In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import missingno as msno
df= pd.read_csv(r'C:\Users\Ceejay\Documents\QVI_transaction_data.csv')
demo= pd.read_csv(r'C:\Users\Ceejay\Documents\QVI_purchase_behaviour.csv')
df.head()
```

Out[1]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TO
0	7/1/2018	47	47142	42540	14	Smiths Crnkle Chip Orgnl Big Bag 380g	2	
1	7/1/2018	55	55073	48884	99	Pringles Sthrn FriedChicken 134g	2	
2	7/1/2018	55	55073	48884	91	CCs Tasty Cheese 175g	2	
3	7/1/2018	58	58351	54374	102	Kettle Mozzarella Basil & Pesto 175g	2	
4	7/1/2018	68	68193	65598	44	Thins Chips Light& Tangy 175g	2	

In [2]:
```
demo.head()
```

Out[2]:

	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

In [3]:
```
df.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  object 
 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(5), object(2)
memory usage: 16.2+ MB
```

DATA CLEANING

The date column datatype will be transformed into a datetime datatype

In [4]:
```
df['DATE']= pd.to_datetime(df.DATE)
```

Getting the number of unique loyalty card number in the transaction dataset

In [5]:
```
df.LYLTY_CARD_NBR.nunique()
```

Out[5]:
```
72637
```

In [6]:
```
df['PROD_NAME'].value_counts()
```

Out[6]:
```
Kettle Mozzarella   Basil & Pesto 175g      3304
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3296
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3269
Tyrrells Crisps     Ched & Chives 165g      3268
Cobs Popd Sea Salt  Chips 110g              3265
                                            ...
RRD Pc Sea Salt     165g                    1431
Woolworths Medium   Salsa 300g              1430
NCC Sour Cream &    Garden Chives 175g      1419
French Fries Potato Chips 175g              1418
WW Crinkle Cut      Original 175g           1410
Name: PROD_NAME, Length: 114, dtype: int64
```

Removing any instance where salsa was brought. First, the column is converted to a string format and then filtered for rows that doesnt contain Salsa.

In [7]:
```
df['PROD_NAME']= df['PROD_NAME'].astype('string')
```

In [8]:
```
count=[]
from collections import Counter
count= Counter([c for c in df['PROD_NAME'] for x in c.split()])
freq= pd.DataFrame([count.keys(), count.values()]).transpose()
freq.columns=['brand', 'count']
freq
```

Out[8]:

	brand	count
0	Smiths Crnkle Chip Orgnl Big Bag 380g	22631
1	Pringles Sthrn FriedChicken 134g	12332
2	CCs Tasty Cheese 175g	6156
3	Kettle Mozzarella Basil & Pesto 175g	19824
4	Thins Chips Light& Tangy 175g	15940
...	...	...
109	Infuzions Mango Chutny Papadums 70g	7535
110	Natural Chip Compny SeaSalt175g	5872
111	Smiths Crinkle Cut Chips Barbecue 170g	8934
112	WW D/Style Chip Sea Salt 200g	8814
113	Natural ChipCo Hony Soy Chckn175g	7300

In [9]:
```
freq.sort_values('brand', ascending= False)
```

Out[9]:

	brand	count
6	Woolworths Mild Salsa 300g	5964
94	Woolworths Medium Salsa 300g	5720
105	Woolworths Cheese Rings 190g	6064
52	WW Supreme Cheese Corn Chips 200g	9054
24	WW Sour Cream &OnionStacked Chips 160g	8898
...	...	...
107	Cheetos Chs & Bacon Balls 190g	8874
2	CCs Tasty Cheese 175g	6156
23	CCs Original 175g	4542
44	CCs Nacho Cheese 175g	5992
45	Burger Rings 220g	4692

114 rows × 2 columns

In [10]:
```
df= df[df['PROD_NAME'].str.contains('Salsa')==False]
```

In [11]:
```
df= df[df['PROD_NAME'].str.contains('salsa')==False]
```

The PROD_NAME column also contains the brand and size of the product bought. Therefore two columns 'PACK_SIZE' and 'BRAND' will be created.

In [12]:
```
df['PACK_SIZE']= df['PROD_NAME'].str.extract(r'(\d+)')
df.head()
```

Out[12]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT
0	2018-07-01	47	47142	42540	14	Smiths Crnkle Chip Orgnl Big Bag 380g	2	
1	2018-07-01	55	55073	48884	99	Pringles Sthrn FriedChicken 134g	2	
2	2018-07-01	55	55073	48884	91	CCs Tasty Cheese 175g	2	
3	2018-07-01	58	58351	54374	102	Kettle Mozzarella Basil & Pesto 175g	2	
4	2018-07-01	68	68193	65598	44	Thins Chips Light& Tangy 175g	2	

In [13]:
```
df['BRAND']= df['PROD_NAME'].str.split(expand= True)[0]
```

In [14]:
```
df.BRAND.value_counts()
```

Out[14]:
```
Kettle        41288
Smiths        27390
Pringles      25102
Doritos       22041
Thins         14075
RRD           11894
Infuzions     11057
WW            10320
Cobs           9693
Tostitos       9471
Twisties       9454
Tyrrells       6442
Grain          6272
Natural        6050
Cheezels       4603
CCs            4551
Red            4427
Dorito         3185
Infzns         3144
Smith          2963
Cheetos        2927
Snbts          1576
Burger         1564
Woolworths     1516
GrnWves        1468
Sunbites       1432
NCC            1419
French         1418
Name: BRAND, dtype: Int64
```

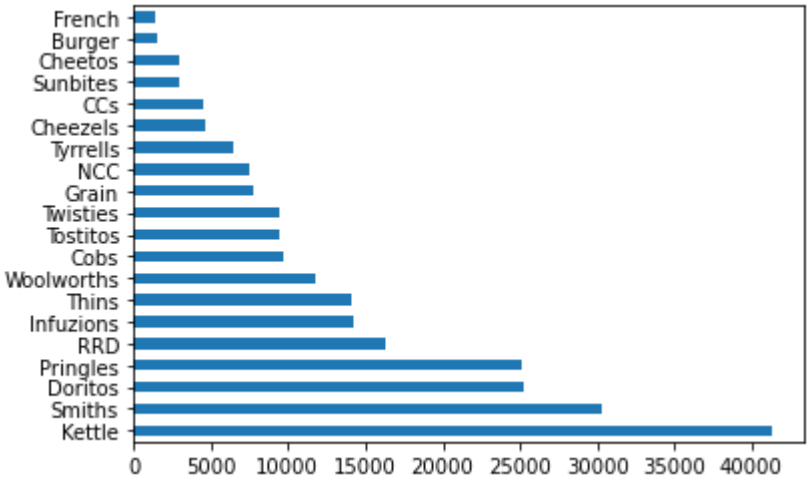
Some chip brands were recorded under different names such as 'Smith' and 'Smiths', 'RED' and 'RRD'. This will be replaced with a single value.

In [15]:
```
df['BRAND']= df['BRAND'].replace('Red', 'RRD')
df['BRAND']= df['BRAND'].replace('Dorito', 'Doritos')
df['BRAND']= df['BRAND'].replace('GrnWves', 'Grain')
df['BRAND']= df['BRAND'].replace('Infzns', 'Infuzions')
df['BRAND']= df['BRAND'].replace('Smith', 'Smiths')
df['BRAND']= df['BRAND'].replace('Natural', 'NCC')
df['BRAND']= df['BRAND'].replace('Snbts', 'Sunbites')
df['BRAND']= df['BRAND'].replace('WW', 'Woolworths')
```

The names of each brands has been corrected, now plotting the counts of each brand...

In [16]:
```
df.BRAND.value_counts().plot(kind= 'barh')
```

Out[16]:
```
<AxesSubplot:>
```



The kettle. Smiths, Doritos and pringles are the most common brands customers buy.

In [17]:
```
df.drop('PROD_NAME', axis= 1, inplace= True)
df.head()
```

Out[17]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES	PACK
0	2018-07-01	47	47142	42540	14	2	11.8	
1	2018-07-01	55	55073	48884	99	2	7.4	
2	2018-07-01	55	55073	48884	91	2	4.2	
3	2018-07-01	58	58351	54374	102	2	10.8	
4	2018-07-01	68	68193	65598	44	2	6.6	

Checking for missing values and duplicated values in the transaction dataset.....

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_QTY          0
TOT_SALES         0
PACK_SIZE         0
BRAND             0
dtype: int64
```

In [19]:
```
df.duplicated().sum()
```

Out[19]: 1

There are no missing values in the dataset, but there is a duplicated value. This will be dropped from the dataset.

In [20]:
```
df.drop_duplicates(inplace= True)
```

Getting the summary statistics of the transaction dataset.....

In [21]:
```
df.describe()
```

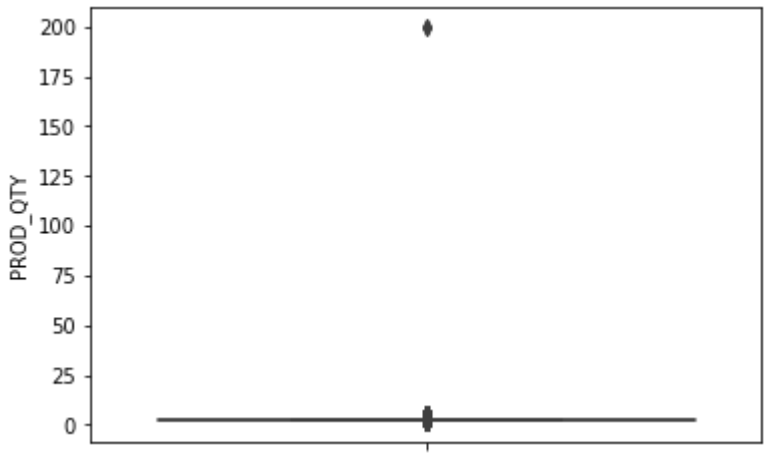
Out[21]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SA
count	246741.000000	2.467410e+05	2.467410e+05	246741.000000	246741.000000	246741.00
mean	135.051212	1.355311e+05	1.351312e+05	56.351835	1.908061	7.32
std	76.787231	8.071542e+04	7.814786e+04	33.695488	0.659832	3.07
min	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.70
25%	70.000000	7.001500e+04	6.756900e+04	26.000000	2.000000	5.80
50%	130.000000	1.303670e+05	1.351840e+05	53.000000	2.000000	7.40
75%	203.000000	2.030840e+05	2.026540e+05	87.000000	2.000000	8.80
max	272.000000	2.373711e+06	2.415841e+06	114.000000	200.000000	650.00

The prod_qty column has a maximum of 200 i.e about 200 items were bought, this should be an outlier.

In [22]:
```
sns.boxplot(y= 'PROD_QTY', data= df)
```

Out[22]: <AxesSubplot:ylabel='PROD_QTY'>



The boxplot shows clearly that the value is an outlier. Lets investigate this.....

In [23]:
```
df[df['PROD_QTY']== 200]
```

Out[23]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
35936	2018-08-19	226	226000	226201	4	200	650.0
234487	2019-05-20	226	226000	226210	4	200	650.0

This particular customer is a wholesale customer, see as they buy in bulk and they only bought twice in two years, his loyalty number will be removed from the analysis.

In [24]:
```
df= df[df['LYLTY_CARD_NBR']!= 226000]
df
```

Out[24]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
0	2018-07-01	47	47142	42540	14	2	11.8
1	2018-07-01	55	55073	48884	99	2	7.4
2	2018-07-01	55	55073	48884	91	2	4.2
3	2018-07-01	58	58351	54374	102	2	10.8
4	2018-07-01	68	68193	65598	44	2	6.6
...	...	...	...	...	...	...	...
264831	2019-06-30	242	242159	246222	36	2	10.8
264832	2019-06-30	244	244213	247339	93	2	7.8
264833	2019-06-30	256	256018	255130	105	2	3.6
264834	2019-06-30	257	257079	256218	71	2	8.6
264835	2019-06-30	265	265006	263307	106	1	3.0

246739 rows × 9 columns

Three columns containing the day, month and year of the transaction will be created, this will help in further analysis

In [25]:
```
df['DAY']= df.DATE.dt.day_name()
df['MONTH']= df.DATE.dt.month
df['YEAR']= df.DATE.dt.year
```

In [26]:
```
df.head()
```

Out[26]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES	PACK
0	2018-07-01	47	47142	42540	14	2	11.8	
1	2018-07-01	55	55073	48884	99	2	7.4	
2	2018-07-01	55	55073	48884	91	2	4.2	
3	2018-07-01	58	58351	54374	102	2	10.8	
4	2018-07-01	68	68193	65598	44	2	6.6	

Getting the count of unique values for the purchase behaviour dataset.....

Data cleaning for the customers dataset.

In [27]:
```
demo.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
```

In [28]:
```
demo.LYLTY_CARD_NBR.nunique()
```

Out[28]:
```
72637
```

There are 72637 unique customer loyalty card number in the purchase behaviour dataset which is the same as the one in the transactions dataset...

In [29]:
```
demo['LIFESTAGE'].value_counts()
```

Out[29]:
```
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
```

In [30]:
```
demo['PREMIUM_CUSTOMER'].value_counts()
```

Out[30]:
```
Mainstream    29245
Budget        24470
Premium       18922
Name: PREMIUM_CUSTOMER, dtype: int64
```

Merging the transaction dataset and the purchase behaviour dataset.....

In [31]:
```
data= df.merge(demo, on = 'LYLTY_CARD_NBR', how= 'inner')
data
```

Out[31]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
0	2018-07-01	47	47142	42540	14	2	11.8
1	2018-07-01	55	55073	48884	99	2	7.4
2	2018-07-01	55	55073	48884	91	2	4.2
3	2018-08-05	55	55073	48885	109	2	7.4
4	2018-09-02	55	55073	48886	89	2	10.8
...	...	...	...	...	...	...	...
246734	2019-06-30	6	6358	6145	87	1	3.8
246735	2019-06-30	90	90228	88954	77	2	8.8
246736	2019-06-30	169	169080	171120	1	2	5.8
246737	2019-06-30	244	244213	247339	93	2	7.8
246738	2019-06-30	265	265006	263307	106	1	3.0

246739 rows × 14 columns

Checking for missing values in the merged dataset

In [32]:
```
data.isnull().sum()
```

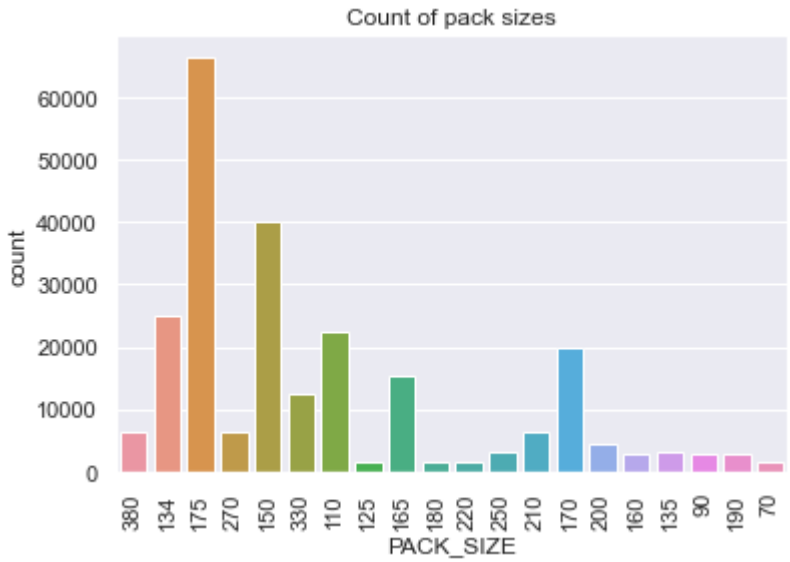
Out[32]:
```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_QTY          0
TOT_SALES         0
PACK_SIZE         0
BRAND             0
DAY               0
MONTH             0
YEAR              0
LIFESTAGE         0
PREMIUM_CUSTOMER  0
dtype: int64
```

There are no missing data in any of the columns

EDA

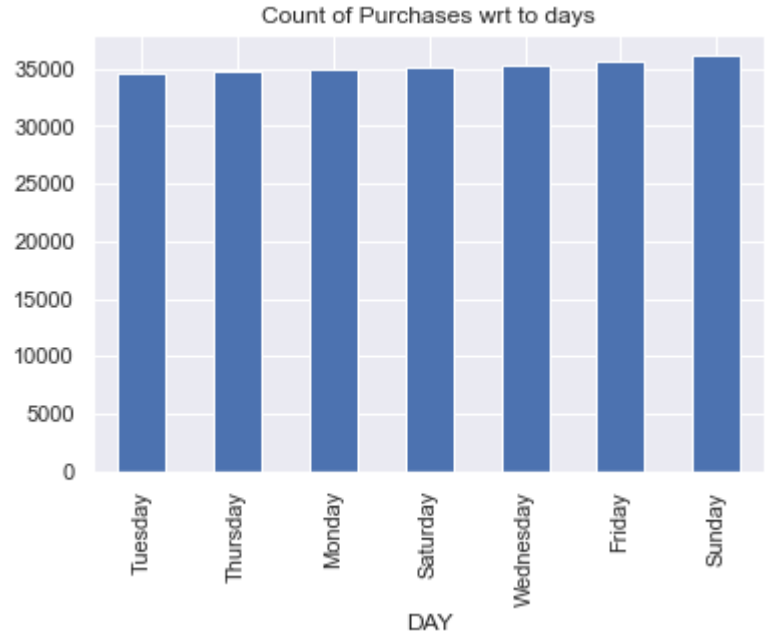
Getting the counts of the various pack sizes.....

In [33]:
```
sns.set()
sns.countplot(x= 'PACK_SIZE', data= data)
plt.xticks(rotation= 90)
plt.title('Count of pack sizes')
plt.show()
```



The maximum size is the 380g and the minimum size is the 70g pack.

In [34]:
```
daily_visit= data.groupby('DAY')['LYLTY_CARD_NBR'].count()
daily_visit.sort_values(ascending= True).plot(kind= 'bar')
plt.title('Count of Purchases wrt to days')
plt.show()
```



Customers tends to purchase chips more on weekends.

In [35]:
```
sns.lineplot(x= 'MONTH', y= 'TOT_SALES', data= data, estimator= np.sum)
plt.title('Total sales for each month')
plt.show()
```

More sales are made during december. This may be due to the holidays, but there is also an increase in sales during the months of March and June. This may also be due to the occurence of the summer and spring holidays.

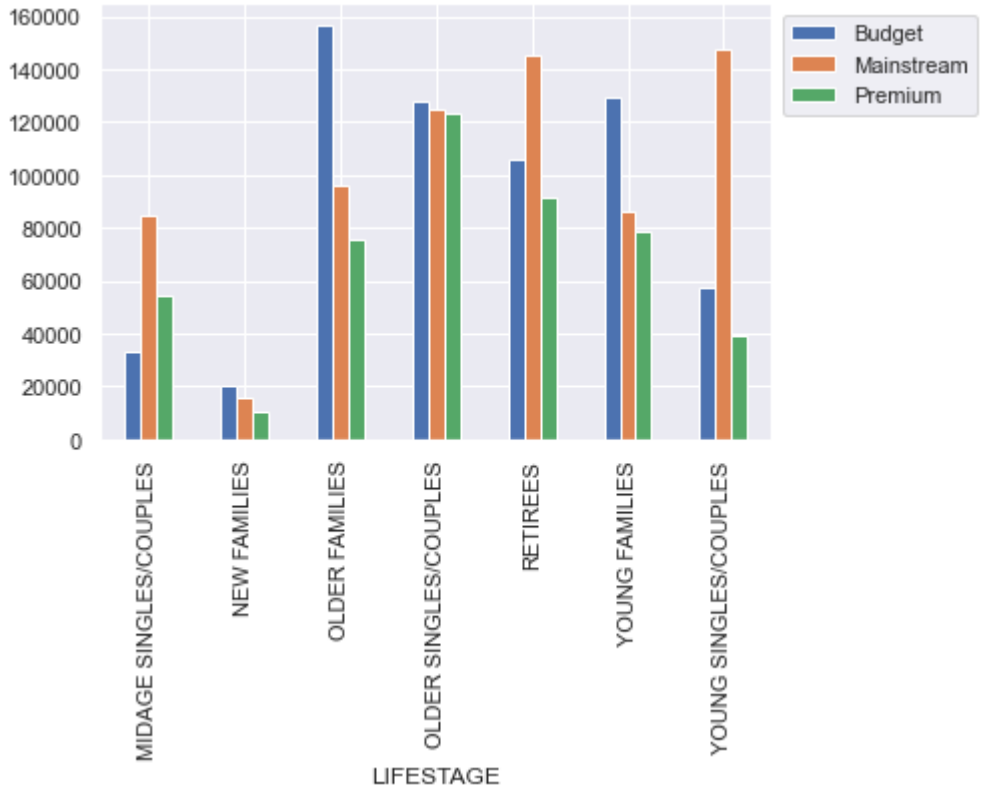
Who spends most on chips?

In [36]:
```
life_prem_cust= pd.pivot_table(index= 'LIFESTAGE', columns= 'PREMIUM_CUSTOMER'
life_prem_cust
```

Out[36]:

PREMIUM_CUSTOMER LIFESTAGE	Budget	Mainstream	Premium
MIDAGE SINGLES/COUPLES	33345.70	84734.25	54443.85
NEW FAMILIES	20607.45	15979.70	10760.80
OLDER FAMILIES	156863.75	96413.55	75242.60
OLDER SINGLES/COUPLES	127833.60	124648.50	123531.55
RETIREES	105916.30	145168.95	91296.65
YOUNG FAMILIES	129717.95	86338.25	78571.70
YOUNG SINGLES/COUPLES	57122.10	147582.20	39052.30

In [37]:
```
life_prem_cust.plot(kind= 'bar')
plt.legend(loc= 'upper left', bbox_to_anchor= (1, 1))
plt.show()
```



Most sales are coming from (older families- budget) customers, the (retirees- mainstream) customers and the (young singles- mainstream) customers.

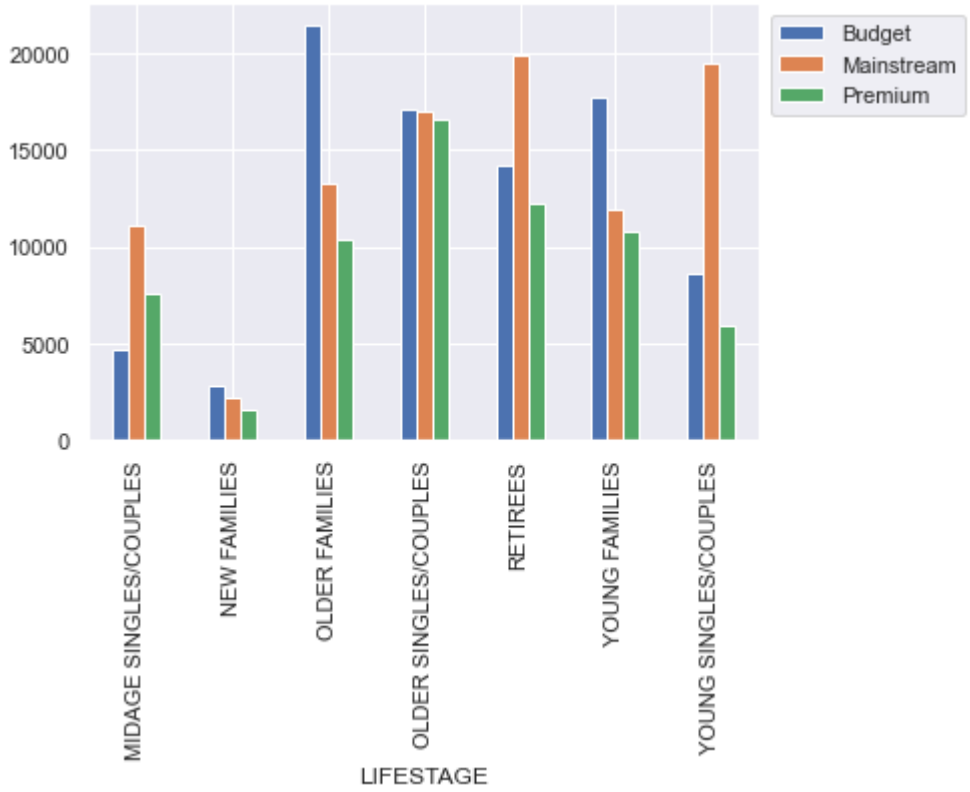
What is the count of customers in each lifestage/premium customer category?

In [38]:
```
life_prem_count= pd.crosstab(data.LIFESTAGE, data.PREMIUM_CUSTOMER)
life_prem_count
```

Out[38]:

PREMIUM_CUSTOMER	Budget	Mainstream	Premium
LIFESTAGE			
MIDAGE SINGLES/COUPLES	4691	11095	7612
NEW FAMILIES	2824	2185	1488
OLDER FAMILIES	21514	13241	10403
OLDER SINGLES/COUPLES	17172	17061	16559
RETIREES	14225	19970	12236
YOUNG FAMILIES	17763	11947	10784
YOUNG SINGLES/COUPLES	8573	19544	5852

In [39]:
```
life_prem_count.plot(kind= 'bar')
plt.legend(loc= 'upper left', bbox_to_anchor= (1, 1))
plt.show()
```



Most of the customers comes from the (older families- budget) customers, the (retirees- mainstream) customers and the (young singles- mainstream) category. This is a major factor for the high sales for each category.

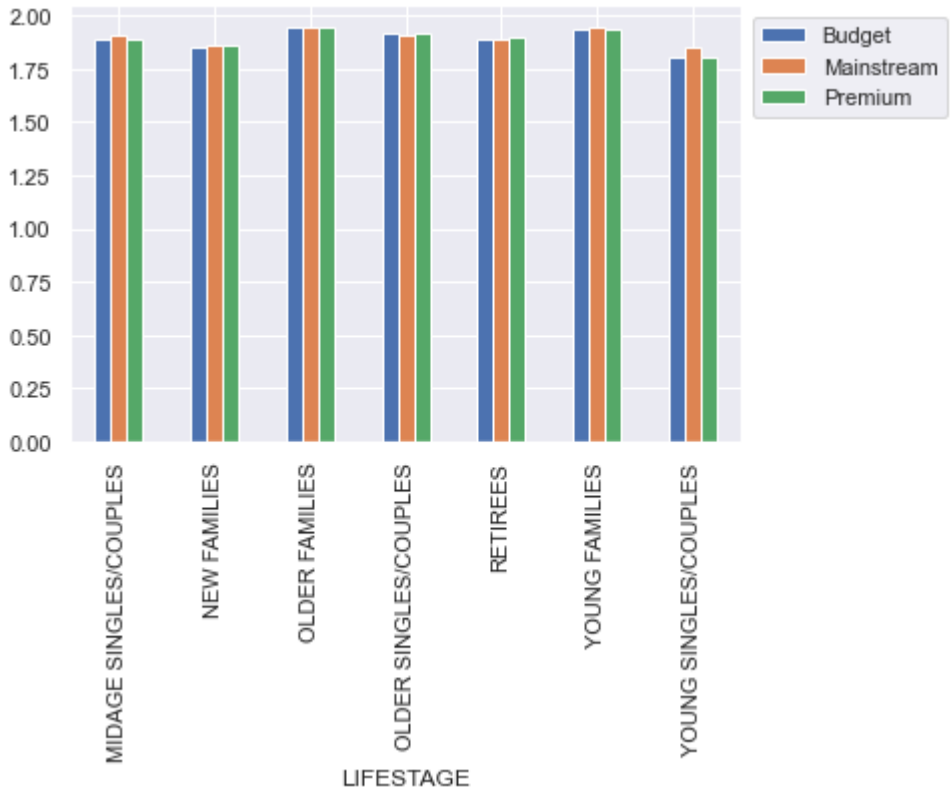
What is the average price per lifestage and premium customer?

In [40]:
```
life_prem_qty= pd.pivot_table(index= 'LIFESTAGE', columns= 'PREMIUM_CUSTOMER',
life_prem_qty
```

Out[40]:

PREMIUM_CUSTOMER	Budget	Mainstream	Premium
LIFESTAGE			
MIDAGE SINGLES/COUPLES	1.893626	1.911942	1.891750
NEW FAMILIES	1.855878	1.858124	1.860887
OLDER FAMILIES	1.945384	1.948795	1.945496
OLDER SINGLES/COUPLES	1.914920	1.911201	1.913944
RETIREES	1.893286	1.886680	1.901438
YOUNG FAMILIES	1.941226	1.941408	1.938149
YOUNG SINGLES/COUPLES	1.808002	1.853510	1.807075

In [41]:
```
life_prem_qty.plot(kind= 'bar')
plt.legend(loc= 'upper left', bbox_to_anchor= (1, 1))
plt.show()
```



Older families and younger families in general tends to buy more quantity of chips.

Average price per units for each segments

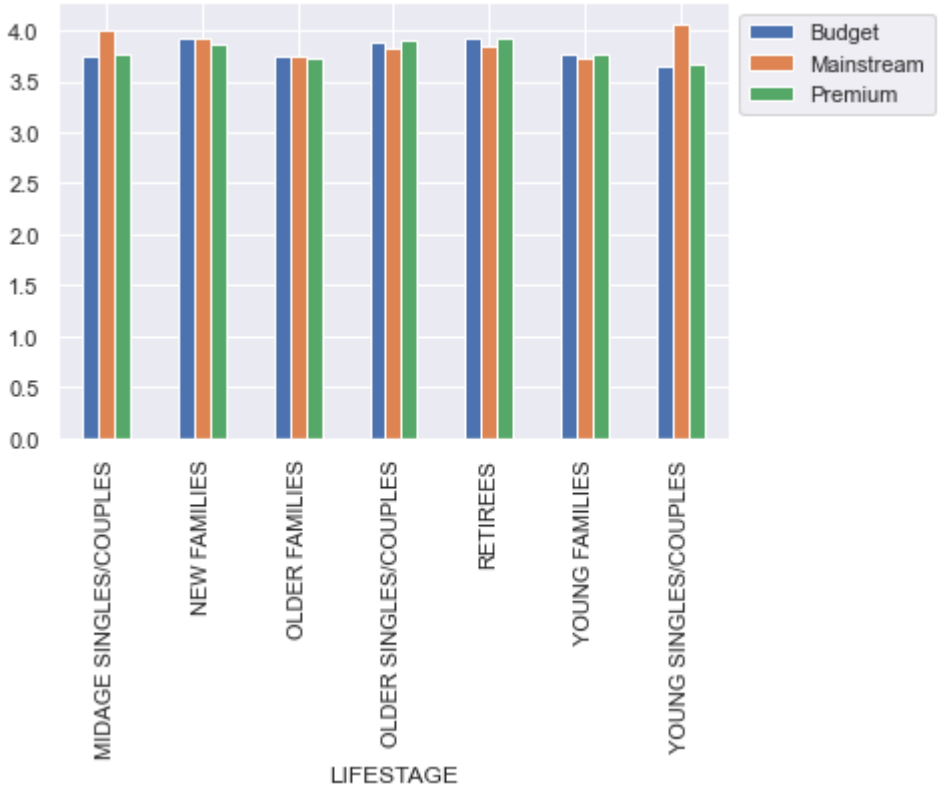
In [42]:
```
data['PPU']= data.TOT_SALES/data.PROD_QTY
```

In [43]:
```
life_prem_ppu= pd.pivot_table(index= 'LIFESTAGE', columns= 'PREMIUM_CUSTOMER',
life_prem_ppu
```

Out[43]:

PREMIUM_CUSTOMER / LIFESTAGE	Budget	Mainstream	Premium
MIDAGE SINGLES/COUPLES	3.743328	3.994241	3.770698
NEW FAMILIES	3.917688	3.916133	3.872110
OLDER FAMILIES	3.745340	3.737077	3.717000
OLDER SINGLES/COUPLES	3.882096	3.814665	3.893236
RETIREES	3.924404	3.844294	3.920942
YOUNG FAMILIES	3.760737	3.724533	3.762150
YOUNG SINGLES/COUPLES	3.657366	4.065642	3.665414

In [44]:
```
life_prem_ppu.plot(kind= 'bar')
plt.legend(loc= 'upper left', bbox_to_anchor= (1, 1))
plt.show()
```



The mainstream segment of the midage and young singles/couples category are more likely to pay more price for a packet of chip than their budget and premium counterpart. This may be due to premium shoppers being more likely to purchase healthy snacks.

Performing a test to see whether the difference is signnificant

In [45]:
```
from scipy.stats import ttest_ind
x= data[['LIFESTAGE', 'PREMIUM_CUSTOMER', 'PPU']]
df1= x.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PPU'].mean()
df1
```

Out[45]:
```
LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget              3.743328
                        Mainstream          3.994241
                        Premium             3.770698
NEW FAMILIES            Budget              3.917688
                        Mainstream          3.916133
                        Premium             3.872110
OLDER FAMILIES          Budget              3.745340
                        Mainstream          3.737077
                        Premium             3.717000
OLDER SINGLES/COUPLES   Budget              3.882096
                        Mainstream          3.814665
                        Premium             3.893236
RETIREES                Budget              3.924404
                        Mainstream          3.844294
                        Premium             3.920942
YOUNG FAMILIES          Budget              3.760737
                        Mainstream          3.724533
                        Premium             3.762150
YOUNG SINGLES/COUPLES   Budget              3.657366
                        Mainstream          4.065642
                        Premium             3.665414
Name: PPU, dtype: float64
```

Mainstream vs budget

In [46]:
```
ttest_ind([3.994, 4.065], [3.743, 3.657], alternative= 'greater')
```

Out[46]:
```
Ttest_indResult(statistic=5.909185145532703, pvalue=0.013731922514446288)
```

The average price per chip is significantly greater for mainstream young singles- midage families than their budget counterpart.

Mainstream vs premium

In [47]:
```
ttest_ind([3.994, 4.065], [3.770, 3.665], alternative= 'greater')
```

Out[47]:
```
Ttest_indResult(statistic=4.923009891431924, pvalue=0.01943550279718279)
```

The average price per chip is significantly greater for mainstream young singles- midage families than the premium counterpart.

What brand does each young singles/couples - mainstream customer segment consumes the most.

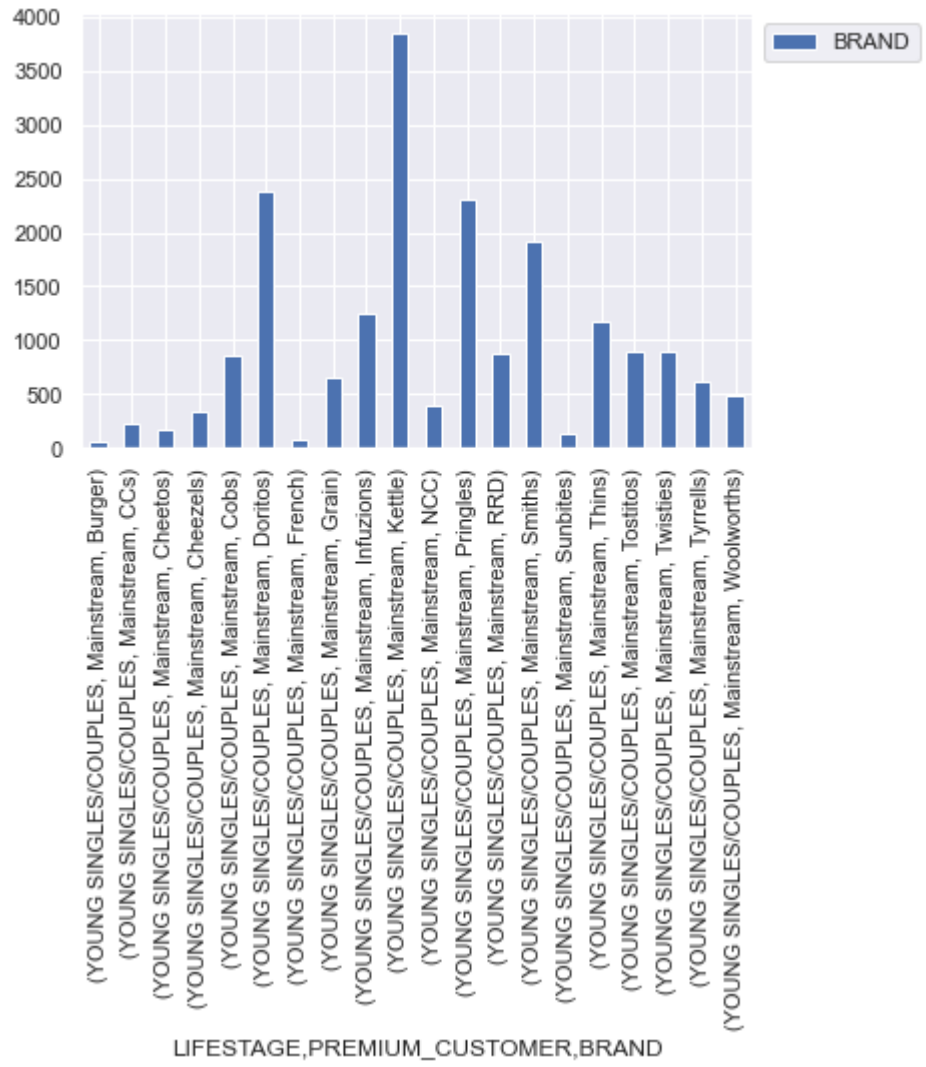
In [48]:
```
df2= data[(data.LIFESTAGE=='YOUNG SINGLES/COUPLES') & (data.PREMIUM_CUSTOMER==
pivot= df2.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER', 'BRAND'])['BRAND'].count(
pivot
```

Out[48]:

LIFESTAGE	PREMIUM_CUSTOMER	BRAND	
YOUNG SINGLES/COUPLES	Mainstream	Burger	62
		CCs	222
		Cheetos	166
		Cheezels	346
		Cobs	864
		Doritos	2379
		French	78
		Grain	646
		Infuzions	1250
		Kettle	3844
		NCC	394
		Pringles	2315
		RRD	875
		Smiths	1921
		Sunbites	128
		Thins	1166
		Tostitos	890
		Twisties	900
		Tyrrells	619
		Woolworths	479

Name: BRAND, dtype: int64

In [49]:
```
pivot.plot(kind= 'bar')
plt.legend(loc= 'upper left', bbox_to_anchor= (1, 1))
plt.show()
```



The kettle, doritos, pringles and smith brand are the brand customers in this segment likes to purchase the most. This is reflective of the whole dataset.

What brand does each midage singles/couples - mainstream customer segment consumes the most.

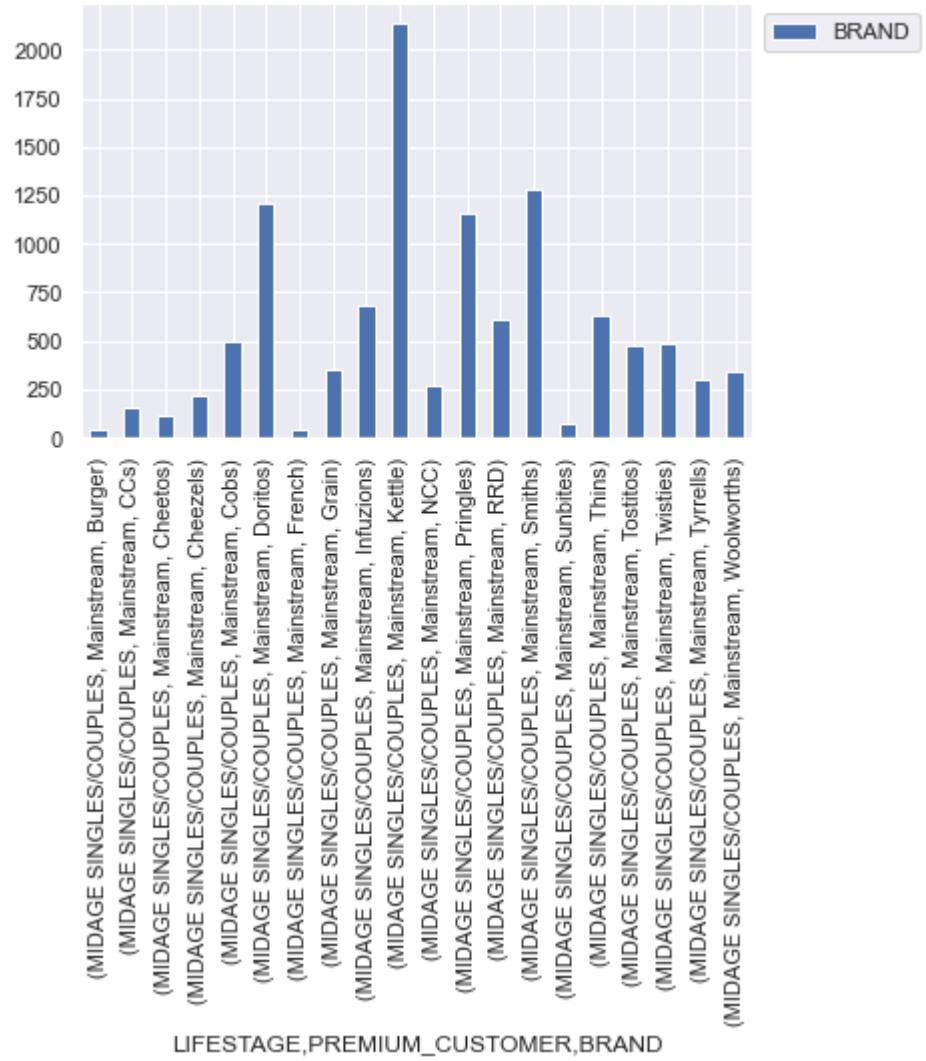
In [50]:
```
df3= data[(data.LIFESTAGE=='MIDAGE SINGLES/COUPLES') & (data.PREMIUM_CUSTOMER=
pivot1= df3.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER', 'BRAND'])['BRAND'].count
pivot1
```

Out[50]:

LIFESTAGE	PREMIUM_CUSTOMER	BRAND	
MIDAGE SINGLES/COUPLES	Mainstream	Burger	48
		CCs	159
		Cheetos	115
		Cheezels	221
		Cobs	495
		Doritos	1210
		French	43
		Grain	355
		Infuzions	679
		Kettle	2136
		NCC	271
		Pringles	1159
		RRD	611
		Smiths	1276
		Sunbites	69
		Thins	635
		Tostitos	479
		Twisties	490
		Tyrrells	298
		Woolworths	346

Name: BRAND, dtype: int64

In [51]:
```
pivot1.plot(kind= 'bar')
plt.legend(loc= 'upper left', bbox_to_anchor= (1, 1))
plt.show()
```



The kettle, doritos, pringles and smith brand are the brand customers in this segment likes to purchase the most. This is just like the young singles/couples segment.

Do customers in the young singles/couples mainstream segment and midage singles/couples mainstream segment tend to buy larger chip size?

In [52]:
```
size= df2.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER', 'PACK_SIZE'])['PACK_SIZE']
size.head(5)
```

Out[52]:
```
LIFESTAGE              PREMIUM_CUSTOMER  PACK_SIZE
YOUNG SINGLES/COUPLES  Mainstream        110          2051
                                         125            59
                                         134          2315
                                         135           290
                                         150          3080
Name: PACK_SIZE, dtype: int64
```

In [53]:
```
size1= df3.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER', 'PACK_SIZE'])['PACK_SIZE'
size1.head(5)
```

Out[53]:
```
LIFESTAGE               PREMIUM_CUSTOMER  PACK_SIZE
MIDAGE SINGLES/COUPLES  Mainstream        110          1124
                                          125            35
                                          134          1159
                                          135           163
                                          150          1777
Name: PACK_SIZE, dtype: int64
```

Both segments tends to buy the 110g, 125g, 134g 135g and 150g pack size.

In [54]:
```
basket= pd.crosstab(data['LYLTY_CARD_NBR'], data['BRAND'])
basket
```

Out[54]:

LYLTY_CARD_NBR	BRAND	Burger	CCs	Cheetos	Cheezels	Cobs	Doritos	French	Grain	Infuzions	K
1000		0	0	0	0	0	0	0	0	0	
1002		0	0	0	0	0	0	0	0	0	
1003		0	0	0	0	0	0	0	1	0	
1004		0	0	0	0	0	0	0	0	0	
1005		0	0	1	0	0	0	0	0	0	
...		...	...	...	...	...	...	...	...	...	...
2370651		0	0	0	0	0	1	0	0	0	
2370701		0	0	0	0	0	0	0	1	0	
2370751		0	0	0	0	0	0	0	0	0	
2370961		0	0	0	0	0	0	0	0	0	
2373711		0	0	0	0	0	0	0	0	0	

71287 rows × 20 columns

In [55]:
```
def reducer(x):
    if x<= 0:
        return 0
    else:
        return 1
basket= basket.applymap(reducer)
basket
```

Out[55]:

BRAND	Burger	CCs	Cheetos	Cheezels	Cobs	Doritos	French	Grain	Infuzions	K
LYLTY_CARD_NBR										
1000	0	0	0	0	0	0	0	0	0	
1002	0	0	0	0	0	0	0	0	0	
1003	0	0	0	0	0	0	0	1	0	
1004	0	0	0	0	0	0	0	0	0	
1005	0	0	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
2370651	0	0	0	0	0	1	0	0	0	
2370701	0	0	0	0	0	0	0	1	0	
2370751	0	0	0	0	0	0	0	0	0	
2370961	0	0	0	0	0	0	0	0	0	
2373711	0	0	0	0	0	0	0	0	0	

71287 rows × 20 columns

In [56]:
```
from mlxtend.frequent_patterns import apriori, association_rules
rules= apriori(basket, min_support= 0.05, use_colnames= True)
```

```
C:\Users\Ceejay\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommo
n.py:111: DeprecationWarning: DataFrames with non-bool types result in worse
computationalperformance and their support might be discontinued in the futur
e.Please use a DataFrame with bool type
  warnings.warn(
```

In [57]:
```
rules
```

Out[57]:

	support	itemsets
0	0.059562	(CCs)
1	0.061919	(Cheezels)
2	0.125745	(Cobs)
3	0.290446	(Doritos)
4	0.101589	(Grain)
5	0.177311	(Infuzions)
6	0.423303	(Kettle)
7	0.093664	(NCC)
8	0.289772	(Pringles)
9	0.180103	(RRD)
10	0.314896	(Smiths)
11	0.176624	(Thins)
12	0.122884	(Tostitos)
13	0.122449	(Twisties)
14	0.085696	(Tyrrells)
15	0.139661	(Woolworths)
16	0.060796	(Cobs, Kettle)
17	0.059450	(Infuzions, Doritos)
18	0.136420	(Doritos, Kettle)
19	0.094533	(Pringles, Doritos)
20	0.052450	(RRD, Doritos)
21	0.094155	(Smiths, Doritos)
22	0.058482	(Thins, Doritos)
23	0.083003	(Infuzions, Kettle)
24	0.058440	(Infuzions, Pringles)
25	0.060474	(Infuzions, Smiths)
26	0.135452	(Pringles, Kettle)
27	0.073842	(RRD, Kettle)
28	0.135130	(Smiths, Kettle)
29	0.083437	(Thins, Kettle)
30	0.058328	(Tostitos, Kettle)
31	0.058748	(Twisties, Kettle)
32	0.057865	(Woolworths, Kettle)
33	0.051566	(Pringles, RRD)
34	0.094449	(Pringles, Smiths)

	support	itemsets
35	0.057430	(Pringles, Thins)
36	0.090311	(RRD, Smiths)
37	0.051580	(Woolworths, RRD)
38	0.060712	(Thins, Smiths)
39	0.069606	(Woolworths, Smiths)
40	0.050177	(Pringles, Doritos, Kettle)

In [58]:
```
association_rules(rules, metric= 'lift', min_threshold= 1).sort_values('confid
```

32	(Pringles)	(Thins)	0.289772	0.176624	0.057430	0.198189	1.122098	0.006
25	(Kettle)	(Thins)	0.423303	0.176624	0.083437	0.197110	1.115988	0.008
15	(Kettle)	(Infuzions)	0.423303	0.177311	0.083003	0.196083	1.105868	0.007
39	(Smiths)	(Thins)	0.314896	0.176624	0.060712	0.192801	1.091590	0.005
19	(Smiths)	(Infuzions)	0.314896	0.177311	0.060474	0.192044	1.083088	0.004
9	(Doritos)	(RRD)	0.290446	0.180103	0.052450	0.180584	1.002673	0.000
45	(Pringles)	(Doritos, Kettle)	0.289772	0.136420	0.050177	0.173162	1.269324	0.010
46	(Doritos)	(Pringles, Kettle)	0.290446	0.135452	0.050177	0.172760	1.275430	0.010
1	(Kettle)	(Cobs)	0.423303	0.125745	0.060796	0.143624	1.142183	0.007
29	(Kettle)	(Twisties)	0.423303	0.122449	0.058748	0.138786	1.133420	0.006
27	(Kettle)	(Tostitos)	0.423303	0.122884	0.058328	0.137792	1.121319	0.006

Summary

- Sales is mainly due to the budget-older families, mainstream- young/singles couples and mainstream- retiree shoppers.
- The high spend for mainstream young singles/ couples and retiree is due to the fact that they are more than other buyers.
- Recommendations can be made for shoppers of different brands e.g Doritos can be recommended for buyers of Kettle and vice versa.