Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

 # Solution for Task 1
This file is a solution for the Task 1 of the Quantium Virtual Internship.

BY-
Divyanshu Dev Awasthi

div.awasthi01@gmail.com

https://www.linkedin.com/in/divdev7/ GitHub https://github.com/divyanshuhub

Loading required libraries and datasets

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from plotly.offline import init_notebook_mode, iplot

init_notebook_mode(connected=True)

import plotly.offline as offline

offline.init_notebook_mode()

import cufflinks as cf

cf.go_offline()
```

Pointing the filePath to where I have downloaded the datasets to and
assigning the data files to data.tables

```
purchase=pd.read_csv("C:\ALL DATA\ML PTOJECT\Quantium\QVI_purchase_behaviour.csv")

purchase

transaction=pd.read_excel("C:\ALL DATA\ML PTOJECT\Quantium\QVI_transaction_data.xlsx")
```

```
transaction
```

```

## Exploratory data analysis
The first step in any analysis is to first understand the data. Let's take a look at each of the datasets provided.

# #TRANSACTION

### #transforming date column
#### A quick search online tells us that CSV and Excel integer dates begin on 30 Dec. 1899

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

transaction

transaction["PROD_NAME"].describe()
```

### #finding the most frequest words

```
import collections

freq=collections.Counter([j  for s in transaction["PROD_NAME"] for j in s.split()])

freq
```

#sorting in decreasing order of the frequency of words

```
fre=pd.DataFrame([freq.keys(),freq.values()],index=['Word','Frequency']).transpose().sort_values(by='Frequency',ascending=False)

fre
```

removing useless words like '170g' & # most frequent words

```
fre=fre[[ s[0] not in  ['0','1','2','3','4','5','6','7','8','9','&'] for s in fre['Word'] ]]

fre
```

#dropping salsa items

There are salsa products in the dataset but we are only interested in the chips category, so let's remove these.
Remove salsa products
```

```
transaction.drop(transaction[[("Salsa" in s)  for s in transaction['PROD_NAME']]].index,inplace=True)

transaction[[("Salsa" in s)  for s in transaction['PROD_NAME']]]
```

#details about transaction dataset

```
transaction.describe()

transaction.info()
```

#number of nulls in each column

```
transaction.isna().sum()
```

There are no nulls in the columns but product quantity appears to have an outlier which we should investigate further. Let's investigate further the case where 200 packets of chips are bought in one transaction.

Removing Anomalies/Outliers

```
transaction[transaction['TOT_SALES']>30]

transaction[transaction['PROD_QTY']>5]
```

```
transaction.drop(labels=transaction[transaction['PROD_QTY']==200].index,inplace=True)

#transaction.drop(labels=transaction[transaction['TOT_SALES']>600].index,inplace=True)

#transaction.drop(labels=transaction[transaction['TXN_ID']>1500000].index,inplace=True)
```

There are two transactions where 200 packets of chips are bought in one transaction and both of these transactions were by the same customer.
Let's see if the customer has had other transactions

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 from further analysis.
```
That's better. Now, let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

**#missing dates**

```
ts=transaction.groupby('DATE').count()

ts
```

Count the number of transactions by date
```

```
ts=transaction.groupby('DATE').count()

ts
```

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.

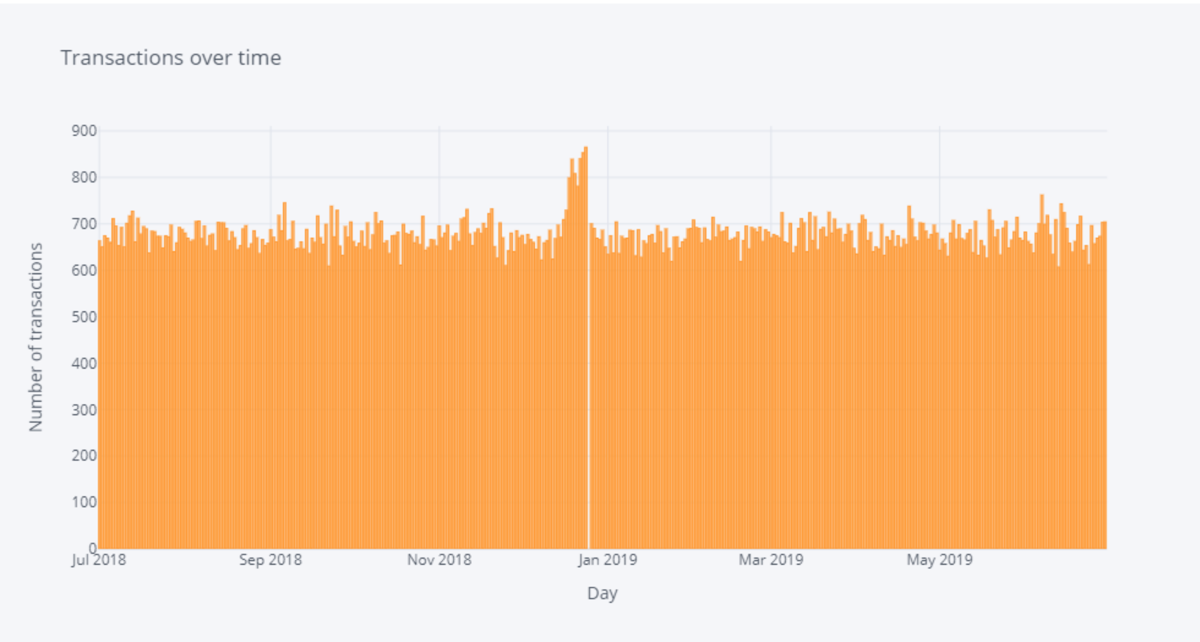
#missing date

```
set(pd.date_range('2018-07-01', end='2019-06-30',freq='D'))-set((ts.index))

ts.loc['2018-12-25']=np.nan#=ts.mean().apply(int)

ts[ts.index=='2018-12-25']
```

Creating a column of dates that includes every day from 1 Jul 2018 to 30 Jun 2019, and joining it onto the data to fill in the missing day.

 # plot showing missing date

```
ts['TXN_ID'].iplot(kind='bar',xTitle='Day',yTitle= "Number of transactions", title = "Transactions over time")
```



We can see that there is an increase in purchases in December and a break in late December
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.

Now we can move on to creating other features such as brand of chips or pack size from PROD_NAME. We will start with pack size.

#Adding features

```
def fun(s):
    a=[]
    for i in s:
        if i in ['0','1','2','3','4','5','6','7','8','9']:
            a.append(i)
    return int("".join(a))


transaction['PACKAGE_SIZE']=transaction['PROD_NAME'].apply(fun)
transaction
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Let's combine these together.

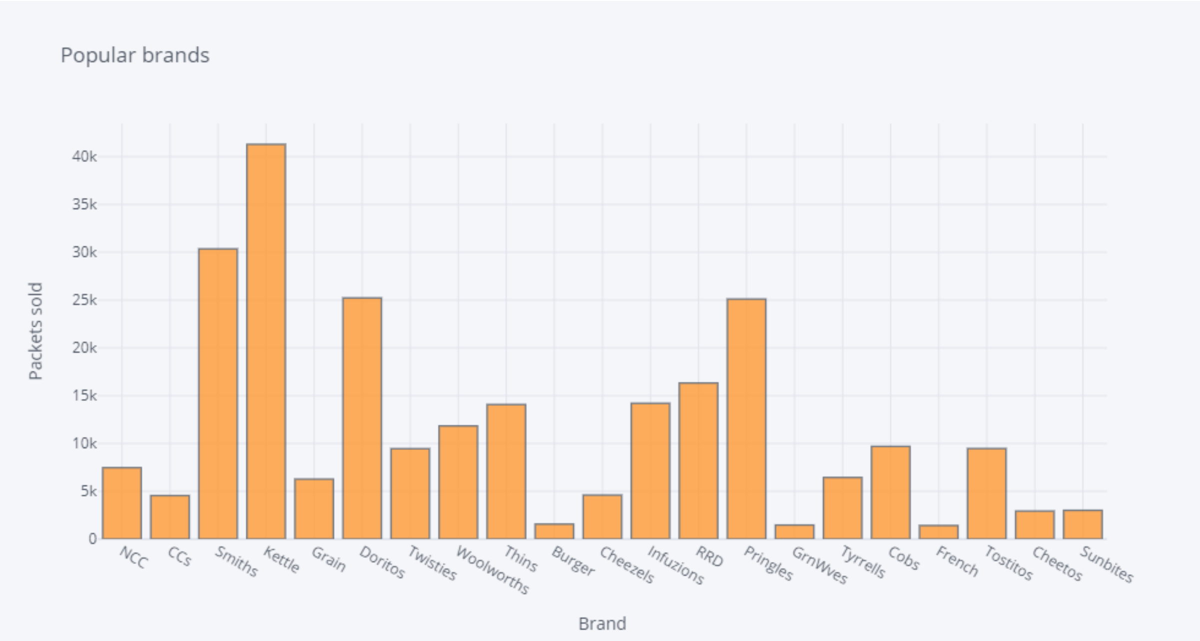
```
transaction['BRAND']=[s.split()[0] for s in transaction['PROD_NAME']]
transaction['BRAND'].replace('Dorito','Doritos',inplace=True)
transaction['BRAND'].replace('Infzns','Infuzions',inplace=True)
transaction['BRAND'].replace('Smith','Smiths',inplace=True)
transaction['BRAND'].replace('Snbts','Sunbites',inplace=True)
transaction['BRAND'].replace('Red','RRD',inplace=True)
transaction['BRAND'].replace('Old','Old El Paso',inplace=True)
transaction['BRAND'].replace('WW','Woolworths',inplace=True)
transaction['BRAND'].replace('Natural','NCC',inplace=True)
```

The largest size is 380g and the smallest size is 70g - seems sensible!

Let's plot a histogram of PACK_SIZE since we know that it is a categorical variable and not a continuous variable even though it is numeric.

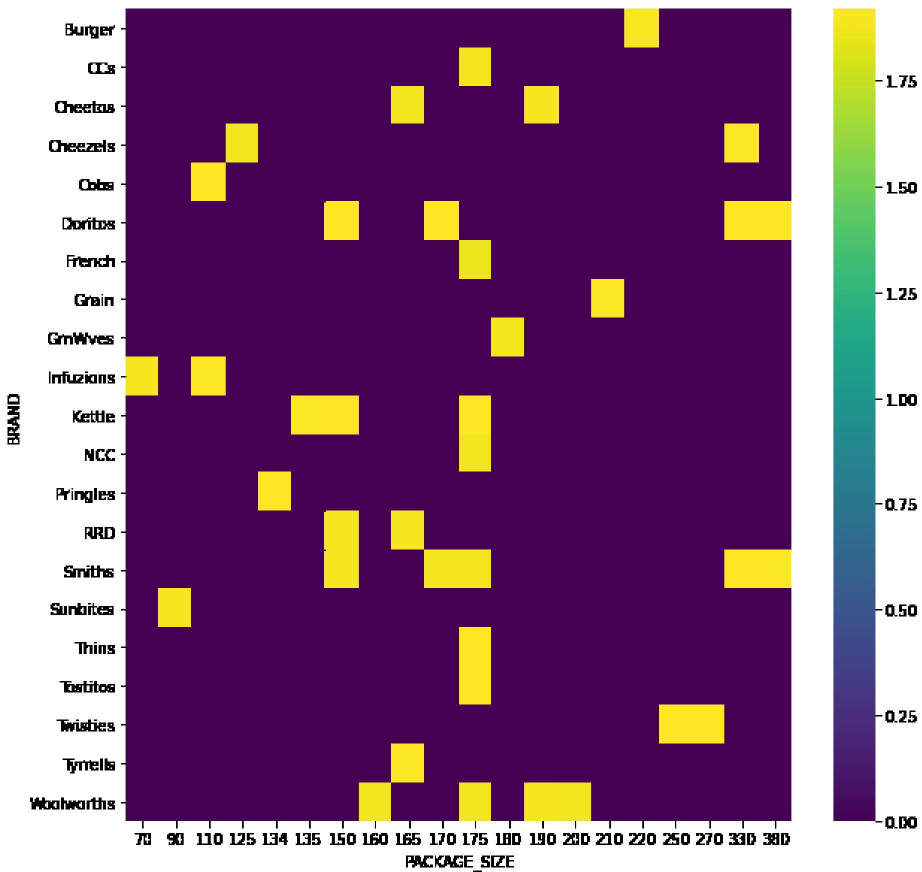
#Histogram for brands

```
transaction['BRAND'].iplot(kind='hist',xTitle='Brand',yTitle='Packets sold',title='Popular brands')
```



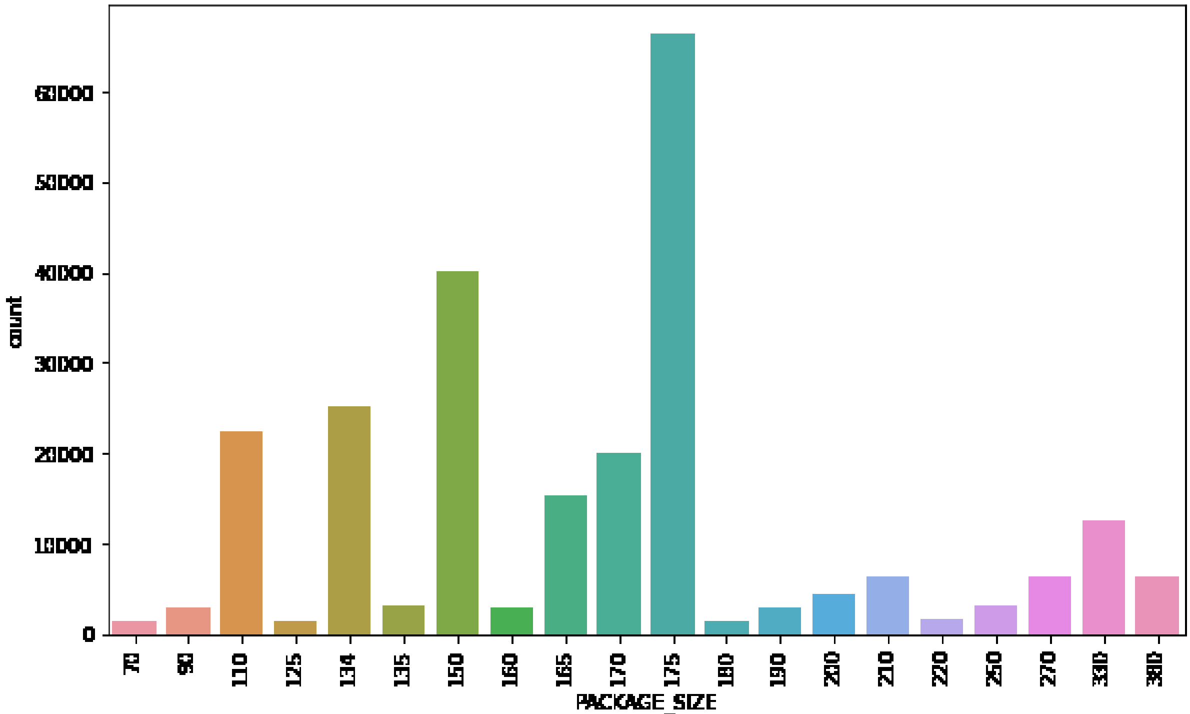
#heatmap showing packet quantity mostly bought according to brand and packet size

```
plt.figure(figsize=(10,10))

sns.heatmap(pd.pivot_table(data=transaction,index='BRAND',columns='PACKAGE_SIZE',values='PROD_QTY').fillna
(0),cmap='viridis')
```



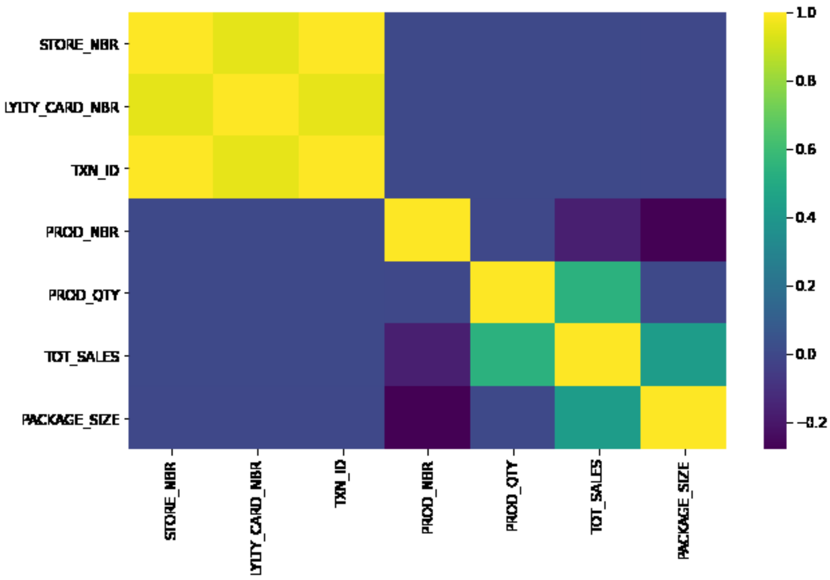
#histogram of packet size

```
plt.figure(figsize=(10,6))

plt.xticks(rotation=90)

sns.countplot(transaction['PACKAGE_SIZE'])
```



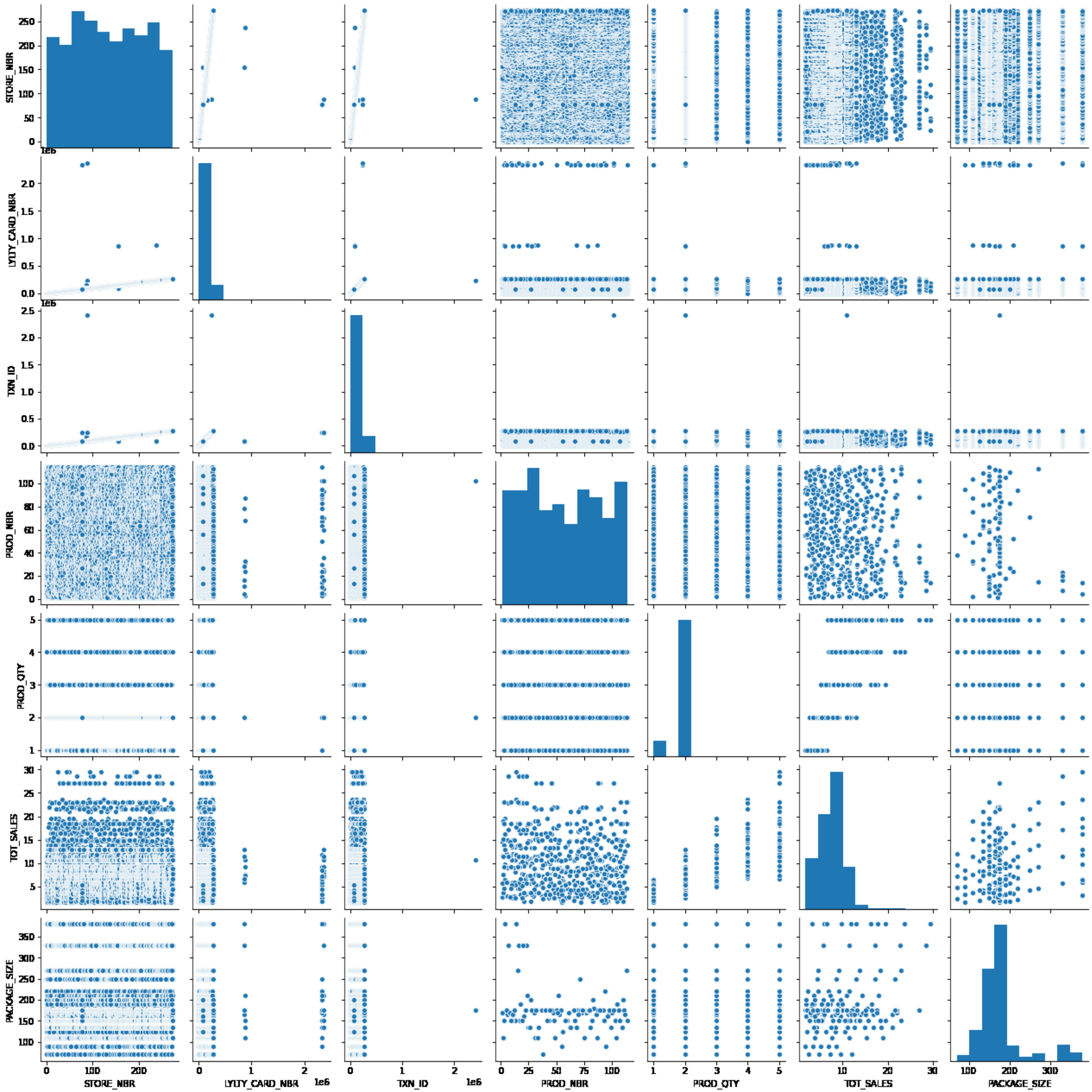
#correlation heatmap

```
plt.figure(figsize=(10,6))

sns.heatmap(transaction.corr(),cmap='viridis')
```



#pairplot

```
sns.pairplot(data=transaction[transaction.columns.drop('PROD_NAME')])
```



CUSTOMER PURCHASE BEHAVIOUR DATA

Now that we are happy with the transaction dataset, let's have a look at the customer dataset.

{Exploratory data analysis}

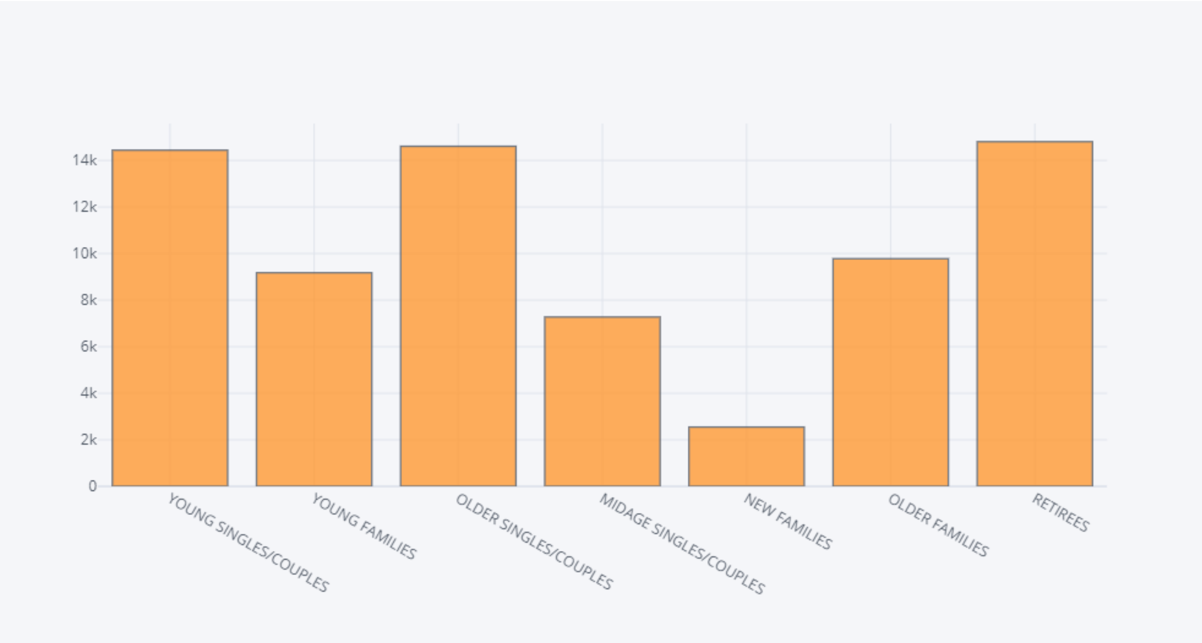
```
purchase['LYLTY_CARD_NBR'].nunique()
```

```
purchase.info()
```

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

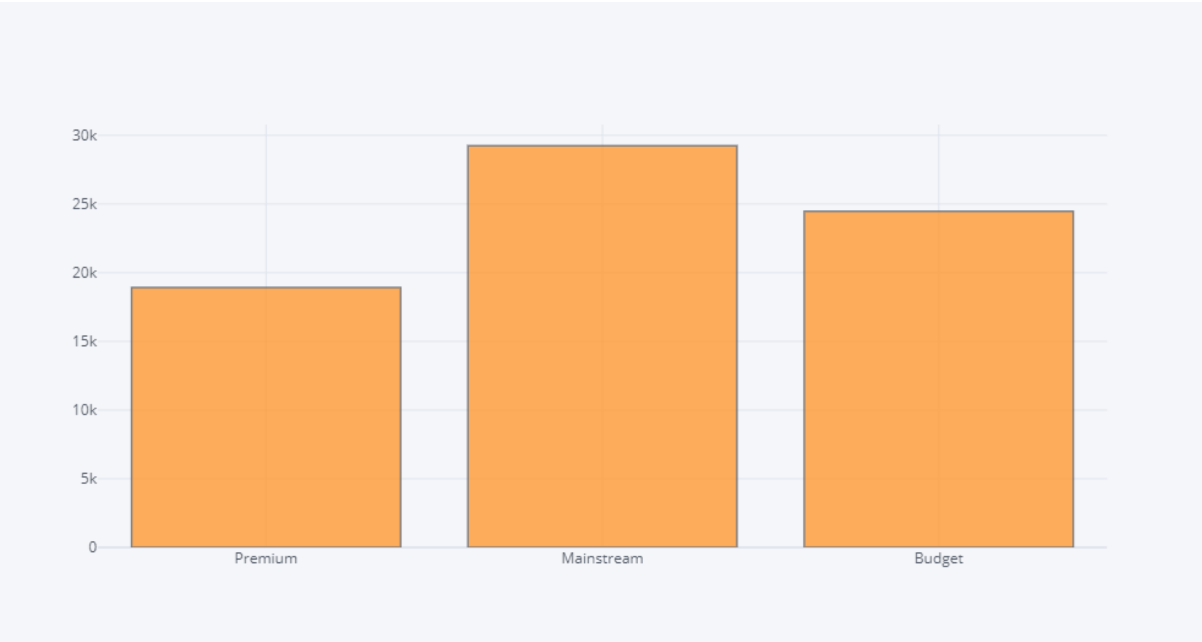
#lifestage distribution among customers

```
purchase['LIFESTAGE'].iplot(kind='hist')
```



#Premium customer distribution among customers

```
purchase['PREMIUM_CUSTOMER'].iplot(kind='hist')

purchase.isna().sum()
```



JOINING DATASETS

```
finaldf=pd.merge(transaction,purchase,on='LYLTY_CARD_NBR')

finaldf.head(2)

finaldf.info()
```

Let's also check if some customers were not matched on by checking for nulls.

```
finaldf.isna().sum()

finaldf.to_csv('Final.csv')
```

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

Total Sales to Different kind of customers:

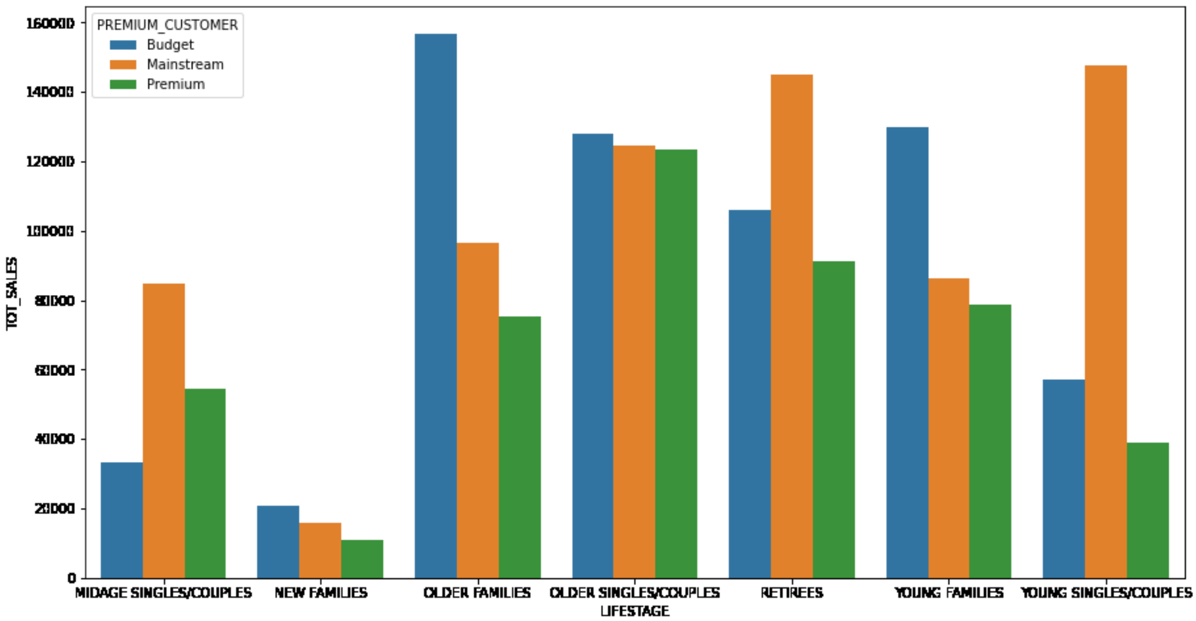
```
finaldf[['TOT_SALES','PREMIUM_CUSTOMER']].groupby('PREMIUM_CUSTOMER').sum().sort_values('TOT_SALES',ascending=False)
```

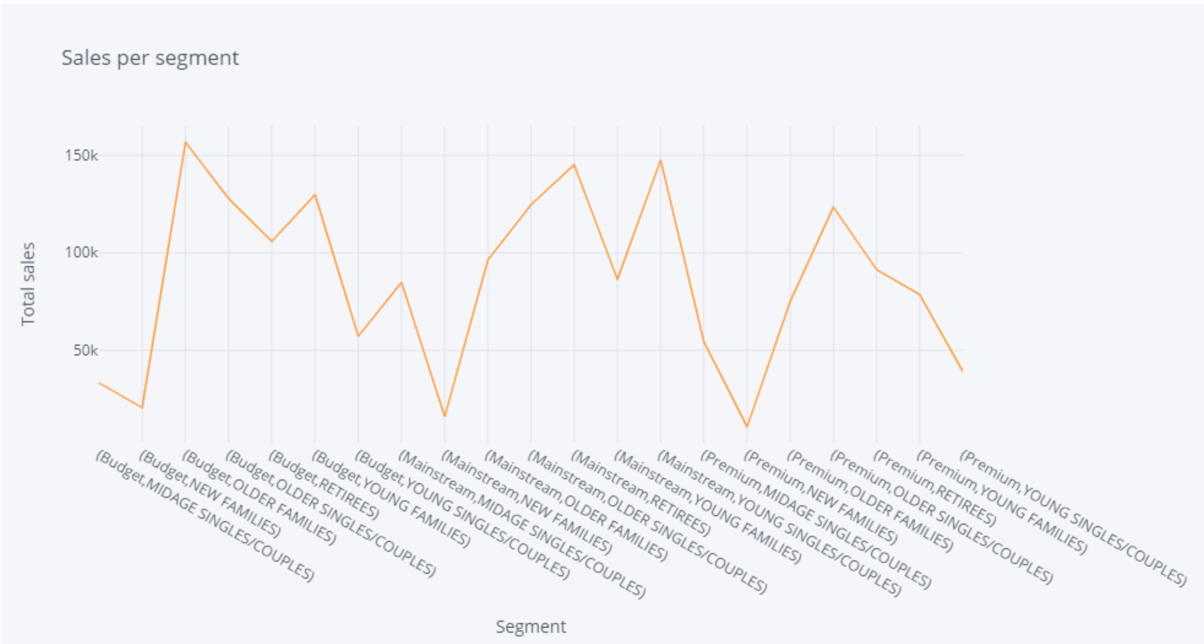
PREMIUM_CUSTOMER	TOT_SALES
Mainstream	700865.40
Budget	631406.85
Premium	472905.45

- Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is

```
a=finaldf[['LIFESTAGE','PREMIUM_CUSTOMER','TOT_SALES']].groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).sum()

a.sort_values('TOT_SALES',ascending=False)
```

```
plt.figure(figsize=(15,8))

sns.barplot(y=a.reset_index()['TOT_SALES'],x=a.reset_index()['LIFESTAGE'],hue=a.reset_index()['PREMIUM_CUSTOMER'])

a.iplot(title="Sales per segment",yTitle='Total sales',xTitle='Segment')
```

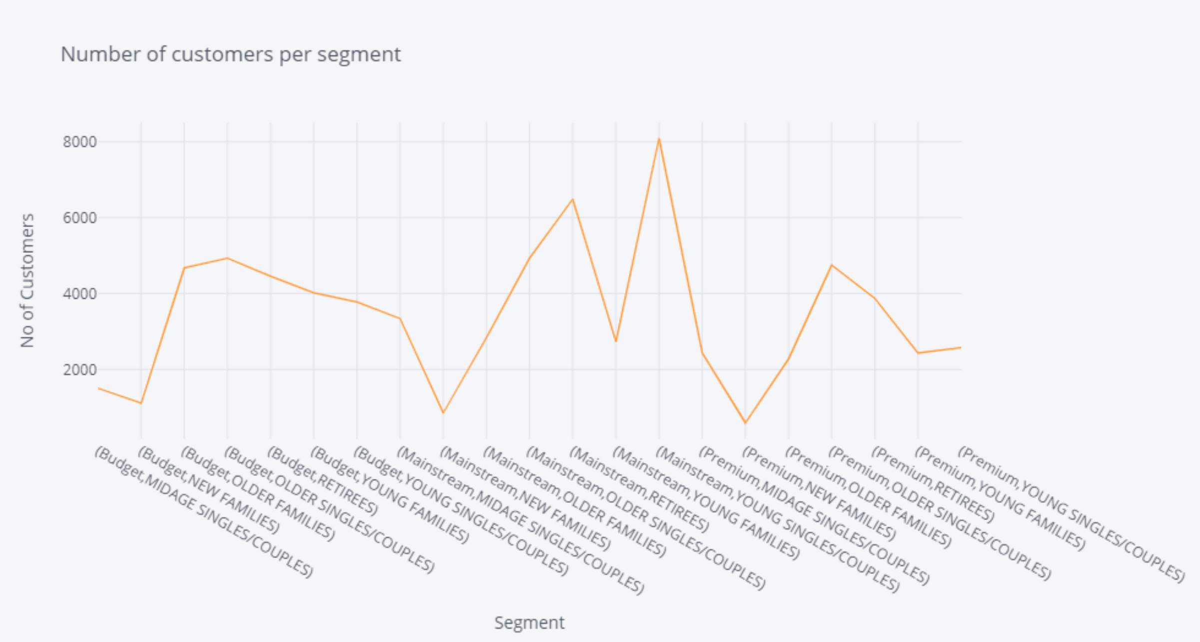




- How many customers are in each segment

```
b=purchase.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).count()
b.columns=['CUSTOMER_COUNT']
b.sort_values('CUSTOMER_COUNT',ascending=False)
```

```
b.iplot(title="Number of customers per segment",yTitle='No of Customers',xTitle='Segment')
```

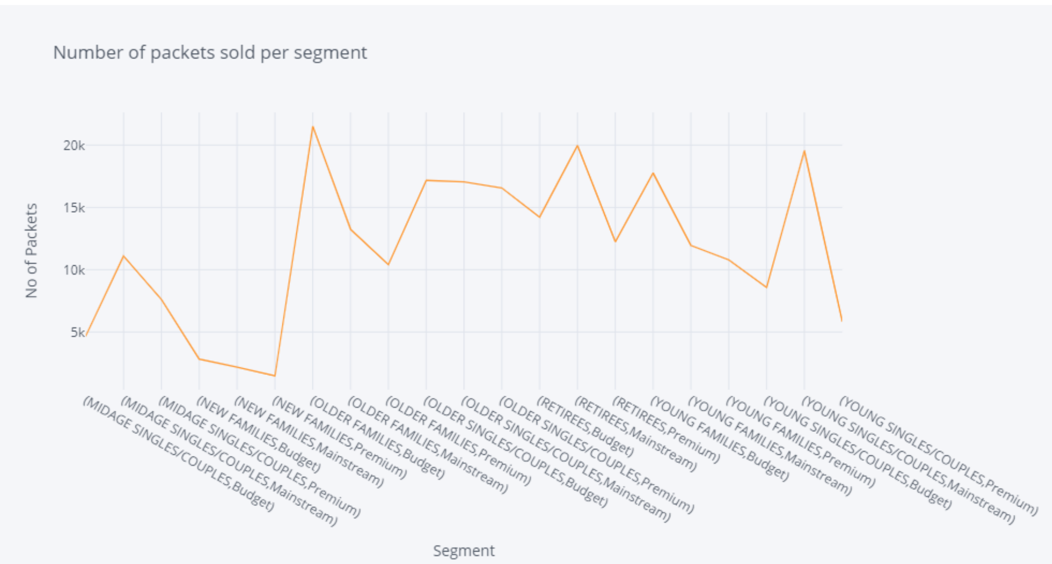


This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment. Higher sales may also be driven by more units of chips being bought per customer.

- How many chips are bought per customer by segment

```
c=finaldf[['LIFESTAGE','PREMIUM_CUSTOMER','TOT_SALES']].groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).count()

c.sort_values('TOT_SALES',ascending=False).head(5)
```

```
c.iplot(title="Number of packets sold per segment",yTitle='No of Packets',xTitle='Segment')
```



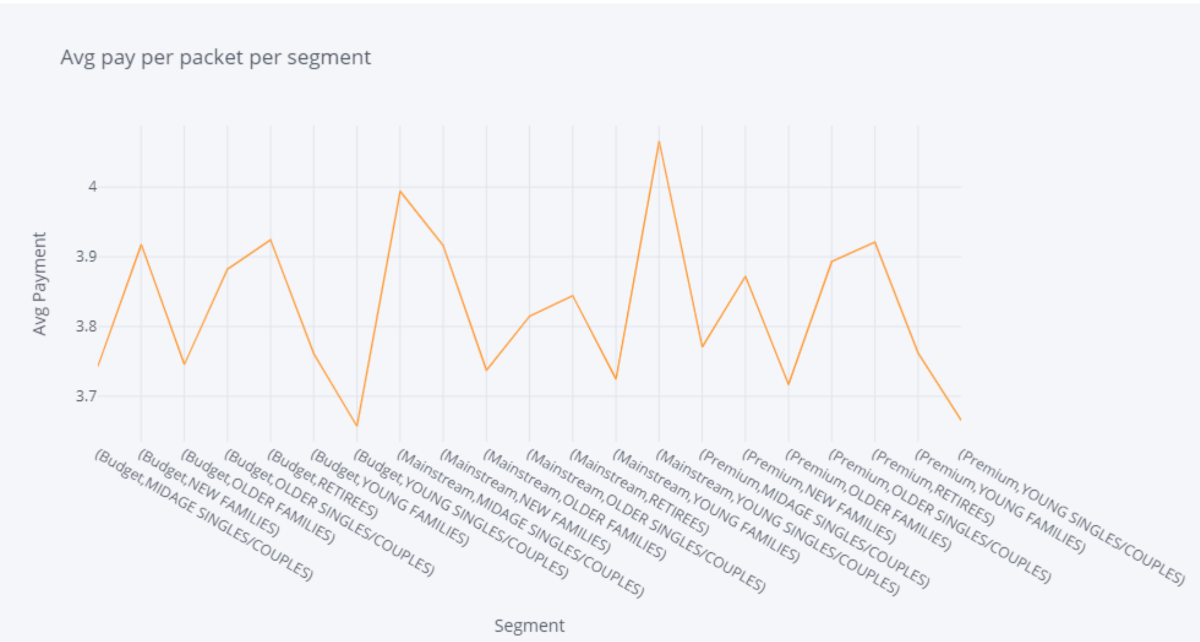
- The customer's total spend over the period to understand what proportion of their grocery spend is on chips

```
transaction1=pd.read_excel("C:\ALL DATA\ML PTOJECT\Quantium\QVI_transaction_data.xlsx")

totsalespercust=transaction1[['LYLTY_CARD_NBR','TOT_SALES']].groupby(['LYLTY_CARD_NBR']).sum().reset_index()

ratio=finaldf[['LYLTY_CARD_NBR','TOT_SALES']].merge(totsalespercust,on='LYLTY_CARD_NBR').rename(columns={'TOT_SALES_x':'TRAN_SALE','TOT_SALES_y':'CUST_TOT_SALE'})

ratio['RATIO']=ratio['TRAN_SALE']/ratio['CUST_TOT_SALE']

ratio.sort_values('RATIO')
```

- What's the average chip price by customer segment

```
finaldf['CHIP_PRICE']=finaldf['TOT_SALES']/finaldf['PROD_QTY']

d=finaldf[['LIFESTAGE','PREMIUM_CUSTOMER','CHIP_PRICE']].groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).mean()

d.sort_values("CHIP_PRICE",ascending=False)
```

```
d.iplot(title="Avg pay per packet per segment",yTitle='Avg Payment',xTitle='Segment')
```



- Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

```
e=finaldf[['LIFESTAGE','PREMIUM_CUSTOMER','TOT_SALES']].groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).count()

e["TOT_SALES"]/(e['TOT_SALES'].sum())
```

```
PREMIUM_CUSTOMER  LIFESTAGE
Budget            MIDAGE SINGLES/COUPLES    0.019012
                  NEW FAMILIES              0.011445
                  OLDER FAMILIES            0.087193
                  OLDER SINGLES/COUPLES     0.069596
                  RETIREES                  0.057652
                  YOUNG FAMILIES            0.071991
                  YOUNG SINGLES/COUPLES     0.034745
Mainstream        MIDAGE SINGLES/COUPLES    0.044966
                  NEW FAMILIES              0.008855
                  OLDER FAMILIES            0.053664
                  OLDER SINGLES/COUPLES     0.069146
                  RETIREES                  0.080935
                  YOUNG FAMILIES            0.048419
                  YOUNG SINGLES/COUPLES     0.079209
Premium           MIDAGE SINGLES/COUPLES    0.030850
                  NEW FAMILIES              0.006031
                  OLDER FAMILIES            0.042162
                  OLDER SINGLES/COUPLES     0.067115
                  RETIREES                  0.049591
                  YOUNG FAMILIES            0.043706
                  YOUNG SINGLES/COUPLES     0.023717
Name: TOT_SALES, dtype: float64
```

Performing an independent t-test between mainstream vs premium and budget middle-age and young singles and couples

Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

#t-test

```
from scipy import stats
```

#Mainstream vs premium

```
stats.ttest_ind([4.065642,3.994241],[3.770698,3.665414])
```

Ttest_indResult(statistic=4.903408005498769, pvalue=0.039164352682153285)

#Mainstream vs budget

```
stats.ttest_ind([4.065642,3.994241],[3.657366,3.743328])
```

Ttest_indResult(statistic=5.898899732826305, pvalue=0.027555775534860754)

The t-test results in a p-value of 0.03 and 0.02, i.e. the unit price for mainstream, young and mid-age singles and couples ARE significantly higher than that of budget or premium, young and midage singles and couples.
#Now we are focussing on the mainstream, young and mid-age singles and couples brands that these two customer segments prefer more than others

Deep diving into specific customer segments for insights
We have found quite a few interesting insights that we can dive deeper into. We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream – young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

```
midage=finaldf[(finaldf['PREMIUM_CUSTOMER']=='Mainstream') & (finaldf['LIFESTAGE']=='MIDAGE SINGLES/COUPLES')]

young=finaldf[(finaldf['PREMIUM_CUSTOMER']=='Mainstream') & (finaldf['LIFESTAGE']=='YOUNG SINGLES/COUPLES')]

print(f"MIDAGE SINGLES/COUPLES\n{midage['BRAND'].value_counts().head(5)}")

print(f"YOUNG SINGLES/COUPLES\n{young['BRAND'].value_counts().head(5)}")
```

```
MIDAGE SINGLES/COUPLES
Kettle       2136
Smiths       1276
Doritos      1210
Pringles     1159
Infuzions     679
Name: BRAND, dtype: int64
YOUNG SINGLES/COUPLES
Kettle       3844
Doritos      2379
Pringles     2315
Smiths       1921
Infuzions    1250
Name: BRAND, dtype: int64
```

#Kettle, Smiths and Doritos are popular among MIDAGE and Kettle, Pringles and Doritos are popular among YOUNG

```
print(f"MIDAGE SINGLES/COUPLES\n{midage['PACKAGE_SIZE'].value_counts().head(5)}")

print(f"YOUNG SINGLES/COUPLES\n{young['PACKAGE_SIZE'].value_counts().head(5)}")
```

```
MIDAGE SINGLES/COUPLES
175    2975
150    1777
134    1159
110    1124
170     882
Name: PACKAGE_SIZE, dtype: int64
YOUNG SINGLES/COUPLES
175    4997
150    3080
134    2315
110    2051
170    1575
Name: PACKAGE_SIZE, dtype: int64
```

#both the segments buy 175g,150g and 134 packets mostly

Association rules

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



basket=finaldf.groupby(['LYLTY_CARD_NBR','BRAND'])['PROD_QTY'].sum().unstack().fillna(0)

basket
```

```
def reducer(x):

    if x <= 0:

        return 0

    else:
```

```
    return 1
basket=basket.applymap(reducer)
basket
```

```
frequent=apriori(basket,0.1,use_colnames=True)
frequent
```

	support	itemsets
0	0.125745	(Cobs)
1	0.290446	(Doritos)
2	0.177311	(Infuzions)
3	0.423303	(Kettle)
4	0.289772	(Pringles)
5	0.180103	(RRD)
6	0.314896	(Smiths)
7	0.176624	(Thins)
8	0.122884	(Tostitos)
9	0.122449	(Twisties)
10	0.139661	(Woolworths)
11	0.136420	(Doritos, Kettle)
12	0.135452	(Pringles, Kettle)
13	0.135130	(Smiths, Kettle)

```
association_rules(frequent,metric='lift',min_threshold=1).sort_values(['support','confidence'],ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Doritos)	(Kettle)	0.290446	0.423303	0.136420	0.469693	1.109591	0.013474	1.087478
1	(Kettle)	(Doritos)	0.423303	0.290446	0.136420	0.322276	1.109591	0.013474	1.046966
2	(Pringles)	(Kettle)	0.289772	0.423303	0.135452	0.467444	1.104279	0.012791	1.082886
3	(Kettle)	(Pringles)	0.423303	0.289772	0.135452	0.319989	1.104279	0.012791	1.044436
4	(Smiths)	(Kettle)	0.314896	0.423303	0.135130	0.429125	1.013754	0.001833	1.010199
5	(Kettle)	(Smiths)	0.423303	0.314896	0.135130	0.319227	1.013754	0.001833	1.006362

Therefore if someone buys Doritos Kettle can be recommended and vice-versa. Same for Pringles and Kettle.

Thanks