Chips Data Analysis

Introduction

The Quantium retail analytics team has been approached by the category manager of Chips, who wants to better understand the types of customers who purchase Chips and their purchasing behaviour within the region.

The insights from this analysis will feed into the supermarket's strategic plan for the chip category in the next half year.

In this analysis I will:

- Examine transaction data to look for inconsistencies, missing data, find outliers, correctly identify category items and numeric data across the different tables.
- Examine customer data to check for similar issues in customer data, look for nulls and merge the transaction data and customer data to prepare for data analysis
- Analyze data and customer segments by defining the metrics and looking at total sales, driver of sales, such as where the highest sales are coming from, etc. We will find interesting insights and trends in the data.
- Deep dive into customer segments by defining recommendations from the aformentioned insights, determine which segments we should target, and if packet sizes and form and overall conclusion based on the analysis.

In [277]:
```
#Import
import pandas as pd
import numpy as np

#Visualization
import seaborn as sns
import matplotlib.pyplot as plt

#Date
from datetime import datetime

#Warnings
import warnings
warnings.filterwarnings('ignore')

#Statistics
from scipy.stats import ttest_ind

#Regular Expression
import re

#Apriori
from mlxtend.frequent_patterns import apriori
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import association_rules

#Others
from collections import Counter
```

In [2]:
```
#Import datasets
transaction_data = pd.read_csv('QVI_transaction_data.csv')
customer_data = pd.read_csv('QVI_purchase_behaviour.csv')
```

Data Cleaning

Goals:

- Change date column to datetime
- Parse brand and pack sizes
- Remove special characters in PROD_NAME
- Remove salsa brands from PROD_NAME
- Find for outliers
- Replace brand names for similar brands

In [39]:
```
customer_data.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 [3]:
```
#View dataset info
transaction_data.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
```

In [4]:
```
transaction_data.head()
```

Out[4]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

In [5]:
```
#Converting serial date to datetime
date_changes = []
for i in transaction_data["DATE"]:
    dt = datetime.fromordinal(datetime(1900,1,1).toordinal() + i-2)
    date_changes.append(dt)

transaction_data["DATE"] = date_changes
```

In [6]:
```
transaction_data.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  datetime64[ns]
 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: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 16.2+ MB
```

In [7]:
```
#Check that we are looking at the right products
transaction_data["PROD_NAME"].unique()
```

Out[7]:
```
array(['Natural Chip        Compny SeaSalt175g',
       'CCs Nacho Cheese    175g',
       'Smiths Crinkle Cut  Chips Chicken 170g',
       'Smiths Chip Thinly  S/Cream&Onion 175g',
       'Kettle Tortilla ChpsHny&Jlpno Chili 150g',
       'Old El Paso Salsa   Dip Tomato Mild 300g',
       'Smiths Crinkle Chips Salt & Vinegar 330g',
       'Grain Waves         Sweet Chilli 210g',
       'Doritos Corn Chip Mexican Jalapeno 150g',
       'Grain Waves Sour    Cream&Chives 210G',
       'Kettle Sensations   Siracha Lime 150g',
       'Twisties Cheese     270g', 'WW Crinkle Cut      Chicken 175g',
       'Thins Chips Light&  Tangy 175g', 'CCs Original 175g',
       'Burger Rings 220g', 'NCC Sour Cream &    Garden Chives 175g',
       'Doritos Corn Chip Southern Chicken 150g',
       'Cheezels Cheese Box 125g', 'Smiths Crinkle      Original 330g',
       'Infzns Crn Crnchers Tangy Gcamole 110g',
       'Kettle Sea Salt     And Vinegar 175g',
       'Smiths Chip Thinly  Cut Original 175g', 'Kettle Original 175g',
       'Red Rock Deli Thai  Chilli&Lime 150g',
       'Pringles Sthrn FriedChicken 134g', 'Pringles Sweet&Spcy BBQ 134g',
       'Red Rock Deli SR    Salsa & Mzzrlla 150g',
       'Thins Chips         Originl saltd 175g',
       'Red Rock Deli Sp    Salt & Truffle 150G',
       'Smiths Thinly       Swt Chli&S/Cream175G', 'Kettle Chilli 175g',
       'Doritos Mexicana    170g',
       'Smiths Crinkle Cut  French OnionDip 150g',
       'Natural ChipCo      Hony Soy Chckn175g',
       'Dorito Corn Chp     Supreme 380g', 'Twisties Chicken270g',
       'Smiths Thinly Cut   Roast Chicken 175g',
       'Smiths Crinkle Cut  Tomato Salsa 150g',
       'Kettle Mozzarella   Basil & Pesto 175g',
       'Infuzions Thai SweetChili PotatoMix 110g',
       'Kettle Sensations   Camembert & Fig 150g',
       'Smith Crinkle Cut   Mac N Cheese 150g',
       'Kettle Honey Soy    Chicken 175g',
       'Thins Chips Seasonedchicken 175g',
       'Smiths Crinkle Cut  Salt & Vinegar 170g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'Doritos Salsa       Medium 300g', 'Kettle 135g Swt Pot Sea Salt',
       'Pringles SourCream  Onion 134g',
       'Doritos Corn Chips  Original 170g',
       'Twisties Cheese     Burger 250g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
       'Woolworths Mild     Salsa 300g',
       'Natural Chip Co     Tmato Hrb&Spce 175g',
       'Smiths Crinkle Cut  Chips Original 170g',
       'Cobs Popd Sea Salt  Chips 110g',
       'Smiths Crinkle Cut  Chips Chs&Onion170g',
       'French Fries Potato Chips 175g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
       'Doritos Corn Chips  Cheese Supreme 170g',
       'Pringles Original   Crisps 134g',
       'RRD Chilli&         Coconut 150g',
       'WW Original Corn    Chips 200g',
       'Thins Potato Chips  Hot & Spicy 175g',
       'Cobs Popd Sour Crm  &Chives Chips 110g',
       'Smiths Crnkle Chip  Orgnl Big Bag 380g',
       'Doritos Corn Chips  Nacho Cheese 170g',
       'Kettle Sensations   BBQ&Maple 150g',
       'WW D/Style Chip     Sea Salt 200g',
       'Pringles Chicken    Salt Crips 134g',
       'WW Original Stacked Chips 160g',
       'Smiths Chip Thinly  CutSalt/Vinegr175g', 'Cheezels Cheese 330g',
       'Tostitos Lightly    Salted 175g',
       'Thins Chips Salt &  Vinegar 175g',
       'Smiths Crinkle Cut  Chips Barbecue 170g', 'Cheetos Puffs 165g',
       'RRD Sweet Chilli &  Sour Cream 165g',
       'WW Crinkle Cut      Original 175g',
       'Tostitos Splash Of  Lime 175g', 'Woolworths Medium   Salsa 300g',
       'Kettle Tortilla ChpsBtroot&Ricotta 150g',
       'CCs Tasty Cheese    175g', 'Woolworths Cheese   Rings 190g',
       'Tostitos Smoked     Chipotle 175g', 'Pringles Barbeque   134g',
       'WW Supreme Cheese   Corn Chips 200g',
       'Pringles Mystery    Flavour 134g',
       'Tyrrells Crisps     Ched & Chives 165g',
       'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
       'Cheetos Chs & Bacon Balls 190g', 'Pringles Slt Vingar 134g',
       'Infuzions SourCream&Herbs Veg Strws 110g',
       'Kettle Tortilla ChpsFeta&Garlic 150g',
       'Infuzions Mango     Chutny Papadums 70g',
       'RRD Steak &         Chimuchurri 150g',
       'RRD Honey Soy       Chicken 165g',
       'Sunbites Whlegrn    Crisps Frch/Onin 90g',
       'RRD Salt & Vinegar  165g', 'Doritos Cheese      Supreme 330g',
       'Smiths Crinkle Cut  Snag&Sauce 150g',
       'WW Sour Cream &OnionStacked Chips 160g',
       'RRD Lime & Pepper   165g',
       'Natural ChipCo Sea  Salt & Vinegr 175g',
       'Red Rock Deli Chikn&Garlic Aioli 150g',
       'RRD SR Slow Rst     Pork Belly 150g', 'RRD Pc Sea Salt     165g',
       'Smith Crinkle Cut   Bolognese 150g', 'Doritos Salsa Mild  300g'],
      dtype=object)
```

In [8]:
```
#Removing special characters
transaction_data_words = transaction_data["PROD_NAME"].str.replace("&", '')
```

In [9]:
```
pattern = r"[0-9]+[gG]"
transaction_data_words = transaction_data["PROD_NAME"].str.replace(pattern, '')
```

In [10]:
```
transaction_data_words.head()
```

Out[10]:
```
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
```

In [11]:
```
words = Counter()
transaction_data_words.str.split().apply(words.update)
print(words)
```

```
Counter({'Chips': 49770, 'Kettle': 41288, '&': 35565, 'Smiths': 28860, 'Salt': 27976, 'Cheese': 2789
0, 'Pringles': 25102, 'Doritos': 24962, 'Crinkle': 23960, 'Corn': 22063, 'Original': 21560, 'Cut': 20
754, 'Chip': 18645, 'Chicken': 18577, 'Salsa': 18094, 'Sea': 14145, 'Thins': 14075, 'Sour': 13882, 'C
risps': 12607, 'Vinegar': 12402, 'Chilli': 12389, 'RRD': 11894, 'Infuzions': 11057, 'Supreme': 10963,
'WW': 10320, 'Cobs': 9693, 'Popd': 9693, 'Tortilla': 9580, 'Tostitos': 9471, 'Twisties': 9454, 'Sensa
tions': 9429, 'Old': 9324, 'El': 9324, 'Paso': 9324, 'Dip': 9324, 'Sweet': 7883, 'Lime': 7852, 'Tomat
o': 7669, 'Cream': 7618, 'Thinly': 7507, 'Tyrrells': 6442, 'And': 6373, 'BBQ': 6351, 'Tangy': 6332,
'Grain': 6272, 'Waves': 6272, 'Lightly': 6248, 'Salted': 6248, 'Soy': 6121, 'Natural': 6050, 'Mild':
6048, 'Red': 5885, 'Rock': 5885, 'Deli': 5885, 'Thai': 4737, 'Burger': 4733, 'Swt': 4718, 'Chives': 4
687, 'Honey': 4661, 'Nacho': 4658, 'Potato': 4647, 'Cheezels': 4603, 'CCs': 4551, 'Woolworths': 4437,
'Mozzarella': 3304, 'Basil': 3304, 'Pesto': 3304, 'ChpsHny&Jlpno': 3296, 'Chili': 3296, 'Swt/Chlli':
3269, '&Sr/Cream': 3269, 'Ched': 3268, 'Pot': 3257, 'Splash': 3252, 'Of': 3252, 'SweetChili': 3242,
'PotatoMix': 3242, 'Crnkle': 3233, 'Orgnl': 3233, 'Big': 3233, 'Bag': 3233, 'Hot': 3229, 'Spicy': 322
9, 'Camembert': 3219, 'Fig': 3219, 'Barbeque': 3210, 'Mexican': 3204, 'Jalapeno': 3204, 'Light&': 318
8, 'Dorito': 3185, 'Chp': 3185, 'Sweet&Spcy': 3177, 'Rib': 3174, 'Prawn': 3174, 'Crackers': 3174, 'So
uthern': 3172, 'SourCream': 3162, 'Onion': 3162, 'Crm': 3159, '&Chives': 3159, 'ChpsBtroot&Ricotta':
3146, 'Smoked': 3145, 'Chipotle': 3145, 'Infzns': 3144, 'Crn': 3144, 'Crnchers': 3144, 'Gcamole': 314
4, 'ChpsFeta&Garlic': 3138, 'SourCream&Herbs': 3134, 'Veg': 3134, 'Strws': 3134, 'Siracha': 3127, 'Ch
nky': 3125, 'Tom': 3125, 'Ht': 3125, 'Mexicana': 3115, 'Seasonedchicken': 3114, 'Med': 3114, 'Myster
y': 3114, 'Flavour': 3114, 'Cream&Chives': 3105, 'Crips': 3104, 'Slt': 3095, 'Vingar': 3095, 'Sthrn':
3083, 'FriedChicken': 3083, 'BBQ&Maple': 3083, 'Rings': 3080, 'ChipCo': 3010, 'SR': 2984, 'Smith': 29
63, 'Cheetos': 2927, 'Medium': 2879, 'French': 2856, 'Snbts': 1576, 'Whlgrn': 1576, 'Cheddr&Mstrd': 1
576, 'Co': 1572, 'Tmato': 1572, 'Hrb&Spce': 1572, 'Vinegr': 1550, 'Tasty': 1539, 'Slow': 1526, 'Rst':
1526, 'Pork': 1526, 'Belly': 1526, 'Roast': 1519, 'Mac': 1512, 'N': 1512, 'Mango': 1507, 'Chutny': 15
07, 'Papadums': 1507, 'Chilli&': 1506, 'Coconut': 1506, 'Snag&Sauce': 1503, 'Sp': 1498, 'Truffle': 14
98, 'Chilli&Lime': 1495, 'Barbecue': 1489, 'Stacked': 1487, '&OnionStacked': 1483, 'Chs&Onion': 1481,
'Chs': 1479, 'Bacon': 1479, 'Balls': 1479, 'S/Cream&Onion': 1473, 'Pepper': 1473, 'D/Style': 1469, 'C
ompny': 1468, 'SeaSalt': 1468, 'GrnWves': 1468, 'Plus': 1468, 'Btroot': 1468, 'Jam': 1468, 'Chli&S/Cr
eam': 1461, 'Hony': 1460, 'Chckn': 1460, 'Mzzrlla': 1458, 'Steak': 1455, 'Chimuchurri': 1455, 'Box':
1454, 'Bolognese': 1451, 'Puffs': 1448, 'Originl': 1441, 'saltd': 1441, 'CutSalt/Vinegr': 1440, 'Onio
nDip': 1438, 'Chikn&Garlic': 1434, 'Aioli': 1434, 'Sunbites': 1432, 'Whlegrn': 1432, 'Frch/Onin': 143
2, 'Pc': 1431, 'NCC': 1419, 'Garden': 1419, 'Fries': 1418})
```

In [12]:
```
words_series = pd.Series(words)
words_series.sort_values(ascending=False)
```

Out[12]:
```
Chips       49770
Kettle      41288
&           35565
Smiths      28860
Salt        27976
            ...  
Sunbites     1432
Pc           1431
NCC          1419
Garden       1419
Fries        1418
Length: 196, dtype: int64
```

In [13]:
```
#There are alot of words with Salsa in them. We want only chips in this dataset so we will remove these
 values
print(words_series[words_series.index == "Salsa"])
```

```
Salsa    18094
dtype: int64
```

In [14]:
```
pattern = r"([sS]alsa)"
transaction_data_ns = transaction_data.drop(transaction_data[transaction_data["PROD_NAME"].str.contains
(pattern)].index)
transaction_data_ns[transaction_data_ns["PROD_NAME"].str.contains(pattern)].shape[0]
```

Out[14]: 0

In [15]:
```
#Now we will find for outliers and NA values
transaction_data_ns.isnull().sum()
```

Out[15]:
```
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
```

In [16]:
```
transaction_data_ns.describe()
```

Out[16]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
count	246742.000000	2.467420e+05	2.467420e+05	246742.000000	246742.000000	246742.000000
mean	135.051098	1.355310e+05	1.351311e+05	56.351789	1.908062	7.321322
std	76.787096	8.071528e+04	7.814772e+04	33.695428	0.659831	3.077828
min	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.700000
25%	70.000000	7.001500e+04	6.756925e+04	26.000000	2.000000	5.800000
50%	130.000000	1.303670e+05	1.351830e+05	53.000000	2.000000	7.400000
75%	203.000000	2.030840e+05	2.026538e+05	87.000000	2.000000	8.800000
max	272.000000	2.373711e+06	2.415841e+06	114.000000	200.000000	650.000000

We can clearly see that there are outliers in PROD_QTY and TOT_SALES

In [17]:
```
#We can examine the cases where 200 products are bought
transaction_data_ns[transaction_data_ns["PROD_QTY"] == 200]
```

Out[17]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

The cases where **PROD_QTY** is 200, is bought by the same customer since the loyalty card number is the same. Let's see the behavior of this customer; if this customer has other large quantity transactions.

In [18]:
```
transaction_data_ns[transaction_data_ns["LYLTY_CARD_NBR"] == 226000]
```

Out[18]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

It looks like this customer has only made two transactions and is not an ordinary retail customer. This customer might be buying for commercial purposes instead. We will remove these two values.

In [19]:
```
transaction_data_ns = transaction_data_ns.drop(transaction_data_ns[transaction_data_ns["LYLTY_CARD_NBR"
] == 226000].index)
transaction_data_ns[transaction_data_ns["LYLTY_CARD_NBR"] == 226000].shape[0]
```

Out[19]: 0

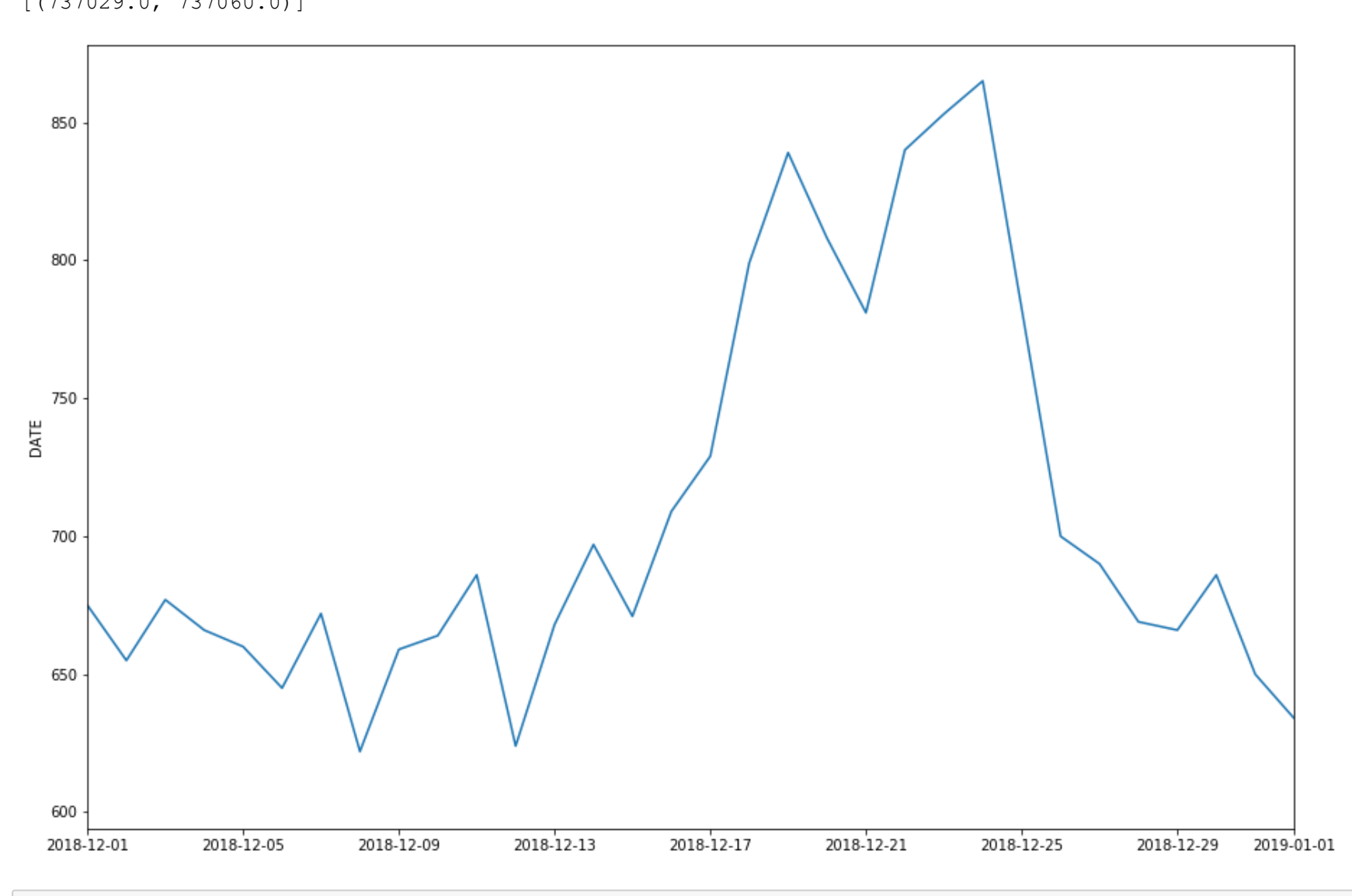
Now we will check the transaction line over time to see if there are any obvious data issues such as missing values

In [20]:
```
transaction_per_date = transaction_data_ns["DATE"].value_counts()
print(transaction_per_date)
```

```
2018-12-24    865
2018-12-23    853
2018-12-22    840
2018-12-19    839
2018-12-20    808
             ... 
2019-06-24    612
2018-10-18    611
2018-11-25    610
2018-09-22    609
2019-06-13    607
Name: DATE, Length: 364, dtype: int64
```

It seems like there is a missing date since there are only 364 rows

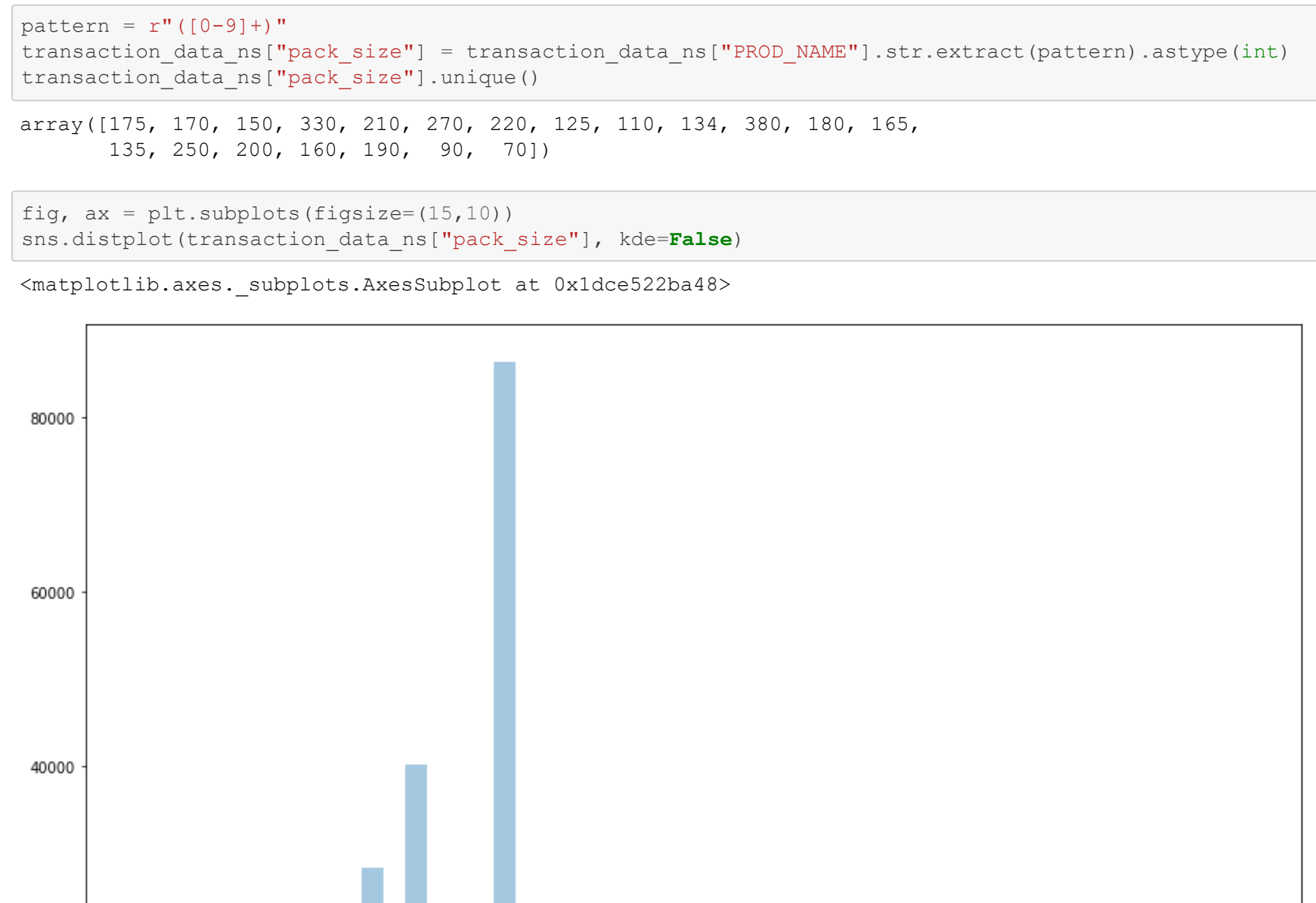
In [21]:
```
fig, ax = plt.subplots(figsize=(15,10))
ax = sns.lineplot(transaction_per_date.index, transaction_per_date, ax=ax)
```



It is obvious that there is an outlier between 2018-11 to 2019-01.

In [22]:
```
fig1, ax1 = plt.subplots(figsize=(15,10))
ax1 = sns.lineplot(transaction_per_date.index, transaction_per_date, ax=ax1)
ax1.set(xlim=(datetime(2018,12,1), datetime(2019,1,1)))
```

Out[22]: [(737029.0, 737060.0)]



In [23]:
```
print(transaction_data_ns[transaction_data_ns["DATE"] == datetime(2018,12,25)].shape[0])
```

```
0
```

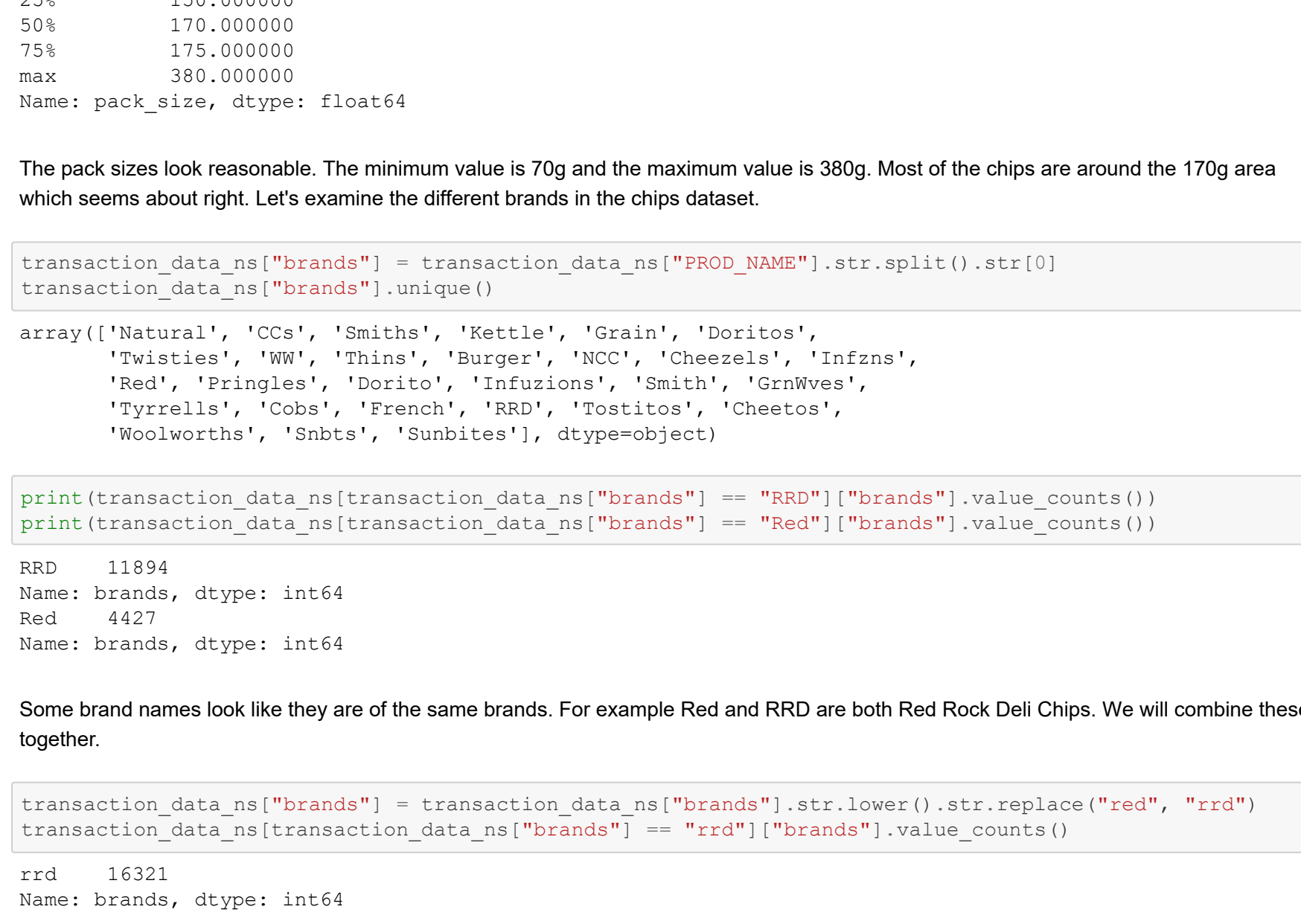
We can see that leading up to christmas, there is a surge in transactions. However, during christmas day, there is 0 purchases. This is due to stores closing during christmas day. This does not count as an outlier. Now we will examine the package size and brands of chips.

In [24]:
```
pattern = r"([0-9]+)"
transaction_data_ns["pack_size"] = transaction_data_ns["PROD_NAME"].str.extract(pattern).astype(int)
transaction_data_ns["pack_size"].unique()
```

Out[24]:
```
array([175, 170, 150, 330, 210, 270, 220, 125, 110, 134, 380, 180, 165,
       135, 250, 200, 160, 190,  90,  70])
```

In [25]:
```
fig, ax = plt.subplots(figsize=(15,10))
sns.distplot(transaction_data_ns["pack_size"], kde=False)
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1dce522ba48>



In [26]:
```
transaction_data_ns["pack_size"].describe()
```

Out[26]:
```
count    246740.000000
mean        175.583521
std          59.432118
min          70.000000
25%         150.000000
50%         170.000000
75%         175.000000
max         380.000000
Name: pack_size, dtype: float64
```

The pack sizes look reasonable. The minimum value is 70g and the maximum value is 380g. Most of the chips are around the 170g area which seems about right. Let's examine the different brands in the chips dataset.

In [27]:
```
transaction_data_ns["brands"] = transaction_data_ns["PROD_NAME"].str.split().str[0]
transaction_data_ns["brands"].unique()
```

Out[27]:
```
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)
```

In [28]:
```
print(transaction_data_ns[transaction_data_ns["brands"] == "RRD"]["brands"].value_counts())
print(transaction_data_ns[transaction_data_ns["brands"] == "Red"]["brands"].value_counts())
```

```
RRD    11894
Name: brands, dtype: int64
Red    4427
Name: brands, dtype: int64
```

Some brand names look like they are of the same brands. For example Red and RRD are both Red Rock Deli Chips. We will combine these together.

In [29]:
```
transaction_data_ns["brands"] = transaction_data_ns["brands"].str.lower().str.replace("red", "rrd")
transaction_data_ns[transaction_data_ns["brands"] == "rrd"]["brands"].value_counts()
```

Out[29]:
```
rrd    16321
Name: brands, dtype: int64
```

In [30]:
```
transaction_data_ns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 246740 entries, 0 to 264835
Data columns (total 10 columns):
 #   Column          Non-Null Count   Dtype         
---  ------          --------------   -----         
 0   DATE            246740 non-null  datetime64[ns]
 1   STORE_NBR       246740 non-null  int64         
 2   LYLTY_CARD_NBR  246740 non-null  int64         
 3   TXN_ID          246740 non-null  int64         
 4   PROD_NBR        246740 non-null  int64         
 5   PROD_NAME       246740 non-null  object        
 6   PROD_QTY        246740 non-null  int64         
 7   TOT_SALES       246740 non-null  float64       
 8   pack_size       246740 non-null  int32         
 9   brands          246740 non-null  object        
dtypes: datetime64[ns](1), float64(1), int32(1), int64(5), object(2)
memory usage: 19.8+ MB
```

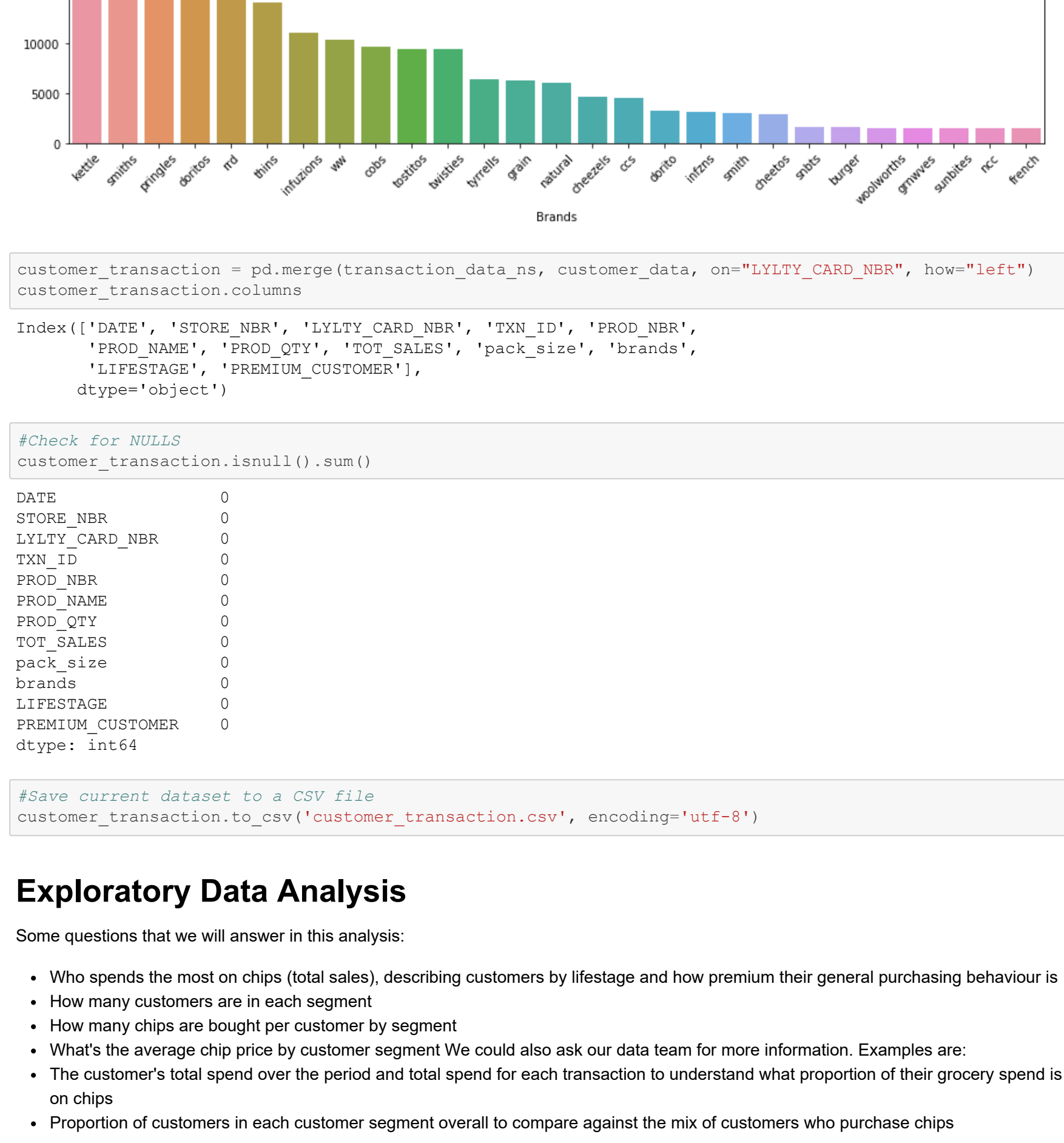
Data Preparation

Goals:

- Show distributions for brands and pack size
- Combine transaction_data and customer_data
- Find for nulls
- Export to CSV

In [38]:
```
#Visualizing brand and pack size distributions
brand = pd.Series(transaction_data_ns["brands"].value_counts())
fig = plt.figure(figsize=(15,15))
ax1 = fig.add_subplot(2,1,1)
ax2 = fig.add_subplot(2,1,2)
sns.distplot(transaction_data_ns["pack_size"], kde=False, ax=ax1).set_title("Distribution of Chip Pack
 Sizes")
sns.barplot(x=brand.index, y=brand, ax=ax2).set_title("Distribution of Chip Brands")
ax1.set(xlabel="Pack Size(grams)")
ax2.set_xticklabels(ax2.get_xticklabels(), rotation=45)
ax2.set(xlabel="Brands", ylabel="")
```

Out[38]: [Text(0, 0.5, ''), Text(0.5, 0, 'Brands')]



In [40]:
```
customer_transaction = pd.merge(transaction_data_ns, customer_data, on="LYLTY_CARD_NBR", how="left")
customer_transaction.columns
```

Out[40]:
```
Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES', 'pack_size', 'brands',
       'LIFESTAGE', 'PREMIUM_CUSTOMER'],
      dtype='object')
```

In [41]:
```
#Check for NULLS
customer_transaction.isnull().sum()
```

Out[41]:
```
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
pack_size           0
brands              0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [45]:
```
#Save current dataset to a CSV file
customer_transaction.to_csv('customer_transaction.csv', encoding='utf-8')
```

Exploratory Data Analysis

Some questions that we will answer in this analysis:

- Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is
- How many customers are in each segment
- How many chips are bought per customer by segment
- What's the average chip price by customer segment We could also ask our data team for more information. Examples are:
- The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips
- Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

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.

In [46]:
```
customer_transaction.head()
```

Out[46]:

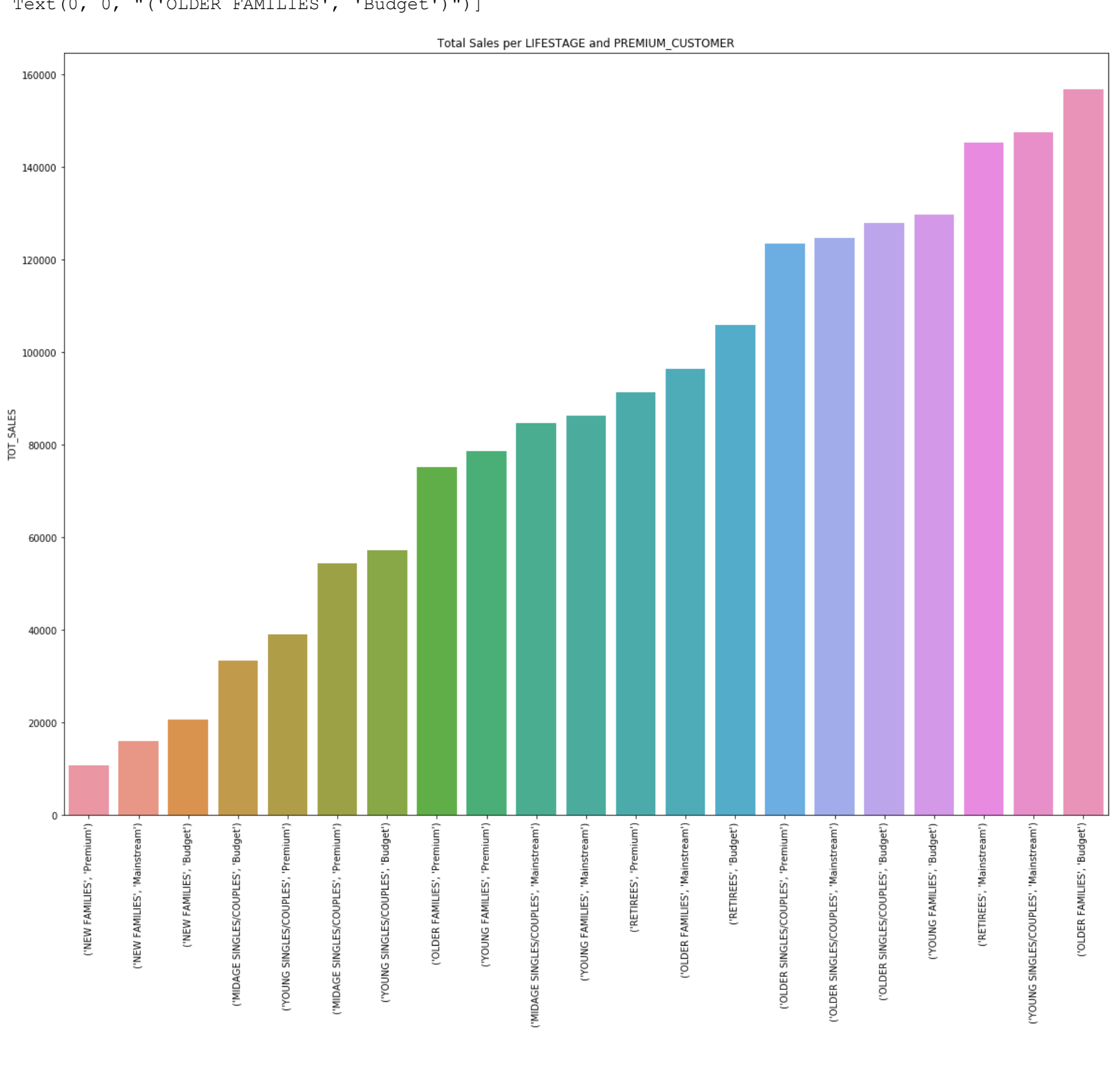
	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	pack_size	brands	LI
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	natural	SINGLES/C
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	175	ccs	SINGLES/C
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	170	smiths	SINGLES/C
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0	175	smiths	SINGLES/C
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8	150	kettle	SINGLES/C

In [68]:
```
#Total Sales per LIFESTAGE
customer_transaction_lifestage_premium = customer_transaction.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"
]).agg("sum")["TOT_SALES"].sort_values(ascending=False)
customer_transaction_lifestage_premium
```

Out[68]:
```
LIFESTAGE               PREMIUM_CUSTOMER
OLDER FAMILIES          Budget              156863.75
YOUNG SINGLES/COUPLES   Mainstream          147582.20
RETIREES                Mainstream          145168.95
YOUNG FAMILIES          Budget              129717.95
OLDER SINGLES/COUPLES   Budget              127833.60
                        Mainstream          124648.50
                        Premium             123537.55
RETIREES                Budget              105916.30
OLDER FAMILIES          Mainstream           96413.55
RETIREES                Premium              91296.65
YOUNG FAMILIES          Mainstream           86338.25
MIDAGE SINGLES/COUPLES  Mainstream           84734.25
YOUNG FAMILIES          Premium              78571.70
OLDER FAMILIES          Premium              75242.60
YOUNG SINGLES/COUPLES   Budget               57122.10
MIDAGE SINGLES/COUPLES  Premium              54443.85
YOUNG SINGLES/COUPLES   Premium              39052.30
MIDAGE SINGLES/COUPLES  Budget               33345.70
NEW FAMILIES            Budget               20607.45
                        Mainstream           15979.70
                        Premium              10760.80
Name: TOT_SALES, dtype: float64
```

```
In [67]: fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(1,1,1)
sns.barplot(x=customer_transaction_lifestage_premium.index, y=customer_transaction_lifestage_premium, a
x=ax).set_title("Total Sales per LIFESTAGE and PREMIUM_CUSTOMER")
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
Out[67]: [Text(0, 0, "('NEW FAMILIES', 'Premium')"),
 Text(0, 0, "('NEW FAMILIES', 'Mainstream')"),
 Text(0, 0, "('NEW FAMILIES', 'Budget')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('OLDER FAMILIES', 'Premium')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Premium')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Mainstream')"),
 Text(0, 0, "('RETIREES', 'Premium')"),
 Text(0, 0, "('OLDER FAMILIES', 'Mainstream')"),
 Text(0, 0, "('RETIREES', 'Budget')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Budget')"),
 Text(0, 0, "('RETIREES', 'Mainstream')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('OLDER FAMILIES', 'Budget')")]
```



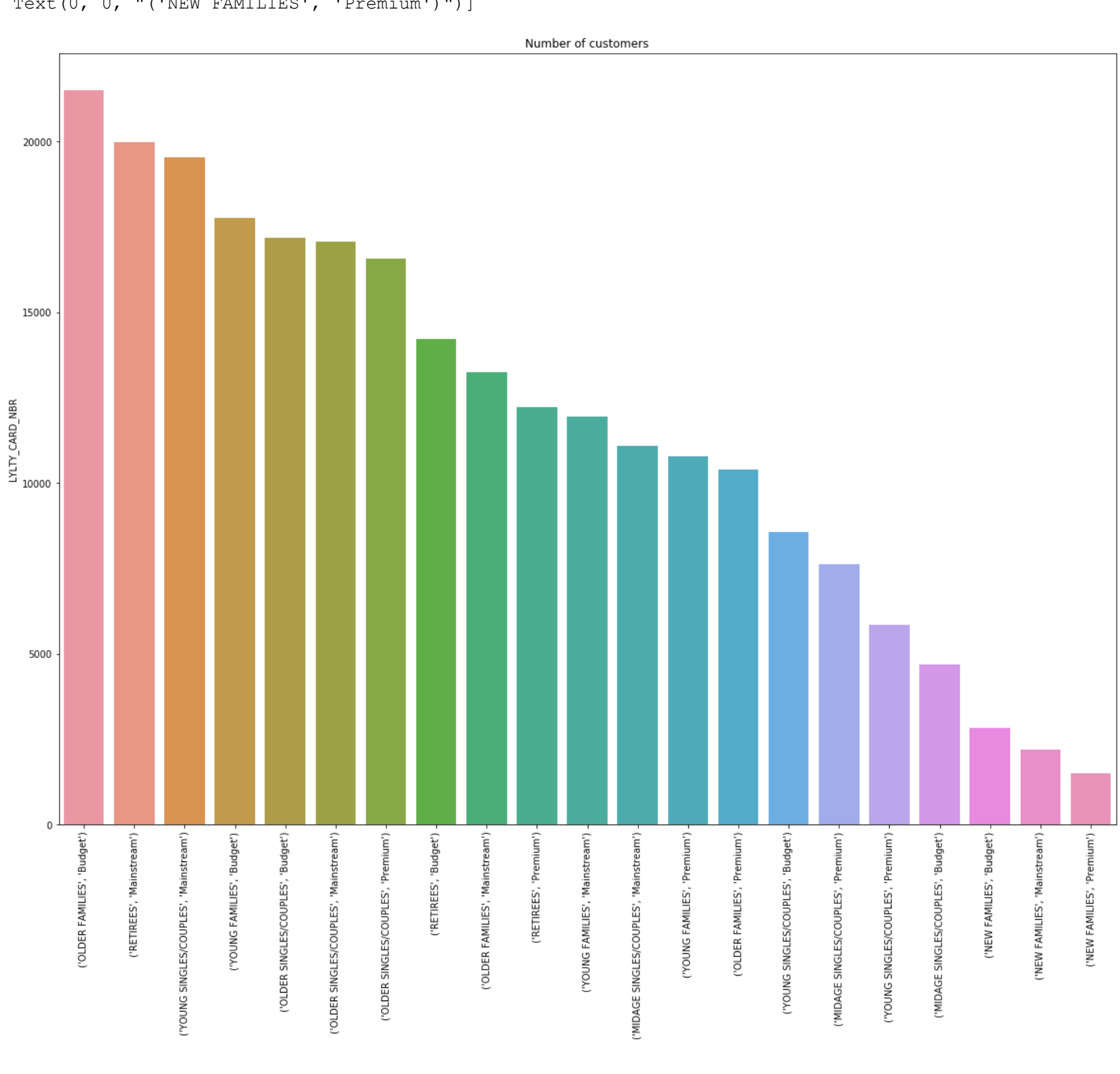
It seems like sales are mostly coming from **Budget-older families**, **Mainstream-young single/couples**, **and Mainstream-Retirees**. Let's see if the higher sales come from there being more customers who buy chips

```
In [72]: customer_count = customer_transaction.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"]).agg("count")["LYLTY_CA
RD_NBR"].sort_values(ascending=False)
customer_count
```

```
Out[72]: LIFESTAGE               PREMIUM_CUSTOMER
OLDER FAMILIES          Budget              21514
RETIREES                Mainstream          19970
YOUNG SINGLES/COUPLES   Mainstream          19544
YOUNG FAMILIES          Budget              17763
OLDER SINGLES/COUPLES   Budget              17172
                        Mainstream          17061
                        Premium             16560
RETIREES                Budget              14225
OLDER FAMILIES          Mainstream          13241
RETIREES                Premium             12236
YOUNG FAMILIES          Mainstream          11947
MIDAGE SINGLES/COUPLES  Mainstream          11095
YOUNG FAMILIES          Premium             10784
OLDER FAMILIES          Premium             10403
YOUNG SINGLES/COUPLES   Budget               8573
MIDAGE SINGLES/COUPLES  Premium              7612
YOUNG SINGLES/COUPLES   Premium              5852
MIDAGE SINGLES/COUPLES  Budget               4691
NEW FAMILIES            Budget               2824
                        Mainstream           2185
                        Premium              1488
Name: LYLTY_CARD_NBR, dtype: int64
```

```
In [74]: fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(1,1,1)
sns.barplot(x=customer_count.index, y=customer_count, ax=ax).set_title("Number of customers")
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
Out[74]: [Text(0, 0, "('OLDER FAMILIES', 'Budget')"),
 Text(0, 0, "('RETIREES', 'Mainstream')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Budget')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('RETIREES', 'Budget')"),
 Text(0, 0, "('OLDER FAMILIES', 'Mainstream')"),
 Text(0, 0, "('RETIREES', 'Premium')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Mainstream')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Premium')"),
 Text(0, 0, "('OLDER FAMILIES', 'Premium')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('NEW FAMILIES', 'Budget')"),
 Text(0, 0, "('NEW FAMILIES', 'Mainstream')"),
 Text(0, 0, "('NEW FAMILIES', 'Premium')")]
```



There seems to be more Mainstream-Retirees and Mainstream-Young/Single Couples who buy chips. This seems to be the main driver for the high sales in these segments. However, this is not the main driver for Budget-Older Families. Higher sales might also be driven by more units of chips being bought per customer.

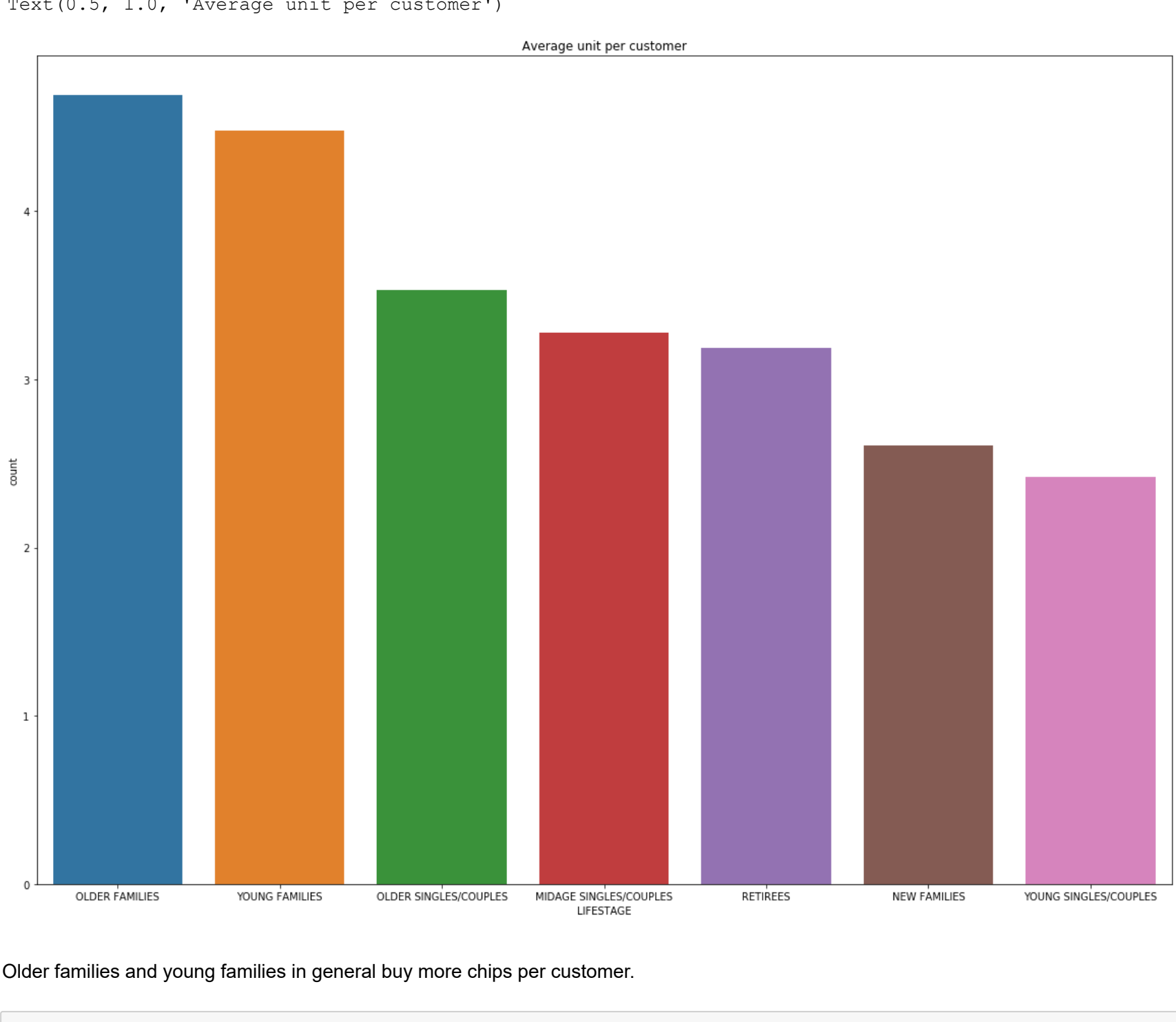
```
In [99]: customer_count = customer_transaction.groupby(["LIFESTAGE", "LYLTY_CARD_NBR"]).agg(["count"])["TOT_SALE
S"]
average_count = customer_mean.groupby("LIFESTAGE").agg("mean").sort_values(by="count", ascending=False)
average_count
```

Out[99]:

	count
LIFESTAGE	
OLDER FAMILIES	4.689304
YOUNG FAMILIES	4.481408
OLDER SINGLES/COUPLES	3.529988
MIDAGE SINGLES/COUPLES	3.276572
RETIREES	3.190038
NEW FAMILIES	2.607143
YOUNG SINGLES/COUPLES	2.418755

```
In [103]: fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(1,1,1)
sns.barplot(x=average_count.index, y=average_count["count"], ax=ax).set_title("Average unit per custom
er")
```

```
Out[103]: Text(0.5, 1.0, 'Average unit per customer')
```



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

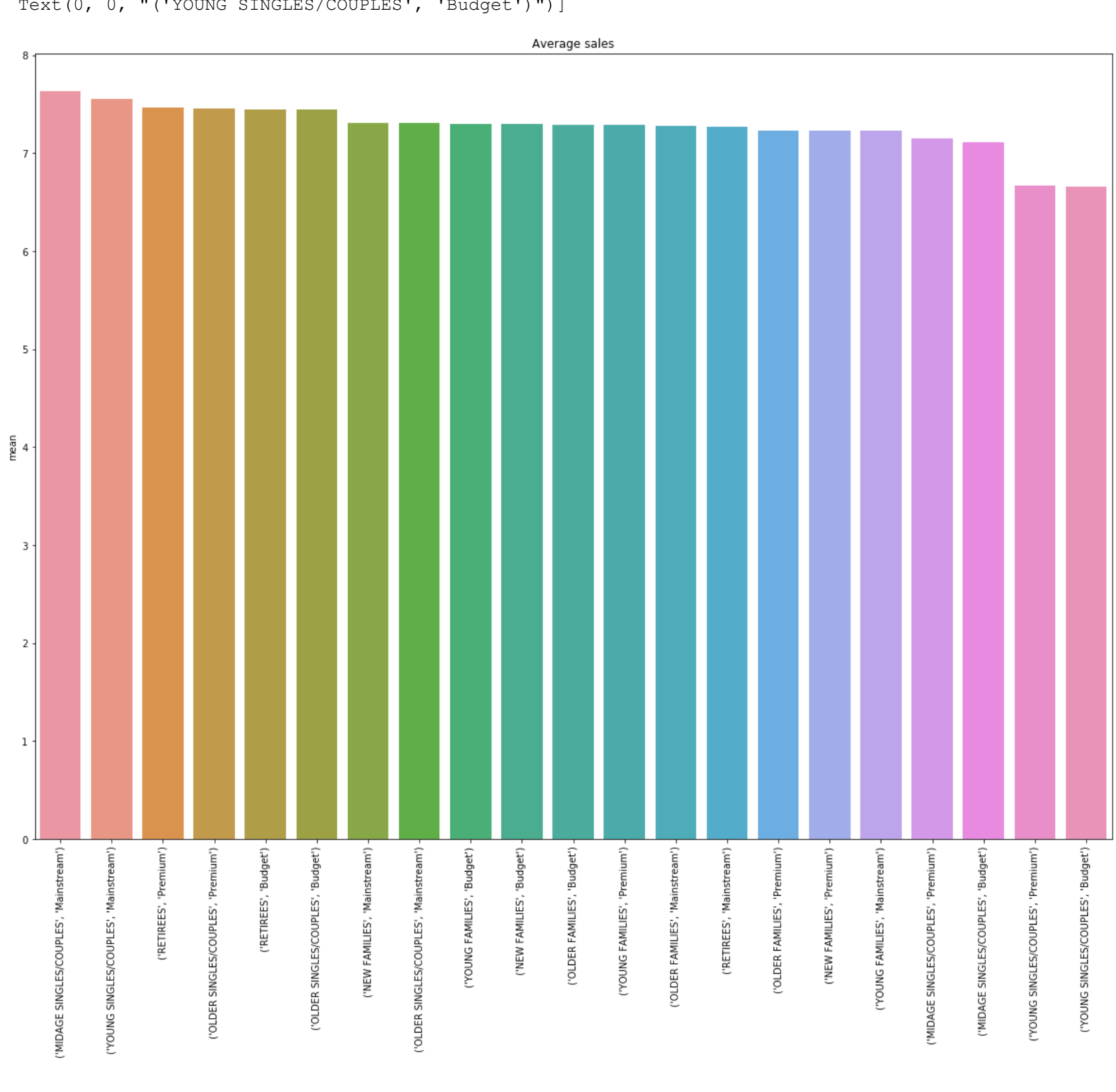
```
In [114]: average_sales = customer_transaction.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"]).agg(["mean"])["TOT_SAL
ES"].sort_values(by="mean", ascending=False)
average_sales
```

Out[114]:

		mean
LIFESTAGE	**PREMIUM_CUSTOMER**	
MIDAGE SINGLES/COUPLES	Mainstream	7.637156
YOUNG SINGLES/COUPLES	Mainstream	7.551279
RETIREES	Premium	7.461315
OLDER SINGLES/COUPLES	Premium	7.459997
RETIREES	Budget	7.445786
OLDER SINGLES/COUPLES	Budget	7.444305
NEW FAMILIES	Mainstream	7.313364
OLDER SINGLES/COUPLES	Mainstream	7.306049
YOUNG FAMILIES	Budget	7.302705
NEW FAMILIES	Budget	7.297256
OLDER FAMILIES	Budget	7.291241
YOUNG FAMILIES	Premium	7.285951
OLDER FAMILIES	Mainstream	7.281440
RETIREES	Mainstream	7.269352
OLDER FAMILIES	Premium	7.232779
NEW FAMILIES	Premium	7.231720
YOUNG FAMILIES	Mainstream	7.226772
MIDAGE SINGLES/COUPLES	Premium	7.152371
	Budget	7.108442
YOUNG SINGLES/COUPLES	Premium	6.673325
	Budget	6.663023

```
In [311]: fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(1,1,1)
sns.barplot(x=average_sales.index, y=average_sales["mean"], ax=ax).set_title("Average sales")
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
Out[311]: [Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('RETIREES', 'Premium')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('RETIREES', 'Budget')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('NEW FAMILIES', 'Mainstream')"),
 Text(0, 0, "('OLDER SINGLES/COUPLES', 'Mainstream')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Budget')"),
 Text(0, 0, "('NEW FAMILIES', 'Budget')"),
 Text(0, 0, "('OLDER FAMILIES', 'Budget')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Premium')"),
 Text(0, 0, "('OLDER FAMILIES', 'Mainstream')"),
 Text(0, 0, "('RETIREES', 'Mainstream')"),
 Text(0, 0, "('OLDER FAMILIES', 'Premium')"),
 Text(0, 0, "('NEW FAMILIES', 'Premium')"),
 Text(0, 0, "('YOUNG FAMILIES', 'Mainstream')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('MIDAGE SINGLES/COUPLES', 'Budget')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Premium')"),
 Text(0, 0, "('YOUNG SINGLES/COUPLES', 'Budget')")]
```



Mainstream Midage and Young Singles/Couples are more willing to pay more per packet of chips compared to their budget or premium counterparts. This is likely due to premium customers buying healthy snacks and when they buy chips, it is usually for entertainment purposes rather than personal consumption. This is supported by lesser premium midage and young/single couples buying chips compared to their mainstream counterparts. Since the difference in average sales isnt large, we can check if this difference is statistically significant. We will use a T-test between mainstream vs premium and budget midage and young single/couples.

```
In [142]: customer_lifestage = customer_transaction[((customer_transaction["LIFESTAGE"] == 'MIDAGE SINGLES/COUPL
ES') | (customer_transaction["LIFESTAGE"] == 'YOUNG SINGLES/COUPLES'))]
customer_mainstream = customer_lifestage[customer_lifestage["PREMIUM_CUSTOMER"] == 'Mainstream']['TOT_
SALES']
customer_budget_premium = customer_lifestage[(customer_lifestage["PREMIUM_CUSTOMER"] == 'Budget') | (c
ustomer_lifestage["PREMIUM_CUSTOMER"] == 'Premium') ]['TOT_SALES']
```

```
In [143]: ttest_ind(customer_mainstream, customer_budget_premium)
```

```
Out[143]: Ttest_indResult(statistic=33.200521751400586, pvalue=1.9916804791117584e-239)
```

From the t-test results, we can see that the p-value is 1.9916804791117584e-239. The unit price for mainstream, young and mid-age singles and couples **are not** significantly higher than that of budget or premium, young and midage couples.

Deep dive into specific customer segments for insights

From prior analysis, we have gathered quite a few interesting insights. We can dive deeper into these insights to target customer segments that contribute to most sales so that we can retain them or further increase sales. We will target the Mainstream- Young Singles/Couples and Midage group for this analysis. Let's find out if these group of people tend to buy a paticular brand of chips.

```
In [223]: customer_lifestage_mainstream = customer_lifestage[customer_lifestage["PREMIUM_CUSTOMER"] == 'Mainstre
am']
```

For this analysis we will use an apriori algorithm. Since we are only focusing on brands, we will only include columns of brands in a new dataset.

```
In [266]: customer_brand = customer_lifestage_mainstream[["LYLTY_CARD_NBR","brands"]]
customer_brand = customer_brand.set_index("LYLTY_CARD_NBR")
customer_brand
```

Out[266]:

	brands
LYLTY_CARD_NBR	
3159	doritos
3294	doritos
4187	kettle
4264	doritos
7036	kettle
...	...
272377	cobs
272389	kettle
272389	pringles
272389	pringles
272391	kettle

30639 rows × 1 columns

```
In [272]: crosstab_customer = pd.crosstab(customer_brand.index, customer_brand["brands"]).rename_axis(None, axis
=1)
crosstab_customer = crosstab_customer.clip(0,1)
crosstab_customer
```

Out[272]:

	burger	ccs	cheetos	cheezels	cobs	dorito	doritos	french	grain	grnwves	...	smith	smiths	snbts	sunbites	thins	tostit
row_0																	
1002	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1005	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	
1010	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	
1018	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1020	0	0	0	0	0	0	0	0	0	1	...	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2330041	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	
2330321	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2370181	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2370651	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	
2373711	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	

11215 rows × 27 columns

```
In [290]: frequent_itemsets = apriori(crosstab_customer, min_support=0.03, use_colnames=True)
frequent_itemsets.sort_values(by="support", ascending=False)
```

Out[290]:

	support	itemsets
8	0.414802	(kettle)
10	0.269906	(pringles)
4	0.244226	(doritos)
12	0.216585	(smiths)
13	0.147214	(thins)
6	0.121534	(infuzions)
15	0.114757	(twisties)
2	0.114668	(cobs)
14	0.113598	(tostitos)
27	0.109050	(pringles, kettle)
11	0.108872	(rrd)
21	0.096300	(kettle, doritos)
29	0.090682	(kettle, smiths)
16	0.078912	(tyrrells)
5	0.075435	(grain)
22	0.065270	(pringles, doritos)
30	0.060901	(thins, kettle)
17	0.057601	(ww)
34	0.055729	(pringles, smiths)
23	0.054124	(doritos, smiths)
25	0.049666	(infuzions, kettle)
1	0.048506	(cheezels)
18	0.046723	(kettle, cobs)
28	0.045742	(rrd, kettle)
31	0.045118	(tostitos, kettle)
9	0.044316	(natural)
32	0.044226	(twisties, kettle)
35	0.042532	(pringles, thins)
38	0.041908	(rrd, smiths)
24	0.039322	(thins, doritos)
7	0.039055	(infzns)
26	0.038966	(infuzions, pringles)
3	0.038520	(dorito)
33	0.033259	(tyrrells, kettle)
36	0.032100	(pringles, tostitos)
0	0.031922	(ccs)
20	0.031922	(infuzions, doritos)
37	0.031297	(pringles, twisties)
39	0.031297	(thins, smiths)
19	0.030584	(pringles, cobs)
40	0.030406	(pringles, kettle, doritos)

From the apriori analysis above, we can see that people who are in the Mainstream-Midage or Young Singles/Couples tend to buy chips from the Kettle brand. Following that, this group of people also like the Pringles brand. Let's see if there are certain brands that are more likely to be bought together.

```
In [297]: association_rules(frequent_itemsets, metric="confidence", min_threshold=0.03).sort_values(by="confiden
ce", ascending=False)
```

Out[297]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
45	(pringles, doritos)	(kettle)	0.065270	0.414802	0.030406	0.465847	1.123060	0.003332	1.095563
30	(tyrrells)	(kettle)	0.078912	0.414802	0.033259	0.421469	1.016074	0.000526	1.011525
20	(rrd)	(kettle)	0.108872	0.414802	0.045742	0.420147	1.012888	0.000582	1.009219
23	(smiths)	(kettle)	0.216585	0.414802	0.090682	0.418691	1.009376	0.000842	1.006690
24	(thins)	(kettle)	0.147214	0.414802	0.060901	0.413689	0.997317	-0.000164	0.998102
14	(infuzions)	(kettle)	0.121534	0.414802	0.049666	0.408657	0.985188	-0.000747	0.989610
1	(cobs)	(kettle)	0.114668	0.414802	0.046723	0.407465	0.982313	-0.000841	0.987618
18	(pringles)	(kettle)	0.269906	0.414802	0.109050	0.404030	0.974033	-0.002907	0.981927
26	(tostitos)	(kettle)	0.113598	0.414802	0.045118	0.397174	0.957504	-0.002002	0.970759
7	(doritos)	(kettle)	0.244226	0.414802	0.096300	0.394304	0.950586	-0.005006	0.966159
28	(twisties)	(kettle)	0.114757	0.414802	0.044226	0.385392	0.929101	-0.003375	0.952150
40	(rrd)	(smiths)	0.108872	0.216585	0.041908	0.384930	1.777272	0.018328	1.273701
16	(infuzions)	(pringles)	0.121534	0.269906	0.038966	0.320616	1.187880	0.006163	1.074641
46	(kettle, doritos)	(pringles)	0.096300	0.269906	0.030406	0.315741	1.169816	0.004414	1.066984
35	(thins)	(pringles)	0.147214	0.269906	0.042532	0.288916	1.070430	0.002798	1.026733
37	(tostitos)	(pringles)	0.113598	0.269906	0.032100	0.282575	1.046936	0.001439	1.017658
44	(pringles, kettle)	(doritos)	0.109050	0.244226	0.030406	0.278823	1.141656	0.003773	1.047972
39	(twisties)	(pringles)	0.114757	0.269906	0.031297	0.272727	1.010451	0.000324	1.003879
9	(doritos)	(pringles)	0.244226	0.269906	0.065270	0.267251	0.990161	-0.000649	0.996376
12	(thins)	(doritos)	0.147214	0.244226	0.039322	0.267111	1.093701	0.003369	1.031225
3	(cobs)	(pringles)	0.114668	0.269906	0.030584	0.266719	0.988189	-0.000366	0.995653
19	(kettle)	(pringles)	0.414802	0.269906	0.109050	0.262898	0.974033	-0.002907	0.990492
4	(infuzions)	(doritos)	0.121534	0.244226	0.031922	0.262656	1.075460	0.002240	1.024994
33	(smiths)	(pringles)	0.216585	0.269906	0.055729	0.257308	0.953321	-0.002729	0.983036
11	(smiths)	(doritos)	0.216585	0.244226	0.054124	0.249897	1.023219	0.001228	1.007560
8	(pringles)	(doritos)	0.269906	0.244226	0.065270	0.241824	0.990161	-0.000649	0.996831
6	(kettle)	(doritos)	0.414802	0.244226	0.096300	0.232158	0.950586	-0.005006	0.984283
10	(doritos)	(smiths)	0.244226	0.216585	0.054124	0.221614	1.023219	0.001228	1.006461
22	(kettle)	(smiths)	0.414802	0.216585	0.090682	0.218616	1.009376	0.000842	1.002599
42	(thins)	(smiths)	0.147214	0.216585	0.031297	0.212598	0.981594	-0.000587	0.994937
32	(pringles)	(smiths)	0.269906	0.216585	0.055729	0.206475	0.953321	-0.002729	0.987260
41	(smiths)	(rrd)	0.216585	0.108872	0.041908	0.193495	1.777272	0.018328	1.104926
13	(doritos)	(thins)	0.244226	0.147214	0.039322	0.161008	1.093701	0.003369	1.016441
34	(pringles)	(thins)	0.269906	0.147214	0.042532	0.157582	1.070430	0.002798	1.012308
25	(kettle)	(thins)	0.414802	0.147214	0.060901	0.146819	0.997317	-0.000164	0.999537
43	(smiths)	(thins)	0.216585	0.147214	0.031297	0.144504	0.981594	-0.000587	0.996833
17	(pringles)	(infuzions)	0.269906	0.121534	0.038966	0.144367	1.187880	0.006163	1.026686
5	(doritos)	(infuzions)	0.244226	0.121534	0.031922	0.130705	1.075460	0.002240	1.010550
49	(doritos)	(pringles, kettle)	0.244226	0.109050	0.030406	0.124498	1.141656	0.003773	1.017644
15	(kettle)	(infuzions)	0.414802	0.121534	0.049666	0.119733	0.985188	-0.000747	0.997955
36	(pringles)	(tostitos)	0.269906	0.113598	0.032100	0.118930	1.046936	0.001439	1.006051
38	(pringles)	(twisties)	0.269906	0.114757	0.031297	0.115956	1.010451	0.000324	1.001357
2	(pringles)	(cobs)	0.269906	0.114668	0.030584	0.113314	0.988189	-0.000366	0.998473
47	(pringles)	(kettle, doritos)	0.269906	0.096300	0.030406	0.112653	1.169816	0.004414	1.018429
0	(kettle)	(cobs)	0.414802	0.114668	0.046723	0.112640	0.982313	-0.000841	0.997714
21	(kettle)	(rrd)	0.414802	0.108872	0.045742	0.110275	1.012888	0.000582	1.001577
27	(kettle)	(tostitos)	0.414802	0.113598	0.045118	0.108770	0.957504	-0.002002	0.994583
29	(kettle)	(twisties)	0.414802	0.114757	0.044226	0.106621	0.929101	-0.003375	0.990893
31	(kettle)	(tyrrells)	0.414802	0.078912	0.033259	0.080181	1.016074	0.000526	1.001379
48	(kettle)	(pringles, doritos)	0.414802	0.065270	0.030406	0.073302	1.123060	0.003332	1.008667

From the association rules analysis, there is a high confidence rate that when a customer buys pringles and doritos, they are likely to buy kettle as well. Next, let's analyze customer preference in regards to pack sizes.

```
In [302]: customer_pack_size = customer_lifestage_mainstream[["LYLTY_CARD_NBR","pack_size"]]
customer_pack_size = customer_pack_size.set_index("LYLTY_CARD_NBR")
customer_pack_size
```

Out[302]:

	pack_size
LYLTY_CARD_NBR	
3159	170
3294	170
4187	175
4264	170
7036	150
...	...
272377	110
272389	150
272389	134
272389	134
272391	135

30639 rows × 1 columns

```
In [303]: crosstab_pack_size = pd.crosstab(customer_pack_size.index, customer_pack_size["pack_size"]).rename_axi
s(None, axis=1)
crosstab_pack_size = crosstab_pack_size.clip(0,1)
crosstab_pack_size
```

Out[303]:

	70	90	110	125	134	135	150	160	165	170	175	180	190	200	210	220	250	270	330	380
row_0																				
1002	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1005	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1010	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1018	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
1020	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2330041	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
2330321	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2370181	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
2370651	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2373711	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

11215 rows × 20 columns

```
In [304]: frequent_itemsets_pack_size = apriori(crosstab_pack_size, min_support=0.03, use_colnames=True)
frequent_itemsets_pack_size.sort_values(by="support", ascending=False)
```

Out[304]:

	support	itemsets
6	0.491485	(175)
3	0.337227	(150)
1	0.269906	(134)
0	0.242265	(110)
5	0.192421	(170)
25	0.166117	(150, 175)
10	0.151939	(330)
4	0.139902	(165)
21	0.130896	(134, 175)
16	0.116184	(110, 175)
29	0.099153	(170, 175)
18	0.088185	(150, 134)
9	0.081676	(270)
11	0.080606	(380)
13	0.079804	(150, 110)
7	0.075435	(210)
28	0.073562	(165, 175)
32	0.072938	(330, 175)
12	0.068926	(134, 110)
24	0.067588	(170, 150)
23	0.054213	(165, 150)
26	0.053678	(330, 150)
37	0.052608	(150, 134, 175)
20	0.051181	(170, 134)
15	0.048506	(170, 110)
35	0.046099	(150, 110, 175)
40	0.041373	(170, 150, 175)
19	0.040660	(165, 134)
22	0.040481	(330, 134)
34	0.039857	(134, 110, 175)
33	0.039768	(380, 175)
2	0.039768	(135)
31	0.039233	(270, 175)
17	0.037896	(330, 110)
8	0.037807	(250)
27	0.034953	(170, 165)
30	0.034686	(210, 175)
14	0.034062	(165, 110)
39	0.034062	(165, 150, 175)
36	0.030673	(170, 110, 175)
38	0.030495	(170, 134, 175)

From the apriori analysis, it is clear that the target customer group prefers to buy chips that are in the 175g pack size. Following this, the target customer group also prefers the 150g pack size. Next, we will check to see if there are any combination of chips that the customers prefer.

```
In [305]: association_rules(frequent_itemsets_pack_size, metric="confidence", min_threshold=0.03).sort_values(by
="confidence", ascending=False)
```

Out[305]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
56	(170, 110)	(175)	0.048506	0.491485	0.030673	0.632353	1.286618	0.006833	1.383162
74	(165, 150)	(175)	0.054213	0.491485	0.034062	0.628289	1.278350	0.007417	1.368041
80	(170, 150)	(175)	0.067588	0.491485	0.041373	0.612137	1.245486	0.008155	1.311070
62	(134, 150)	(175)	0.088185	0.491485	0.052608	0.596562	1.213796	0.009266	1.260455
68	(170, 134)	(175)	0.051181	0.491485	0.030495	0.595819	1.212284	0.005340	1.258137
...	...	...	...	...	...	...	...	...	...
39	(175)	(270)	0.491485	0.081676	0.039233	0.079826	0.977344	-0.000909	0.997989
37	(175)	(210)	0.491485	0.075435	0.034686	0.070573	0.935555	-0.002389	0.994769
79	(175)	(165, 150)	0.491485	0.054213	0.034062	0.069303	1.278350	0.007417	1.016214
61	(175)	(170, 110)	0.491485	0.048506	0.030673	0.062409	1.286618	0.006833	1.014828
73	(175)	(170, 134)	0.491485	0.051181	0.030495	0.062046	1.212284	0.005340	1.011584

86 rows × 9 columns

From this we can see that when a customer buys a bag of chips that there is a 63% chance that a customer who buys a chip in pack size 170,110 they will proceed to buy a chip with a pack size of 175. Given the lift value of 1.29, it shows that this association has a high probability of occuring.

Conclusion

To recap, sales has been mainly due to Budget-Old Families, Mainstream-Young/Singles couple, and Mainstream-Retirees shoppers. We found that the high spend in chips for mainstream young singles/couples and retirees is due to there being more of them than other buyers. Mainstream, midage and young singles and couples are also more likely to pay more per packet of chips. This is due to the impulse buying behavior of these group of people.

From the insights that we have gathered, the company should focus on targeting the Mainstream-Young and Midage Singles/Couples. We saw that the majority of the sales between 2018 and 2019 were made by these group of people. Furthermore, through the apriori analysis, Chips should focus on selling chips with the Kettle brand. If Chips were to create a sale or promo, they should pair their Kettle chips with Pringles and Doritos. This is due to its high confidence and lift value. In regards to pack sizes, Chips should sell Kettle brands that are in the 175g category.