task1

April 14, 2024

Quantium Data Analytics Virtual Experience Program

0.0.1 Task 1

0.1 Data preparation and customer analytics

Conduct analysis on your client's transaction dataset and identify customer purchasing behaviours to generate insights and provide commercial recommendations.

Here is the background information on your task

You are part of Quantium's retail analytics team and have been approached by your client, the Category Manager for Chips, who wants to better understand the types of customers who purchase Chips and their purchasing behaviour within the region.

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

Here is your task

We need to present a strategic recommendation to Julia that is supported by data which she can then use for the upcoming category review however to do so we need to analyse the data to understand the current purchasing trends and behaviours. The client is particularly interested in customer segments and their chip purchasing behaviour. Consider what metrics would help describe the customers' purchasing behaviour.

To get started, download the resource csv data files below and begin performing high level data checks such as:

- Creating and interpreting high level summaries of the data
- Finding outliers and removing these (if applicable)
- Checking data formats and correcting (if applicable)

You will also want to derive extra features such as pack size and brand name from the data and define metrics of interest to enable you to draw insights on who spends on chips and what drives spends for each customer segment. Remember our end goal is to form a strategy based on the findings to provide a clear recommendation to Julia the Category Manager so make sure your insights can have a commercial application.

LIFESTAGE: Customer attribute that identifies whether a customer has a family or not and what point in life they are at e.g. are their children in pre-school/primary/secondary school.

PREMIUM_CUSTOMER: Customer segmentation used to differentiate shoppers by the price point of products they buy and the types of products they buy. It is used to identify whether customers may spend more for quality or brand or whether they will purchase the cheapest options.

[1]:
```
#   Importing the necessary libraries/modules.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import xlrd
%matplotlib inline

#   Ignoring any warnings.

import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

[2]:
```
transaction_data=pd.read_excel("QVI_transaction_data.xlsx") #   Reading the␣
↪Excel Workbook file into a pandas.DataFrame.
transaction_data
```

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

                                        PROD_NAME  PROD_QTY  TOT_SALES
0         Natural Chip        Compny SeaSalt175g         2        6.0
1                       CCs Nacho Cheese    175g         3        6.3
2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
3         Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
…                                           …        …         …
264831   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264832             Tostitos Splash Of  Lime 175g         1        4.4
264833                  Doritos Mexicana    170g         2        8.8
264834   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264835             Tostitos Splash Of  Lime 175g         2        8.8
```

```
[264836 rows x 8 columns]
```

[3]:
```python
transaction_data.info() #   Getting a concise summary of the pandas.DataFrame.
```

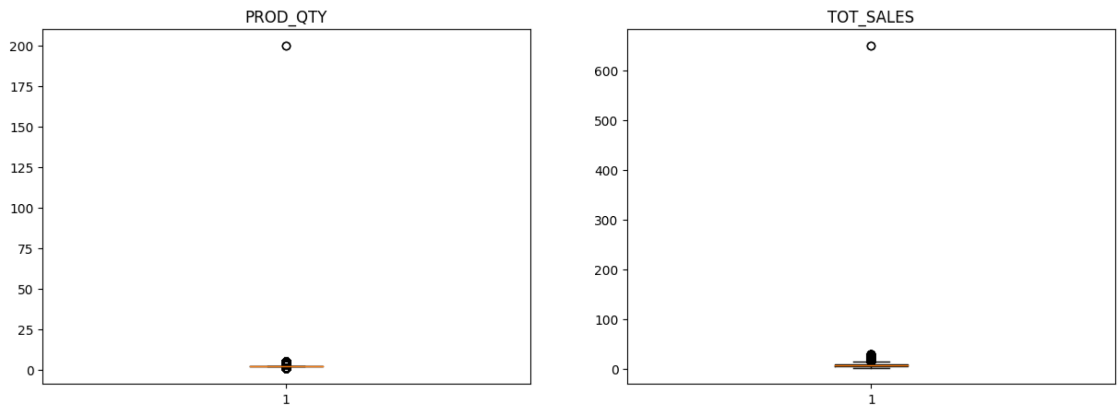
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            264836 non-null  int64
 1   STORE_NBR       264836 non-null  int64
 2   LYLTY_CARD_NBR  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
```

[4]:
```python
transaction_data.isnull().sum() #   Checking for any null values in the pandas.
 ↪DataFrame.
```

[4]:
```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
dtype: int64
```

Of all the columns in the pandas.DataFrame, we know that the outliers are likely (if at all) in the PROD_NAME and PROD_QTY column.

[5]:
```python
#   Checking for any outliers in the pandas.DataFrame using a box plot of the␣
 ↪PROD_QTY and TOT_SALES column.

figure, axis=plt.subplots(1, 2, figsize=(15, 5))
axis[0].boxplot(transaction_data["PROD_QTY"])
axis[1].boxplot(transaction_data["TOT_SALES"])
axis[0].set_title("PROD_QTY")
axis[1].set_title("TOT_SALES")
plt.show()
```



From the visualised box plot, we know that there are outliers present in the PROD_QTY and TOT_SALES column, which we can handle according to their abnormal values to normalise the dataset.

[6]:
```python
#   Removing the outliers from the pandas.DataFrame.

transaction_data=transaction_data[transaction_data["PROD_QTY"]<100]
transaction_data=transaction_data[transaction_data["TOT_SALES"]<500]
transaction_data=transaction_data.reset_index(drop=True)     #   Resetting the index of the pandas.DataFrame.
transaction_data
```

[6]:
```
         DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0       43390          1            1000       1         5
1       43599          1            1307     348        66
2       43605          1            1343     383        61
3       43329          2            2373     974        69
4       43330          2            2426    1038       108
…         …        …              …       …       …
264829  43533        272          272319  270088        89
264830  43325        272          272358  270154        74
264831  43410        272          272379  270187        51
264832  43461        272          272379  270188        42
264833  43365        272          272380  270189        74

                                       PROD_NAME  PROD_QTY  TOT_SALES
0         Natural Chip        Compny SeaSalt175g         2        6.0
1                       CCs Nacho Cheese    175g         3        6.3
2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
3         Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
…                                            …       …        …
264829   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
```

```
264830             Tostitos Splash Of  Lime 175g         1       4.4
264831                  Doritos Mexicana    170g         2       8.8
264832   Doritos Corn Chip Mexican Jalapeno 150g         2       7.8
264833             Tostitos Splash Of  Lime 175g         2       8.8

[264834 rows x 8 columns]
```

As we can see, removing the outliers decreased the pandas.DataFrame down two rows. Of course, this isn't a significant difference, but removing these outliers may allow us to get slightly more accurate analysis results.

Unfortunately, we can also see that the DATE column in the dataset is in the Microsoft Excel serial time format, which is the number of days since the number of days since 1st Janurary 1900, so it's better to convert it to the appropriate datetime format that's more familiar to us.

```
[7]: date=transaction_data["DATE"].tolist()  #   Storing the DATE column as a list.

     #   Converting the Microsoft Excel serial date format to the datetime format.

     for i in range(len(date)):
         date[i]=xlrd.xldate_as_datetime(date[i], 0)

     transaction_data["DATE"]=date   #   Replacing the DATE column with its␣
      ↪corresponding datetime format entries.
     transaction_data
```

```
[7]:               DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
     0       2018-10-17          1            1000       1         5
     1       2019-05-14          1            1307     348        66
     2       2019-05-20          1            1343     383        61
     3       2018-08-17          2            2373     974        69
     4       2018-08-18          2            2426    1038       108
     …              …        …              …       …        …
     264829 2019-03-09        272          272319  270088        89
     264830 2018-08-13        272          272358  270154        74
     264831 2018-11-06        272          272379  270187        51
     264832 2018-12-27        272          272379  270188        42
     264833 2018-09-22        272          272380  270189        74

                                          PROD_NAME  PROD_QTY  TOT_SALES
     0         Natural Chip        Compny SeaSalt175g         2        6.0
     1                       CCs Nacho Cheese    175g         3        6.3
     2         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
     3         Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
     4       Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
     …                                          …       …        …
     264829   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
     264830             Tostitos Splash Of  Lime 175g         1        4.4
```

```
264831                  Doritos Mexicana    170g          2          8.8
264832   Doritos Corn Chip Mexican Jalapeno 150g          2          7.8
264833             Tostitos Splash Of  Lime 175g          2          8.8

[264834 rows x 8 columns]
```

With our transaction dataset in the appropriate format, we can start with cleaning the purchase behaviour dataset.

[8]:
```
purchase_behaviour=pd.read_csv("QVI_purchase_behaviour.csv")    #   Reading the␣
↪CSV file into a pandas.DataFrame.
purchase_behaviour
```

[8]:
```
       LYLTY_CARD_NBR               LIFESTAGE PREMIUM_CUSTOMER
0                1000   YOUNG SINGLES/COUPLES          Premium
1                1002   YOUNG SINGLES/COUPLES       Mainstream
2                1003          YOUNG FAMILIES           Budget
3                1004   OLDER SINGLES/COUPLES       Mainstream
4                1005  MIDAGE SINGLES/COUPLES       Mainstream
…                 …                      …                …
72632         2370651  MIDAGE SINGLES/COUPLES       Mainstream
72633         2370701          YOUNG FAMILIES       Mainstream
72634         2370751          YOUNG FAMILIES          Premium
72635         2370961          OLDER FAMILIES           Budget
72636         2373711   YOUNG SINGLES/COUPLES       Mainstream

[72637 rows x 3 columns]
```

[9]:
```
purchase_behaviour.info()  #   Getting a concise summary of the pandas.
↪DataFrame.
```

```
<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
```

[10]:
```
purchase_behaviour.isnull().sum()   #   Checking for any null values in the␣
↪pandas.DataFrame.
```

[10]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
```

```
dtype: int64
```

Since the dataset seems satisfactory, we can go ahead and merge the two pandas.DataFrames into one on the basis of a factor between them, the LYLTY_CARD_NBR column, for further analysis.

[11]:
```
dataframe=pd.merge(transaction_data, purchase_behaviour, on="LYLTY_CARD_NBR")  
 ↪#   Merging the two pandas.DataFrames into one on the basis of the 
 ↪LYLTY_CARD_NBR column.
dataframe
```

[11]:
```
              DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0       2018-10-17          1            1000       1         5
1       2019-05-14          1            1307     348        66
2       2018-11-10          1            1307     346        96
3       2019-03-09          1            1307     347        54
4       2019-05-20          1            1343     383        61
…              …          …              …       …         …
264829 2019-03-09        272          272319  270088        89
264830 2018-08-13        272          272358  270154        74
264831 2018-11-06        272          272379  270187        51
264832 2018-12-27        272          272379  270188        42
264833 2018-09-22        272          272380  270189        74

                                       PROD_NAME  PROD_QTY  TOT_SALES  \
0         Natural Chip        Compny SeaSalt175g         2        6.0
1                       CCs Nacho Cheese    175g         3        6.3
2               WW Original Stacked Chips 160g         2        3.8
3                              CCs Original 175g         1        2.1
4         Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
…                                            …       …         …
264829   Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264830              Tostitos Splash Of  Lime 175g         1        4.4
264831                  Doritos Mexicana    170g         2        8.8
264832   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264833              Tostitos Splash Of  Lime 175g         2        8.8

                    LIFESTAGE PREMIUM_CUSTOMER
0         YOUNG SINGLES/COUPLES          Premium
1        MIDAGE SINGLES/COUPLES           Budget
2        MIDAGE SINGLES/COUPLES           Budget
3        MIDAGE SINGLES/COUPLES           Budget
4        MIDAGE SINGLES/COUPLES           Budget
…                          …              …
264829    YOUNG SINGLES/COUPLES          Premium
264830    YOUNG SINGLES/COUPLES          Premium
264831    YOUNG SINGLES/COUPLES          Premium
264832    YOUNG SINGLES/COUPLES          Premium
```

```
264833    YOUNG SINGLES/COUPLES          Premium

[264834 rows x 10 columns]
```

Let's find the total number of distinct products the customers purchased over the recorded duration before moving forward.

```
[12]: unique_products=list(dataframe["PROD_NAME"].unique())    #   Storing the␣
       ↪distinct products from the pandas.DataFrame into a list.
      print("Total Distinct Products:", len(unique_products))
```

```
Total Distinct Products: 114
```

As we can see, these 114 products aren't all entirely from different brands, as some of them are from the same brand but have different package sizes. Hence, it'd be useful for us to separate the product names from the package sizes into separate columns in the pandas.DataFrame, along with the corresponding brand names.

```
[13]: dataframe["PROD_NAME_CLEAN"]=dataframe["PROD_NAME"].str.replace("\d+g", "") #  ␣
       ↪Removing the package sizes from the product names, and storing them in a␣
       ↪separate column.
      dataframe["PROD_SIZE"]=dataframe["PROD_NAME"].str.extract("(\d+)")    #  ␣
       ↪Extracting the package sizes from the product names, and storing them in a␣
       ↪separate column.
      dataframe["PROD_NAME"]=dataframe["PROD_NAME_CLEAN"] #   Assigning the␣
       ↪PROD_NAME_CLEAN column to the PROD_NAME column.
      dataframe=dataframe.drop("PROD_NAME_CLEAN", axis=1) #   Dropping the␣
       ↪PROD_NAME_CLEAN column from the pandas.DataFrame.
      dataframe["BRAND_NAME"]=dataframe["PROD_NAME"].str.split().str[0]    #  ␣
       ↪Extracting the brand names from the product names, and storing them in a␣
       ↪separate column.
      dataframe=dataframe.loc[:, ["DATE", "STORE_NBR", "LYLTY_CARD_NBR", "TXN_ID",␣
       ↪"PROD_NBR", "PROD_NAME", "PROD_SIZE", "BRAND_NAME", "PROD_QTY", "TOT_SALES",␣
       ↪"LIFESTAGE", "PREMIUM_CUSTOMER"]]    #   Rearranging the columns of the␣
       ↪pandas.DataFrame.
      dataframe
```

```
[13]:               DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
      0       2018-10-17          1            1000       1         5
      1       2019-05-14          1            1307     348        66
      2       2018-11-10          1            1307     346        96
      3       2019-03-09          1            1307     347        54
      4       2019-05-20          1            1343     383        61
      …              …          …               …       …         …
      264829  2019-03-09        272          272319  270088        89
      264830  2018-08-13        272          272358  270154        74
      264831  2018-11-06        272          272379  270187        51
      264832  2018-12-27        272          272379  270188        42
```

```
264833 2018-09-22        272          272380  270189        74

                                  PROD_NAME PROD_SIZE BRAND_NAME  PROD_QTY  \
0        Natural Chip        Compny SeaSalt       175    Natural         2
1                      CCs Nacho Cheese           175        CCs         3
2                WW Original Stacked Chips        160         WW         2
3                           CCs Original          175        CCs         1
4         Smiths Crinkle Cut  Chips Chicken       170     Smiths         2
…                                        …        …          …         …
264829  Kettle Sweet Chilli And Sour Cream        175     Kettle         2
264830            Tostitos Splash Of  Lime        175   Tostitos         1
264831                 Doritos Mexicana           170    Doritos         2
264832  Doritos Corn Chip Mexican Jalapeno        150    Doritos         2
264833            Tostitos Splash Of  Lime        175   Tostitos         2

        TOT_SALES              LIFESTAGE PREMIUM_CUSTOMER
0             6.0   YOUNG SINGLES/COUPLES          Premium
1             6.3  MIDAGE SINGLES/COUPLES           Budget
2             3.8  MIDAGE SINGLES/COUPLES           Budget
3             2.1  MIDAGE SINGLES/COUPLES           Budget
4             2.9  MIDAGE SINGLES/COUPLES           Budget
…             …                      …              …
264829       10.8   YOUNG SINGLES/COUPLES          Premium
264830        4.4   YOUNG SINGLES/COUPLES          Premium
264831        8.8   YOUNG SINGLES/COUPLES          Premium
264832        7.8   YOUNG SINGLES/COUPLES          Premium
264833        8.8   YOUNG SINGLES/COUPLES          Premium

[264834 rows x 12 columns]
```

[14]:
```
dataframe.isnull().sum()    #   Checking for any null values in the pandas.
↪DataFrame.
```

[14]:
```
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_SIZE           0
BRAND_NAME          0
PROD_QTY            0
TOT_SALES           0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

[15]:

```python
dataframe=dataframe.sort_values(by="DATE")  #   Sorting the pandas.DataFrame in␣
 ↪ascending order of the DATE column.
dataframe=dataframe.reset_index(drop=True)  #   Resetting the index of the␣
 ↪pandas.DataFrame.
dataframe
```

[15]:

```
              DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0       2018-07-01         27           27181   24218        70
1       2018-07-01        191          191099  192367       103
2       2018-07-01        257          257010  255769        24
3       2018-07-01         48           48129   43842       114
4       2018-07-01        203          203013  202339        23
…              …          …              …       …         …
264829 2019-06-30         67           67129   64592        57
264830 2019-06-30        133          133121  136776        44
264831 2019-06-30        257          257195  256935        83
264832 2019-06-30         45           45057   40739        91
264833 2019-06-30        199          199122  198088        42

                                      PROD_NAME PROD_SIZE BRAND_NAME  PROD_QTY  \
0        Tyrrells Crisps     Lightly Salted        165   Tyrrells         2
1           RRD Steak &          Chimuchurri        150        RRD         2
2          Grain Waves          Sweet Chilli        210      Grain         2
3          Kettle Sensations   Siracha Lime        150     Kettle         2
4                            Cheezels Cheese        330   Cheezels         2
…                                          …         …          …         …
264829  Old El Paso Salsa   Dip Tomato Mild        300        Old         2
264830            Thins Chips Light&  Tangy        175      Thins         2
264831         WW D/Style Chip      Sea Salt        200         WW         2
264832                     CCs Tasty Cheese        175        CCs         2
264833   Doritos Corn Chip Mexican Jalapeno        150    Doritos         2

        TOT_SALES               LIFESTAGE PREMIUM_CUSTOMER
0             8.4                RETIREES           Budget
1             5.4          YOUNG FAMILIES           Budget
2             7.2          YOUNG FAMILIES          Premium
3             9.2          OLDER FAMILIES       Mainstream
4            11.4  MIDAGE SINGLES/COUPLES           Budget
…               …                       …                …
264829       10.2   OLDER SINGLES/COUPLES       Mainstream
264830        6.6          OLDER FAMILIES       Mainstream
264831        3.8          YOUNG FAMILIES       Mainstream
264832        4.2   OLDER SINGLES/COUPLES          Premium
264833        7.8            NEW FAMILIES          Premium

[264834 rows x 12 columns]
```

Now that we have the pandas.DataFrame sorted according to the date, we can analyse the dataset with regards to the change over time. However, before we can do that, we need to make sure that the DATE column contains no missing values for unrecorded dates.

[16]:
```
pd.date_range(start="2018-07-01", end="2019-06-30").
 ↪difference(dataframe["DATE"])    #   Checking for any missing dates in the␣
 ↪pandas.DataFrame.
```

[16]:
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

As suspected, there *is* one unrecorded date and that's for Christmas Day, since most stores are closed during that time. Hence, we can fill in the value for this as having zero sales on the date.

[17]:
```
#   Adding an entry for the missing date in the pandas.DataFrame.

dataframe=dataframe.append({"DATE": pd.to_datetime("2018-12-25"), "STORE_NBR":␣
 ↪0, "LYLTY_CARD_NBR": 0, "TXN_ID": 0, "PROD_NBR": 0, "PROD_NAME": "None",␣
 ↪"BRAND_NAME": "None", "PROD_SIZE": "0g", "PROD_QTY": 0, "TOT_SALES": 0,␣
 ↪"LIFESTAGE": "None", "PREMIUM_CUSTOMER": "None"}, ignore_index=True)
dataframe=dataframe.sort_values(by="DATE")  #   Sorting the pandas.DataFrame in␣
 ↪ascending order of the DATE column.
dataframe=dataframe.reset_index(drop=True)  #   Resetting the index of the␣
 ↪pandas.DataFrame.
dataframe
```

[17]:
```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0      2018-07-01         27           27181   24218        70
1      2018-07-01        180          180179  182143        46
2      2018-07-01        164          164069  164212        56
3      2018-07-01        179          179216  180709        24
4      2018-07-01         18           18221   15451        80
…             …        …              …       …       …
264830 2019-06-30        230          230022  232028        77
264831 2019-06-30        101          101071  100462        12
264832 2019-06-30        141          141226  142472        47
264833 2019-06-30        162          162118  162544        42
264834 2019-06-30         27           27288   24377        25

                                  PROD_NAME PROD_SIZE BRAND_NAME  PROD_QTY  \
0       Tyrrells Crisps     Lightly Salted       165   Tyrrells         2
1                          Kettle Original       175     Kettle         2
2                      Cheezels Cheese Box       125   Cheezels         2
3         Grain Waves         Sweet Chilli       210      Grain         2
4        Natural ChipCo Sea  Salt & Vinegr       175    Natural         1
…                                        …         …          …       …
264830    Doritos Corn Chips  Nacho Cheese       170    Doritos         2
264831  Natural Chip Co      Tmato Hrb&Spce       175    Natural         2
264832        Doritos Corn Chips  Original       170    Doritos         2
```

```
264833  Doritos Corn Chip Mexican Jalapeno        150    Doritos         2
264834           Pringles SourCream  Onion        134   Pringles         2

        TOT_SALES              LIFESTAGE PREMIUM_CUSTOMER
0             8.4               RETIREES           Budget
1            10.8               RETIREES          Premium
2             4.2         OLDER FAMILIES          Premium
3             7.2               RETIREES          Premium
4             3.0  MIDAGE SINGLES/COUPLES          Premium
…             …                     …              …
264830        8.8   YOUNG SINGLES/COUPLES       Mainstream
264831        6.0   YOUNG SINGLES/COUPLES       Mainstream
264832        8.8   YOUNG SINGLES/COUPLES       Mainstream
264833        7.8  MIDAGE SINGLES/COUPLES       Mainstream
264834        7.4   YOUNG SINGLES/COUPLES           Budget

[264835 rows x 12 columns]
```

[18]:

```python
dataframe.loc[dataframe["DATE"]=="2018-12-25"]  #   Checking if the missing␣
 ↪date has been added to the pandas.DataFrame.
```

[18]:

```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR PROD_NAME  \
129324 2018-12-25          0               0       0         0      None

       PROD_SIZE BRAND_NAME  PROD_QTY  TOT_SALES LIFESTAGE PREMIUM_CUSTOMER
129324        0g       None         0        0.0      None             None
```

With our missing date sorted, let's start by visualising the change in total sales over the entire recorded duration.

[19]:

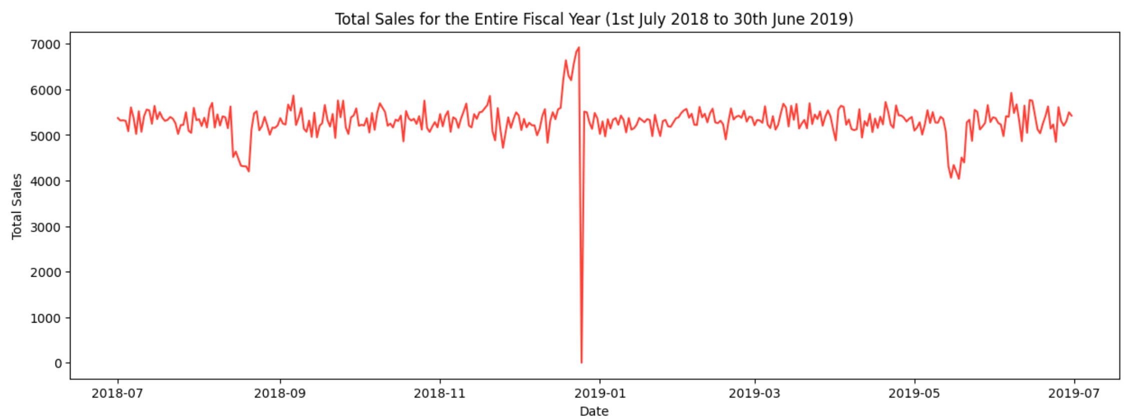
```python
date_sales=dataframe.groupby("DATE")["TOT_SALES"].sum().reset_index()   #  ␣
 ↪Grouping the pandas.DataFrame by the TOT_SALES column for each recorded date.
date_sales
```

[19]:

```
           DATE  TOT_SALES
0    2018-07-01     5372.2
1    2018-07-02     5315.4
2    2018-07-03     5321.8
3    2018-07-04     5309.9
4    2018-07-05     5080.9
..          …         …
360  2019-06-26     5305.0
361  2019-06-27     5202.8
362  2019-06-28     5299.6
363  2019-06-29     5497.6
364  2019-06-30     5423.4
```

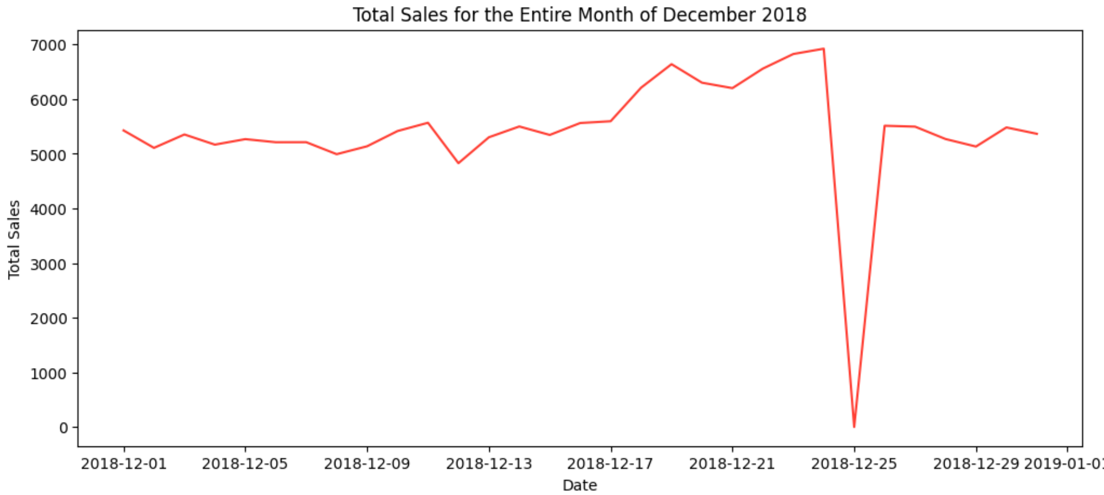
```
[365 rows x 2 columns]
```

[20]:
```
#   Plotting a line graph of the total sales for each date over the entire
↪recorded duration.

plt.figure(figsize=(15, 5))
plt.plot(date_sales["DATE"], date_sales["TOT_SALES"], color="#ff3f34")
plt.title("Total Sales for the Entire Fiscal Year (1st July 2018 to 30th June
↪2019)")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.show()
```



As we can see from the line graph, the sales drop to zero on a certain date, which is 25th December 2018 — Christmas Day, which we manually set to zero. However, the sales also reached an all-time high right before that, so we would need to analyse the transaction data for December 2018 to find out more about the sales.

[21]:
```
#   Plotting a line graph of the total sales for each recorded date during
↪December 2018.

plt.figure(figsize=(12, 5))
plt.plot(date_sales["DATE"][date_sales["DATE"].dt.month==12],
↪date_sales["TOT_SALES"][date_sales["DATE"].dt.month==12], color="#ff3f34")
plt.title("Total Sales for the Entire Month of December 2018")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.show()
```



As suspected, the sales reached an all-time high the day before Christmas Day, which makes sense because people tend to purchase food items more when approaching holiday season. We can also see a consistent rise in the line graph between 21st December and 24th December, which means that these are the dates the store could target with promotions and discounts to increase the sales even more.

If the store *does* want to target these dates, it would be important to know which package sizes sell the most to create promotions and discounts around them.

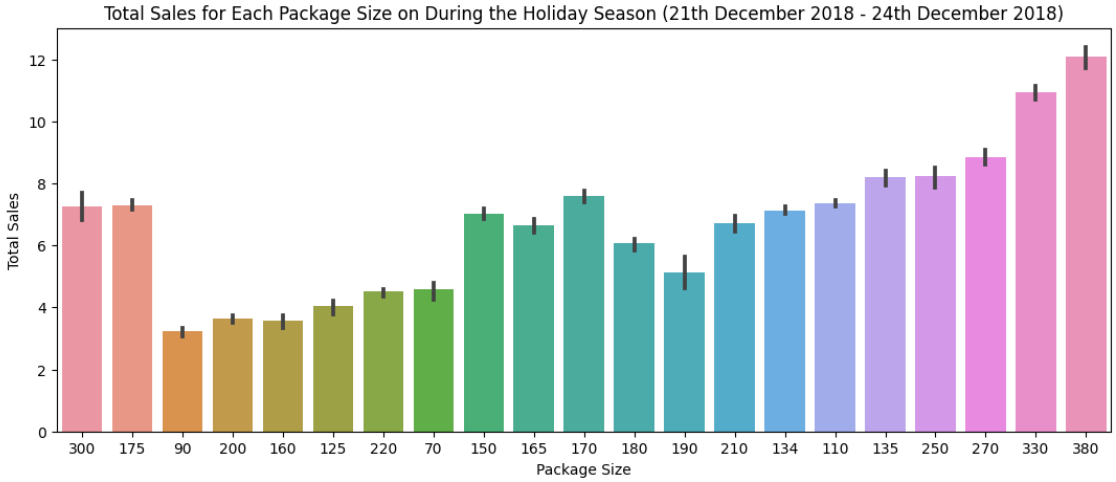
[22]:
```
holiday_sales=dataframe[(dataframe["DATE"]>="2018-12-21") &␣
↪(dataframe["DATE"]<="2018-12-24")]  #   Extracting entries for the dates␣
↪between 21st December 2018 and 24th December 2018.
holiday_sales=holiday_sales.sort_values(by="TOT_SALES") #   Sorting the pandas.
↪DataFrame in ascending order of the TOT_SALES column.
holiday_sales=holiday_sales.reset_index(drop=True)  #   Resetting the index of␣
↪the pandas.DataFrame.
holiday_sales
```

[22]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	2018-12-24	38	38005	34012	35	
1	2018-12-22	127	127448	130458	76	
2	2018-12-22	136	136114	138499	35	
3	2018-12-23	255	255077	254619	76	
4	2018-12-22	186	186218	188613	76	
…	…	…	…	…	…	
3608	2018-12-24	40	40152	36819	4	
3609	2018-12-24	217	217332	217772	4	
3610	2018-12-23	238	238351	243296	4	
3611	2018-12-24	3	3270	2289	4	
3612	2018-12-21	250	250213	252361	4	

```
                          PROD_NAME PROD_SIZE  BRAND_NAME  PROD_QTY  TOT_SALES  \
0        Woolworths Mild      Salsa        300  Woolworths         1        1.5
1        Woolworths Medium    Salsa        300  Woolworths         1        1.5
2        Woolworths Mild      Salsa        300  Woolworths         1        1.5
3        Woolworths Medium    Salsa        300  Woolworths         1        1.5
4        Woolworths Medium    Salsa        300  Woolworths         1        1.5
…                               …         …           …       …        …
3608  Dorito Corn Chp      Supreme        380      Dorito         2       13.0
3609  Dorito Corn Chp      Supreme        380      Dorito         2       13.0
3610  Dorito Corn Chp      Supreme        380      Dorito         2       13.0
3611  Dorito Corn Chp      Supreme        380      Dorito         2       13.0
3612  Dorito Corn Chp      Supreme        380      Dorito         2       13.0

                   LIFESTAGE PREMIUM_CUSTOMER
0             OLDER FAMILIES           Budget
1             OLDER FAMILIES       Mainstream
2      OLDER SINGLES/COUPLES       Mainstream
3             YOUNG FAMILIES           Budget
4                   RETIREES       Mainstream
…                        …              …
3608  OLDER SINGLES/COUPLES       Mainstream
3609         YOUNG FAMILIES       Mainstream
3610         YOUNG FAMILIES           Budget
3611         YOUNG FAMILIES           Budget
3612  YOUNG SINGLES/COUPLES       Mainstream

[3613 rows x 12 columns]
```

[23]:
```
#   Plotting a bar graph of the total sales for each package size between 21st␣
↪December 2018 and 24th December 2018.

plt.figure(figsize=(13, 5))
sns.barplot(x="PROD_SIZE", y="TOT_SALES", data=holiday_sales)
plt.title("Total Sales for Each Package Size on During the Holiday Season (21th␣
↪December 2018 - 24th December 2018)")
plt.xlabel("Package Size")
plt.ylabel("Total Sales")
plt.show()
```



It seems like customers mostly purchased the 380 gramme package size (the largest one in the store) when approaching the holiday season.

Additionally, we can also find the brands that sold the most during the particular dates for brand-specific campaigns.

[24]:
```
holiday_brands=holiday_sales.groupby("BRAND_NAME")["TOT_SALES"].sum().
↪reset_index().sort_values(by="TOT_SALES", ascending=False).head(5)    # ␣
↪Grouping the pandas.DataFrame by the TOT_SALES column for each brand, and␣
↪sorting it in descending order of the TOT_SALES column.
holiday_brands=holiday_brands.reset_index(drop=True)    #   Resetting the index␣
↪of the pandas.DataFrame.
holiday_brands
```

[24]:

	BRAND_NAME	TOT_SALES
0	Kettle	4940.0
1	Doritos	2948.5
2	Smiths	2914.5
3	Pringles	2290.3
4	Thins	1343.1

We can see that KETTLE® was the highest-selling brand during the holiday season, so it'd be wise to surround promotions and discounts around it to drive sales even more.

Let's see if our holiday season statistics match with the ones during the entire duration of the recorded sales.

[25]:
```
package_sales=dataframe.groupby("PROD_SIZE")["TOT_SALES"].sum().reset_index().
↪sort_values(by="TOT_SALES", ascending=False)    #   Grouping the pandas.
↪DataFrame by the TOT_SALES column for each package size, and sorting it in␣
↪descending order of the TOT_SALES column.
```

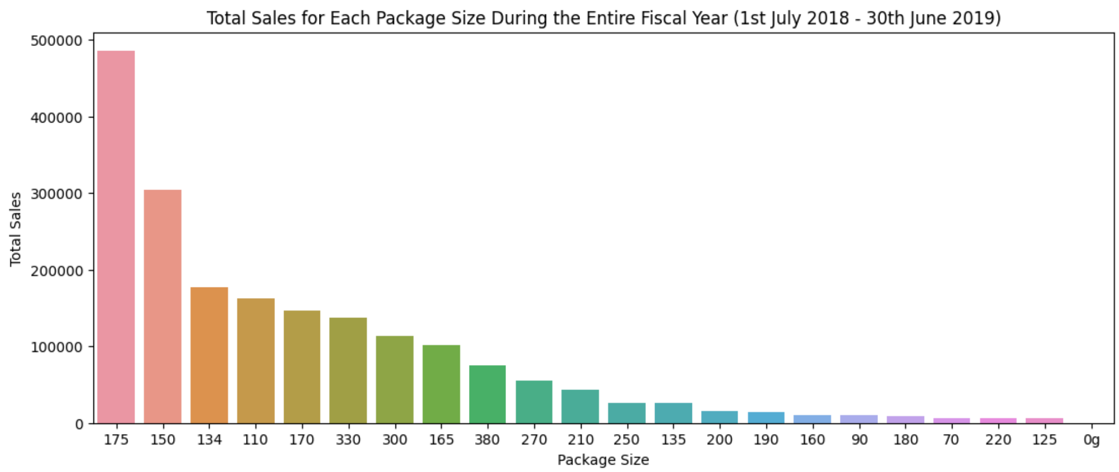
```
package_sales=package_sales.reset_index(drop=True)     #   Resetting the index␣
 ↪of the pandas.DataFrame.
package_sales
```

[25]:

	PROD_SIZE	TOT_SALES
0	175	485437.4
1	150	304288.5
2	134	177655.5
3	110	162765.4
4	170	146673.0
5	330	136794.3
6	300	113330.6
7	165	101360.6
8	380	75419.6
9	270	55425.4
10	210	43048.8
11	250	26096.7
12	135	26090.4
13	200	16007.5
14	190	14412.9
15	160	10647.6
16	90	9676.4
17	180	8568.4
18	70	6852.0
19	220	6831.0
20	125	5733.0
21	0g	0.0

[26]:

```
#   Plotting a bar graph of the total sales for each package size during the␣
 ↪entire recorded duration.

plt.figure(figsize=(13, 5))
sns.barplot(x="PROD_SIZE", y="TOT_SALES", data=package_sales)
plt.title("Total Sales for Each Package Size During the Entire Fiscal Year (1st␣
 ↪July 2018 - 30th June 2019)")
plt.xlabel("Package Size")
plt.ylabel("Total Sales")
plt.show()
```



As we can see, the 175 gramme package size was the highest-selling one over the entire duration of the recorded sales, and even that by nearly 37% from the second highest-selling package size. Hence, it's clear that the 175 gramme package size is a customer favourite!

Likewise, we can also check for the highest-selling brands during the entire duration of the recorded sales.

[27]:
```
brands_sales=dataframe.groupby("BRAND_NAME")["TOT_SALES"].sum().reset_index().
  ↪sort_values(by="TOT_SALES", ascending=False).head(5)  #   Grouping the␣
  ↪pandas.DataFrame by the TOT_SALES column for the top five brands, and␣
  ↪sorting it in descending order of the TOT_SALES column.
brands_sales=brands_sales.reset_index(drop=True)     #   Resetting the index of␣
  ↪the pandas.DataFrame.
brands_sales
```

[27]:

	BRAND_NAME	TOT_SALES
0	Kettle	390239.8
1	Smiths	210076.8
2	Doritos	201538.9
3	Pringles	177655.5
4	Old	90785.1

Just like the holiday season sales, KETTLE® remained the highest-selling brand during the entire duration of the recorded sales.

With the brand and product analysis done, we can move onto the customer analysis now. The first part would be analyse which sort of customers are the most loyal to the store, which would also be the ones that have the most purchases from it.

[28]:
```
dataframe["LIFESTAGE"].value_counts()    #   Finding the number of entries for␣
  ↪each entry in the LIFESTAGE column in the pandas.DataFrame.
```

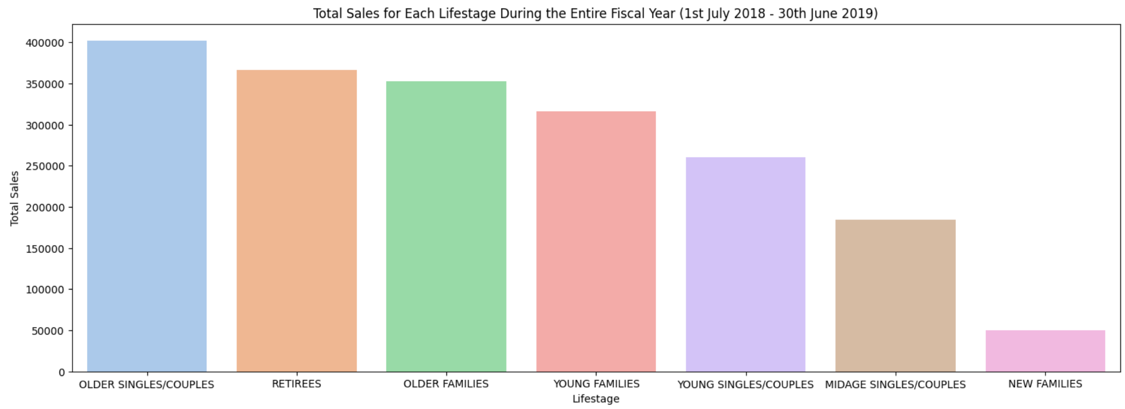
[28]:
```
OLDER SINGLES/COUPLES     54479
RETIREES                  49763
OLDER FAMILIES            48594
YOUNG FAMILIES            43592
YOUNG SINGLES/COUPLES     36377
MIDAGE SINGLES/COUPLES    25110
NEW FAMILIES               6919
None                          1
Name: LIFESTAGE, dtype: int64
```

[29]:
```
customer_sales=dataframe.groupby("LIFESTAGE")["TOT_SALES"].sum().reset_index().
  ↪sort_values(by="TOT_SALES", ascending=False).head(7) #   Grouping the pandas.
  ↪DataFrame by the TOT_SALES column for each lifestage, and sorting it in␣
  ↪descending order of the TOT_SALES column.
customer_sales=customer_sales.reset_index(drop=True)     #   Resetting the index␣
  ↪of the pandas.DataFrame.
customer_sales
```

[29]:

	LIFESTAGE	TOT_SALES
0	OLDER SINGLES/COUPLES	402426.75
1	RETIREES	366470.90
2	OLDER FAMILIES	352467.20
3	YOUNG FAMILIES	316160.10
4	YOUNG SINGLES/COUPLES	260405.30
5	MIDAGE SINGLES/COUPLES	184751.30
6	NEW FAMILIES	50433.45

[30]:
```
#   Plotting a bar graph of the total sales for each lifestage during the␣
  ↪entire recorded duration.

plt.figure(figsize=(18, 6))
sns.barplot(x="LIFESTAGE", y="TOT_SALES", data=customer_sales, palette="pastel")
plt.title("Total Sales for Each Lifestage During the Entire Fiscal Year (1st␣
  ↪July 2018 - 30th June 2019)")
plt.xlabel("Lifestage")
plt.ylabel("Total Sales")
plt.show()
```



It seems like OLDER SINGLES/COUPLES are the most loyal customers of the store and NEW FAMILIES are the least. Interestingly, we can see a decreasing trend of purchases according to age in the first half of the bar graph, with customers that are the most likely to spend the most time at home also having the most purchases, even though snack items wouldn't logically be associated with an age demographic.

[31]:
```
lifestage_sales=dataframe.groupby(["LIFESTAGE", "DATE"])["TOT_SALES"].sum().
↪reset_index()  #   Grouping the pandas.DataFrame by the TOT_SALES column for␣
↪each lifestage and date.
lifestage_sales=lifestage_sales[lifestage_sales["LIFESTAGE"]!="None"]   #  ␣
↪Removing the entries with the lifestage "None" from the pandas.DataFrame.
lifestage_sales
```

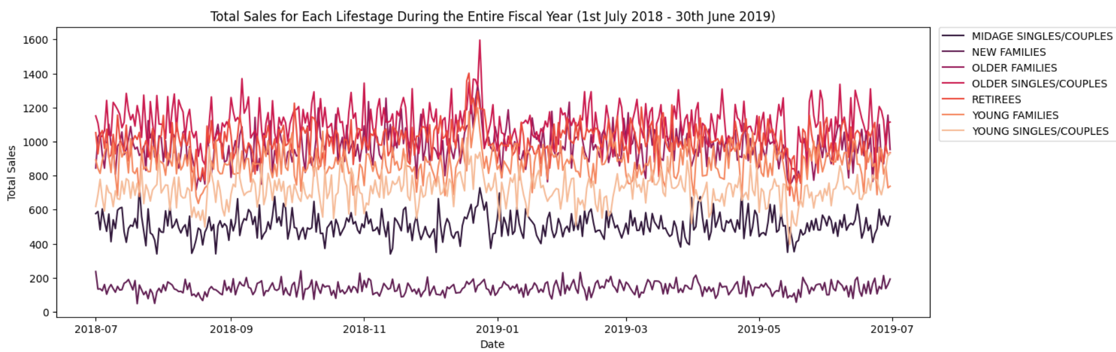
[31]:

	LIFESTAGE	DATE	TOT_SALES
0	MIDAGE SINGLES/COUPLES	2018-07-01	576.8
1	MIDAGE SINGLES/COUPLES	2018-07-02	589.5
2	MIDAGE SINGLES/COUPLES	2018-07-03	482.2
3	MIDAGE SINGLES/COUPLES	2018-07-04	604.5
4	MIDAGE SINGLES/COUPLES	2018-07-05	531.6
…	…	…	…
2544	YOUNG SINGLES/COUPLES	2019-06-26	687.4
2545	YOUNG SINGLES/COUPLES	2019-06-27	743.4
2546	YOUNG SINGLES/COUPLES	2019-06-28	840.7
2547	YOUNG SINGLES/COUPLES	2019-06-29	924.5
2548	YOUNG SINGLES/COUPLES	2019-06-30	929.9

[2548 rows x 3 columns]

[32]:
```
#   Plotting a multi-line graph of the total sales for each lifestage during␣
↪the entire recorded duration.

plt.figure(figsize=(15, 5))
```

```
sns.lineplot(x="DATE", y="TOT_SALES", hue="LIFESTAGE", data=lifestage_sales, palette="rocket")
plt.title("Total Sales for Each Lifestage During the Entire Fiscal Year (1st July 2018 - 30th June 2019)")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.legend(bbox_to_anchor=(1.01, 1), loc=2, borderaxespad=0.)
plt.show()
```



Like the holiday season statistics, we can see an increase in sales right before Christmas Day for all age demographics, except NEW FAMILIES, which remains consistent throughout the entire recorded duration. As new families are more inclined toward their careers and developing their newly established home, it's unlikely for them to spend on snack items frequently.

Let's see what sort of purchase behaviour each age demographic has!

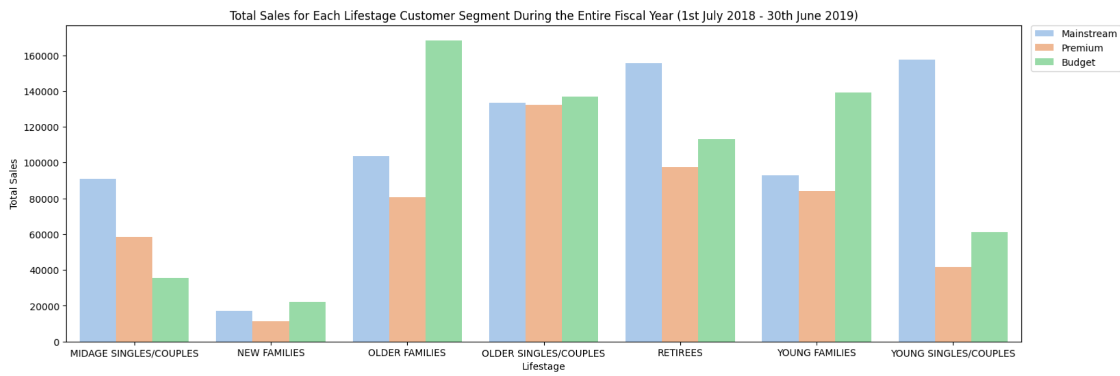
[33]:
```
lifestage_segment=dataframe.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["TOT_SALES"].sum().reset_index().sort_values(by=["LIFESTAGE", "TOT_SALES"], ascending=[True, False]) # Grouping the pandas.DataFrame by the TOT_SALES column for each lifestage and customer segment, and sorting it in descending order of the TOT_SALES column.
lifestage_segment=lifestage_segment[lifestage_segment["LIFESTAGE"]!="None"] # Removing the entries with the lifestage "None" from the pandas.DataFrame.
lifestage_segment=lifestage_segment.reset_index(drop=True)  #   Resetting the index of the pandas.DataFrame.
lifestage_segment
```

[33]:

	LIFESTAGE	PREMIUM_CUSTOMER	TOT_SALES
0	MIDAGE SINGLES/COUPLES	Mainstream	90803.85
1	MIDAGE SINGLES/COUPLES	Premium	58432.65
2	MIDAGE SINGLES/COUPLES	Budget	35514.80
3	NEW FAMILIES	Budget	21928.45
4	NEW FAMILIES	Mainstream	17013.90
5	NEW FAMILIES	Premium	11491.10
6	OLDER FAMILIES	Budget	168363.25

```
7           OLDER FAMILIES      Mainstream  103445.55
8           OLDER FAMILIES         Premium   80658.40
9    OLDER SINGLES/COUPLES          Budget  136769.80
10   OLDER SINGLES/COUPLES      Mainstream  133393.80
11   OLDER SINGLES/COUPLES         Premium  132263.15
12                RETIREES      Mainstream  155677.05
13                RETIREES          Budget  113147.80
14                RETIREES         Premium   97646.05
15          YOUNG FAMILIES          Budget  139345.85
16          YOUNG FAMILIES      Mainstream   92788.75
17          YOUNG FAMILIES         Premium   84025.50
18   YOUNG SINGLES/COUPLES      Mainstream  157621.60
19   YOUNG SINGLES/COUPLES          Budget   61141.60
20   YOUNG SINGLES/COUPLES         Premium   41642.10
```

[34]:
```python
#   Plotting a bar graph of the total sales for each lifestage and whether it is a Premium, Mainstream, or Budget customer during the entire recorded duration.

plt.figure(figsize=(18, 6))
sns.barplot(x="LIFESTAGE", y="TOT_SALES", hue="PREMIUM_CUSTOMER", data=lifestage_segment, palette="pastel")
plt.title("Total Sales for Each Lifestage Customer Segment During the Entire Fiscal Year (1st July 2018 - 30th June 2019)")
plt.xlabel("Lifestage")
plt.ylabel("Total Sales")
plt.legend(bbox_to_anchor=(1.01, 1), loc=2, borderaxespad=0.)
plt.show()
```



We can see that MIDAGE SINGLES/COUPLES had the highest Mainstream and Premium purchases of all their purchases, while all the others had the least Premium purchases, which means that *this* would be the age demographic to target for payment plans and promotions to drive sales even more since they're more likely to pay more per packet of chips than the others.

Now that we know which age demographic the store should target to drive sales more, let's find which brand and package size each customer segment for MIDAGE SINGLES/COUPLES is the most inclined to.

[35]:
```
lifestage_brands=dataframe.groupby(["LIFESTAGE", "BRAND_NAME",␣
↪"PREMIUM_CUSTOMER", "PROD_SIZE"])["TOT_SALES"].sum().reset_index().
↪sort_values(by=["LIFESTAGE", "TOT_SALES"], ascending=[True, False])  #  ␣
↪Grouping the pandas.DataFrame by the TOT_SALES column for each lifestage and␣
↪brand, and sorting it in descending order of the TOT_SALES column.
lifestage_brands=lifestage_brands[lifestage_brands["LIFESTAGE"]!="None"]    #  ␣
↪Removing the entries with the lifestage "None" from the pandas.DataFrame.
lifestage_brands=lifestage_brands.reset_index(drop=True)    #   Resetting the␣
↪index of the pandas.DataFrame.
midage=lifestage_brands[lifestage_brands["LIFESTAGE"]=="MIDAGE SINGLES/
↪COUPLES"]    #   Storing the entries for MIDAGE SINGLES/COUPLES into a␣
↪separate pandas.DataFrame.
midage
```

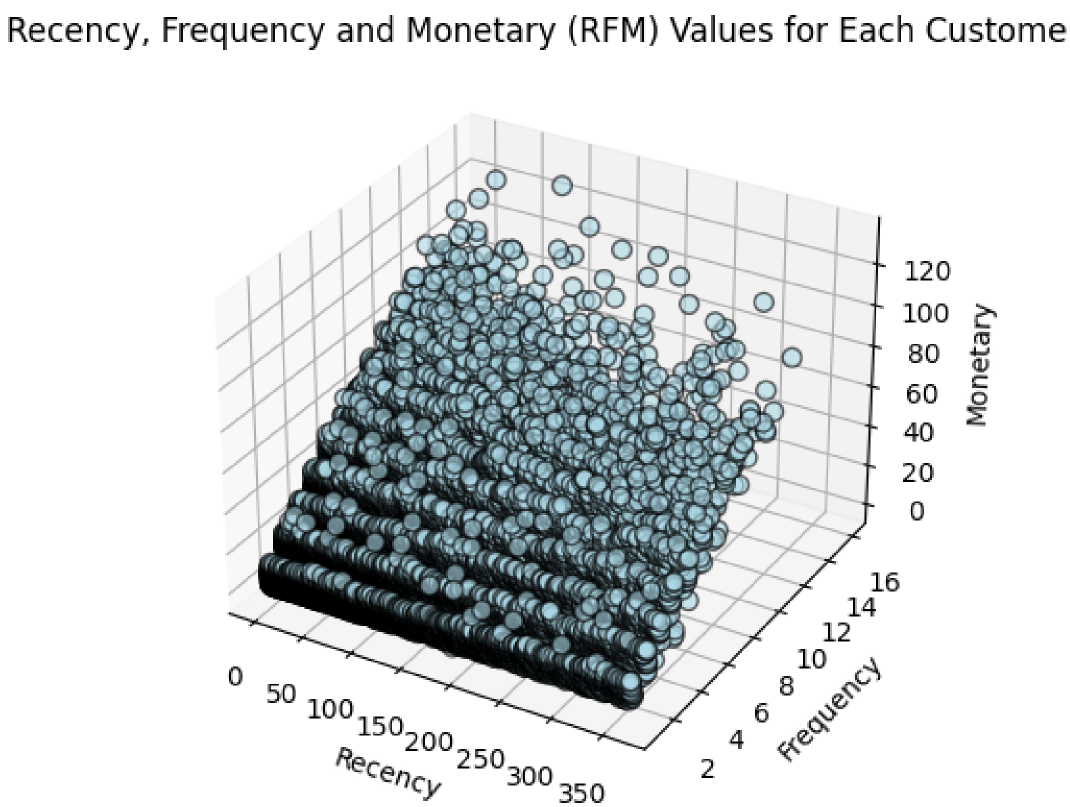
[35]:
```
                 LIFESTAGE  BRAND_NAME PREMIUM_CUSTOMER PROD_SIZE  TOT_SALES
0    MIDAGE SINGLES/COUPLES      Kettle       Mainstream       175    10557.0
1    MIDAGE SINGLES/COUPLES      Kettle       Mainstream       150     8381.2
2    MIDAGE SINGLES/COUPLES    Pringles       Mainstream       134     8177.0
3    MIDAGE SINGLES/COUPLES      Kettle          Premium       175     5815.8
4    MIDAGE SINGLES/COUPLES    Pringles          Premium       134     5538.9
..                      …          …                …         …          …
133  MIDAGE SINGLES/COUPLES       Snbts       Mainstream        90      120.7
134  MIDAGE SINGLES/COUPLES    Cheezels           Budget       125      105.0
135  MIDAGE SINGLES/COUPLES    Sunbites       Mainstream        90      103.7
136  MIDAGE SINGLES/COUPLES    Sunbites           Budget        90       96.9
137  MIDAGE SINGLES/COUPLES  Woolworths           Budget       190       81.0

[138 rows x 5 columns]
```

With this, we can see that MIDAGE SINGLES/COUPLES prefer KETTLE® and 175 gramme package size the most in both the Mainstream and Premium customer segment.

- **Recency, Frequency and Monetary (RFM) Analysis:** Recency, Frequency and Monetary (RFM) analysis is a marketing technique used to quantitatively rank and group customers based on the recency, frequency and monetary total of their recent transactions to identify the best customers and perform targeted marketing campaigns. This can help us identify customers who are most valuable to the store, as well as those who may be at risk of churning.

[36]:
```
#   Creating a new pandas.DataFrame with the Recency, Frequency and Monetary␣
↪(RFM) values for each customer based on the LYLTY_CARD_NBR.

rfm=dataframe.groupby("LYLTY_CARD_NBR")["TOT_SALES"].agg(["sum", "count"]).
↪reset_index()    #   Grouping the pandas.DataFrame by the TOT_SALES column␣
↪for each customer, and aggregating the sum and count.
```

```
rfm.columns=["LYLTY_CARD_NBR", "MONETARY", "FREQUENCY"] #   Renaming the columns of the pandas.DataFrame.
rfm["RECENCY"]=(datetime.datetime.strptime("2019-06-30", "%Y-%m-%d")-dataframe.groupby("LYLTY_CARD_NBR")["DATE"].max()).dt.days #   Calculating the Recency value for each customer.
rfm=rfm.dropna()    #   Dropping the null values from the pandas.DataFrame.
rfm=rfm.reset_index(drop=True)  #   Resetting the index of the pandas.DataFrame.

#   Plotting a three-dimensional scatter graph of the Recency, Frequency and Monetary (RFM) values for each customer.

figure=plt.figure(figsize=(15, 5))
axis=figure.add_subplot(111, projection="3d")
axis.scatter(rfm["RECENCY"], rfm["FREQUENCY"], rfm["MONETARY"], c="lightblue", s=50, alpha=0.6, edgecolors="black", linewidth=1, marker="o")
axis.set_xlabel("Recency")
axis.set_ylabel("Frequency")
axis.set_zlabel("Monetary")
plt.title("Recency, Frequency and Monetary (RFM) Values for Each Customer")
plt.show()
```



From the Recency, Frequency and Monetary (RFM) analysis, there don't seem to be many customers at risk of churning, but the scatter graph does suggest that the oldest customers may be most valuable to the store and the recent ones may likely be at risk of churning.

[37]:
```
dataframe.to_csv("QVI_data.csv", index=False)    #   Saving the finalised pandas.
 ↪DataFrame as a CSV file.
```

0.1.1 Conclusion:

- Generally, sales gradually increase during the holiday season and are the highest the day before Christmas Day, but suddenly decrease right after, so this would be the ideal time for any promotional campaigns or discounts.
- The 380 gramme package size, also the largest in the store, is the highest-selling package size during the holiday season with KETTLE® being the highest-selling brand.
- KETTLE® is the also the highest-selling brand during the entire year, but the 175 gramme package size is the highest-selling package size, on average, with a difference of nearly 37% from the second highest-selling package size.
- OLDER SINGLES/COUPLES are the most loyal customers of the store and NEW FAMILIES are the least.
- MIDAGE SINGLES/COUPLES had the highest Mainstream and Premium purchases of all their purchases, while all the others had the least Premium purchases, which means that they're more likely to pay more per packet of chips than the others.
- MIDAGE SINGLES/COUPLES prefer KETTLE® and 175 gramme package size the most in both the Mainstream and Premium customer segment.
- Oldest customers may be most valuable to the store and the recent ones may likely be at risk of churning.