Quantium_Data_Analytics

March 28, 2023

1 Regional Analysis of Chips Sales and Customer Behavior

1.1 Analyst: Albert Dellor

1.2 Email: dell.datascience@gmail.com

1.3 Table of Contents

Introduction

Data Wrangling

Exploratory Data Analysis

Strategy & Conclusions

2 Introduction

This project aims at analyzing the purchasing behavior of customers who buy chips, which includes identifying the frequency, quantity, and types of chips they purchase in an effort to inform and drive strategy for supermarket's chips division for the next half year.

2.1 Data description

This dataset was made available by Quantium as part of their Data Analytics Virtual Experience program. It consists of transaction dataset; QVI_transaction_data.xlsx and and behaviour dataset; QVI_purchase_behaviour.csv data sets.

QVI_purchase_behaviour.csv has 726373 by 3 rows and columns respectively.

- `LYLTY_CARD_NBR`: Loyalty card number of customers
- `LIFESTAGE`: life stage of customers
- `PREMIUM_CUSTOMER`: purchasing status of customers

QVI_transaction_data.xlsx has 264836 by 8 rows and columns respectively.

- `DATE`: Data of transaction
- `STORE_NBR`: Store within which transaction took place
- `LYLTY_CARD_NBR`: Loyaity card number of customer
- `TXN_ID`: Transaction Identification
- `PROD_NBR`: Product number
- `PROD_NAME`: Product name
- `PROD_QTY`: Quantity of product purchased

- `TOT_SALES`: Total cost of product sold

3 Data Wrangling

3.1 Import all packages and set plots to be embedded inline

[430]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import requests
from IPython.display import Image
from wordcloud import WordCloud
from collections import Counter
import matplotlib.dates as dates
%matplotlib inline
from scipy.signal import find_peaks
from scipy.stats import ttest_ind
```

3.2 Loading the dataset: Get the URL of dataset. Create request, download transaction and purchase behaviour datasets

[246]:
```
urls = ['https://cdn.theforage.com/vinternships/companyassets/32A6DqtsbF7LbKdcq/
 ↪QVI_transaction_data.xlsx',
        'https://cdn.theforage.com/vinternships/companyassets/32A6DqtsbF7LbKdcq/
 ↪QVI_purchase_behaviour.csv']

for url in urls:
    data = requests.get(url)
    with open(url.split('/')[-1], mode = 'wb') as file:
        file.write(data.content)
```

3.3 Loading Datasets

[431]:
```
transactions = pd.read_excel('QVI_transaction_data.xlsx')
purchase_behavor = pd.read_csv('QVI_purchase_behaviour.csv')
```

3.4 Data Assessment

3.4.1 Visual assessment and programmatic assessments were conducted to detect quality and tidiness issues with both dataset

- Visual assessment of data in Microsoft excell
- Programmatic assessment in python

3.5 Assessment: QVI_purchase_behaviour.csv

[432]:
```
purchase_behavor.sample(5)
```

[432]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
19539	71211	OLDER SINGLES/COUPLES	Mainstream
57776	217304	RETIREES	Budget
35574	130332	OLDER SINGLES/COUPLES	Mainstream
61733	233016	RETIREES	Mainstream
66442	248480	RETIREES	Budget

3.6 Dimensions of data set

[433]:
```
print('Rows: {}\nColumns: {}'.format(purchase_behavor.shape[0]\
                                     ,purchase_behavor.shape[1]))
```

```
Rows: 72637
Columns: 3
```

3.7 Missing values and data types

[434]:
```
purchase_behavor.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
```

3.8 Number of unique values in PREMIUM_CUSTOMER and LIFESTAGE

[435]:
```
purchase_behavor.PREMIUM_CUSTOMER.nunique(),\
purchase_behavor.LIFESTAGE.nunique()
```

[435]: (3, 7)

[436]:
```
purchase_behavor.PREMIUM_CUSTOMER.unique().tolist()
```

[436]: ['Premium', 'Mainstream', 'Budget']

[437]:
```
purchase_behavor.LIFESTAGE.unique().tolist()
```

[437]:
```
['YOUNG SINGLES/COUPLES',
 'YOUNG FAMILIES',
 'OLDER SINGLES/COUPLES',
 'MIDAGE SINGLES/COUPLES',
 'NEW FAMILIES',
 'OLDER FAMILIES',
 'RETIREES']
```

3.8.1 NB:

The data consists of **72637 rows** by **3 columns**. There are **no missing vaules**. However, **PREMIUM_CUSTOMER** and **LIFESTAGE** columns have few number of unique values, thus can be converted from string object to categorical date type as they hold no ordinal value.

3.9 Assessment: QVI_transaction_data.xlsx

[438]:
```
transactions.sample(5)
```

[438]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
212404	43342	170	170239	171981	74	
95451	43505	213	213178	213054	55	
233353	43445	134	134120	137847	114	
215232	43410	231	231343	235064	102	
61846	43581	114	114114	117579	35	

	PROD_NAME	PROD_QTY	TOT_SALES
212404	Tostitos Splash Of Lime 175g	2	8.8
95451	Snbts Whlgrn Crisps Cheddr&Mstrd 90g	2	3.4
233353	Kettle Sensations Siracha Lime 150g	2	9.2
215232	Kettle Mozzarella Basil & Pesto 175g	2	10.8
61846	Woolworths Mild Salsa 300g	2	3.0

3.10 Dimensions of data set

[439]:
```
print('Rows: {}\nColumns: {}'.format(transactions.shape[0],\
                                     transactions.shape[1]))
```

```
Rows: 264836
Columns: 8
```

3.11 Missing values and data types

[440]:
```
transactions.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
```

3.12 Descriptive statistics

[441]:
```
transactions.describe()[['PROD_QTY','TOT_SALES']]
```

[441]:

	PROD_QTY	TOT_SALES
count	264836.000000	264836.000000
mean	1.907309	7.304200
std	0.643654	3.083226
min	1.000000	1.500000
25%	2.000000	5.400000
50%	2.000000	7.400000
75%	2.000000	9.200000
max	200.000000	650.000000

3.13 check for duplicates.

The only unique identifier in dataset is TXN_ID (transaction id). Thus is used to find duplicates

[442]:
```
transactions[transactions.duplicated()]
```

[442]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
124845	43374	107	107024	108462	45	Smiths Thinly Cut	Roast Chicken 175g	2	6.0

3.14 Further investigate the transaction ID

[443]:
```
transactions.query("TXN_ID ==␣
↪108462")[['TXN_ID','PROD_NAME','PROD_QTY','TOT_SALES']]
```

[443]:

	TXN_ID	PROD_NAME	PROD_QTY	TOT_SALES
124843	108462	Smiths Thinly Cut	Roast Chicken 175g	2	6.0

```
124844  108462          Cheetos Chs & Bacon Balls 190g         2        6.6
124845  108462  Smiths Thinly Cut   Roast Chicken 175g         2        6.0
```

3.14.1 NB

There are **no missing data**. **PROD_QTY and TOT_SALES have 75% of entire data being under 2 and 9.2 respectively**, however the maximun values are 200 and 650 respectively, there might be outliers present. Further investigation reveal that there was double recording of a transaction involving **Smiths Thinly Cut Roast Chicken 175g**.

3.15 Summary of assessment

Transaction data (unclean)

3.15.1 Tidiness issues

1. Product name contains product mass variable as well, thus must be separated into respective columns as each variable must form exclusive column to comply with data tidiness standards.
2. Presence of duplicate transaction involving Smiths Thinly Cut Roast Chicken 175g. All entries must be unique. Thus duplicates are dropped.
3. Absence of unit cost column.

3.15.2 Quality issues

4. PROD_QTY and TOT_SALES may have outliers.
5. product name has inconsistent spacing thus must be formated to comply with data quality standards.
6. Product name may contain irrelevant products other than chips.
7. date format is in Excel serial number instead of pandas datetime object.
8. convert `PREMIUM_CUSTOMER` and `LIFESTAGE` columns to categorical data type.

Purchase behavior data(clean)

3.16 Data Cleaning

In this sectionall the data issues outlined in assesment stage are cleaned.

[444]:
```
# make copies of both data sets
purchase_behavor_copy = purchase_behavor.copy()
transactions_copy = transactions.copy()
```

3.16.1 Issue #1: separate product mass and product name into deparate columns

[445]:
```
regex = r'(\d+)(g|G)'
transactions_copy['PACK_SIZE'] = pd.to_numeric(
                                    transactions_copy.PROD_NAME\
                                    .str.extract(regex)[0])
```

3.16.2 Issue #2: Remove duplicates in data set

[446]:
```
transactions_copy.drop_duplicates(inplace = True)
```

3.16.3 Issue #3: Add a unit cost column

[447]:
```
transactions_copy['UNIT_COST'] = np.divide(transactions_copy.
↪TOT_SALES,transactions_copy.PROD_QTY)
```

3.16.4 Issue #7: Format date from Excel serial number to pandas datetime object

[448]:
```
transactions_copy.DATE = pd.to_datetime(transactions_copy.DATE, unit='d',\
                         origin=pd.Timestamp('1899-12-30'))
```

3.16.5 Issue #4: PROD_QTY and TOT_SALES may have outliers

[449]:
```
transactions_copy.nlargest(6,'TOT_SALES')[['PROD_NAME','PROD_QTY','TOT_SALES']]
```

[449]:

	PROD_NAME	PROD_QTY	TOT_SALES
69762	Dorito Corn Chp Supreme 380g	200	650.0
69763	Dorito Corn Chp Supreme 380g	200	650.0
5179	Smiths Crnkle Chip Orgnl Big Bag 380g	5	29.5
55558	Smiths Crnkle Chip Orgnl Big Bag 380g	5	29.5
69496	Smiths Crnkle Chip Orgnl Big Bag 380g	5	29.5
117850	Smiths Crnkle Chip Orgnl Big Bag 380g	5	29.5

looking at the 6 largest TOT_SALES, its evident that 650 are outliers. To further comfirm this, it is general accepted that any data point that falls outside the range of Q1 - 1.5 x IQR to Q3 + 1.5 x IQR is considered a potential outlier.

[450]:
```
stats = transactions_copy.describe()
```

[451]:
```
def outlier(column):
    Q1 = stats.loc['25%'][column]
    Q3 = stats.loc['75%'][column]
    IQR = Q3 - Q1
    low_range = Q1 - (1.5 * IQR)
    upper_range = Q3 + (1.5 * IQR)
```

```
    print("For {} an outlier is any value outside the range of {:.2f} and {:.
↪2f}".\
    format(column,low_range,upper_range))
```

[452]:
```
outlier('PROD_QTY')
outlier('TOT_SALES')
```

```
For PROD_QTY an outlier is any value outside the range of 2.00 and 2.00
For TOT_SALES an outlier is any value outside the range of -0.30 and 14.90
```

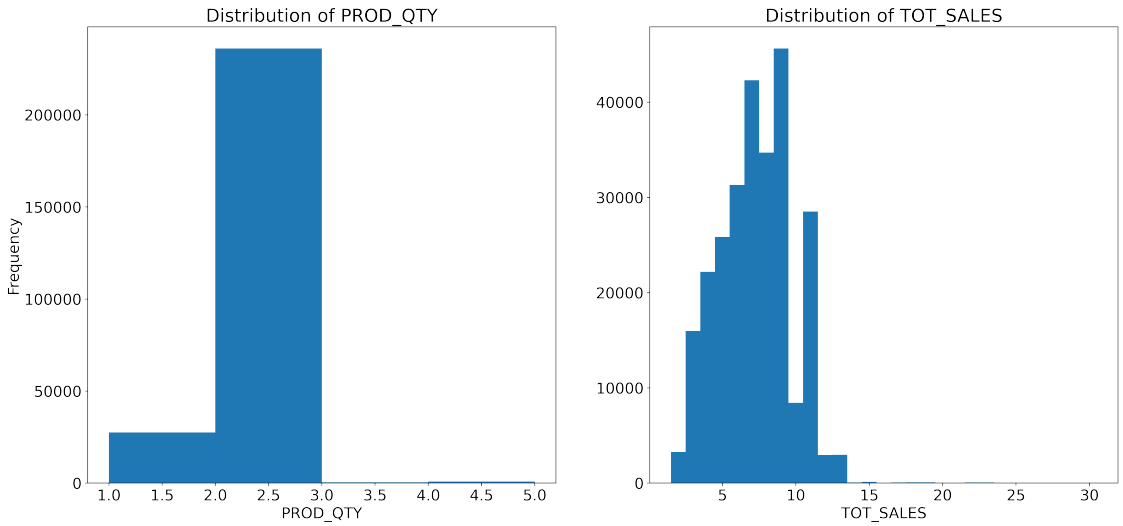
3.16.6 Looking at the descriptive statistics of both datasets

[453]:
```
pd.merge(pd.DataFrame(transactions_copy.PROD_QTY.describe()),\
         pd.DataFrame(transactions_copy.TOT_SALES.describe()),\
         on=pd.DataFrame(transactions_copy.TOT_SALES.describe()).index).
↪rename(columns={'key_0':'index'})
```

[453]:

	index	PROD_QTY	TOT_SALES
0	count	264835.000000	264835.000000
1	mean	1.907308	7.304205
2	std	0.643655	3.083231
3	min	1.000000	1.500000
4	25%	2.000000	5.400000
5	50%	2.000000	7.400000
6	75%	2.000000	9.200000
7	max	200.000000	650.000000

[454]:
```
plt.figure(figsize=(22,10),dpi=400)
plt.rcParams['font.size'] = 18
plt.subplot(1,2,1)
bin = np.arange(1,5+1,1);
plt.hist(transactions_copy.PROD_QTY,bins=bin);
plt.xlabel('PROD_QTY');
plt.ylabel('Frequency');
plt.title('Distribution of PROD_QTY');

plt.subplot(1,2,2)
bin = np.arange(1.5, 30+1.5,1);
plt.hist(transactions_copy.TOT_SALES, bins=bin);
plt.xlabel('TOT_SALES');
plt.title('Distribution of TOT_SALES');
```



It is evident that a customer with loyalty card number 226000, purchased large orders on two occasions in 2018 and 2019, thus creating the outliers for PROD_QTY and corresponding TOT_SALES .

Additionally, the histograms clearly show that a majority of the data is located between 1 to 3 for PROD_QTY and 1 to 15 for TOT_SALES.

Therefore is is safe to say that 200 and 650 for PROD_QTY and TOT_SALES are outliers respectively and are removed from the datase

3.16.7 Outlier is droped from data set

[455]:
```
index = transactions_copy.query("TOT_SALES == 650").index.values
transactions_copy.drop(index,axis=0, inplace=True)
```

3.16.8 Issue 5: format spacing in product name

[456]:
```
transactions_copy['PROD_NAME'] = transactions_copy.PROD_NAME\
                                    .str.replace(regex, '',regex=True)\
                                    .str.replace(r'\s\s+', ' ',regex=True)\
                                    .str.replace('&', ' & ',regex=True)
```

3.16.9 Issue 6: examine products summary,rid it of irrelevant products

[457]:
```
def wordcloud(data):
    wordcloud = WordCloud(width = 1600, height = 1600,
                background_color ='white',
                min_font_size = 10).generate_from_frequencies(data)
    plt.figure(figsize = (16,8), facecolor = None,dpi=400)
    plt.imshow(wordcloud)
```

```
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.show()
```

3.16.10 Summary of individual PROD_NAME words with wordcloud

[458]:
```
PROD_NAME = transactions_copy.PROD_NAME.str.replace("&","")
words = ' '.join(PROD_NAME.values).split()
data = Counter(words)
wordcloud(data)
```

Taking a quick glance, reveal that the column generally contains chips products. However the presence of some questionable words like **Salsa, salt, Mexican, Rock, Sea** raise doubts and may

indicate that not all products are chips. Next entire PROD_NAMEs are summarised to get better sense of products.

3.16.11 Summary of entire PROD_NAME products in wordcloud

[459]:
```
products = transactions_copy.PROD_NAME.value_counts().to_dict()
wordcloud(products)
```

It turns out that previously alarming words like **Sea, salt, Mexican, Rock etc.** are actually part of chips product name or branding. With the exception of **salsa** as it clearly doesn't qualify as chips and must be removed from column.

3.16.12 Identify all salsa entries

[460]:
```
salsa_df = transactions_copy[transactions_copy.PROD_NAME.\
                             str.contains('Salsa', case=False)]
total = salsa_df.PROD_NAME.value_counts().values.sum()
print('There are {} instances of irrelevant salsa products\
in dataset.\n'.format(total))
pd.DataFrame(salsa_df.PROD_NAME.value_counts()).reset_index().\
                            rename(columns={'index':'Product','PROD_NAME':
 ↪'Counts'})
```

```
There are 18094 instances of irrelevant salsa productsin dataset.
```

[460]:

	Product	Counts
0	Old El Paso Salsa Dip Chnky Tom Ht	3125
1	Old El Paso Salsa Dip Tomato Med	3114
2	Old El Paso Salsa Dip Tomato Mild	3085
3	Woolworths Mild Salsa	1491
4	Doritos Salsa Mild	1472
5	Smiths Crinkle Cut Tomato Salsa	1470
6	Red Rock Deli SR Salsa & Mzzrlla	1458
7	Doritos Salsa Medium	1449
8	Woolworths Medium Salsa	1430

3.16.13 Remove 18094 instances of irrelevant salsa products from dataset

[461]:
```
salsa_index = salsa_df.index.tolist()
transactions_copy.drop(index=salsa_index, inplace=True)
print('After removing {} irrelevant sala entries,initial {} total\
                    entries reduced to {}'.\
                    format(total,transactions.shape[0],transactions_copy.
 ↪shape[0]))
```

```
After removing 18094 irrelevant sala entries,initial 264836 total
entries reduced to 246739
```

3.16.14 Assess the top 20 most frequent words in descending order

[462]:
```
PROD_NAME = transactions_copy.PROD_NAME.str.replace("&","")
words = ' '.join(PROD_NAME.values).split()
word_counts = Counter(words)
sorted_words = sorted(word_counts.items(), key=lambda x: x[1], reverse=True)
for word, count in sorted_words[:20]:
    print(f'{word}: {count}')
```

```
Chips: 49770
Kettle: 41288
```

```
Salt: 27976
Cheese: 27890
Smiths: 27389
Pringles: 25102
Crinkle: 22490
Corn: 22061
Doritos: 22041
Original: 21560
Cut: 19283
Chip: 18645
Chicken: 18576
Chilli: 15390
Sea: 14145
Thins: 14075
Sour: 13882
Crisps: 12607
Vinegar: 12402
RRD: 11894
```

3.16.15 Issue #8: convert PREMIUM_CUSTOMER and LIFESTAGE columns to categorical data type

[463]:
```
purchase_behavor_copy.PREMIUM_CUSTOMER = purchase_behavor_copy.PREMIUM_CUSTOMER.
  ↪astype('category')
purchase_behavor_copy.LIFESTAGE = purchase_behavor_copy.LIFESTAGE.
  ↪astype('category')
```

[464]:
```
purchase_behavor_copy.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  category
 2   PREMIUM_CUSTOMER  72637 non-null  category
dtypes: category(2), int64(1)
memory usage: 709.9 KB
```

3.16.16 Finally merge two data sets together

[465]:
```
clean_data = pd.merge(transactions_copy, purchase_behavor_copy,␣
  ↪left_on='LYLTY_CARD_NBR',\
                      right_on='LYLTY_CARD_NBR', how='left')
```

3.16.17 Dimension of final dataset

[466]:
```
clean_data.shape
```

[466]:
```
(246739, 12)
```

3.16.18 missing values in dataset

[467]:
```
clean_data.isnull().sum()
```

[467]:
```
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
UNIT_COST           0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

3.16.19 Save clean dataset

[468]:
```
clean_data.to_csv('wrangled_data.csv',index=False)
```

4 Exploratory Data Analysis

4.1 Load final datase

[469]:
```
df = pd.read_csv('wrangled_data.csv',parse_dates=['DATE'])
```

4.2 Assess final dataset

[470]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246739 entries, 0 to 246738
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   DATE              246739 non-null  datetime64[ns]
 1   STORE_NBR         246739 non-null  int64
 2   LYLTY_CARD_NBR    246739 non-null  int64
 3   TXN_ID            246739 non-null  int64
```

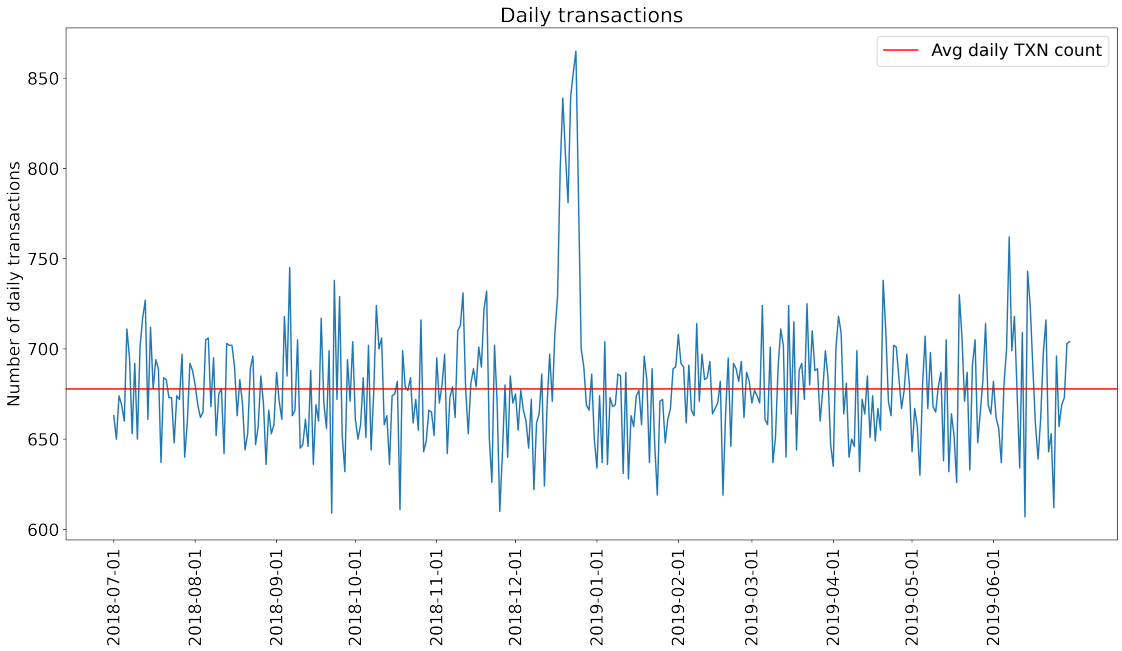
```
 4   PROD_NBR          246739 non-null  int64
 5   PROD_NAME         246739 non-null  object
 6   PROD_QTY          246739 non-null  int64
 7   TOT_SALES         246739 non-null  float64
 8   PACK_SIZE         246739 non-null  int64
 9   UNIT_COST         246739 non-null  float64
 10  LIFESTAGE         246739 non-null  object
 11  PREMIUM_CUSTOMER  246739 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(6), object(3)
memory usage: 22.6+ MB
```

4.3 What is the daily number of transaction over time?

4.3.1 Analyse daily number of transaction over time

[471]:
```
date_txn = df.groupby('DATE')['TXN_ID'].count()
date = df.groupby(pd.Grouper(key='DATE', freq='MS'))['TXN_ID'].count()
avg_daily = date_txn.values.mean()
print('The average daily number of transaction is {}'.format(avg_daily))

plt.rcParams['font.size'] = 18
plt.figure(figsize=(20,10), dpi = 400)
plt.plot(date_txn.index,date_txn.values);
plt.axhline(y= avg_daily, color = 'red', label = f'Avg daily TXN count');
plt.legend()
plt.xticks(rotation=90,ticks=date.index);
plt.ylabel('Number of daily transactions');
plt.title('Daily transactions');
```

```
The average daily number of transaction is 677.8543956043956
```



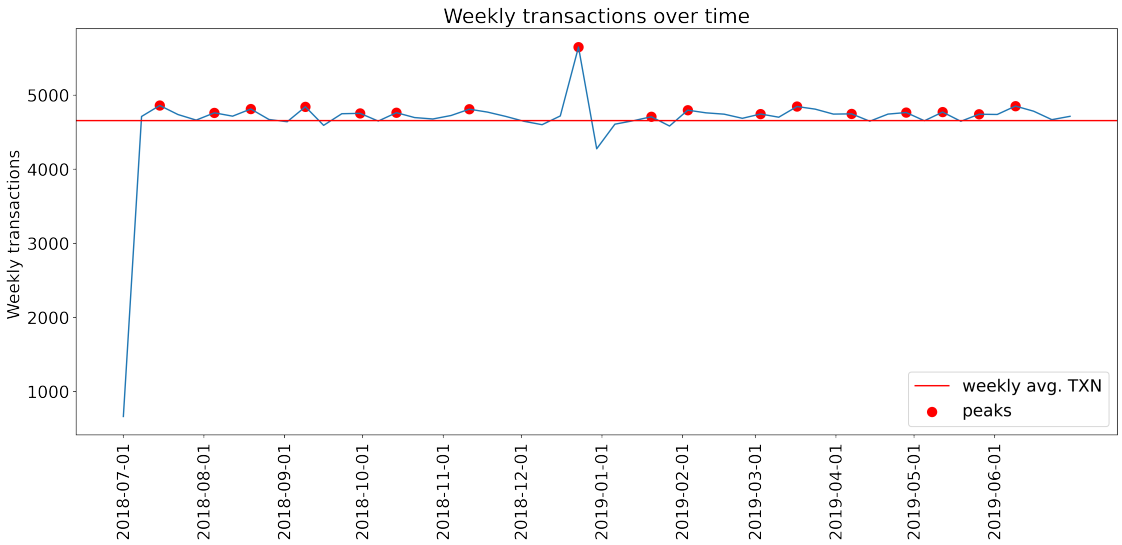
Analysing daily number of transaction over time, reveal that daily average transaction hover around $678 with little deviation throughout thr year except in early december, where daily average transctions shoots up to a little above 859, before quickly returning to normal levels.

4.4 What is the weekly number of transaction over time?

4.4.1 Analyse weekly transactions

[472]:
```
date_TXN = df.groupby(pd.Grouper(key='DATE', freq='1W'))['TXN_ID'].count()
avg = date_TXN.values.mean()
print('The mean weekly total transaction is {}'.format(avg))
date= df.groupby(pd.Grouper(key='DATE', freq='MS'))['TXN_ID'].count()
xticks = date.index

plt.figure(figsize=(20,8), dpi = 400)
plt.plot(date_TXN.index,date_TXN.values);
plt.xticks(rotation = 90);
plt.ylabel('Weekly transactions ');
plt.title('Weekly transactions over time');
plt.axhline(y= avg, color = 'red', label = 'weekly avg. TXN');
peaks, _ = find_peaks(date_TXN.values)
plt.scatter(date_TXN.index[peaks], date_TXN.values[peaks], color='red', s=100,␣
 ↪label= 'peaks')
plt.xticks(xticks);
plt.legend(loc= 4);
```

```
The mean weekly total transaction is 4655.452830188679
```

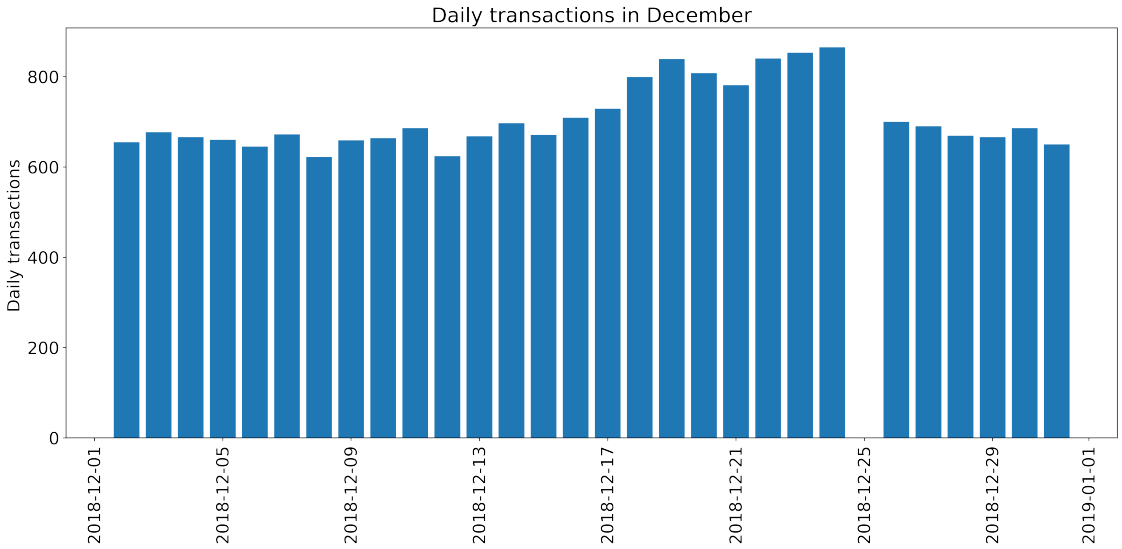


Also similar threnrd is recorded in weekly total transaction, which remain fairly stable around the mean of 4655 up until early december, which records a drastic increase in sales before crushing to below average in early January, and finally recovering in February. This customer purchase behavour is investigated further.

4.5 What is the daily transactions in December alone?

4.5.1 Analyse daily transactions in December

[473]:
```
plt.figure(figsize=(20,8), dpi = 400)
christmas_TXN = df.query("DATE > '2018-12-01' & DATE < '2019-01-01'")
christmas_trend = christmas_TXN.groupby(pd.Grouper(key='DATE',␣
 ↪freq='1D'))['TXN_ID'].count()
plt.bar(christmas_trend.index,christmas_trend.values);
plt.xticks(rotation=90);
plt.ylabel('Daily transactions ');
plt.title('Daily transactions in December');
```



A further look into the trend confirms that there daily transactions increase leading up tp the **25th** (christmas day) before crashing to 0, as shops close for christmas. They then reopen on the 26th and trasactions recover steadily to average levels.

4.6 What is the most sold pack size?

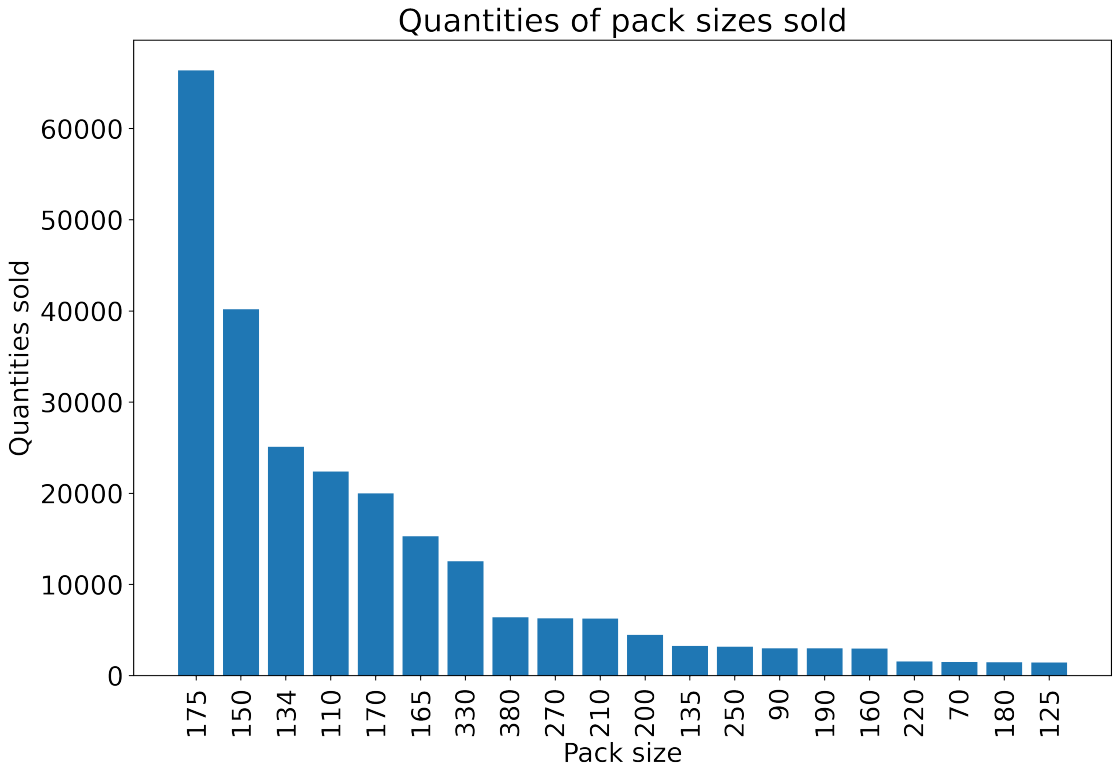
4.6.1 Analyse pack size

[474]:
```
pd.DataFrame(df.PACK_SIZE.describe())
```

[474]:

	PACK_SIZE
count	246739.000000
mean	175.583523
std	59.432239
min	70.000000
25%	150.000000
50%	170.000000
75%	175.000000
max	380.000000

[475]:
```
color = sb.color_palette()[0]
plt.figure(figsize=(12,8), dpi = 400)
pack_size = df.PACK_SIZE.value_counts()
pack_size
plt.bar(pack_size.index.astype(str),pack_size.values);
plt.xticks(rotation=90);
plt.ylabel('Quantities sold');
plt.xlabel('Pack size');
plt.title('Quantities of pack sizes sold');
```



Out of all the pack sizes, the ost purhased among customers are size 175g, 150, 134, 110, 170and 165. With size 175g packsizes being almost doubled that of size 150. Therefore management should stock more of these sizes and less of size 200, 135, 250, 90, 190, 160,220, 70, 180, 125.

4.7 What are the customer's favourite brands ?

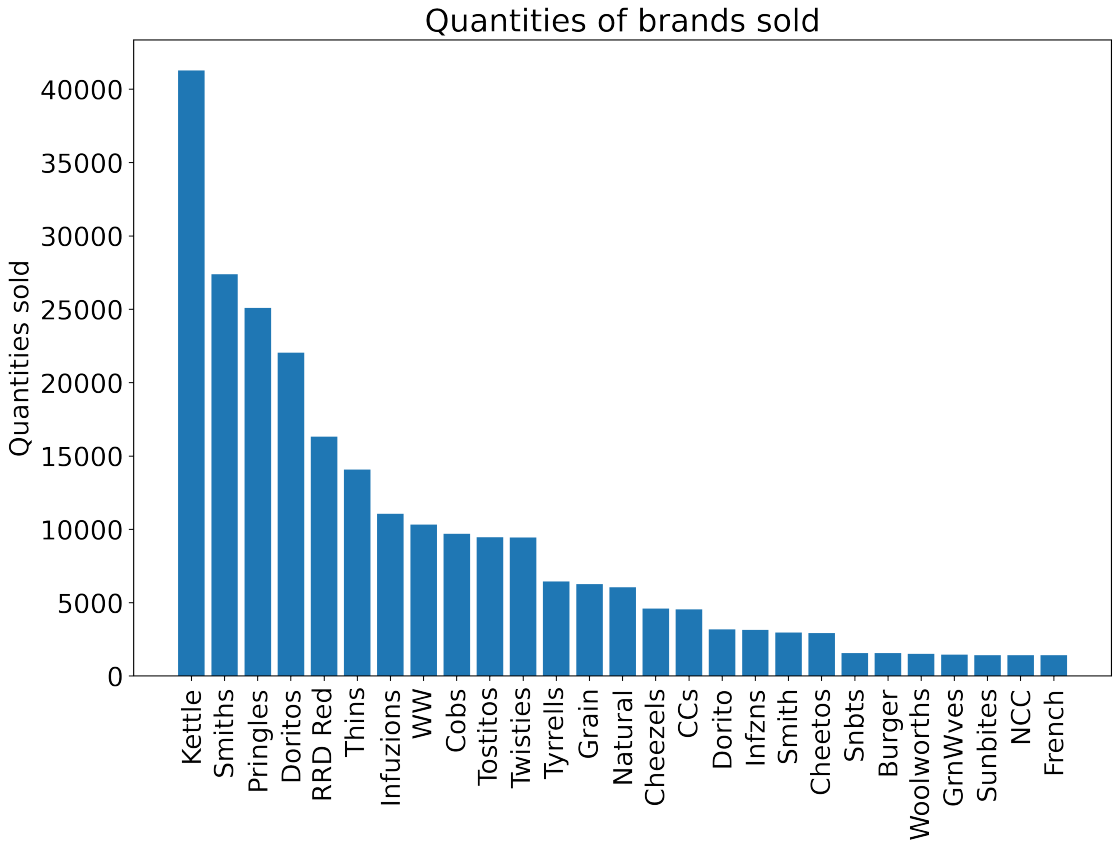
4.7.1 Create brand column and consilidate similar brands together

[476]:
```
df['PROD_BRAND'] = pd.Series([x.split()[0] for x in df.PROD_NAME.values])
```

4.7.2 Combine RED and RRD brands as both are Red Rock Deli chips

[477]:
```
df['PROD_BRAND'] = df['PROD_BRAND'].str.replace('Red|RRD','RRD Red',regex=True)
```

[478]:
```
plt.figure(figsize=(12,8), dpi = 400)
brands = df['PROD_BRAND'].value_counts()
plt.bar(brands.index,brands.values);
plt.xticks(rotation=90);
plt.ylabel('Quantities sold');
plt.title('Quantities of brands sold');
```



Kettle, Smiths, Pringles and Doritos are the top 4 brands most prefered by customers as evident in the large numbers sold. Therefore, a strategy would be to stock more of these and reduce stocking brands like french, NCC, Sunbites, GrnWves, Woolworths , burger and snbts.

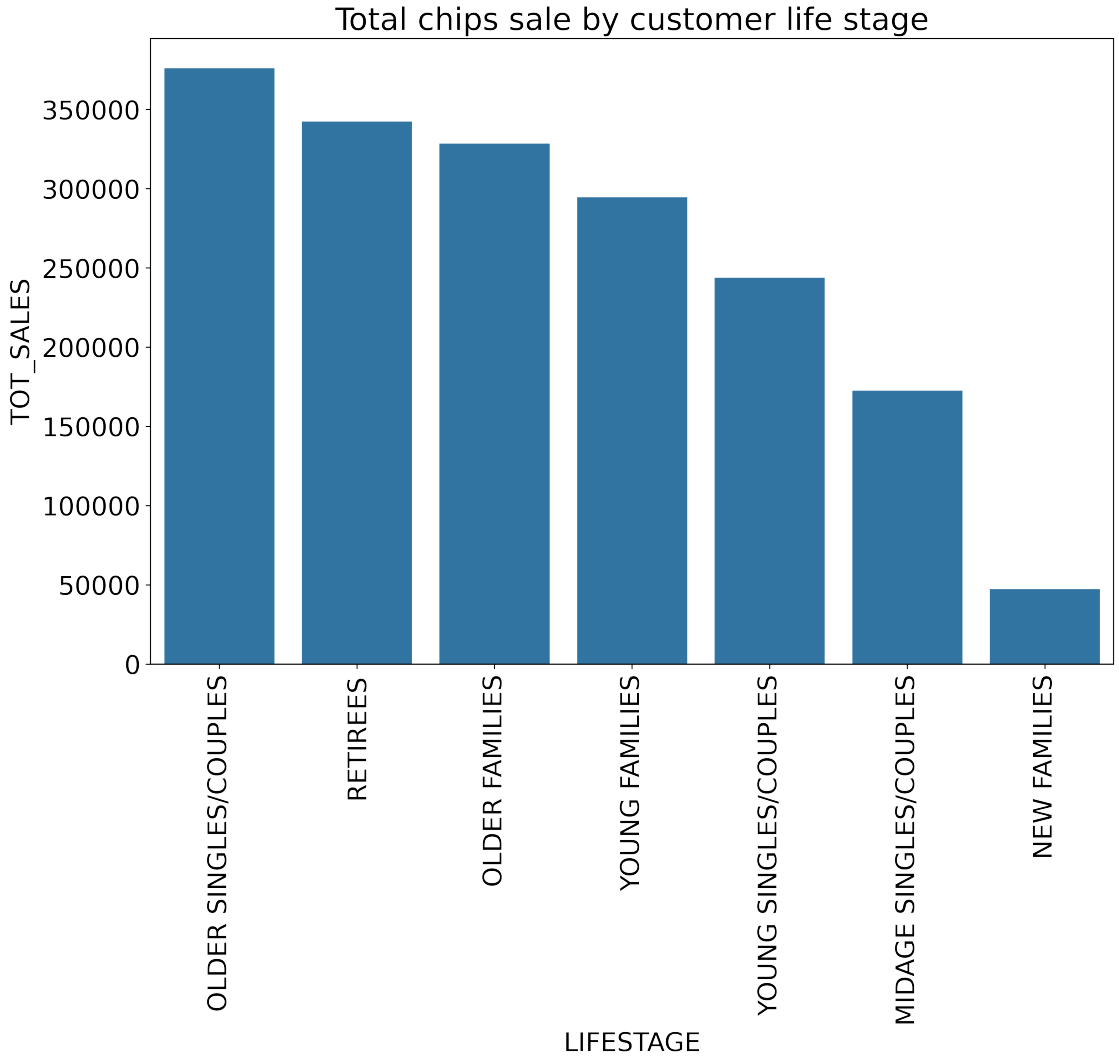
4.8 Who spends the most on chips (total sales) ?

4.8.1 Analyse total sales of customer by LIFESTAGE

[479]:
```
life_stage_spender = df.groupby(['LIFESTAGE'])['TOT_SALES'].sum().
 ↪sort_values(ascending=False).reset_index()
display(life_stage_spender)

plt.figure(figsize=(12,8), dpi = 400)
sb.barplot(x=life_stage_spender.LIFESTAGE,y=life_stage_spender.
 ↪TOT_SALES,color=color)
plt.xticks(rotation=90);
plt.title('Total chips sale by customer life stage');
```

```
                LIFESTAGE  TOT_SALES
0   OLDER SINGLES/COUPLES  376013.65
1                RETIREES  342381.90
```

```
2          OLDER FAMILIES  328519.90
3          YOUNG FAMILIES  294627.90
4   YOUNG SINGLES/COUPLES  243756.60
5  MIDAGE SINGLES/COUPLES  172523.80
6            NEW FAMILIES   47347.95
```

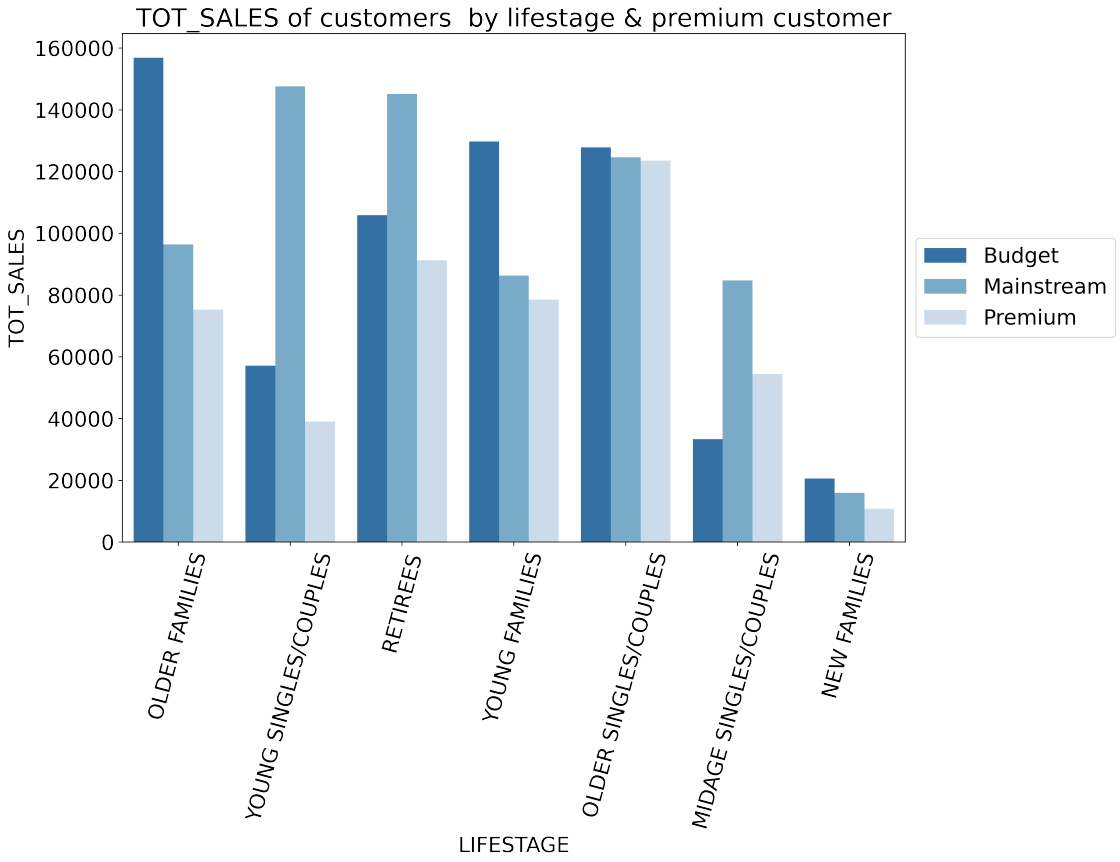


The top 3 buyers in (descending order) are the older single/couples, retirees and older families. However, this information does not reveal much about the types of chips they purchase. This is further explored.

4.9 What kind of chips are they spending on (total sales) ?

4.9.1 Analyse total sales of customer by LIFESTAGE and PREMIUM_CUSTOMER

[480]:
```
premium_status = df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['TOT_SALES']\
                 .sum().sort_values(ascending=False).reset_index()
plt.figure(figsize=(12,8), dpi= 400);
sb.barplot(x=premium_status.LIFESTAGE,y=premium_status.TOT_SALES,\
              hue=premium_status.PREMIUM_CUSTOMER,palette='Blues_r');
plt.xticks(rotation=75);
plt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));
plt.title('TOT_SALES of customers  by lifestage & premium customer');
```



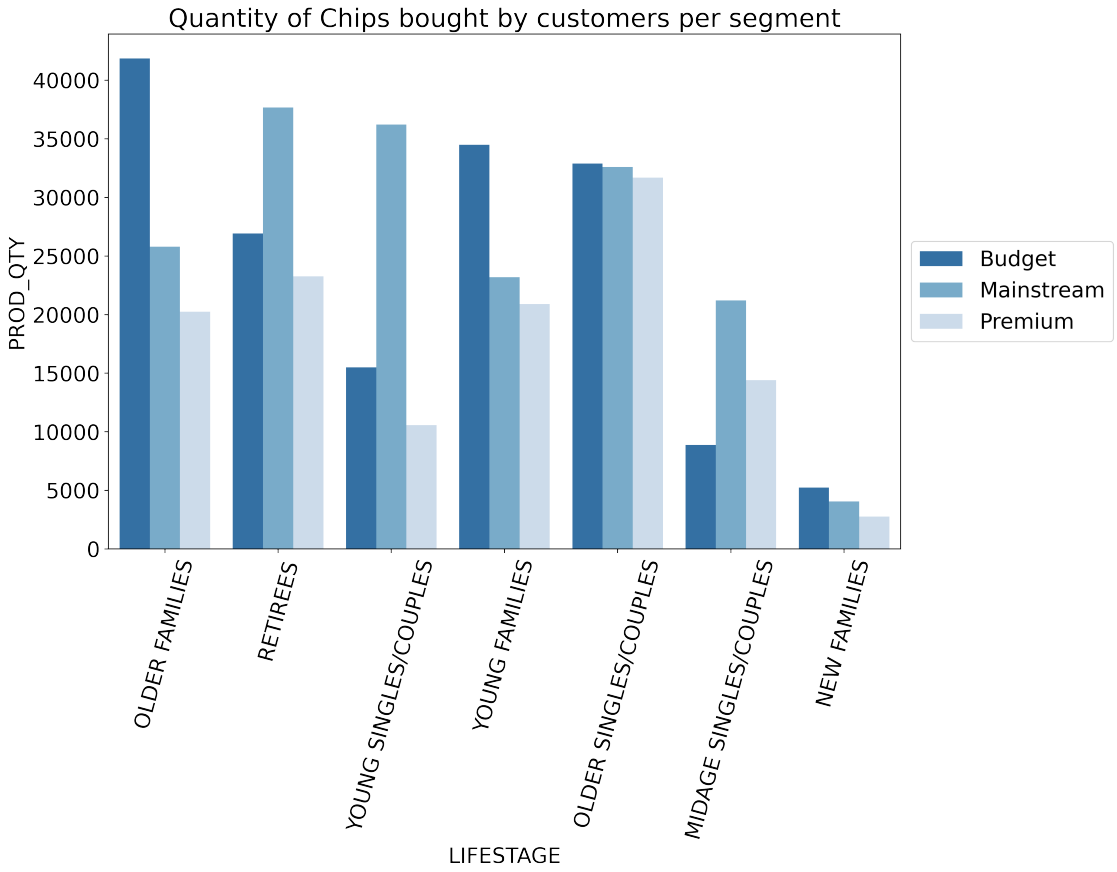
A further examination of purchase status, reveal that sales are coming mainly from Budget-older-families, Mainstream-young-singles/couples, Mainstream-retirees and Budget-young-families. Therefore these are the primary clients.

4.9.2 Analyse quantities of chips purchase by customer by life stage and purchase class

[481]:
```
premium_status_qty = df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['PROD_QTY']\
                .sum().sort_values(ascending=False).reset_index()
plt.figure(figsize=(12,8), dpi= 400);
sb.barplot(x=premium_status_qty.LIFESTAGE,y=premium_status_qty.PROD_QTY,\
             hue=premium_status_qty.PREMIUM_CUSTOMER,palette='Blues_r');
plt.xticks(rotation=75);
plt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));
plt.title('Quantity of Chips bought by customers per segment');
premium_status_qty
```

[481]:

	LIFESTAGE	PREMIUM_CUSTOMER	PROD_QTY
0	OLDER FAMILIES	Budget	41853
1	RETIREES	Mainstream	37677
2	YOUNG SINGLES/COUPLES	Mainstream	36225
3	YOUNG FAMILIES	Budget	34482
4	OLDER SINGLES/COUPLES	Budget	32883
5	OLDER SINGLES/COUPLES	Mainstream	32607
6	OLDER SINGLES/COUPLES	Premium	31693
7	RETIREES	Budget	26932
8	OLDER FAMILIES	Mainstream	25804
9	RETIREES	Premium	23266
10	YOUNG FAMILIES	Mainstream	23194
11	MIDAGE SINGLES/COUPLES	Mainstream	21213
12	YOUNG FAMILIES	Premium	20901
13	OLDER FAMILIES	Premium	20239
14	YOUNG SINGLES/COUPLES	Budget	15500
15	MIDAGE SINGLES/COUPLES	Premium	14400
16	YOUNG SINGLES/COUPLES	Premium	10575
17	MIDAGE SINGLES/COUPLES	Budget	8883
18	NEW FAMILIES	Budget	5241
19	NEW FAMILIES	Mainstream	4060
20	NEW FAMILIES	Premium	2769



Consequently,the product quantity purchased closely mirrows the trend of total sales, further validating the findings.

4.10 How many chips are bought per customer by segment ?

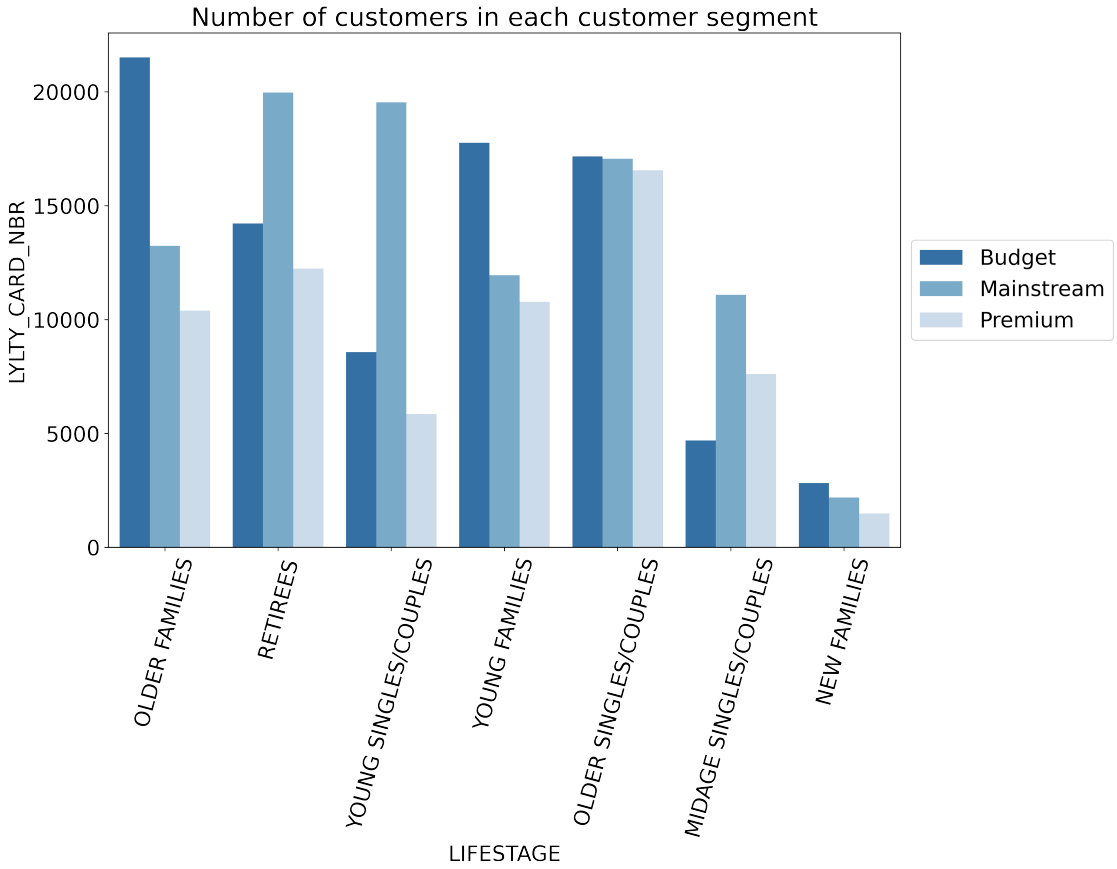
4.10.1 Analyse purchase status of each customer segments

[482]:
```
customers = df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['LYLTY_CARD_NBR']\
                .count().sort_values(ascending=False).reset_index()
plt.figure(figsize=(12,8), dpi= 400)
sb.barplot(x=customers.LIFESTAGE,y=customers.LYLTY_CARD_NBR,\
       hue=customers.PREMIUM_CUSTOMER,palette='Blues_r');
plt.xticks(rotation=75);
plt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));
plt.title('Number of customers in each customer segment');
customers
```

[482]:

	LIFESTAGE	PREMIUM_CUSTOMER	LYLTY_CARD_NBR
0	OLDER FAMILIES	Budget	21514
1	RETIREES	Mainstream	19970

2	YOUNG SINGLES/COUPLES	Mainstream	19544
3	YOUNG FAMILIES	Budget	17763
4	OLDER SINGLES/COUPLES	Budget	17172
5	OLDER SINGLES/COUPLES	Mainstream	17061
6	OLDER SINGLES/COUPLES	Premium	16559
7	RETIREES	Budget	14225
8	OLDER FAMILIES	Mainstream	13241
9	RETIREES	Premium	12236
10	YOUNG FAMILIES	Mainstream	11947
11	MIDAGE SINGLES/COUPLES	Mainstream	11095
12	YOUNG FAMILIES	Premium	10784
13	OLDER FAMILIES	Premium	10403
14	YOUNG SINGLES/COUPLES	Budget	8573
15	MIDAGE SINGLES/COUPLES	Premium	7612
16	YOUNG SINGLES/COUPLES	Premium	5852
17	MIDAGE SINGLES/COUPLES	Budget	4691
18	NEW FAMILIES	Budget	2824
19	NEW FAMILIES	Mainstream	2185
20	NEW FAMILIES	Premium	1488



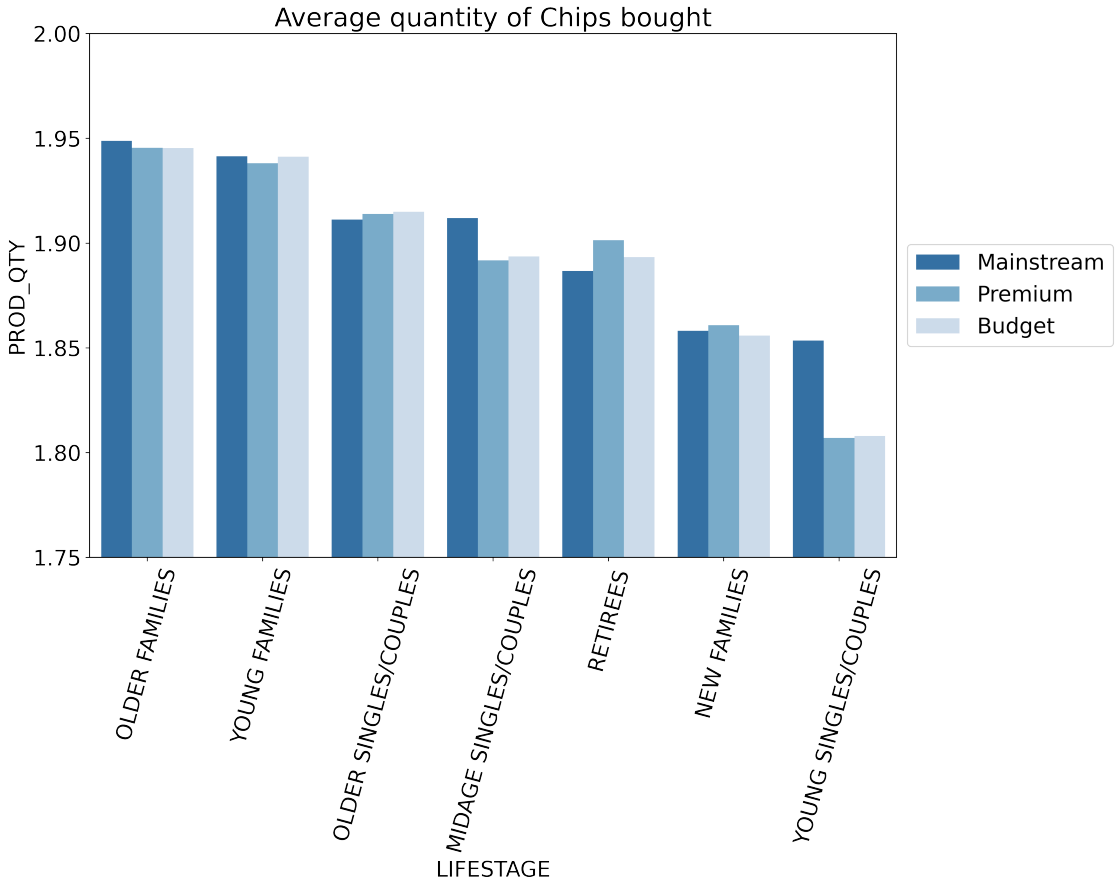
There are more Budget-older-families, Mainstream-retirees and Mainstream-young-singles/couples who buy chips. This reflects in the total product quantities purchased and ultimately also reflects the total sales. Higher sales may also be driven by more units of chips being bought per customer. This is looked at next.

4.11 What is the average number of chips bought per customer by segment?

4.11.1 Analyse average number of chips bought per customer by segment

[483]:
```
premium_status_qty = df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['PROD_QTY']\
                .mean().sort_values(ascending=False).reset_index()
display(premium_status_qty)
plt.figure(figsize=(12,8), dpi= 400);
sb.barplot(x=premium_status_qty.LIFESTAGE,y=premium_status_qty.PROD_QTY,\
             hue=premium_status_qty.PREMIUM_CUSTOMER,palette='Blues_r');
plt.xticks(rotation=75);
plt.ylim([1.75,2])
plt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));
plt.title('Average quantity of Chips bought');
```

	LIFESTAGE	PREMIUM_CUSTOMER	PROD_QTY
0	OLDER FAMILIES	Mainstream	1.948795
1	OLDER FAMILIES	Premium	1.945496
2	OLDER FAMILIES	Budget	1.945384
3	YOUNG FAMILIES	Mainstream	1.941408
4	YOUNG FAMILIES	Budget	1.941226
5	YOUNG FAMILIES	Premium	1.938149
6	OLDER SINGLES/COUPLES	Budget	1.914920
7	OLDER SINGLES/COUPLES	Premium	1.913944
8	MIDAGE SINGLES/COUPLES	Mainstream	1.911942
9	OLDER SINGLES/COUPLES	Mainstream	1.911201
10	RETIREES	Premium	1.901438
11	MIDAGE SINGLES/COUPLES	Budget	1.893626
12	RETIREES	Budget	1.893286
13	MIDAGE SINGLES/COUPLES	Premium	1.891750
14	RETIREES	Mainstream	1.886680
15	NEW FAMILIES	Premium	1.860887
16	NEW FAMILIES	Mainstream	1.858124
17	NEW FAMILIES	Budget	1.855878
18	YOUNG SINGLES/COUPLES	Mainstream	1.853510
19	YOUNG SINGLES/COUPLES	Budget	1.808002
20	YOUNG SINGLES/COUPLES	Premium	1.807075



Although, there are more Mainstream-young-singles/couples and Mainstream-retirees who buy chips and making up the 2nd and 3rd most sales, they buy less units per customer than the other segments and therefore are leaving a lot on the table in unrealizes sales. Thus there is more potential to increase sales from this segment. In effect, a good strategy would be for stores to increase the number of units sold to this customers segments to realise increase sales. If achieved, they can reach sales similar to the older-budget-families who make up the most sales by buying more units per customer.

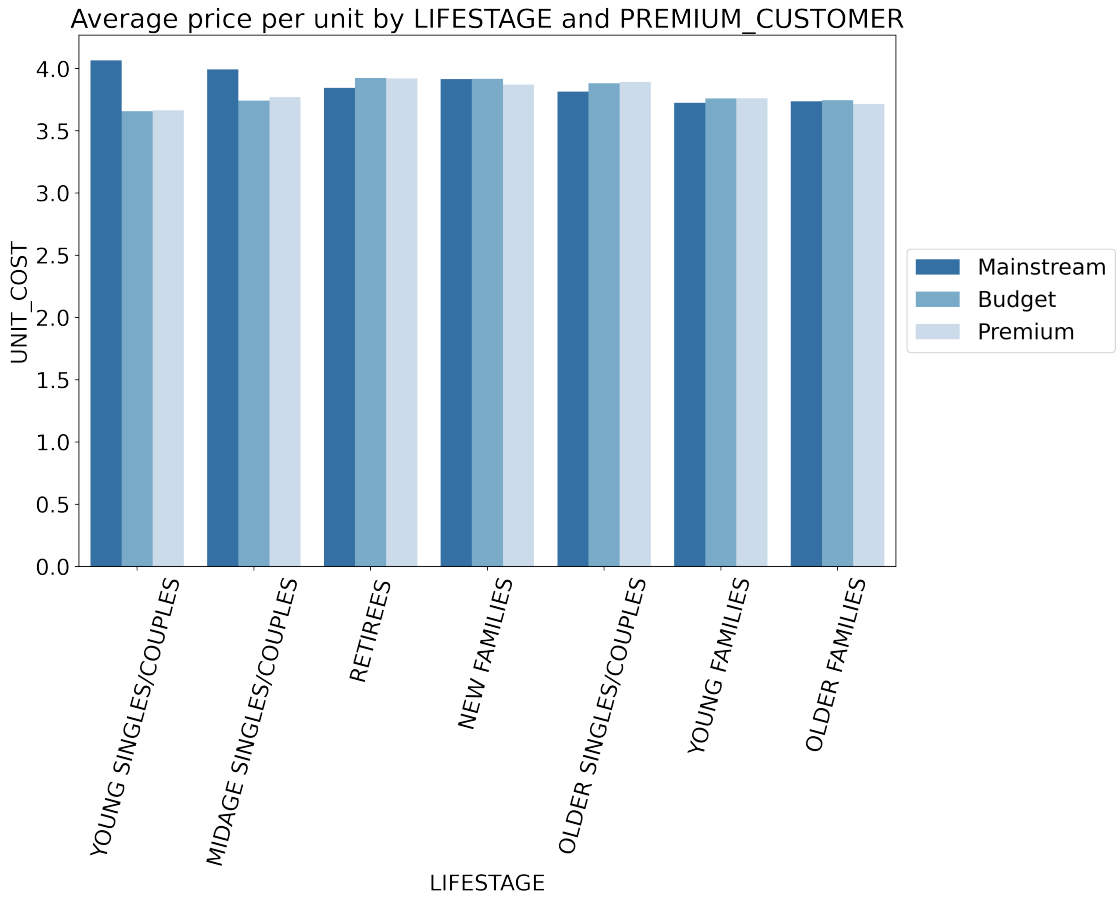
4.12 What is the average price per unit by LIFESTAGE and PREMIUM_CUSTOMER ?

4.12.1 Analyse average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

[484]:
```
premium_status_qty = df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['UNIT_COST']\
                .mean().sort_values(ascending=False).reset_index()
display(premium_status_qty)
plt.figure(figsize=(12,8), dpi= 400);
sb.barplot(x=premium_status_qty.LIFESTAGE,y=premium_status_qty.UNIT_COST,\
             hue=premium_status_qty.PREMIUM_CUSTOMER,palette='Blues_r');
```

```
plt.xticks(rotation=75);
# plt.ylim([0.15,0.35])
plt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));
plt.title('Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER');
```

	LIFESTAGE	PREMIUM_CUSTOMER	UNIT_COST
0	YOUNG SINGLES/COUPLES	Mainstream	4.065642
1	MIDAGE SINGLES/COUPLES	Mainstream	3.994241
2	RETIREES	Budget	3.924404
3	RETIREES	Premium	3.920942
4	NEW FAMILIES	Budget	3.917688
5	NEW FAMILIES	Mainstream	3.916133
6	OLDER SINGLES/COUPLES	Premium	3.893236
7	OLDER SINGLES/COUPLES	Budget	3.882096
8	NEW FAMILIES	Premium	3.872110
9	RETIREES	Mainstream	3.844294
10	OLDER SINGLES/COUPLES	Mainstream	3.814665
11	MIDAGE SINGLES/COUPLES	Premium	3.770698
12	YOUNG FAMILIES	Premium	3.762150
13	YOUNG FAMILIES	Budget	3.760737
14	OLDER FAMILIES	Budget	3.745340
15	MIDAGE SINGLES/COUPLES	Budget	3.743328
16	OLDER FAMILIES	Mainstream	3.737077
17	YOUNG FAMILIES	Mainstream	3.724533
18	OLDER FAMILIES	Premium	3.717000
19	YOUNG SINGLES/COUPLES	Premium	3.665414
20	YOUNG SINGLES/COUPLES	Budget	3.657366



Mainstream midage and **young singles and couples** are more willing to pay more per packet of chips compared to their budget and premium counterparts. As discusses, prior, this is more incentive to increase sales to **young singles and couples** as they tend to buy less quantities despite their willingness to pay more.

4.13 How significant are the differences between these two categories ?

4.13.1 Independent T-test between mainstream (of midrange and young singles) vs. premium and budget (of midrange and young)

Alternate hypothesis:the unit price for mainstream, young and mid-age singles and couples **ARE** significantly higher than that of budget or premium, young and midage singles and couples

Null hypothesis : 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 singles and couples

- if p-value < 0.01 then we can reject the Null hypothesis
- if p-value > 0.01 then we cannot reject the Null the null hypothesis

[485]:
```
mainstream = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES','YOUNG SINGLES/
↪COUPLES']")\
```

```
        .query("PREMIUM_CUSTOMER == 'Mainstream'")\
        .UNIT_COST
premium_budget = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES','YOUNG␣
 ↪SINGLES/COUPLES']")\
        .query("PREMIUM_CUSTOMER == ['Budget','Premium']")\
        .UNIT_COST
```

[486]:
```
def ans(s,p):
        if p < 0.01:
            different = 'ARE'
            print('The t-test results in a p-value of {}, i.e. the unit price␣
 ↪for mainstream, young and mid-age singles and couples {} significantly␣
 ↪higher than that of budget or premium, young and midage singles and couples.
 ↪'.format(p,different))
        else:
            different = 'ARE NOT'
            print('The t-test results in a p-value of {}, i.e. the unit price␣
 ↪for mainstream,\ young and mid-age singles and couples {} significantly␣
 ↪higher than that of budget or premium, young and midage singles and couples.
 ↪'.format(p,different))

s,p = ttest_ind(mainstream,premium_budget)

ans(s,p)
```

```
The t-test results in a p-value of 2.235645611549355e-309, i.e. the unit price
for mainstream, young and mid-age singles and couples ARE significantly higher
than that of budget or premium, young and midage singles and couples.
```

4.14 What are the most prefered brands for these two segments ?

4.14.1 Most prefered brands for mainstream MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES (in descending order)

[487]:
```
mainstream = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES','YOUNG SINGLES/
 ↪COUPLES']")\
             .query("PREMIUM_CUSTOMER == 'Mainstream'")
stats = pd.DataFrame(mainstream.PROD_BRAND.value_counts()).reset_index()\
                                    .rename(columns={'index':
 ↪'PROD_BRAND','PROD_BRAND':'TXN_Counts'})
stats
```

[487]:

	PROD_BRAND	TXN_Counts
0	Kettle	5980
1	Pringles	3474
2	Doritos	3148
3	Smiths	2966

```
4        Thins        1801
5      RRD Red        1486
6    Infuzions        1484
7     Twisties        1390
8     Tostitos        1369
9         Cobs        1359
10    Tyrrells         917
11       Grain         883
12          WW         721
13    Cheezels         567
14     Natural         534
15      Infzns         445
16      Dorito         441
17         CCs         381
18     Cheetos         281
19       Smith         231
20         NCC         131
21      French         121
22     GrnWves         118
23      Burger         110
24       Snbts         107
25  Woolworths         104
26    Sunbites          90
```

4.14.2 Most prefered brands for Budget & Premium MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES (in descending order)

[488]:
```
premium_budget = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES','YOUNG␣
 ↪SINGLES/COUPLES']")\
        .query("PREMIUM_CUSTOMER == ['Budget','Premium']")
stats2 = pd.DataFrame(premium_budget.PROD_BRAND.value_counts()).reset_index()\
                                    .rename(columns={'index':
 ↪'PROD_BRAND','PROD_BRAND':'TXN_Counts'})
stats2
```

[488]:
```
    PROD_BRAND  TXN_Counts
0       Kettle        3968
1       Smiths        3112
2     Pringles        2599
3      Doritos        2153
4      RRD Red        2000
5        Thins        1474
6           WW        1441
7    Infuzions        1138
8         Cobs         998
9     Twisties         936
```

```
10    Tostitos         923
11     Natural         785
12    Tyrrells         649
13         CCs         646
14       Grain         643
15    Cheezels         490
16       Smith         374
17      Infzns         349
18     Cheetos         348
19      Dorito         331
20      Burger         220
21  Woolworths         213
22    Sunbites         208
23       Snbts         205
24      French         190
25         NCC         171
26     GrnWves         164
```

[489]:
```
print('It can be seen that the top 5 most prefered brands for mainstream MIDAGE␣
 ↪SINGLES/COUPLES and YOUNG SINGLES/COUPLES  (in decreasing order) are: {}.
 ↪While that for Budget,Premium are {}'\
    .format(stats.index[:5].values, stats2.index[:5].values))
```

```
It can be seen that the top 5 most prefered brands for mainstream MIDAGE
SINGLES/COUPLES and YOUNG SINGLES/COUPLES  (in decreasing order) are: [0 1 2 3
4].While that for Budget,Premium are [0 1 2 3 4]
```

Both customer segments have a preference for similar brands as their top 5 brands are almost the same.

[490]:
```
print('It can be seen that the top 5 least prefered brands for mainstream␣
 ↪MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES  (in decreasing order) are:␣
 ↪{}.While that for Budget,Premium are {}.'\
    .format(stats.index[-5:].values, stats2.index[-5:].values))
```

```
It can be seen that the top 5 least prefered brands for mainstream MIDAGE
SINGLES/COUPLES and YOUNG SINGLES/COUPLES  (in decreasing order) are: [22 23 24
25 26].While that for Budget,Premium are [22 23 24 25 26].
```

Therefore,it would be beneficial to increase stocks of Kettle, Pringles, Doritos, Smiths, Thins and RRD Red brands as they are most prefered by customer segments while reducing stock of GrnWves, Burger, Snbts, Woolworths, Sunbites, French, NCC.

[491]:
```
mainstream_sizes = mainstream.PACK_SIZE.value_counts()
pd.DataFrame(mainstream_sizes).reset_index()\
                      .rename(columns={'index':'STORE_NBR','PACK_SIZE':
 ↪'TXN_Counts'})
```

[491]:

	STORE_NBR	TXN_Counts
0	175	7972
1	150	4857
2	134	3474
3	110	3175
4	170	2457
5	330	1863
6	165	1743
7	270	963
8	380	936
9	210	883
10	135	453
11	250	427
12	200	311
13	190	264
14	160	229
15	90	197
16	180	118
17	70	113
18	220	110
19	125	94

[492]:
```
premium_budget_size = premium_budget.STORE_NBR.value_counts().head(20)
pd.DataFrame(premium_budget_size).reset_index()\
                    .rename(columns={'index':'STORE_NBR','STORE_NBR':
 ↪'TXN_Counts'})
```

[492]:

	STORE_NBR	TXN_Counts
0	247	197
1	153	192
2	165	189
3	23	182
4	94	172
5	201	171
6	175	170
7	217	168
8	57	167
9	55	166
10	128	165
11	157	164
12	95	164
13	236	160
14	164	158
15	110	157
16	79	156
17	7	156
18	65	156

```
19        237         154
```

[493]:
```
print('Additionlly, the top 5 most prefered sizes by mainstream MIDAGE SINGLES/
 ↪COUPLES and YOUNG SINGLES/COUPLES  (in decreasing order) are: {}.While that␣
 ↪for Budget,Premium are {}. The least frequented shop are {} and {}␣
 ↪respectively'\
    .format(mainstream_sizes.index[:5]\
    .values, premium_budget_size.index[:5]\
    .values,mainstream_sizes.index[-5:]\
    .values, premium_budget_size.index[-5:].values ))
```

```
Additionlly, the top 5 most prefered sizes by mainstream MIDAGE SINGLES/COUPLES
and YOUNG SINGLES/COUPLES  (in decreasing order) are: [175 150 134 110
170].While that for Budget,Premium are [247 153 165  23  94]. The least
frequented shop are [ 90 180  70 220 125] and [110  79   7  65 237] respectively
```

Therefore, stock more of sizes 175 150 134 110 170 of the top 5 brands of mainstream MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES. And for Budget and Premium products, stock more of sizes 247 153 165 23 94 as they could result in increase sales.

4.15 which stores are doing the most transactions ?

4.15.1 Analyse transactions per store (focused on top 20)

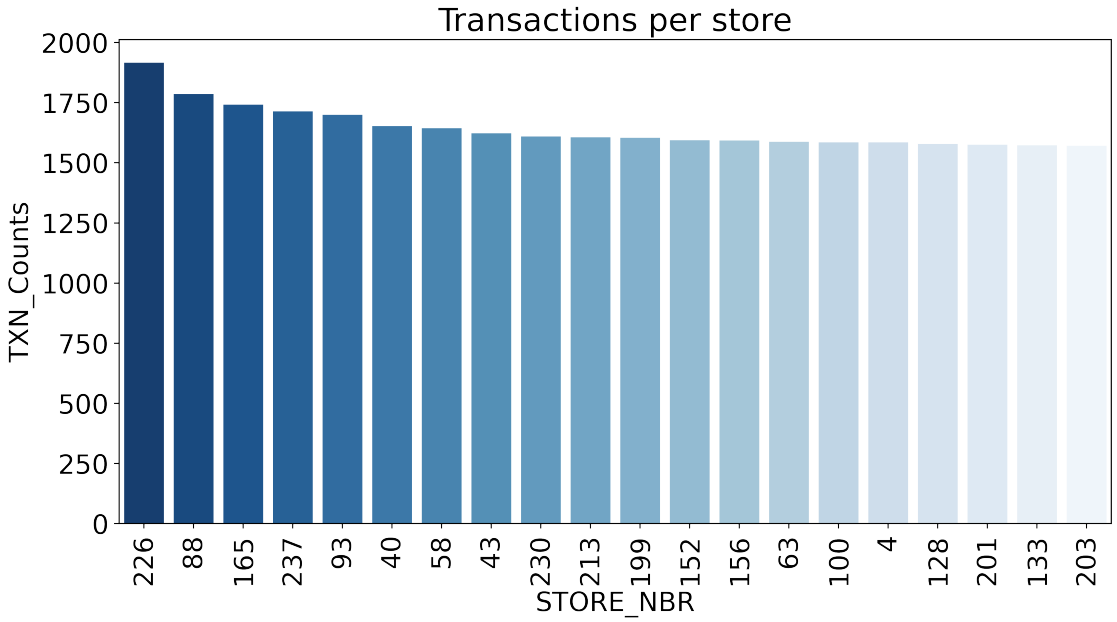
[494]:
```
store = pd.DataFrame(df.STORE_NBR.value_counts(ascending=False)\
                     .head(20))\
                     .reset_index()\
                     .rename(columns={'index':'STORE_NBR','STORE_NBR':
 ↪'TXN_Counts'})

store.STORE_NBR = store.STORE_NBR.astype('str')
store
```

[494]:

	STORE_NBR	TXN_Counts
0	226	1916
1	88	1786
2	165	1741
3	237	1714
4	93	1699
5	40	1653
6	58	1644
7	43	1622
8	230	1609
9	213	1606
10	199	1604
11	152	1594
12	156	1592
13	63	1587

14	100	1585
15	4	1585
16	128	1578
17	201	1575
18	133	1573
19	203	1570

[495]:
```
plt.figure(figsize=(12,6), dpi =400)
sb.barplot(x=store.STORE_NBR,y= store.TXN_Counts, palette='Blues_r');
plt.xticks(rotation = 90);
plt.title('Transactions per store');
```



The top 5 stores (descending order) with the most transactions are store number: 226, 88, 165, 237 and 93. Therefore you can start by to applying the strategies tp these stores that see the most transcations.

5 Strategy & Conclusions

Data Analytics Report:

Introduction: After analyzing the dataset of the purchasing behavior of customers buying chips in a particular region. I present my findings to Julia - Category Manager, to fascilitate better understanding on the types of customers who purchase chips, their purchasing behavior and to unearth insigths to drive strategy and increase sales.

Findings: The daily average transaction of chips hovered around $678 throughout the year except in early December, where it shot up to a little above $859 before returning to normal levels. Similar trend was observed in weekly total transactions, which remained stable around the mean of 4655

up until early December, which recorded a drastic increase in sales before crashing to below average in early January and recovering in February.

Further analysis of the trend confirms that the daily transactions increase leading up to the 25th (Christmas day) before crashing to zero, as shops close for Christmas. They then reopen on the 26th and transactions recover steadily to average levels. Out of all the pack sizes, the most purchased pack sizes among customers are 175g, 150g, 134g, 110g, 170g, and 165g. Kettle, Smiths, Pringles, and Doritos are the top 4 brands most preferred by customers.

The top 3 buyers in descending order are older single/couples, retirees, and older families. However, this information does not reveal much about the types of chips they purchase. Further examination of purchase status reveals that sales are coming mainly from Budget-older-families, Mainstream-young-singles/couples, Mainstream-retirees, and Budget-young-families. The product quantity purchased closely mirrors the trend of total sales, further validating the findings.

Although there are more Mainstream-young-singles/couples and Mainstream-retirees who buy chips and make up the 2nd and 3rd most sales, they buy fewer units per customer than the other segments. Therefore there is more potential to increase sales from this segment. A good strategy would be for stores to increase the number of units sold to this customer segment to realize increased sales.

The average price per packet of chips is higher for Mainstream midage and young singles/couples compared to their budget and premium counterparts. Independent t-test results in a p-value of 2.235645611549355e-309, i.e. the unit price for mainstream, young and mid-age singles and couples is significantly higher than that of budget or premium, young and midage singles and couples.

`Conclusion`:

Based on the findings, the Category Manager should prioritize stocking the top 4 preferred brands (Kettle, Pringles, Doritos, Smiths) over the less popular brands (GrnWves, Burger, Snbts, Woolworths, Sunbites, French, NCC). Also, the main customer segments driving sales are Budget-older-families, Mainstream-young-singles/couples, Mainstream-retirees, and Budget-young-families, thus they should be the primary targets for marketing and promotions.

Further more, to increase sales, it is advised to stock more of the preferred chip sizes (175, 150, 134, 110, 170) for mainstream MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES. Also, for Budget and Premium products, sizes 247, 153, 165, 23, and 94 are recommended.

It also worth mentioning that, stores should increase the number of units sold to the Mainstream-young-singles/couples and Mainstream-retirees segments. Additionally, since Mainstream midage and young singles/couples are more willing to pay more per packet of chips, it may be beneficial to slightly increase prices for these segments.

Lastly, it is suggested to pilot these recommendations at the top-selling stores (226, 88, 165, 237, and 93) to test their effectiveness before implementing them across all stores.