritos, Cobs)
19	0.041943	(Pringles, Cobs)
28	0.041073	(Doritos, Twisties)
40	0.040975	(Kettle, Tyrrells)
52	0.040821	(Smiths, Tostitos)
48	0.040737	(Woolworths, Pringles)
29	0.040568	(Doritos, Woolworths)
46	0.040540	(Pringles, Tostitos)
47	0.040484	(Pringles, Twisties)
20	0.040400	(Smiths, Cobs)
27	0.040386	(Doritos, Tostitos)
11	0.040330	(Sunbites)

```
rules = association_rules(freq_itemset1, metric='lift').sort_values(['support', 'lift','confidence'], ascending= False
rules.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarnin
  and should_run_async(code)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	I
0	(Doritos)	(Kettle)	0.290446	0.423303	0.136420	0.469693	1.109591	
1	(Kettle)	(Doritos)	0.423303	0.290446	0.136420	0.322276	1.109591	
2	(Pringles)	(Kettle)	0.289772	0.423303	0.135452	0.467444	1.104279	
3	(Kettle)	(Pringles)	0.423303	0.289772	0.135452	0.319989	1.104279	
5	(Kettle)	(Smiths)	0.423303	0.314896	0.135130	0.319227	1.013754	

```
size_count = MS_mysc.groupby(['LYLTY_CARD_NBR', 'pack_size'])['PROD_QTY'].sum().unstack()

basket_size = size_count.applymap(lambda x:1 if x > 0 else 0)
basket_size
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarnin
  and should_run_async(code)
```

pack_size	70.0	90.0	110.0	125.0	134.0	135.0	150.0	160.0	165.0	170.0	1
LYLTY_CARD_NBR											
1000	0	0	0	0	0	0	0	0	0	0	
1002	0	0	0	0	0	0	1	0	0	0	
1003	0	0	0	0	0	0	0	0	0	0	
1004	0	0	0	0	0	0	0	1	0	0	