jvyadjo1f

December 12, 2023

1 QUANTIUM TASK 1 SOLUTION

[19]:
```
# Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
from scipy.stats import ttest_ind
import warnings
warnings.filterwarnings("ignore")
```

2 Examining Transaction Data

[23]:
```
### reading the table from file
Data_transaction = pd.read_excel("QVI_transaction_data.xlsx")
Data_transaction
```

[23]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	43390	1	1000	1	5	
1	43599	1	1307	348	66	
2	43605	1	1343	383	61	
3	43329	2	2373	974	69	
4	43330	2	2426	1038	108	
…	…	…	…	…	…	
264831	43533	272	272319	270088	89	
264832	43325	272	272358	270154	74	
264833	43410	272	272379	270187	51	
264834	43461	272	272379	270188	42	
264835	43365	272	272380	270189	74	

	PROD_NAME	PROD_QTY	TOT_SALES
0	Natural Chip Compny SeaSalt175g	2	6.0
1	CCs Nacho Cheese 175g	3	6.3
2	Smiths Crinkle Cut Chips Chicken 170g	2	2.9

```
3         Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
…                                          …       …        …
264831   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264832             Tostitos Splash Of  Lime 175g         1        4.4
264833                  Doritos Mexicana    170g         2        8.8
264834   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264835             Tostitos Splash Of  Lime 175g         2        8.8

[264836 rows x 8 columns]
```

3 To Check Missing Data

[24]:
```
Data_transaction.isnull().sum()
```

[24]:
```
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
```

> No Missing Data in the datasets

[26]:
```
# Look for duplicated TXN_ID
Data_transaction[Data_transaction.duplicated(['TXN_ID'])].head()
```

[26]:
```
      DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
42   43605         55           55073   48887       113
377  43475          7            7364    7739        20
419  43391         12           12301   10982        93
476  43351         16           16427   14546        81
511  43315         19           19272   16683        31

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

[27]:
```
# Select the first duplicated TXN_ID
Data_transaction.loc[Data_transaction['TXN_ID'] == 48887, :]
```

[27]:

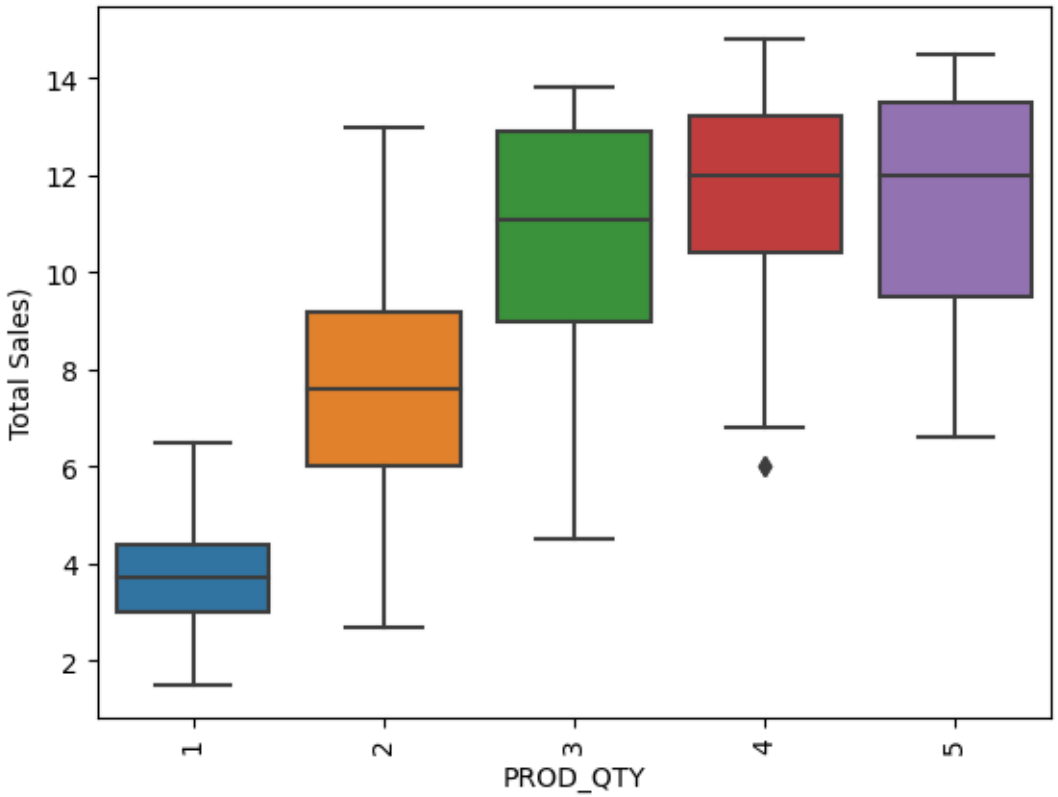
	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
41	43605	55	55073	48887	4	
42	43605	55	55073	48887	113	

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

4 To Check Outliers and Treat them

[28]:
```
def remove_outlier_IQR(df):
    Q1=df.quantile(0.25)
    Q3=df.quantile(0.75)
    IQR=Q3-Q1
    df_final=df[~((df<(Q1-1.5*IQR)) | (df>(Q3+1.5*IQR)))]
    return df_final
df_outlier_removed=remove_outlier_IQR(Data_transaction.TOT_SALES)
df_outlier_removed=pd.DataFrame(df_outlier_removed)
ind_diff=Data_transaction.index.difference(df_outlier_removed.index)

for i in range(0, len(ind_diff),1):
    df_final=Data_transaction.drop([ind_diff[i]])
    Data_transaction =df_final

sns.boxplot(y='TOT_SALES', x='PROD_QTY',data=Data_transaction)
plt.xticks(rotation=90)
plt.ylabel('Total Sales)')
```

[28]:
```
Text(0, 0.5, 'Total Sales)')
```



[29]:
```
print("Shape of dataset after treating outliers:",Data_transaction.shape)
```

```
Shape of dataset after treating outliers: (264258, 8)
```

Comparing the two boxplots, it can be seen that the outliers were removed

[30]:
```
# Convert Date column into DATE format
origin = pd.Timestamp("30/12/1899")
Data_transaction["DATE"] = Data_transaction["DATE"].apply(lambda x: origin + pd.
↪Timedelta(days=x))
Data_transaction
```

[30]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	2018-10-17	1	1000	1	5	
1	2019-05-14	1	1307	348	66	
2	2019-05-20	1	1343	383	61	
4	2018-08-18	2	2426	1038	108	
5	2019-05-19	4	4074	2982	57	
…	…	…	…	…	…	
264831	2019-03-09	272	272319	270088	89	
264832	2018-08-13	272	272358	270154	74	
264833	2018-11-06	272	272379	270187	51	

```
264834 2018-12-27        272          272379  270188        42
264835 2018-09-22        272          272380  270189        74

                                      PROD_NAME  PROD_QTY  TOT_SALES
0         Natural Chip        Compny SeaSalt175g         2        6.0
1                      CCs Nacho Cheese    175g         3        6.3
2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
5       Old El Paso Salsa   Dip Tomato Mild 300g         1        5.1
…                                          …       …         …
264831   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264832             Tostitos Splash Of  Lime 175g         1        4.4
264833                  Doritos Mexicana    170g         2        8.8
264834   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264835             Tostitos Splash Of  Lime 175g         2        8.8

[264258 rows x 8 columns]
```

[31]:
```
Data_transaction['PROD_NAME'].value_counts()
```

[31]:
```
PROD_NAME
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3285
Kettle Mozzarella   Basil & Pesto 175g      3280
Tyrrells Crisps     Ched & Chives 165g      3264
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3260
Cobs Popd Sea Salt  Chips 110g              3259
                                            …
Woolworths Medium   Salsa 300g              1430
RRD Pc Sea Salt     165g                    1429
French Fries Potato Chips 175g              1418
NCC Sour Cream &    Garden Chives 175g      1416
WW Crinkle Cut      Original 175g           1410
Name: count, Length: 114, dtype: int64
```

[32]:
```
# The customer only wants insights on chips category.
Data_transaction = Data_transaction[Data_transaction["PROD_NAME"].str.
 ↪contains("Salsa")==False]
Data_transaction
```

[32]:
```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0      2018-10-17          1            1000       1         5
1      2019-05-14          1            1307     348        66
2      2019-05-20          1            1343     383        61
4      2018-08-18          2            2426    1038       108
6      2019-05-16          4            4149    3333        16
…           …        …              …      …       …
264831 2019-03-09        272          272319  270088        89
```

```
264832 2018-08-13        272          272358  270154        74
264833 2018-11-06        272          272379  270187        51
264834 2018-12-27        272          272379  270188        42
264835 2018-09-22        272          272380  270189        74

                                      PROD_NAME  PROD_QTY  TOT_SALES
0         Natural Chip        Compny SeaSalt175g         2        6.0
1                     CCs Nacho Cheese    175g         3        6.3
2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
6       Smiths Crinkle Chips Salt & Vinegar 330g         1        5.7
…                                          …       …         …
264831   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264832             Tostitos Splash Of  Lime 175g         1        4.4
264833                  Doritos Mexicana    170g         2        8.8
264834   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264835             Tostitos Splash Of  Lime 175g         2        8.8

[246204 rows x 8 columns]
```

[33]:
```
Data_transaction['PROD_NAME'].value_counts()
```

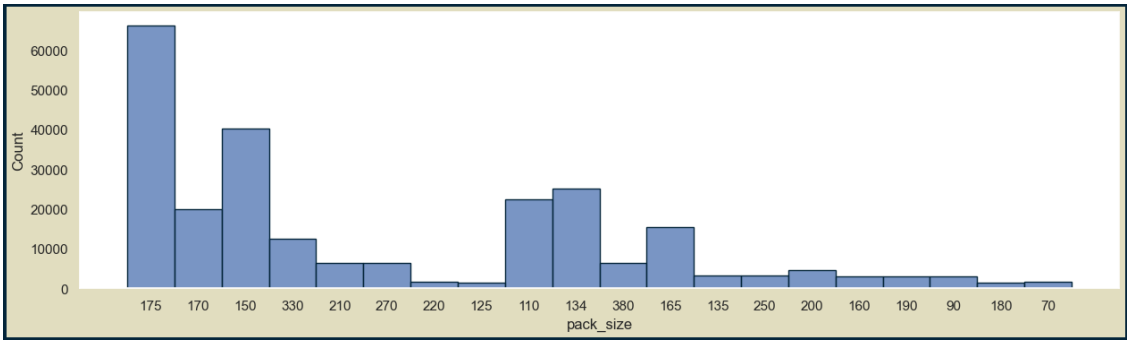
[33]:
```
PROD_NAME
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3285
Kettle Mozzarella   Basil & Pesto 175g      3280
Tyrrells Crisps     Ched & Chives 165g      3264
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3260
Cobs Popd Sea Salt  Chips 110g              3259
                                            …
Sunbites Whlegrn    Crisps Frch/Onin 90g    1432
RRD Pc Sea Salt     165g                    1429
French Fries Potato Chips 175g              1418
NCC Sour Cream &    Garden Chives 175g      1416
WW Crinkle Cut      Original 175g           1410
Name: count, Length: 105, dtype: int64
```

[34]:
```
# Extracting pack size from the Product
import re
def find_number(text):
    num = re.findall(r'[0-9]+',text)
    return " ".join(num)
Data_transaction['pack_size']=Data_transaction['PROD_NAME'].apply(lambda x:␣
 ↪find_number(x))
Data_transaction
```

[34]:
```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0      2018-10-17          1            1000       1         5
```

```
1      2019-05-14          1            1307     348        66
2      2019-05-20          1            1343     383        61
4      2018-08-18          2            2426    1038       108
6      2019-05-16          4            4149    3333        16
…             …        …              …      …       …
264831 2019-03-09        272          272319  270088        89
264832 2018-08-13        272          272358  270154        74
264833 2018-11-06        272          272379  270187        51
264834 2018-12-27        272          272379  270188        42
264835 2018-09-22        272          272380  270189        74

                                       PROD_NAME  PROD_QTY  TOT_SALES  \
0         Natural Chip        Compny SeaSalt175g         2        6.0
1                       CCs Nacho Cheese    175g         3        6.3
2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
6       Smiths Crinkle Chips Salt & Vinegar 330g         1        5.7
…                                          …       …         …
264831   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264832             Tostitos Splash Of  Lime 175g         1        4.4
264833                  Doritos Mexicana    170g         2        8.8
264834   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264835             Tostitos Splash Of  Lime 175g         2        8.8

       pack_size
0            175
1            175
2            170
4            150
6            330
…            …
264831       175
264832       175
264833       170
264834       150
264835       175

[246204 rows x 9 columns]
```

[35]:
```
# Histogram showing the number of transactions by pack size

fig = plt.figure(figsize = (15,4), linewidth=5, edgecolor="#04253a", facecolor␣
↪= '#e1ddbf')
sns.set(rc = {'figure.figsize':(15,4)})
ax = sns.histplot(data=Data_transaction, x="pack_size",  edgecolor="#04253a")
ax.set_facecolor("#ffffff")
```



175g is the highest selling pack size for the chips followed by 150g.

[37]:
```
# Column for brand names
Data_transaction['Brand Name'] = Data_transaction['PROD_NAME'].str.split(' ').
 ↪str[0]
```

[38]:
```
# Duplication or similar brands
Data_transaction['Brand Name'].value_counts()
```

[38]:
```
Brand Name
Kettle        41141
Smiths        27340
Pringles      25052
Doritos       21975
Thins         14049
RRD           11880
Infuzions     11035
WW            10320
Cobs           9669
Tostitos       9443
Twisties       9420
Tyrrells       6428
Grain          6265
Natural        6037
Cheezels       4583
CCs            4551
Red            4427
Dorito         3175
Infzns         3138
Smith          2963
Cheetos        2926
Snbts          1576
Burger         1564
Woolworths     1516
GrnWves        1465
```

```
Sunbites       1432
French         1418
NCC            1416
Name: count, dtype: int64
```

Some Brands are similar like RRD and Red Rock Deli , WW and Woolworths,NCC and Natural Chip Company etc.Let us combine them together as they are a single unit.

[39]:
```
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('Red','RRD')
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('Woolworths','WW')
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('INFUZIONS','INFZNS')
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('SMITHS','SMITH')
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('SUNBITES','SNBTS')
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('DORITOS','DORITO')
Data_transaction['Brand Name'] = Data_transaction['Brand Name'].str.
 ↪replace('GRNWVES','GRAIN')
```

[40]:
```
Data_transaction['Brand Name'].value_counts()
```

[40]:
```
Brand Name
Kettle       41141
Smiths       27340
Pringles     25052
Doritos      21975
RRD          16307
Thins        14049
WW           11836
Infuzions    11035
Cobs          9669
Tostitos      9443
Twisties      9420
Tyrrells      6428
Grain         6265
Natural       6037
Cheezels      4583
CCs           4551
Dorito        3175
Infzns        3138
Smith         2963
Cheetos       2926
Snbts         1576
```

```
Burger        1564
GrnWves       1465
Sunbites      1432
French        1418
NCC           1416
Name: count, dtype: int64
```

Combined the similar brands.

5 Examining customer data

[42]:
```
Data_customer = pd.read_csv("QVI_purchase_behaviour.csv")
Data_customer
```

[42]:
```
       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
…                 …                       …                …
72632         2370651  MIDAGE SINGLES/COUPLES       Mainstream
72633         2370701          YOUNG FAMILIES       Mainstream
72634         2370751          YOUNG FAMILIES          Premium
72635         2370961          OLDER FAMILIES           Budget
72636         2373711   YOUNG SINGLES/COUPLES       Mainstream

[72637 rows x 3 columns]
```

6 To check for null Values in the data

[43]:
```
Data_customer.isnull().sum()
```

[43]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

7 Categorise Numeric and Categorical Data

[44]:
```
Data_customer_numerics_only = Data_customer.select_dtypes(include=np.number)
Data_customer_cat = set(Data_customer.columns) -␣
 ↪set(Data_customer_numerics_only)
```

[45]:
```
print("Numeric Columns:\n",list(Data_customer_numerics_only))
print("Categorical Columns:\n",Data_customer_cat)
```

```
Numeric Columns:
 ['LYLTY_CARD_NBR']
Categorical Columns:
 {'LIFESTAGE', 'PREMIUM_CUSTOMER'}
```

[46]:
```
# Merging two dataframes
df4 = pd.merge(Data_transaction,Data_customer)
df4
```

[46]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	2018-10-17	1	1000	1	5	
1	2019-05-14	1	1307	348	66	
2	2018-11-10	1	1307	346	96	
3	2019-03-09	1	1307	347	54	
4	2019-05-20	1	1343	383	61	
…	…	…	…	…	…	
246199	2019-03-09	272	272319	270088	89	
246200	2018-08-13	272	272358	270154	74	
246201	2018-11-06	272	272379	270187	51	
246202	2018-12-27	272	272379	270188	42	
246203	2018-09-22	272	272380	270189	74	

	PROD_NAME	PROD_QTY	TOT_SALES	\
0	Natural Chip Compny SeaSalt175g	2	6.0	
1	CCs Nacho Cheese 175g	3	6.3	
2	WW Original Stacked Chips 160g	2	3.8	
3	CCs Original 175g	1	2.1	
4	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	
…	…	…	…	
246199	Kettle Sweet Chilli And Sour Cream 175g	2	10.8	
246200	Tostitos Splash Of Lime 175g	1	4.4	
246201	Doritos Mexicana 170g	2	8.8	
246202	Doritos Corn Chip Mexican Jalapeno 150g	2	7.8	
246203	Tostitos Splash Of Lime 175g	2	8.8	

	pack_size	Brand Name	LIFESTAGE	PREMIUM_CUSTOMER
0	175	Natural	YOUNG SINGLES/COUPLES	Premium
1	175	CCs	MIDAGE SINGLES/COUPLES	Budget
2	160	WW	MIDAGE SINGLES/COUPLES	Budget
3	175	CCs	MIDAGE SINGLES/COUPLES	Budget
4	170	Smiths	MIDAGE SINGLES/COUPLES	Budget
…	…	…	…	…
246199	175	Kettle	YOUNG SINGLES/COUPLES	Premium
246200	175	Tostitos	YOUNG SINGLES/COUPLES	Premium

```
246201       170    Doritos   YOUNG SINGLES/COUPLES          Premium
246202       150    Doritos   YOUNG SINGLES/COUPLES          Premium
246203       175   Tostitos   YOUNG SINGLES/COUPLES          Premium

[246204 rows x 12 columns]
```

[47]:
```
# Check if some customers were not matched on by checking for nulls.
df4.isnull().sum()
```

[47]:
```
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
Brand Name          0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

[48]:
```
pd.date_range(start = '2018-07-01', end = '2019-06-30').difference(df4['DATE'])
```

[48]:
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

7.1 We have a missing date on Christmas Day. This makes sense because most retail stores are closed that day

[49]:
```
# Create a new dataframe which contains the total sale for each date

df5 = pd.pivot_table(df4, values = 'TOT_SALES', index = 'DATE', aggfunc = 'sum')
df5.head()
```

[49]:
```
            TOT_SALES
DATE
2018-07-01     4920.1
2018-07-02     4877.0
2018-07-03     4954.7
2018-07-04     4968.1
2018-07-05     4682.0
```

[50]:
```
df6 = pd.DataFrame(index = pd.date_range(start = '2018-07-01', end =␣
 ↪'2019-06-30'))
df6['TOT_SALES'] = 0
len(df6)
```

[50]: 365

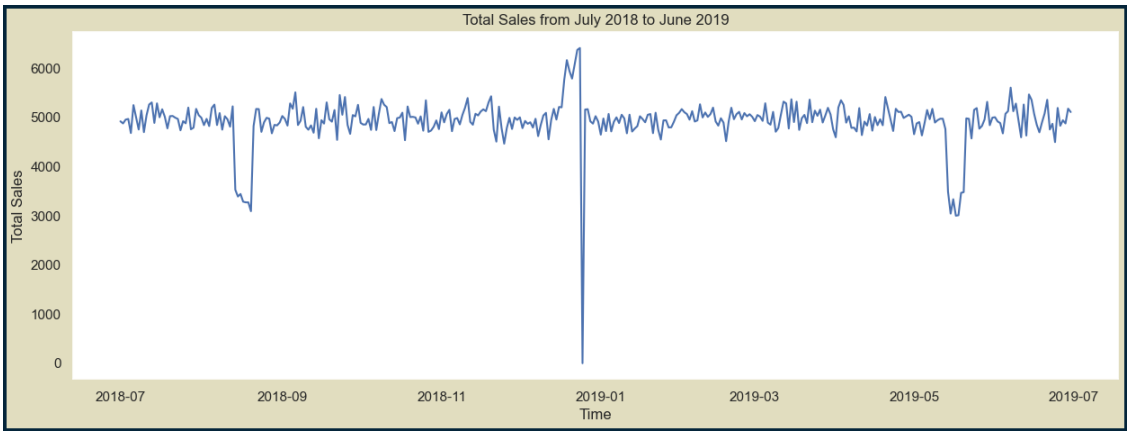
[51]:
```
z = df5 + df6
z.fillna(0, inplace = True)

z.index.name = 'Date'
z.rename(columns = {'TOT_SALES': 'Total Sales'}, inplace = True)
z.head()
```

[51]:

Date	Total Sales
2018-07-01	4920.1
2018-07-02	4877.0
2018-07-03	4954.7
2018-07-04	4968.1
2018-07-05	4682.0

[52]:
```
timeline = z.index
graph = z['Total Sales']



fig, ax = plt.subplots(figsize = (15, 5), linewidth=5, edgecolor="#04253a",␣
 ↪facecolor = '#e1ddbf')

ax.plot(timeline, graph)

ax.set_facecolor("#ffffff")

date_form = DateFormatter("%Y-%m")
ax.xaxis.set_major_formatter(date_form)
plt.title('Total Sales from July 2018 to June 2019')
plt.xlabel('Time')
plt.ylabel('Total Sales')
```

[52]: Text(0, 0.5, 'Total Sales')



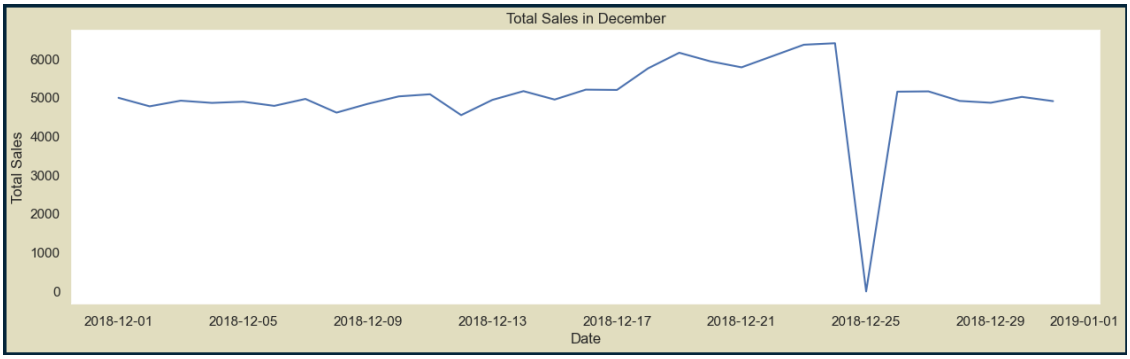
8 Sales during December and Christmas Day

[53]:
```
# December month only

z_december = z[(z.index < "2019-01-01") & (z.index > "2018-11-30")]
z_december.head()
```

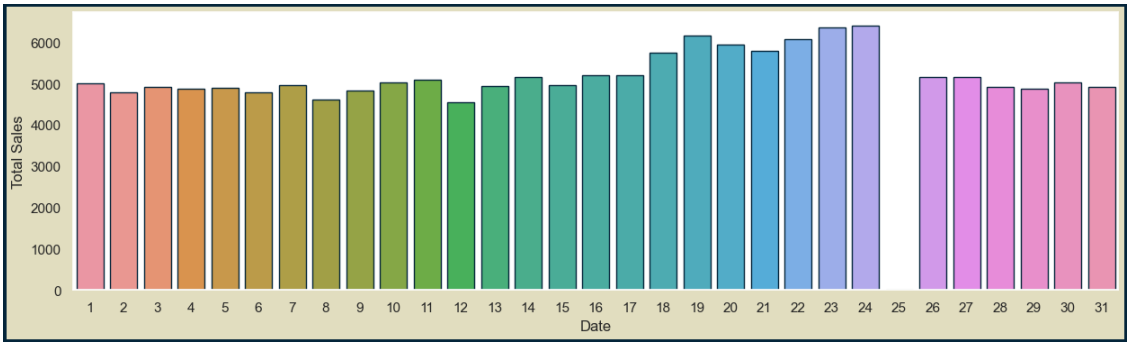
[53]:
```
            Total Sales
Date
2018-12-01       5000.9
2018-12-02       4781.1
2018-12-03       4927.0
2018-12-04       4869.4
2018-12-05       4900.5
```

[54]:
```
plt.figure(figsize = (15, 5))
fig = plt.figure(linewidth=5, edgecolor="#04253a", facecolor = '#e1ddbf')

ax = sns.lineplot(data= z_december, x= 'Date', y = 'Total Sales')
ax.set_facecolor("#ffffff")

plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.title('Total Sales in December')
```

[54]:
```
Text(0.5, 1.0, 'Total Sales in December')

<Figure size 1500x500 with 0 Axes>
```



[55]:
```
# Reset index

z_december.reset_index(drop = True, inplace = True)
z_december.head()
z_december['Date'] = z_december.index + 1
z_december.head()
fig = plt.figure(linewidth=5, edgecolor="#04253a", facecolor = '#e1ddbf')
ax = sns.barplot(x = 'Date', y ='Total Sales', data = z_december, ␣
 ↪edgecolor="#04253a")
ax.set_facecolor("#ffffff")
```



Sales have increased till the day before Christmas i.e. 2018-12-24 and there are no transaction records on 25th of December because of the Holiday and also the sales went down after Christmas.

9 Data analysis on customer segments

10 1. Who spends the most on chips i.e. describing customers by lifestage and premimum category?

[56]:
```
df4
```

[56]:
```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0      2018-10-17          1            1000       1         5
1      2019-05-14          1            1307     348        66
2      2018-11-10          1            1307     346        96
3      2019-03-09          1            1307     347        54
4      2019-05-20          1            1343     383        61
…             …        …              …       …       …
246199 2019-03-09        272          272319  270088        89
246200 2018-08-13        272          272358  270154        74
246201 2018-11-06        272          272379  270187        51
246202 2018-12-27        272          272379  270188        42
246203 2018-09-22        272          272380  270189        74

                                    PROD_NAME  PROD_QTY  TOT_SALES  \
0        Natural Chip        Compny SeaSalt175g         2        6.0
1                     CCs Nacho Cheese    175g         3        6.3
2               WW Original Stacked Chips 160g         2        3.8
3                            CCs Original 175g         1        2.1
4        Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
…                                        …       …         …
246199  Kettle Sweet Chilli And Sour Cream 175g         2       10.8
246200            Tostitos Splash Of  Lime 175g         1        4.4
246201                 Doritos Mexicana    170g         2        8.8
246202  Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
246203            Tostitos Splash Of  Lime 175g         2        8.8

       pack_size Brand Name              LIFESTAGE PREMIUM_CUSTOMER
0            175    Natural   YOUNG SINGLES/COUPLES          Premium
1            175        CCs  MIDAGE SINGLES/COUPLES           Budget
2            160         WW  MIDAGE SINGLES/COUPLES           Budget
3            175        CCs  MIDAGE SINGLES/COUPLES           Budget
4            170     Smiths  MIDAGE SINGLES/COUPLES           Budget
…          …         …                       …               …
246199       175     Kettle   YOUNG SINGLES/COUPLES          Premium
246200       175   Tostitos   YOUNG SINGLES/COUPLES          Premium
246201       170    Doritos   YOUNG SINGLES/COUPLES          Premium
246202       150    Doritos   YOUNG SINGLES/COUPLES          Premium
246203       175   Tostitos   YOUNG SINGLES/COUPLES          Premium

[246204 rows x 12 columns]
```

[57]:
```
df4['LIFESTAGE'].value_counts()
```

[57]:
```
LIFESTAGE
OLDER SINGLES/COUPLES     50677
RETIREES                  46342
OLDER FAMILIES            45042
YOUNG FAMILIES            40395
YOUNG SINGLES/COUPLES     33917
MIDAGE SINGLES/COUPLES    23342
NEW FAMILIES               6489
Name: count, dtype: int64
```

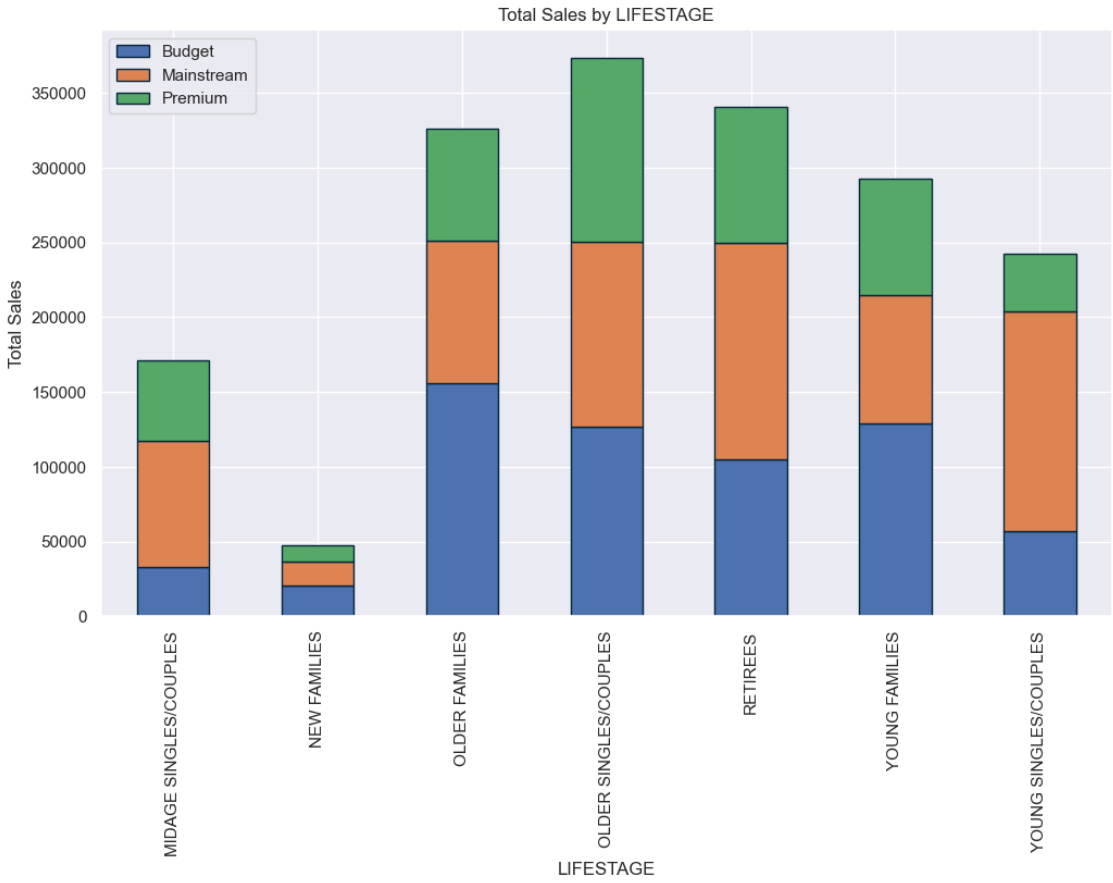
[58]:
```
df4['PREMIUM_CUSTOMER'].value_counts()
```

[58]:
```
PREMIUM_CUSTOMER
Mainstream    94839
Budget        86567
Premium       64798
Name: count, dtype: int64
```

[59]:
```
df8 = pd.DataFrame(df4.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).TOT_SALES.
 ↪sum())
fig = plt.figure(linewidth=5, edgecolor="#04253a", facecolor = '#e1ddbf')
df8.unstack().plot(kind = 'bar', stacked = True, figsize = (12, 7), title =␣
 ↪'Total Sales by LIFESTAGE' , edgecolor="#04253a")
plt.ylabel('Total Sales')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

[59]:
```
<matplotlib.legend.Legend at 0x1c0b053e6d0>
```

```
<Figure size 1500x400 with 0 Axes>
```



The sales are high for Budget - Older Families, Mainstream-young singles/couples, Mainstream - retirees and premium - Older Single/Couples.

11 2. How many customers are there in each segment?

[60]:
```
df9 = pd.DataFrame(df4.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE']).
↪LYLTY_CARD_NBR.nunique())
df9.rename(columns = {'LYLTY_CARD_NBR': 'Number of Customers'}, inplace = True)
```

[61]:
```
df9 = df9.sort_values(by = 'Number of Customers', ascending = False).head(10)
```

[62]:
```
df9
```

[62]:

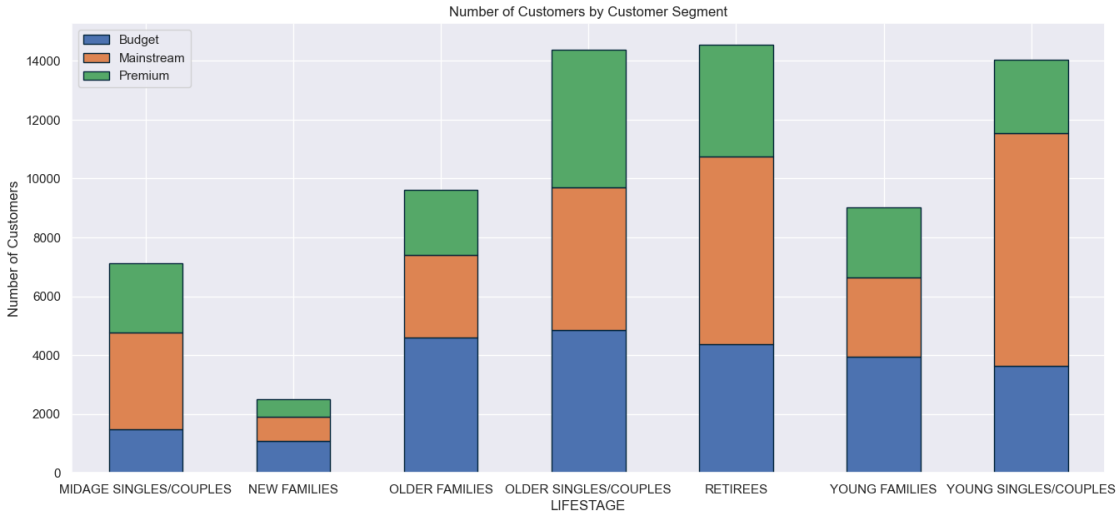
PREMIUM_CUSTOMER	LIFESTAGE	Number of Customers
Mainstream	YOUNG SINGLES/COUPLES	7905
	RETIREES	6357
	OLDER SINGLES/COUPLES	4853
Budget	OLDER SINGLES/COUPLES	4846

```
Premium     OLDER SINGLES/COUPLES     4681
Budget      OLDER FAMILIES            4606
            RETIREES                  4382
            YOUNG FAMILIES            3951
Premium     RETIREES                  3811
Budget      YOUNG SINGLES/COUPLES     3644
```

[63]:
```
df9 = pd.DataFrame(df4.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).
↪LYLTY_CARD_NBR.nunique())
```

[64]:
```
df9.unstack().plot(kind='bar', stacked = True , rot=0 , figsize = (16, 7),␣
↪title = 'Number of Customers by Customer Segment',  edgecolor="#04253a")
plt.ylabel('Number of Customers')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

[64]:
```
<matplotlib.legend.Legend at 0x1c0b06189d0>
```



There are more mainstream young singles/couples and retirees. This contributes to more chips sales in these segments however this is not major driver for the budget older families segment.

12 3. How many chips are bought per customer by segment?

[74]:
```
# Average units per customer by PREMIUM_CUSTOMER and LIFESTAGE

df10 = pd.DataFrame(df4.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).
↪LYLTY_CARD_NBR.nunique())
df10.rename(columns={'LYLTY_CARD_NBR':'Customers'},inplace=True)
df10.sort_values(by='Customers',ascending=False, inplace=True)
```

```
df10
```

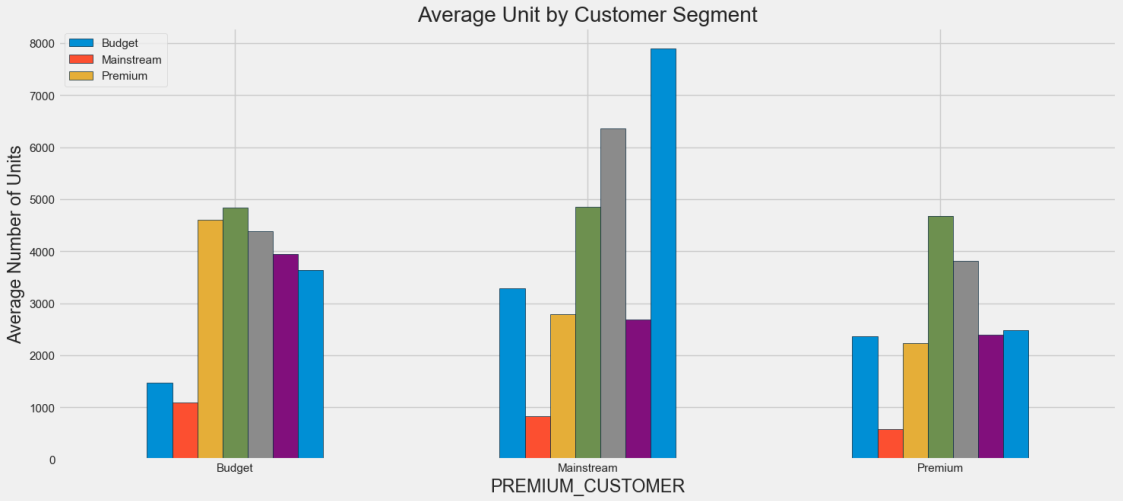
[74]:

PREMIUM_CUSTOMER	LIFESTAGE	Customers
Mainstream	YOUNG SINGLES/COUPLES	7905
	RETIREES	6357
	OLDER SINGLES/COUPLES	4853
Budget	OLDER SINGLES/COUPLES	4846
Premium	OLDER SINGLES/COUPLES	4681
Budget	OLDER FAMILIES	4606
	RETIREES	4382
	YOUNG FAMILIES	3951
Premium	RETIREES	3811
Budget	YOUNG SINGLES/COUPLES	3644
Mainstream	MIDAGE SINGLES/COUPLES	3294
	OLDER FAMILIES	2788
	YOUNG FAMILIES	2683
Premium	YOUNG SINGLES/COUPLES	2479
	YOUNG FAMILIES	2397
	MIDAGE SINGLES/COUPLES	2369
	OLDER FAMILIES	2230
Budget	MIDAGE SINGLES/COUPLES	1472
	NEW FAMILIES	1087
Mainstream	NEW FAMILIES	830
Premium	NEW FAMILIES	575

Let us visualize the cust_num dataframe by bar chart to have a better look at the different segments numbers. So far, we have used seaborn and plotly for drawing graphs. For this visualization, we will use pandas to draw a bar graph with from the cust_num dataframe which is consisting of MultiIndex.

[87]:
```
df10.unstack().plot(kind = 'bar', figsize = (16, 7), rot = 0, title = 'Average␣
↪Unit by Customer Segment',  edgecolor="#04253a")
plt.ylabel('Average Number of Units')
plt.legend(['Budget', 'Mainstream', 'Premium'], loc = 2)
```

[87]:
```
<matplotlib.legend.Legend at 0x1c0b025ccd0>
```

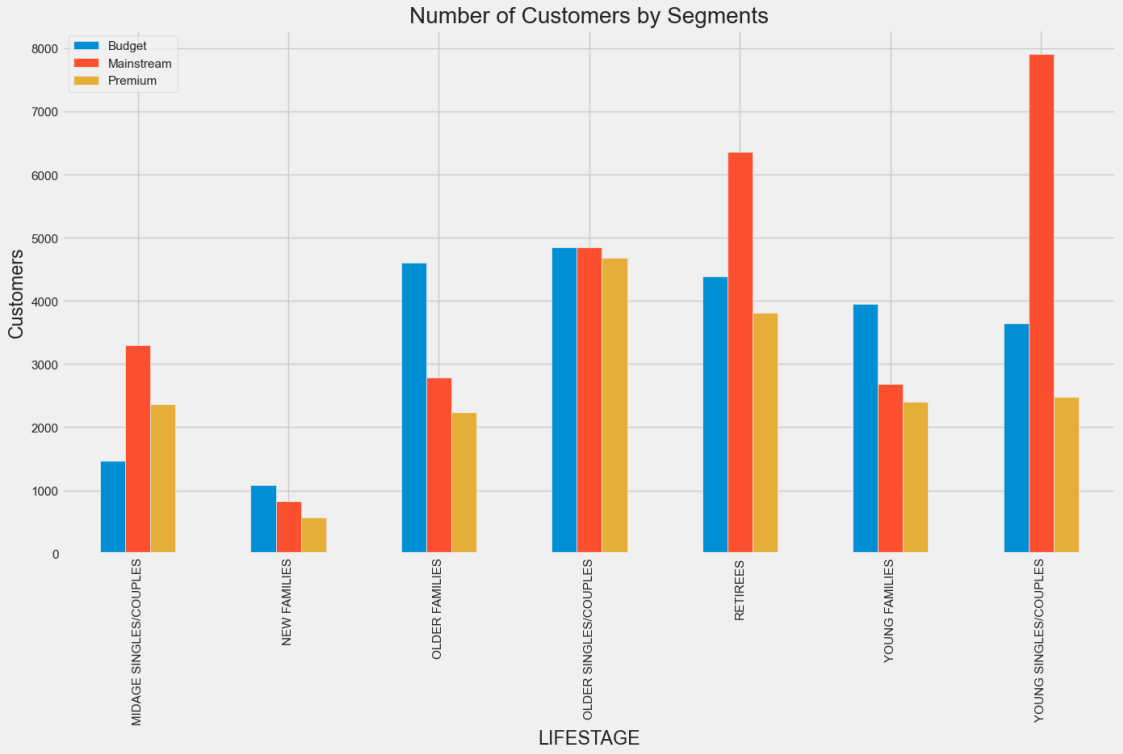


For all the three Lifestages, Older families and Young Families buy more chips per customer.

13 4. What's the average chip price by customer segment?

[86]:
```
# Average price per unit by PREMIUM_CUSTOMER and LIFESTAGE

df11 = pd.DataFrame(df4.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).
 ↪LYLTY_CARD_NBR.nunique())
matplotlib.style.use('fivethirtyeight')
df11.unstack().plot(kind='bar', figsize=(15,8), title='Number of Customers by␣
 ↪Segments')
plt.ylabel('Customers')
plt.legend(['Budget', 'Mainstream', 'Premium'])
```

[86]:
```
<matplotlib.legend.Legend at 0x1c0b0430810>
```



Mainstream midage singles/couples and young singles/couples pay more per packet of chips compared to other segments

[88]:
```
# Perform an independent t-test between mainstream vs non-mainstream midage and young singles/couples to test this difference

# Create a new dataframe pricePerUnit
pricePerUnit = df4

# Create a new column under pricePerUnit called PRICE
pricePerUnit['PRICE'] = pricePerUnit['TOT_SALES'] / pricePerUnit['PROD_QTY']

# Let's have a look
pricePerUnit.head()
```

[88]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	2018-10-17	1	1000	1	5	
1	2019-05-14	1	1307	348	66	
2	2018-11-10	1	1307	346	96	
3	2019-03-09	1	1307	347	54	
4	2019-05-20	1	1343	383	61	

	PROD_NAME	PROD_QTY	TOT_SALES	pack_size	\

```
0  Natural Chip        Compny SeaSalt175g         2        6.0       175
1                CCs Nacho Cheese    175g         3        6.3       175
2          WW Original Stacked Chips 160g         2        3.8       160
3                       CCs Original 175g         1        2.1       175
4  Smiths Crinkle Cut  Chips Chicken 170g         2        2.9       170
```

	Brand Name	LIFESTAGE	PREMIUM_CUSTOMER	PRICE
0	Natural	YOUNG SINGLES/COUPLES	Premium	3.00
1	CCs	MIDAGE SINGLES/COUPLES	Budget	2.10
2	WW	MIDAGE SINGLES/COUPLES	Budget	1.90
3	CCs	MIDAGE SINGLES/COUPLES	Budget	2.10
4	Smiths	MIDAGE SINGLES/COUPLES	Budget	1.45

14 Customer Segments based on Price per Unit

[95]:
```
df4.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).TOT_SALES.sum()
```

[95]:
```
LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget               33140.10
                        Mainstream           84144.30
                        Premium              54127.15
NEW FAMILIES            Budget               20541.25
                        Mainstream           15921.45
                        Premium              10720.20
OLDER FAMILIES          Budget              155857.25
                        Mainstream           95708.20
                        Premium              74677.30
OLDER SINGLES/COUPLES   Budget              126983.90
                        Mainstream          123893.40
                        Premium             122793.85
RETIREES                Budget              105230.10
                        Mainstream          144576.35
                        Premium              90815.00
YOUNG FAMILIES          Budget              128898.00
                        Mainstream           85719.55
                        Premium              78125.60
YOUNG SINGLES/COUPLES   Budget               56933.40
                        Mainstream          146908.90
                        Premium              38930.60
Name: TOT_SALES, dtype: float64
```

15 Let us further explore and target the segment Mainstream and young singles/couples that contributes most to the sale

16 Let's find out if they tend to buy a particular brand of chip

[96]:
```
df4
```

[96]:
```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0      2018-10-17          1            1000       1         5
1      2019-05-14          1            1307     348        66
2      2018-11-10          1            1307     346        96
3      2019-03-09          1            1307     347        54
4      2019-05-20          1            1343     383        61
…             …        …              …      …       …
246199 2019-03-09        272          272319  270088        89
246200 2018-08-13        272          272358  270154        74
246201 2018-11-06        272          272379  270187        51
246202 2018-12-27        272          272379  270188        42
246203 2018-09-22        272          272380  270189        74

                                    PROD_NAME  PROD_QTY  TOT_SALES  \
0        Natural Chip        Compny SeaSalt175g         2        6.0
1                     CCs Nacho Cheese    175g         3        6.3
2               WW Original Stacked Chips 160g         2        3.8
3                           CCs Original 175g         1        2.1
4        Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
…                                        …       …         …
246199  Kettle Sweet Chilli And Sour Cream 175g         2       10.8
246200           Tostitos Splash Of  Lime 175g         1        4.4
246201                Doritos Mexicana    170g         2        8.8
246202  Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
246203           Tostitos Splash Of  Lime 175g         2        8.8

       pack_size Brand Name              LIFESTAGE PREMIUM_CUSTOMER  PRICE
0            175    Natural   YOUNG SINGLES/COUPLES          Premium   3.00
1            175        CCs  MIDAGE SINGLES/COUPLES           Budget   2.10
2            160         WW  MIDAGE SINGLES/COUPLES           Budget   1.90
3            175        CCs  MIDAGE SINGLES/COUPLES           Budget   2.10
4            170     Smiths  MIDAGE SINGLES/COUPLES           Budget   1.45
…          …         …                       …               …    …
246199       175     Kettle   YOUNG SINGLES/COUPLES          Premium   5.40
246200       175   Tostitos   YOUNG SINGLES/COUPLES          Premium   4.40
246201       170    Doritos   YOUNG SINGLES/COUPLES          Premium   4.40
246202       150    Doritos   YOUNG SINGLES/COUPLES          Premium   3.90
246203       175   Tostitos   YOUNG SINGLES/COUPLES          Premium   4.40

[246204 rows x 13 columns]
```

[97]:
```
dfa = df4[(df4['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') &␣
 ↪(df4['PREMIUM_CUSTOMER'] == 'Mainstream')]
dfb = df4[(df4['LIFESTAGE'] != 'YOUNG SINGLES/COUPLES') &␣
 ↪(df4['PREMIUM_CUSTOMER'] != 'Mainstream')]
```

[98]:
```
dfa_quantity = dfa['PROD_QTY'].sum()
dfa_quantity
```

[98]:
```
36074
```

[99]:
```
dfb_quantity = dfb['PROD_QTY'].sum()
dfb_quantity
```

[99]:
```
262133
```

[100]:
```
dfa_quantity_brand = dfa.groupby(['Brand Name'])['PROD_QTY'].sum()
dfa_quantity_brand
```

[100]:
```
Brand Name
Burger        106
CCs           405
Cheetos       291
Cheezels      651
Cobs         1609
Dorito        569
Doritos      3867
French        143
Grain        1055
GrnWves       125
Infuzions    1797
Infzns        541
Kettle       7106
NCC           132
Natural       578
Pringles     4306
RRD          1582
Smith         239
Smiths       3240
Snbts         126
Sunbites      104
Thins        2182
Tostitos     1640
Twisties     1664
Tyrrells     1143
WW            873
Name: PROD_QTY, dtype: int64
```

[101]:
```
dfb_quantity_brand = dfb.groupby(['Brand Name'])['PROD_QTY'].sum()
dfb_quantity_brand
```

[101]:
```
Brand Name
Burger        1723
CCs           4861
Cheetos       3094
Cheezels      4926
Cobs         10085
Dorito        3379
Doritos      23079
French        1504
Grain         6525
GrnWves       1618
Infuzions    11768
Infzns        3290
Kettle       43564
NCC           1587
Natural       6522
Pringles     26502
RRD          17671
Smith         3259
Smiths       29344
Snbts         1780
Sunbites      1544
Thins        14999
Tostitos     10032
Twisties      9903
Tyrrells      6727
WW           12847
Name: PROD_QTY, dtype: int64
```

[102]:
```
# To check the brand affinity

dfa_affinity = dfa_quantity_brand/dfa_quantity
dfb_affinity = dfb_quantity_brand/dfb_quantity

affinity = (dfa_affinity/dfb_affinity).sort_values(ascending = False)

df_affinity= pd.DataFrame({'Brand Name':affinity.index, 'Affinity':affinity.
 ↪values})
df_affinity
```

[102]:

	Brand Name	Affinity
0	Tyrrells	1.234674
1	Dorito	1.223634
2	Twisties	1.220995

```
3     Doritos  1.217544
4      Infzns  1.194892
5    Tostitos  1.187911
6      Kettle  1.185291
7    Pringles  1.180654
8       Grain  1.174896
9        Cobs  1.159331
10  Infuzions  1.109616
11      Thins  1.057109
12   Cheezels  0.960316
13     Smiths  0.802330
14     French  0.690901
15    Cheetos  0.683440
16        RRD  0.650538
17    Natural  0.643983
18        CCs  0.605420
19        NCC  0.604400
20    GrnWves  0.561383
21      Smith  0.532894
22      Snbts  0.514373
23         WW  0.493787
24   Sunbites  0.489456
25     Burger  0.447042
```

Tyrell chips seems to be the most popular brand for Mainstream young singles/couples almost 23% more than the rest of the audience whereas Burger Rings were purchased less than 56% less as compared to rest of the people.

17 Find out if our target segment tends to buy larger packs of chips.

[103]:
```
dfa_quantity_pack = dfa.groupby(['pack_size'])['PROD_QTY'].sum()
dfb_quantity_pack = dfb.groupby(['pack_size'])['PROD_QTY'].sum()
```

[104]:
```
# To Check Pack Affinity
dfa_affinity_pack = dfa_quantity_pack/dfa_quantity
dfb_affinity_pack = dfb_quantity_pack/dfb_quantity

affinity_pack = (dfa_affinity_pack/dfb_affinity_pack).sort_values(ascending =␣
↪False)

df_affinity_pack = pd.DataFrame({'Pack Size':affinity_pack.index, 'Affinity':
↪affinity_pack.values})
df_affinity_pack
```

[104]:

	Pack Size	Affinity
0	270	1.276508
1	380	1.252702
2	330	1.210177
3	110	1.187112
4	134	1.180654
5	210	1.174896
6	135	1.127825
7	250	1.113282
8	170	1.008014
9	150	0.962471
10	175	0.940317
11	165	0.905054
12	190	0.616927
13	180	0.561383
14	160	0.523389
15	125	0.502890
16	90	0.502799
17	200	0.485132
18	70	0.482680
19	220	0.447042

[105]:
```
Product_name = df4[(df4['pack_size'] == '270')]['PROD_NAME']

Product_name.unique()
```

[105]:
```
array(['Twisties Cheese     270g', 'Twisties Chicken270g'], dtype=object)
```

Mainstream young singles/couples are 27% more likely to purchase a 270g pack of Twisties Cheese chips which reflects the higher volume of sales.

[106]:
```
Product_name_least = df4[(df4['pack_size'] == '220')]['PROD_NAME']

Product_name_least.unique()
```

[106]:
```
array(['Burger Rings 220g'], dtype=object)
```

Mainstream young singles/couples are 56% less likely to purchase a 220g pack of Burger Rings which reflects the lower volume of sales.

18 Views and Recommendations:

1. Older Families: Focus on the Budget segment. Strength: Frequent purchase. We can give promotions that encourages more frequency of purchase. Strength: High quantity of chips purchased per visit. We can give promotions that encourage them to buy more quantity of chips per purchase.
2. Young Singles/Couples: Focus on the Mainstream segment. This segment is the only segment

that had Doritos as their 2nd most purchased brand (after Kettle). To specifically target this segment it might be a good idea to collaborate with Doritos merchant to do some branding promotion catered to "Young Singles/Couples - Mainstream" segment. Strength: Population quantity. We can spend more effort on making sure our promotions reach them, and it reaches them frequently.

3. Retirees: Focus on the Mainstream segment. Strength: Population quantity. Again, since their population quantity is the contributor to the high total sales, we should spend more effort on making sure our promotions reaches as many of them as possible and frequent.

4. General: All segments has Kettle as the most frequently purchased brand, and 175gr (regardless of brand) followed by 150gr as the preferred chip size. When promoting chips in general to all segments it is good to take advantage of these two points.