

# Quantum\_Data\_Analytics

March 27, 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 Quantum as part of their Data Analytics Virtual Experience program. It consists of transaction dataset; QVI\_transaction\_data.xlsx 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: Date of transaction
- STORE\_NBR: Store within which transaction took place
- LYLTY\_CARD\_NBR: Loyalty 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

```
[343]: 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.theforge.com/vinternships/companyassets/32A6DqtsbF7LbKdcq/
˓→QVI_transaction_data.xlsx',
            'https://cdn.theforge.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

```
[344]: transactions = pd.read_excel('QVI_transaction_data.xlsx')
purchase_behavior = 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

```
[345]: purchase_behavior.sample(5)
```

```
[345]:   LYLTY_CARD_NBR          LIFESTAGE PREMIUM_CUSTOMER
 9637           35492  OLDER SINGLES/COUPLES      Premium
 14975          55086  MIDAGE SINGLES/COUPLES Mainstream
 35781          131230        RETIREES      Premium
 72157          271081    OLDER FAMILIES      Budget
 14235          52344  YOUNG SINGLES/COUPLES      Premium
```

### 3.6 Dimensions of data set

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

Rows: 72637

Columns: 3

### 3.7 Missing values and data types

```
[347]: purchase_behavior.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

```
[348]: purchase_behavior.PREMIUM_CUSTOMER.nunique(),\
purchase_behavior.LIFESTAGE.nunique()
```

```
[348]: (3, 7)
```

```
[349]: purchase_behavior.PREMIUM_CUSTOMER.unique().tolist()
```

```
[349]: ['Premium', 'Mainstream', 'Budget']
```

```
[350]: purchase_behavior.LIFESTAGE.unique().tolist()
```

```
[350]: ['YOUNG SINGLES/COUPLES',
 'YOUNG FAMILIES',
 'OLDER SINGLES/COUPLES',
 'MIDAGE SINGLES/COUPLES',
 'NEW FAMILIES',
 'OLDER FAMILIES',
 'RETIREEES']
```

### 3.8.1 NB:

The data consists of **72637 rows** by **3 columns**. There are **no missing values**. 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

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

```
[351]:      DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
41732    43290        105          105065  105657       111
87338    43412         93          93054   90953       105
75046    43283        123          123346  127512       26
167181   43282        214          214255  213856       80
74708    43421        114          114121  117618       62

                                         PROD_NAME  PROD_QTY  TOT_SALES
41732    Smiths Chip Thinly Cut Original 175g           2       6.0
87338    Woolworths Cheese Rings 190g           2       3.6
75046    Pringles Sweet&Spcy BBQ 134g           2       7.4
167181   Natural ChipCo Sea Salt & Vinegr 175g           1       3.0
74708    Pringles Mystery Flavour 134g           2       7.4
```

## 3.10 Dimensions of data set

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

Rows: 264836  
Columns: 8

## 3.11 Missing values and data types

```
[353]: 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
1	STORE_NBR	264836	non-null
2	LYLTY_CARD_NBR	264836	non-null
3	TXN_ID	264836	non-null
4	PROD_NBR	264836	non-null
5	PROD_NAME	264836	non-null
6	PROD_QTY	264836	non-null
7	TOT_SALES	264836	non-null

dtypes: float64(1), int64(6), object(1)  
memory usage: 16.2+ MB

### 3.12 Descriptive statistics

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

	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

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

	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
124845							2	6.0

### 3.14 Further investigate the transaction ID

```
[356]: transactions.query("TXN_ID == " +  
    "108462")[['TXN_ID', 'PROD_NAME', 'PROD_QTY', 'TOT_SALES']]
```

	TXN_ID	PROD_NAME	PROD_QTY	TOT_SALES
124843	108462	Smiths Thinly Cut	Roast Chicken	175g
124843	108462		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 maximum 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 formatted 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 section all the data issues outlined in assessment stage are cleaned.

```
[357]: # make copies of both data sets
purchase_behavior_copy = purchase_behavior.copy()
transactions_copy = transactions.copy()
```

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

```
[358]: 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

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

### 3.16.3 Issue #3: Add a unit cost column

```
[360]: 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

```
[361]: 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

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

```
[362]:
```

		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  $Q1 - 1.5 \times IQR$  to  $Q3 + 1.5 \times IQR$  is considered a potential outlier.

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

```
[364]: 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}\n".
format(column,low_range,upper_range))

```

[365]: outlier('PROD\_QTY')  
outlier('TOT\_SALES')

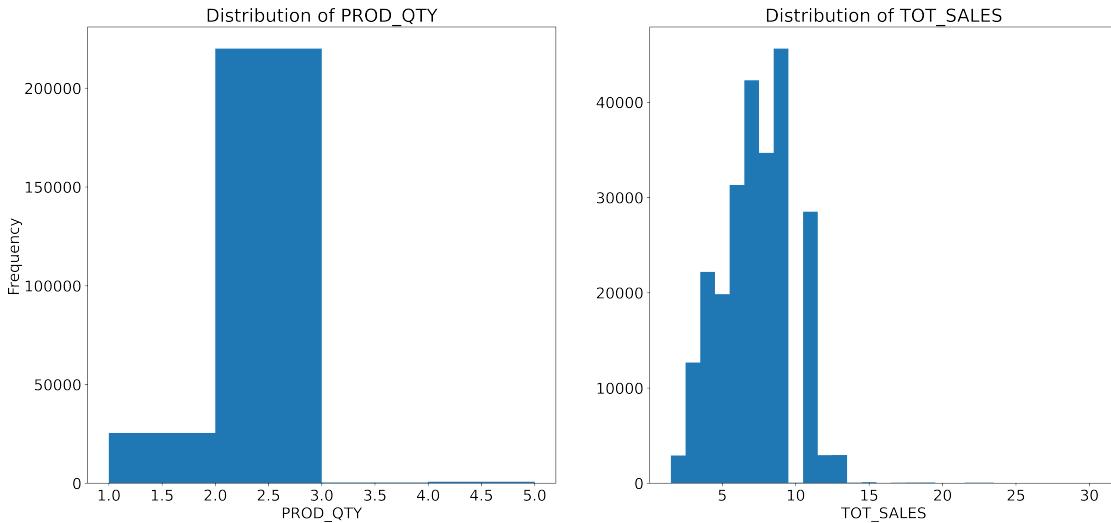
For PROD\_QTY an outlier is any value outside the range of 2.00 and 2.00  
For TOT\_SALES an outlier is any value outside the range of -0.30 and 14.90

### 3.16.6 Looking at the descriptive statistics of both datasets

[366]: pd.merge(pd.DataFrame(transactions\_copy.PROD\_QTY.describe()),\n pd.DataFrame(transactions\_copy.TOT\_SALES.describe()),\n on=pd.DataFrame(transactions\_copy.TOT\_SALES.describe()).index).\n rename(columns={'key\_0':'index'})

	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

[417]: plt.figure(figsize=(22,10),dpi=1000)  
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 it is safe to say that 200 and 650 for PROD\_QTY and TOT\_SALES are outliers respectively and are removed from the dataset.

### 3.16.7 Outlier is dropped from data set

```
[368]: 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

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

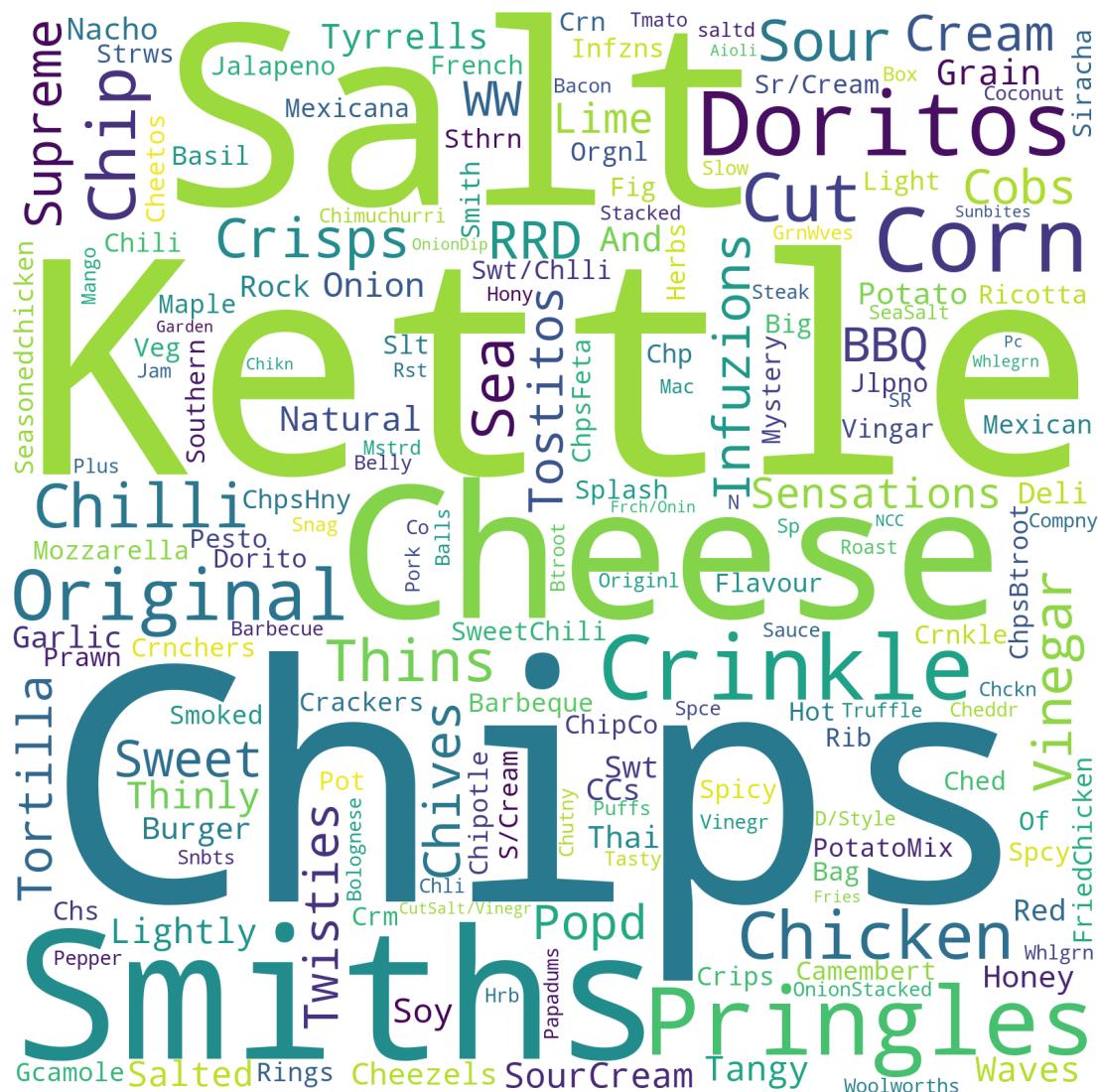
### 3.16.9 Issue 6: examine products summary, rid it of irrelevant products

```
[412]: 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=1000)
    plt.imshow(wordcloud)
```

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

### **3.16.10 Summary of individual PROD\_NAME words with wordcloud**

```
[413]: 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

```
[414]: 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

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

There are 0 instances of irrelevant salsa products in dataset.

```
[415]: Empty DataFrame\nColumns: [Product, Counts]\nIndex: []
```

### 3.16.13 Remove 18094 instances of irrelevant salsa products from dataset

```
[416]: salsa_index = salsa_df.index.tolist()\ntransactions_copy.drop(index=salsa_index, inplace=True)\nprint('After removing {} irrelevant salsa entries, initial {} total\\n'\\n    'entries reduced to {}'\\n    .format(total, transactions_copy.shape[0], transactions_copy.\n        shape[0]))
```

After removing 0 irrelevant salsa entries, initial 264836 total  
entries reduced to 246739

### 3.16.14 Assess the top 20 most frequent words in descending order

```
[375]: PROD_NAME = transactions_copy.PROD_NAME.str.replace('&', '')\nwords = ' '.join(PROD_NAME.values).split()\nword_counts = Counter(words)\nsorted_words = sorted(word_counts.items(), key=lambda x: x[1], reverse=True)\nfor word, count in sorted_words[:20]:\n    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

```
[376]: purchase_behavior_copy.PREMIUM_CUSTOMER = purchase_behavior_copy.PREMIUM_CUSTOMER.
         ↪astype('category')
purchase_behavior_copy.LIFESTAGE = purchase_behavior_copy.LIFESTAGE.
         ↪astype('category')
```

```
[377]: purchase_behavior_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

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

### 3.16.17 Dimension of final dataset

```
[379]: clean_data.shape
```

```
[379]: (246739, 12)
```

### 3.16.18 missing values in dataset

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

```
[380]: 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

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

## 4 Exploratory Data Analysis

### 4.1 Load final dataset

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

### 4.2 Assess final dataset

```
[383]: 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

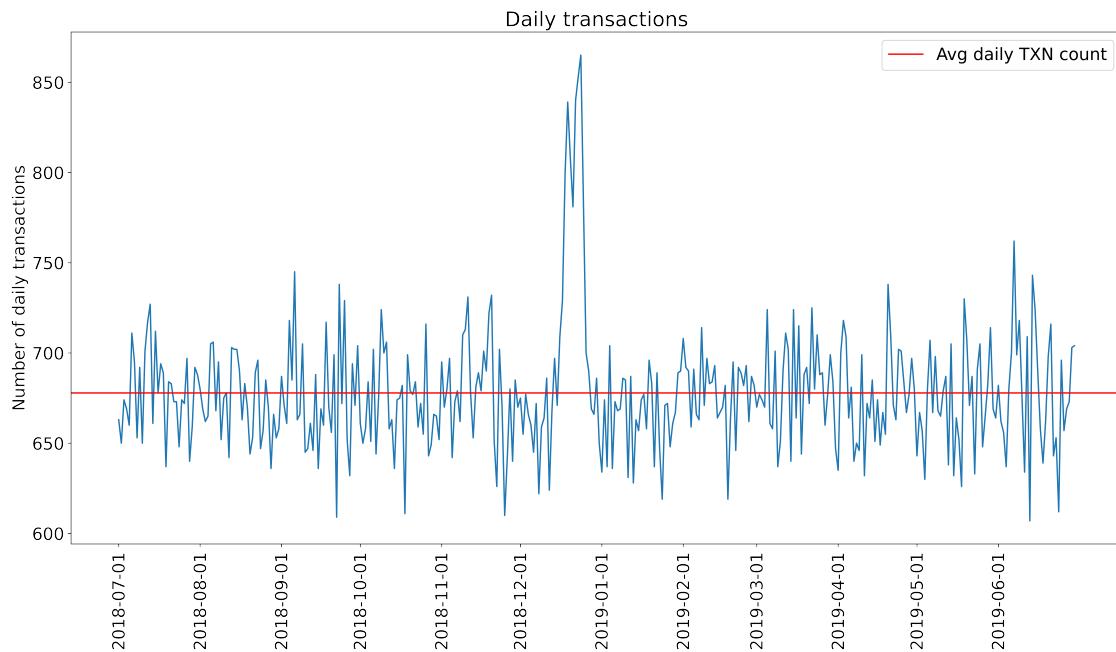
```

[418]: 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 = 1000)
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 tran-

sctions shoots up to a little above 859, before quickly returning to normal levels.

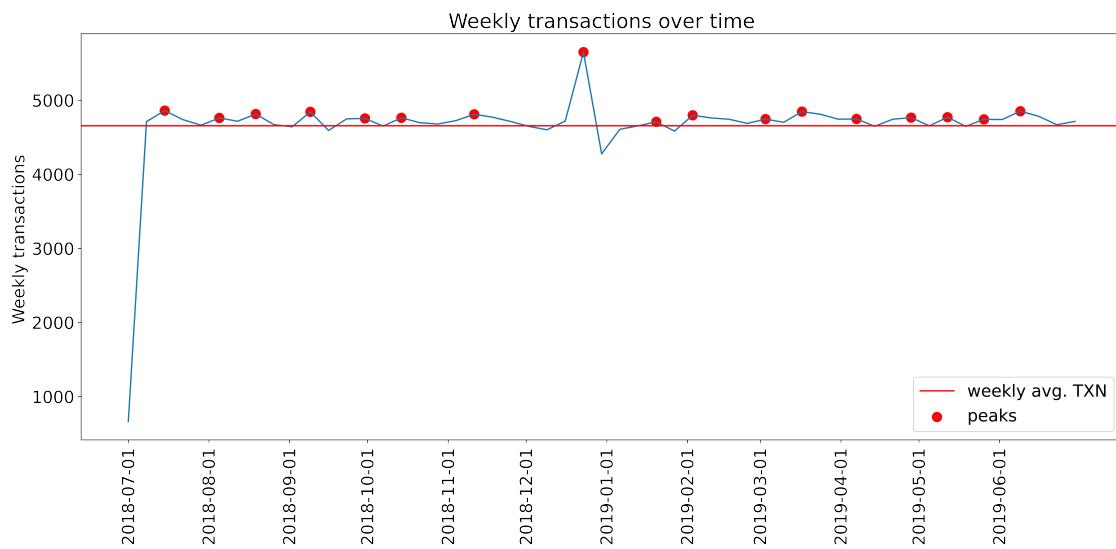
#### 4.4 What is the weekly number of transaction over time?

##### 4.4.1 Analyse weekly transactions

```
[419]: 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 = 1000)
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

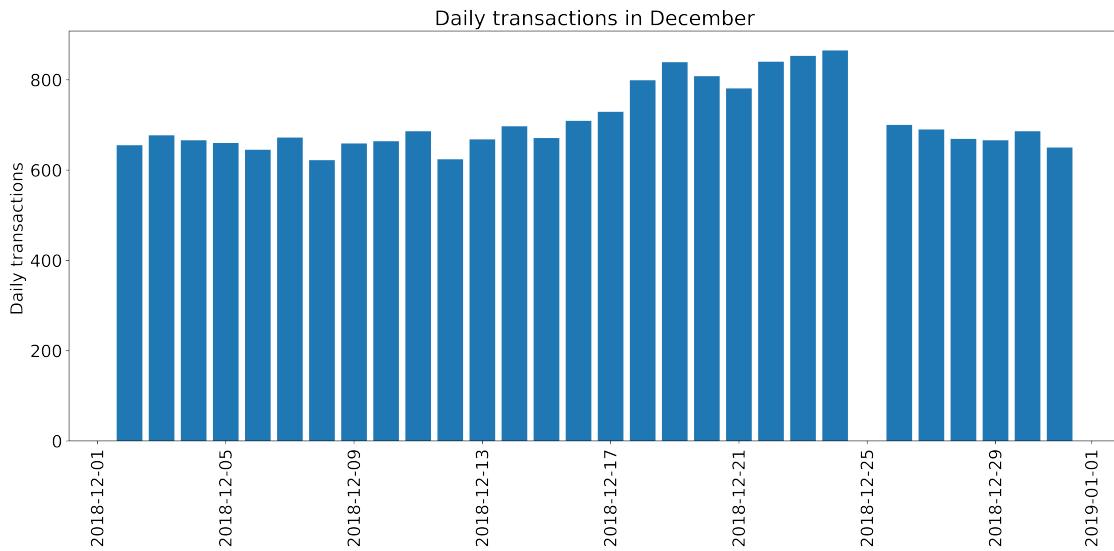


Also similar thrend is recorded in weekly total transaction, which remain fairly stable around the mean of 4655 up until early december, which records a drastic increase in sales before crushing to below average in early January, and finally recovering in February. This customer purchase behaviour is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

## 4.5 What is the daily transactions in December alone?

### 4.5.1 Analyse daily transactions in December

```
[420]: plt.figure(figsize=(20,8), dpi = 1000)
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?

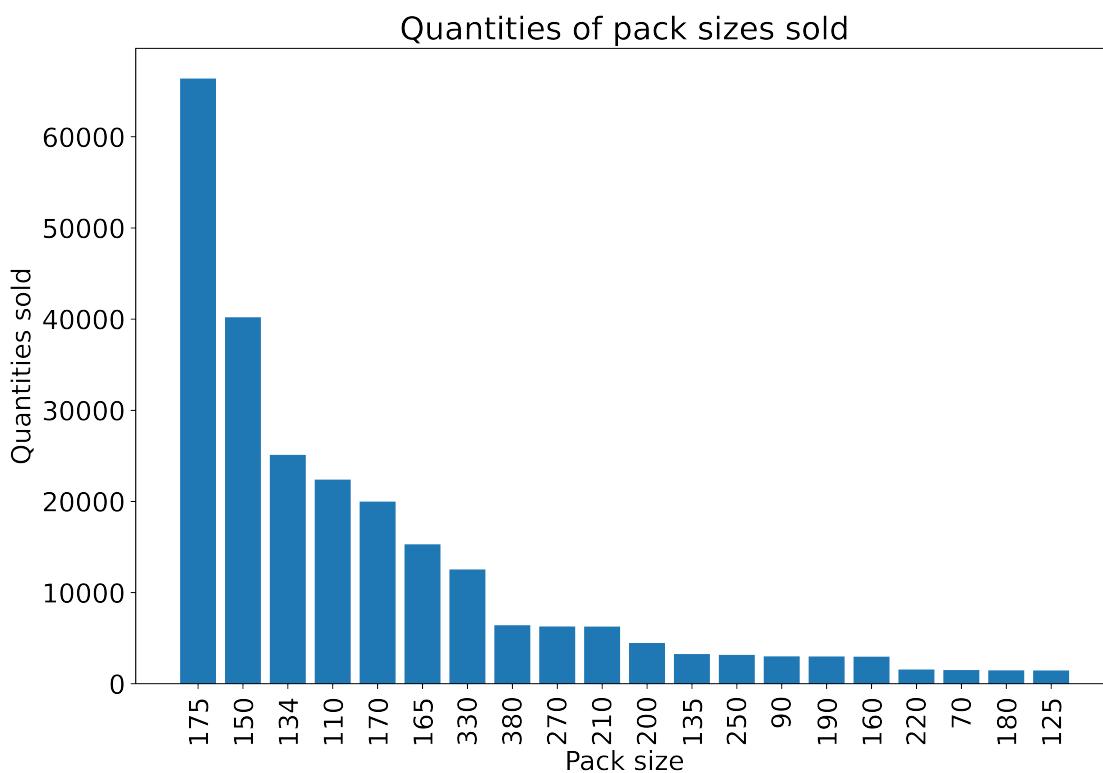
### 4.6.1 Analyse pack size

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

```
[387]:          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
```

```
[421]: color = sb.color_palette()[0]
plt.figure(figsize=(12,8), dpi = 1000)
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 ones purchased among customers are size 175g, 150, 134, 110, 170 and 165. With size 175g pack sizes 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 ?

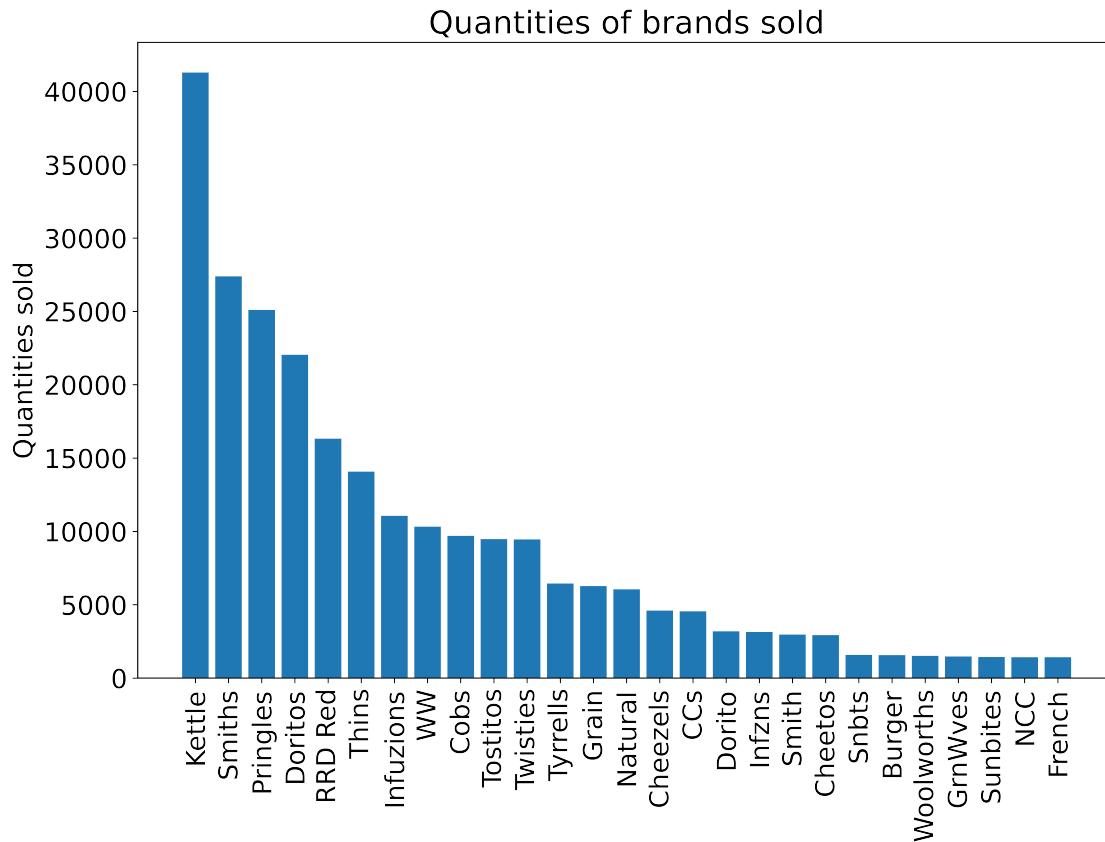
### 4.7.1 Create brand column and consolidate similar brands together

```
[389]: 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

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

```
[422]: plt.figure(figsize=(12,8), dpi = 1000)
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 preferred 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.

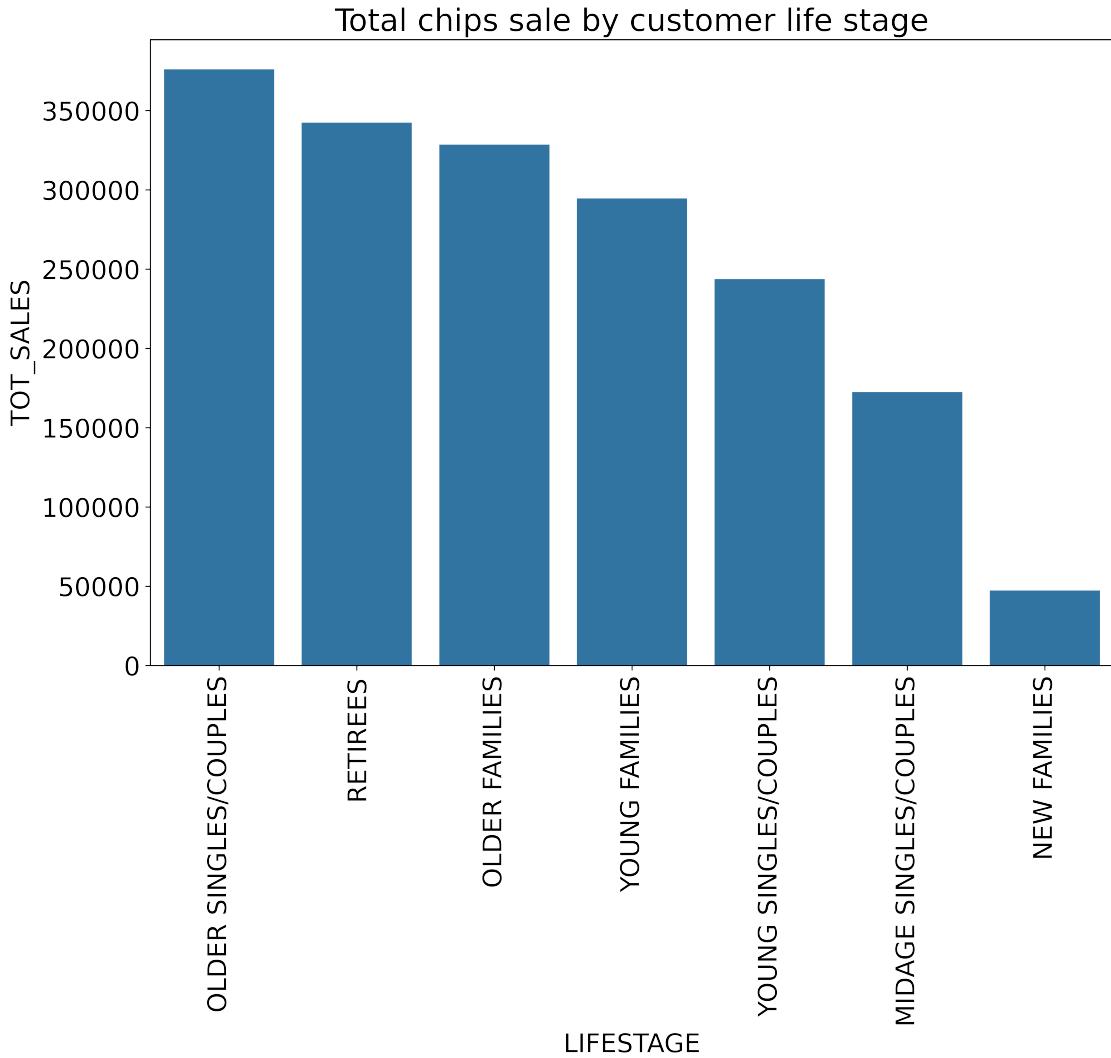
## 4.8 Who spends the most on chips (total sales) ?

### 4.8.1 Analyse total sales of customer by LIFESTAGE

```
[423]: 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 = 1000)
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



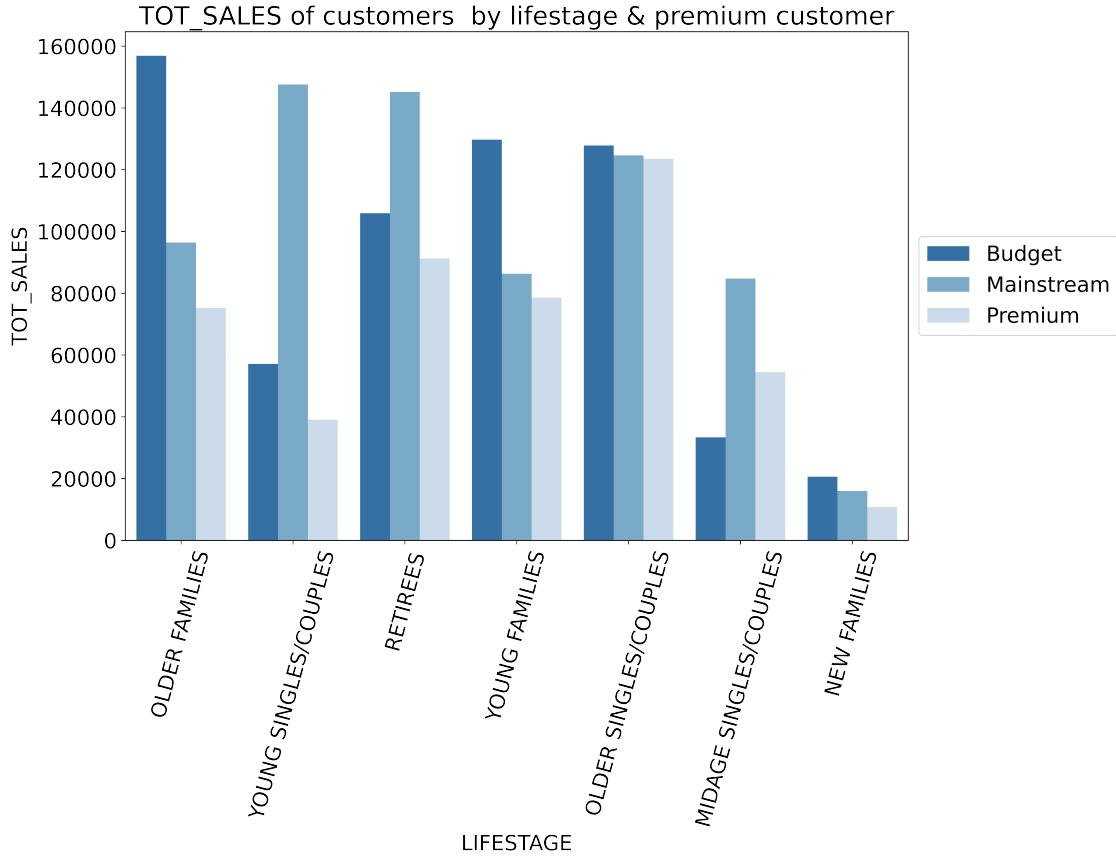
The top 3 buyers in (descending order) are the older single/couples, retirees and older families. However, this information does not reveal much about the types of chips they purchase. This is further explored.

## 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

```
[424]: premium_status = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES']\
         .sum().sort_values(ascending=False).reset_index()
plt.figure(figsize=(12,8), dpi= 1000);
sb.barplot(x=premium_status.LIFESTAGE,y=premium_status.TOT_SALES,\n            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');
```



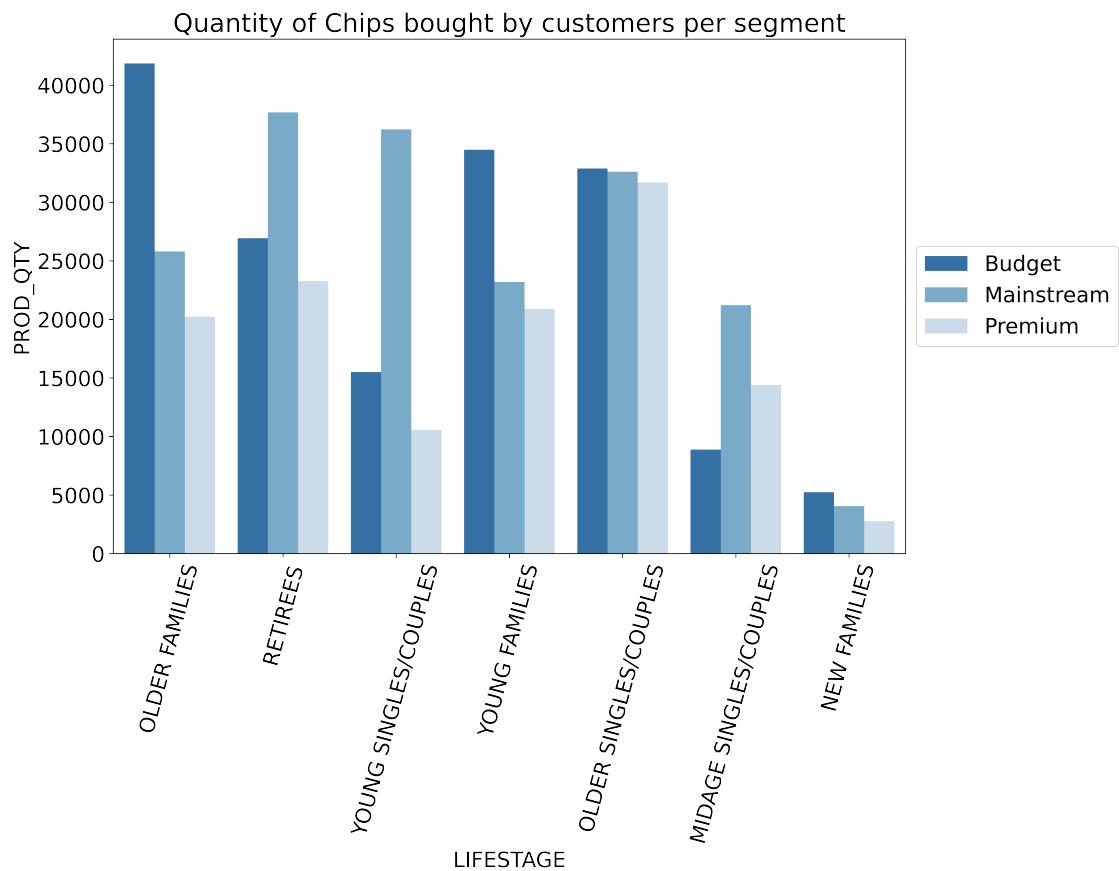
A further examination of purchase status, reveal that sales are coming mainly from Budget-older-families, Mainstream-young-singles/couples, Mainstream-retirees and Budget-young-families. Therefore these are the primary clients.

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

```
[425]: premium_status_qty = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PROD_QTY']\n        .sum().sort_values(ascending=False).reset_index()\n\nplt.figure(figsize=(12,8), dpi= 1000);\nsb.barplot(x=premium_status_qty.LIFESTAGE,y=premium_status_qty.PROD_QTY,\n            hue=premium_status_qty.PREMIUM_CUSTOMER,palette='Blues_r');\nplt.xticks(rotation=75);\nplt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));\nplt.title('Quantity of Chips bought by customers per segment');\npremium_status_qty
```

[425] :

	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 mirrors the trend of total sales, further validating the findings.

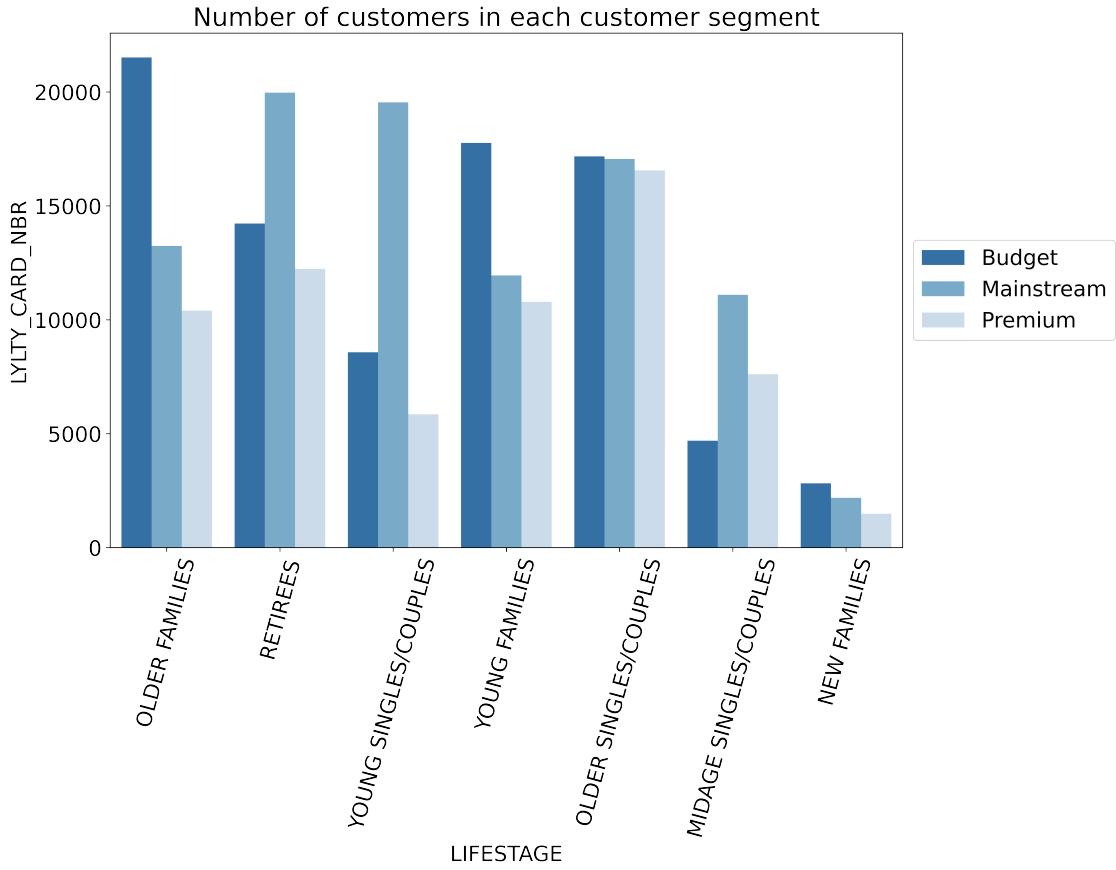
## 4.10 How many chips are bought per customer by segment ?

### 4.10.1 Analyse purchase status of each customer segments

```
[426]: customers = df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])['LYLTY_CARD_NBR']\n        .count().sort_values(ascending=False).reset_index()\nplt.figure(figsize=(12,8), dpi= 1000)\nsb.barplot(x=customers.LIFESTAGE,y=customers.LYLTY_CARD_NBR,\n            hue=customers.PREMIUM_CUSTOMER,palette='Blues_r');\nplt.xticks(rotation=75);\nplt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));\nplt.title('Number of customers in each customer segment');\ncustomers
```

```
[426]:
```

	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.

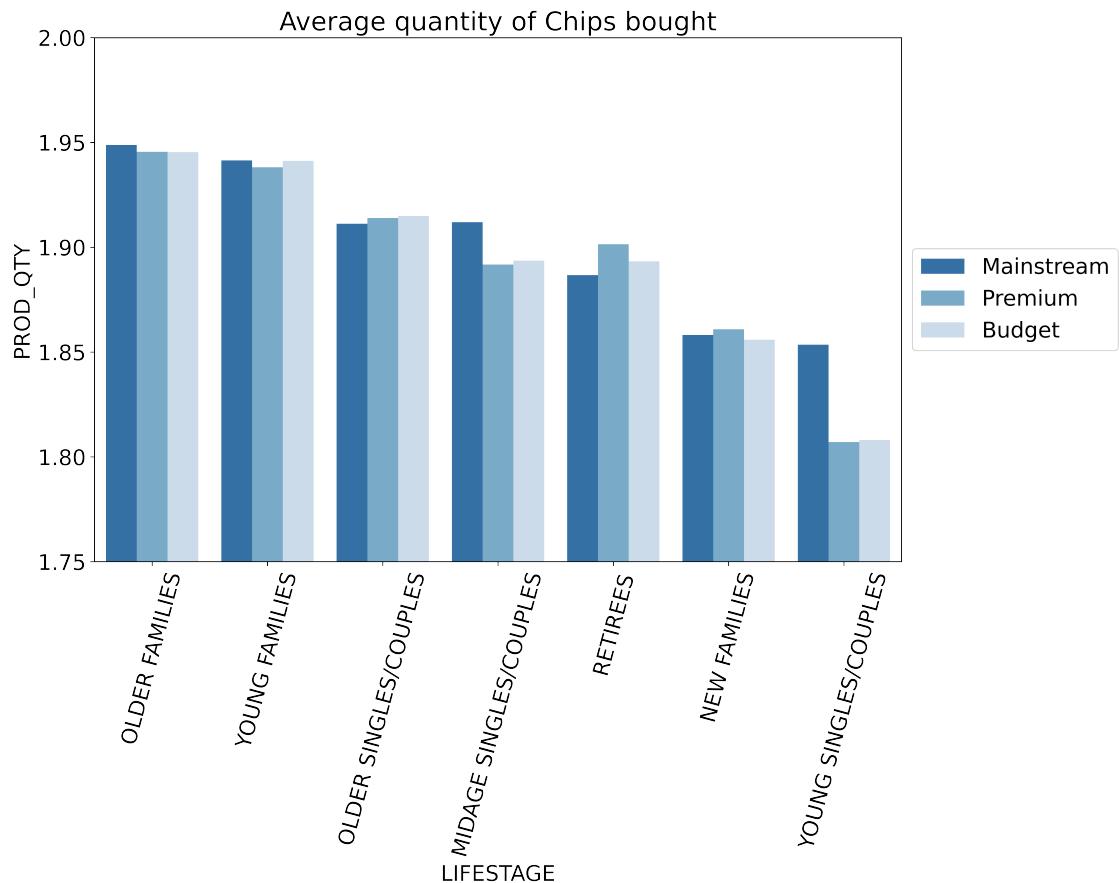
#### 4.11 What is the average number of chips bought per customer by segment?

##### 4.11.1 Analyse average number of chips bought per customer by segment

```
[427]: premium_status_qty = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PROD_QTY']\n        .mean().sort_values(ascending=False).reset_index()\n\ndisplay(premium_status_qty)\nplt.figure(figsize=(12,8), dpi= 1000);\nsb.barplot(x=premium_status_qty.LIFESTAGE,y=premium_status_qty.PROD_QTY,\n            hue=premium_status_qty.PREMIUM_CUSTOMER,palette='Blues_r');\nplt.xticks(rotation=75);\nplt.ylim([1.75,2])\nplt.legend(loc = 6, bbox_to_anchor = (1.0, 0.5));\nplt.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



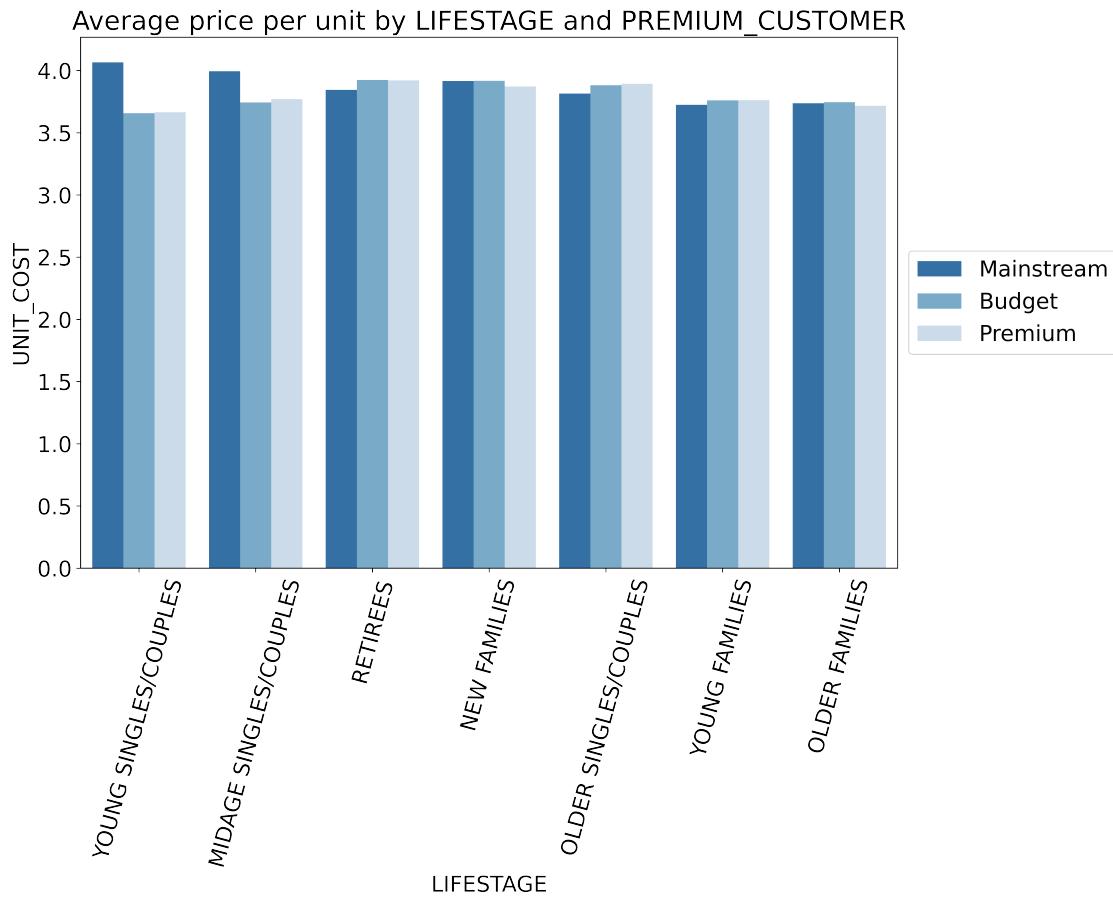
Although, there are more Mainstream-young-singles/couples and Mainstream-retirees who buy chips and making up the 2nd and 3rd most sales, they buy less units per customer than the other segments and therefore are leaving a lot on the table in unrealized sales. Thus there is more potential to increase 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

```
[428]: 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= 1000);
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



**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 discussed, 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 hypothesis

```
[398]: mainstream = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES', 'YOUNG SINGLES/\n    ↵COUPLES']")\
    .query("PREMIUM_CUSTOMER == 'Mainstream'")\
    .UNIT_COST
premium_budget = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES', 'YOUNG\n    ↵SINGLES/COUPLES']")\
    .query("PREMIUM_CUSTOMER == ['Budget', 'Premium']")\
    .UNIT_COST
```

```
[399]: 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\n    ↵for mainstream, young and mid-age singles and couples {} significantly\n    ↵higher than that of budget or premium, young and midage singles and couples.\n    ↵'.format(p,different))
    else:
        different = 'ARE NOT'
        print('The t-test results in a p-value of {}, i.e. the unit price\n    ↵for mainstream, young and mid-age singles and couples {} significantly\n    ↵higher than that of budget or premium, young and midage singles and couples.\n    ↵'.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 preferred brands for these two segments ?

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

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

	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	Infnzns	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 preferred brands for Budget & Premium MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES (in descending order)

```
[410]: premium_budget = df.query("LIFESTAGE == ['MIDAGE SINGLES/COUPLES', 'YOUNG SINGLES/COUPLES'])\n        .query("PREMIUM_CUSTOMER == ['Budget', 'Premium'])\nstats2 = pd.DataFrame(premium_budget.PROD_BRAND.value_counts()).reset_index()\n        .rename(columns={'index':\n        'PROD_BRAND', 'PROD_BRAND':'TXN_Counts'})\nstats2
```

	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

```
[402]: print('It can be seen that the top 5 most prefered brands for mainstream MIDAGE\n'
      ↪SINGLES/COUPLES and YOUNG SINGLES/COUPLES (in decreasing order) are: {}.\n'
      ↪While that for Budget,Premium are {}.'\n'
      .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: ['Kettle' 'Pringles' 'Doritos' 'Smiths' 'Thins']. While that for Budget,Premium are ['Kettle' 'Smiths' 'Pringles' 'Doritos' 'RRD Red']

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

```
[403]: print('It can be seen that the top 5 least prefered brands for mainstream\n'
      ↪MIDAGE SINGLES/COUPLES and YOUNG SINGLES/COUPLES (in decreasing order) are:{}.\n'
      ↪While that for Budget,Premium are {}.'\n'
      .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: ['GrnWves' 'Burger' 'Snbts' 'Woolworths' 'Sunbites']. While that for Budget,Premium are ['Sunbites' 'Snbts' 'French' 'NCC' 'GrnWves'].

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.

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

	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

```
[405]: 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'})
```

	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

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

Additonlly, 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 ?

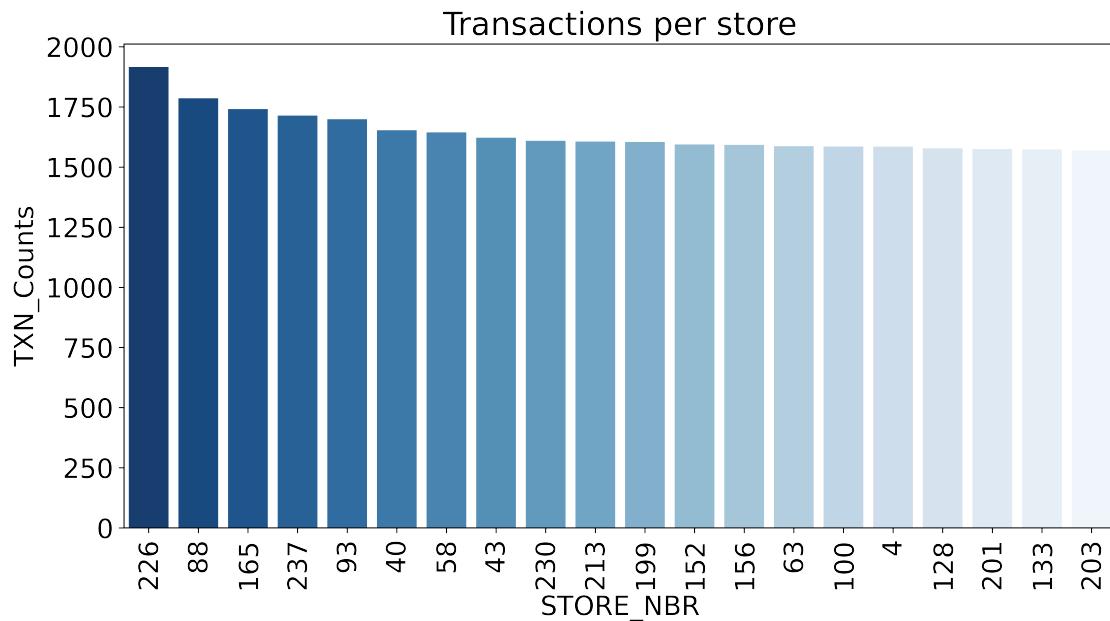
### 4.15.1 Analyse transactions per store (focused on top 20)

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

	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

```
[429]: plt.figure(figsize=(12,6), dpi =1000)
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 applying the strategies to these stores that see the most transactions.

## 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 facilitate better

understanding on the types of customers who purchase chips, their purchasing behavior and to unearth insights 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.