quantium

January 28, 2024

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
#import sheets
purchase_behavior = pd.read_csv("C:/Users/dorwi/Desktop/QVI_purchase_behaviour.
  ↪csv")
transaction_data = pd.read_excel("C:/Users/dorwi/Desktop/QVI_transaction_data.
  ↪xlsx")
```

1 Exploring Purchase Behavior dataset

[4]:
```
#view first ten rows
purchase_behavior.head(10)
```

[4]:
```
   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
5            1007   YOUNG SINGLES/COUPLES           Budget
6            1009            NEW FAMILIES          Premium
7            1010   YOUNG SINGLES/COUPLES       Mainstream
8            1011   OLDER SINGLES/COUPLES       Mainstream
9            1012          OLDER FAMILIES       Mainstream
```

[5]:
```
#count num of rows and cols
purchase_behavior.shape
```

[5]:
```
(72637, 3)
```

[6]:
```
#check info about cols and rows
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
```

1.1 Purchase Behavior dataset - cleaning

[3]:

```
#drop rows with N/A
purchase_behavior.dropna()
```

[3]:

```
       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]
```

[4]:

```
#remove duplicates
purchase_behavior.drop_duplicates()
```

[4]:

```
       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]:
```
#check unique values in 'LIFESTAGE' and 'PREMIUM_CUSTOMER' to check for typos/
 ↪errors

lifestage_unique_values = purchase_behavior.LIFESTAGE.unique()
premium_customer_unique_values = purchase_behavior.PREMIUM_CUSTOMER.unique()

print(f" 1. LIFESTAGE : \n \t {lifestage_unique_values} \n \n 2.␣
 ↪PREMIUM_CUSTOMER : \n \t {premium_customer_unique_values}")
```

```
1. LIFESTAGE :
        ['YOUNG SINGLES/COUPLES' 'YOUNG FAMILIES' 'OLDER SINGLES/COUPLES'
'MIDAGE SINGLES/COUPLES' 'NEW FAMILIES' 'OLDER FAMILIES' 'RETIREES']

2. PREMIUM_CUSTOMER :
        ['Premium' 'Mainstream' 'Budget']
```

2 2. Exploring Transaction Data

[10]:
```
#see top 10 rows
transaction_data.head(10)
```

[10]:
```
    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
5  43604          4            4074    2982        57
6  43601          4            4149    3333        16
7  43601          4            4196    3539        24
8  43332          5            5026    4525        42
9  43330          7            7150    6900        52

                                  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
5  Old El Paso Salsa   Dip Tomato Mild 300g         1        5.1
6  Smiths Crinkle Chips Salt & Vinegar 330g         1        5.7
7     Grain Waves         Sweet Chilli 210g         1        3.6
8   Doritos Corn Chip Mexican Jalapeno 150g         1        3.9
9     Grain Waves Sour    Cream&Chives 210G         2        7.2
```

```
[11]: #count num of rows and cols
      transaction_data.shape
```

```
[11]: (264836, 8)
```

```
[12]: #make sure datatypes are good
      transaction_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            264836 non-null  int64
 1   STORE_NBR       264836 non-null  int64
 2   LYLTY_CARD_NBR  264836 non-null  int64
 3   TXN_ID          264836 non-null  int64
 4   PROD_NBR        264836 non-null  int64
 5   PROD_NAME       264836 non-null  object
 6   PROD_QTY        264836 non-null  int64
 7   TOT_SALES       264836 non-null  float64
dtypes: float64(1), int64(6), object(1)
memory usage: 16.2+ MB
```

2.1 Data Cleaning for Transaction Data

```
[5]: transaction_data.dropna()
```

```
[5]:          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]
```

```
[6]: transaction_data.drop_duplicates()
```

```
[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
…       …        …             …     …       …
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

[264835 rows x 8 columns]
```

```
[15]: #check unique values for 'PROD_NAME' to check for typos/errors

prod_name = transaction_data.PROD_NAME.unique()
prod_name_count = len(prod_name)

print(f" unique values for PROD_NAME : {prod_name} \n \n")
print(f" number of unique values : {prod_name_count}")
```

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


number of unique values : 114
```

[7]:

```
#Change 'DATE' datatype from int64 to datetime64
transaction_data['DATE'] = pd.to_datetime(transaction_data['DATE'],␣
 ↪origin='1899-12-30', unit='D')
```

[8]:

```
transaction_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
```

```
 0   DATE            264836 non-null  datetime64[ns]
 1   STORE_NBR       264836 non-null  int64
 2   LYLTY_CARD_NBR  264836 non-null  int64
 3   TXN_ID          264836 non-null  int64
 4   PROD_NBR        264836 non-null  int64
 5   PROD_NAME       264836 non-null  object
 6   PROD_QTY        264836 non-null  int64
 7   TOT_SALES       264836 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 16.2+ MB
```

[9]:

```
#check for outliers
transaction_data.describe()
#see that there are outliers in 'PROD_QTY' and 'TOT_SALES' based on std and␣
 ↪mean relative to max
#other columns are not for analysis
```

[9]:

```
                                DATE      STORE_NBR  LYLTY_CARD_NBR  \
count                         264836  264836.00000    2.648360e+05
mean   2018-12-30 00:52:12.879215616     135.08011    1.355495e+05
min              2018-07-01 00:00:00       1.00000    1.000000e+03
25%              2018-09-30 00:00:00      70.00000    7.002100e+04
50%              2018-12-30 00:00:00     130.00000    1.303575e+05
75%              2019-03-31 00:00:00     203.00000    2.030942e+05
max              2019-06-30 00:00:00     272.00000    2.373711e+06
std                              NaN      76.78418    8.057998e+04

             TXN_ID       PROD_NBR       PROD_QTY      TOT_SALES
count  2.648360e+05  264836.000000  264836.000000  264836.000000
mean   1.351583e+05      56.583157       1.907309       7.304200
min    1.000000e+00       1.000000       1.000000       1.500000
25%    6.760150e+04      28.000000       2.000000       5.400000
50%    1.351375e+05      56.000000       2.000000       7.400000
75%    2.027012e+05      85.000000       2.000000       9.200000
max    2.415841e+06     114.000000     200.000000     650.000000
std    7.813303e+04      32.826638       0.643654       3.083226
```

[10]:

```
#Sort by DESC  for 'PROD_QTY' to see outliers
transaction_data.sort_values(by='PROD_QTY', ascending=False, inplace=True)
transaction_data.head(10)
```

[10]:

```
             DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
69762  2018-08-19        226          226000  226201         4
69763  2019-05-20        226          226000  226210         4
217237 2019-05-18        201          201060  200202        26
238333 2018-08-14        219          219004  218018        25
238471 2019-05-19        261          261331  261111        87
```

```
228749 2019-05-19        232          232138  235978       109
117802 2019-05-19        176          176471  177469        17
228711 2018-08-17        205          205149  204215         1
238397 2019-05-18        238          238337  243243        28
238395 2019-05-19        238          238250  242874        88

                                      PROD_NAME  PROD_QTY  TOT_SALES
69762           Dorito Corn Chp     Supreme 380g       200      650.0
69763           Dorito Corn Chp     Supreme 380g       200      650.0
217237             Pringles Sweet&Spcy BBQ 134g         5       18.5
238333           Pringles SourCream  Onion 134g         5       18.5
238471  Infuzions BBQ Rib   Prawn Crackers 110g         5       19.0
228749                 Pringles Barbeque   134g         5       18.5
117802       Kettle Sensations   BBQ&Maple 150g         5       23.0
228711  Smiths Crinkle Cut  Chips Barbecue 170g         5       14.5
238397     Thins Potato Chips  Hot & Spicy 175g         5       16.5
238395         Kettle Honey Soy    Chicken 175g         5       27.0
```

```
[11]: #removing outliers for 'PROD_QTY' and 'TOT_SALES' in the dataset
      transaction_data = transaction_data[transaction_data['PROD_QTY'] != 200]

      transaction_data.head(10)
```

```
[11]:              DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
217237 2019-05-18        201          201060  200202        26
238333 2018-08-14        219          219004  218018        25
238471 2019-05-19        261          261331  261111        87
228749 2019-05-19        232          232138  235978       109
117802 2019-05-19        176          176471  177469        17
228711 2018-08-17        205          205149  204215         1
238397 2019-05-18        238          238337  243243        28
238395 2019-05-19        238          238250  242874        88
228668 2018-08-15        167          167417  169238        68
117790 2018-08-17        172          172182  173875        55

                                      PROD_NAME  PROD_QTY  TOT_SALES
217237             Pringles Sweet&Spcy BBQ 134g         5       18.5
238333           Pringles SourCream  Onion 134g         5       18.5
238471  Infuzions BBQ Rib   Prawn Crackers 110g         5       19.0
228749                 Pringles Barbeque   134g         5       18.5
117802       Kettle Sensations   BBQ&Maple 150g         5       23.0
228711  Smiths Crinkle Cut  Chips Barbecue 170g         5       14.5
238397     Thins Potato Chips  Hot & Spicy 175g         5       16.5
238395         Kettle Honey Soy    Chicken 175g         5       27.0
228668     Pringles Chicken    Salt Crips 134g         5       18.5
117790     Snbts Whlgrn Crisps Cheddr&Mstrd 90g         5        8.5
```

[12]:
```
transaction_data.describe()
```

[12]:
```
                                DATE      STORE_NBR  LYLTY_CARD_NBR  \
count                         264834  264834.000000    2.648340e+05
mean   2018-12-30 00:52:10.292938752     135.079423    1.355488e+05
min              2018-07-01 00:00:00       1.000000    1.000000e+03
25%              2018-09-30 00:00:00      70.000000    7.002100e+04
50%              2018-12-30 00:00:00     130.000000    1.303570e+05
75%              2019-03-31 00:00:00     203.000000    2.030940e+05
max              2019-06-30 00:00:00     272.000000    2.373711e+06
std                              NaN      76.784063    8.057990e+04

             TXN_ID       PROD_NBR       PROD_QTY      TOT_SALES
count  2.648340e+05  264834.000000  264834.000000  264834.000000
mean   1.351576e+05      56.583554       1.905813       7.299346
min    1.000000e+00       1.000000       1.000000       1.500000
25%    6.760050e+04      28.000000       2.000000       5.400000
50%    1.351365e+05      56.000000       2.000000       7.400000
75%    2.026998e+05      85.000000       2.000000       9.200000
max    2.415841e+06     114.000000       5.000000      29.500000
std    7.813292e+04      32.826444       0.343436       2.527241
```

3 Grouping both Datasets

[13]:
```
combined_dataset = pd.merge(transaction_data, purchase_behavior,␣
 ↪on='LYLTY_CARD_NBR', how="inner")
```

[14]:
```
combined_dataset.head(10)
```

[14]:
```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2019-05-18        201          201060  200202        26
1 2018-12-11        201          201060  200200        33
2 2019-05-26        201          201060  200203        90
3 2019-01-14        201          201060  200201       108
4 2018-08-14        219          219004  218018        25
5 2018-12-13        219          219004  218019        42
6 2019-04-01        219          219004  218020        38
7 2019-05-19        261          261331  261111        87
8 2019-05-14        261          261331  261110        40
9 2018-07-22        261          261331  261106        46

                                  PROD_NAME  PROD_QTY  TOT_SALES  \
0              Pringles Sweet&Spcy BBQ 134g         5       18.5
1   Cobs Popd Swt/Chlli &Sr/Cream Chips 110g         2        7.6
2         Tostitos Smoked     Chipotle 175g         2        8.8
3   Kettle Tortilla ChpsHny&Jlpno Chili 150g         1        4.6
```

```
4          Pringles SourCream  Onion 134g         5       18.5
5   Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
6   Infuzions Mango     Chutny Papadums 70g         2        4.8
7   Infuzions BBQ Rib   Prawn Crackers 110g         5       19.0
8          Thins Chips Seasonedchicken 175g         2        6.6
9                      Kettle Original 175g         2       10.8

              LIFESTAGE PREMIUM_CUSTOMER
0        YOUNG FAMILIES          Premium
1        YOUNG FAMILIES          Premium
2        YOUNG FAMILIES          Premium
3        YOUNG FAMILIES          Premium
4  YOUNG SINGLES/COUPLES       Mainstream
5  YOUNG SINGLES/COUPLES       Mainstream
6  YOUNG SINGLES/COUPLES       Mainstream
7  YOUNG SINGLES/COUPLES       Mainstream
8  YOUNG SINGLES/COUPLES       Mainstream
9  YOUNG SINGLES/COUPLES       Mainstream
```

4 Metrics of Interest

1. Does customer segment (REMIUM_CUSTOMER) affect total sales for each LIFESTAGE?

[15]:

```python
# 7 diff LIFESTAGE
# At most 3 types of customer segment per LIFESTAGE
# at most 21 values

# Create a list that holds all LIFESTAGE values
lifestage_unique_list = combined_dataset['LIFESTAGE'].unique()
# Create another list that holds all PREMIUM_CUSTOMER values
customer_type_list = combined_dataset['PREMIUM_CUSTOMER'].unique()

for lifestage in lifestage_unique_list:
    print(lifestage)
    for customer_type in customer_type_list:

        filtered_df = combined_dataset[(combined_dataset['LIFESTAGE'] ==␣
 ↪lifestage) & (combined_dataset['PREMIUM_CUSTOMER'] == customer_type)]

        total_sales = filtered_df['TOT_SALES'].sum()

        print(f" \t Customer Segment: {customer_type} ... Total Sales:␣
 ↪${round(total_sales, 2)}")
    print("\n")
```

```
YOUNG FAMILIES
```

```
         Customer Segment: Premium … Total Sales: $84025.5
         Customer Segment: Mainstream … Total Sales: $92788.75
         Customer Segment: Budget … Total Sales: $139345.85


YOUNG SINGLES/COUPLES
         Customer Segment: Premium … Total Sales: $41642.1
         Customer Segment: Mainstream … Total Sales: $157621.6
         Customer Segment: Budget … Total Sales: $61141.6


OLDER SINGLES/COUPLES
         Customer Segment: Premium … Total Sales: $132263.15
         Customer Segment: Mainstream … Total Sales: $133393.8
         Customer Segment: Budget … Total Sales: $136769.8


MIDAGE SINGLES/COUPLES
         Customer Segment: Premium … Total Sales: $58432.65
         Customer Segment: Mainstream … Total Sales: $90803.85
         Customer Segment: Budget … Total Sales: $35514.8


OLDER FAMILIES
         Customer Segment: Premium … Total Sales: $80658.4
         Customer Segment: Mainstream … Total Sales: $103445.55
         Customer Segment: Budget … Total Sales: $168363.25


RETIREES
         Customer Segment: Premium … Total Sales: $97646.05
         Customer Segment: Mainstream … Total Sales: $155677.05
         Customer Segment: Budget … Total Sales: $113147.8


NEW FAMILIES
         Customer Segment: Premium … Total Sales: $11491.1
         Customer Segment: Mainstream … Total Sales: $17013.9
         Customer Segment: Budget … Total Sales: $21928.45
```

Analysis : It seems that customer segment does affect total sales for a given lifestage that a customer is in. Suprisingly, customers labed as premium contributed the least in sales compared to mainstream and budget customers for a given customer's life stage.

The life stage that customers contributed the most to sales is OLDER SINGLES/COUPLES

The life stage that customers contributed the least to sales is NEW FAMILIES

Combining all customer life stages, the customer segment with the greatest total sales is Mainstream customers : $578053.55

Doing the aggregation, the customer segment with the lowest total sales is Premium customers : $397021.8

2. does chip pack weight influence sales ?

[16]:

```
#Extract weight of product into its own column
combined_dataset['PROD_WEIGHT'] = combined_dataset['PROD_NAME'].str.
 ↪extract('(\d+)g').astype(float)
combined_dataset.head(5)
```

[16]:

```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2019-05-18        201          201060  200202        26
1 2018-12-11        201          201060  200200        33
2 2019-05-26        201          201060  200203        90
3 2019-01-14        201          201060  200201       108
4 2018-08-14        219          219004  218018        25

                                  PROD_NAME  PROD_QTY  TOT_SALES  \
0              Pringles Sweet&Spcy BBQ 134g         5       18.5
1  Cobs Popd Swt/Chlli &Sr/Cream Chips 110g         2        7.6
2         Tostitos Smoked     Chipotle 175g         2        8.8
3  Kettle Tortilla ChpsHny&Jlpno Chili 150g         1        4.6
4            Pringles SourCream  Onion 134g         5       18.5

               LIFESTAGE PREMIUM_CUSTOMER  PROD_WEIGHT
0         YOUNG FAMILIES          Premium        134.0
1         YOUNG FAMILIES          Premium        110.0
2         YOUNG FAMILIES          Premium        175.0
3         YOUNG FAMILIES          Premium        150.0
4  YOUNG SINGLES/COUPLES       Mainstream        134.0
```

[17]:

```
#visualize the range of the weight of products
combined_dataset['PROD_WEIGHT'].describe()
```

[17]:

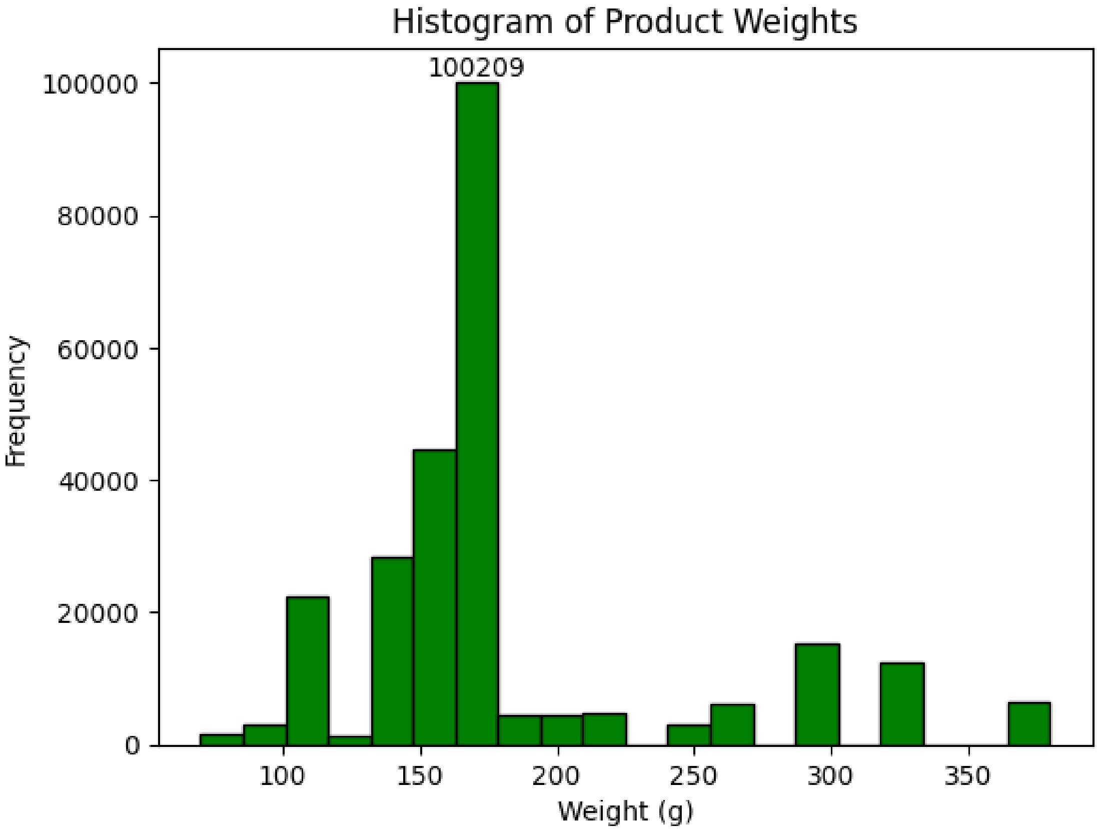
```
count    258770.000000
mean        182.324276
std          64.955035
min          70.000000
25%         150.000000
50%         170.000000
```

```
75%           175.000000
max           380.000000
Name: PROD_WEIGHT, dtype: float64
```

[18]:
```
n, bins, patches = plt.hist(combined_dataset['PROD_WEIGHT'], bins=20,␣
 ↪edgecolor="black", color="green")
plt.title('Histogram of Product Weights')
plt.xlabel('Weight (g)')
plt.ylabel('Frequency')
max_height = max(n)

# Annotate the bars with the bin count
for count, rect in zip(n, patches):
    if rect.get_height() == max_height:  # only label the tallest bar
        plt.annotate(f'{int(count)}',  # The label text
                     (rect.get_x() + rect.get_width() / 2, rect.get_height()), ␣
 ↪# The position of the label
                     ha='center', va='bottom')  # Center alignment

plt.show()
```



Analysis : There is clear evidence that chip pack weight influences sales. Both the descriptive statistics of the column as well as the histogram shows that the majority of chip sales weigh roughly between 150g to 185g, indicating that this is a weight that most consumers are purchasing.

3. Does chip brand influence sales ?

[19]:

```
#Extract brand name from PROD_NAME into a new column
combined_dataset['PROD_BRAND'] = combined_dataset['PROD_NAME'].str.
 ↪extract(r'^(\w+)')

#Validate regex
combined_dataset['PROD_BRAND'].unique()
```

[19]:

```
array(['Pringles', 'Cobs', 'Tostitos', 'Kettle', 'Doritos', 'Infuzions',
       'Thins', 'Woolworths', 'Natural', 'Smiths', 'Tyrrells', 'Snbts',
       'RRD', 'Grain', 'Old', 'WW', 'Cheetos', 'Smith', 'CCs', 'NCC',
       'Twisties', 'Dorito', 'Cheezels', 'Infzns', 'Red', 'Sunbites',
       'Burger', 'French', 'GrnWves'], dtype=object)
```

[20]:

```
#Change misspelled / acronyms of brand name to make it all uniform
corrections = {
    r'\bDorito\b': 'Doritos',
    r'\bGrnWves\b': 'GrainWaves',
    r'\bGrain\b': 'GrainWaves',
    r'\bWW\b': 'Woolworths',
    r'\bRRD\b': 'RedRockDeli',
    r'\bRed\b': 'RedRockDeli',
    r'\bSnbts\b': 'SunBites',
    r'\bInfzns\b': 'Infuzions',
    r'\bInfzns\b': 'Infuzions',
    r'\bNatural\b': 'NaturalChipCo',
    r'\bNCC\b': 'NaturalChipCo',
    r'\bOld\b': 'Old El Paso',
    r'\bSmith\b': 'Smiths',
    r'\bSunbites\b': 'SunBites',
}

combined_dataset['PROD_BRAND'] = combined_dataset['PROD_BRAND'].
 ↪replace(corrections, regex=True)

#validate column names
combined_dataset['PROD_BRAND'].unique()
```

[20]:

```
array(['Pringles', 'Cobs', 'Tostitos', 'Kettle', 'Doritos', 'Infuzions',
       'Thins', 'Woolworths', 'NaturalChipCo', 'Smiths', 'Tyrrells',
       'SunBites', 'RedRockDeli', 'GrainWaves', 'Old El Paso', 'Cheetos',
       'CCs', 'Twisties', 'Cheezels', 'Burger', 'French'], dtype=object)
```

[21]:
```
combined_dataset.head()
```

[21]:
```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2019-05-18        201          201060  200202        26
1 2018-12-11        201          201060  200200        33
2 2019-05-26        201          201060  200203        90
3 2019-01-14        201          201060  200201       108
4 2018-08-14        219          219004  218018        25

                                  PROD_NAME  PROD_QTY  TOT_SALES  \
0              Pringles Sweet&Spcy BBQ 134g         5       18.5
1  Cobs Popd Swt/Chlli &Sr/Cream Chips 110g         2        7.6
2         Tostitos Smoked     Chipotle 175g         2        8.8
3  Kettle Tortilla ChpsHny&Jlpno Chili 150g         1        4.6
4            Pringles SourCream  Onion 134g         5       18.5

               LIFESTAGE PREMIUM_CUSTOMER  PROD_WEIGHT PROD_BRAND
0         YOUNG FAMILIES          Premium        134.0   Pringles
1         YOUNG FAMILIES          Premium        110.0       Cobs
2         YOUNG FAMILIES          Premium        175.0   Tostitos
3         YOUNG FAMILIES          Premium        150.0     Kettle
4  YOUNG SINGLES/COUPLES       Mainstream        134.0   Pringles
```

[22]:
```
#agregate total sales per brand
brand_list = combined_dataset['PROD_BRAND'].unique()

brand_sales = {}

for brand in brand_list:
    #filtered_df = combined_dataset['PROD_BRAND'] == brand
    filtered_df = combined_dataset[(combined_dataset['PROD_BRAND'] == brand)]

    total_sales = filtered_df['TOT_SALES'].sum()

    # print(f"\t{brand} ... ${round(total_sales, 2)}")

    brand_sales[brand] = total_sales


sorted_brand_sales = dict(sorted(brand_sales.items(), key=lambda item: item[1],␣
 ↪reverse=True))

for brand, sales in sorted_brand_sales.items():
    print(f"\t{brand} ... ${round(sales, 2)}")
```

```
        Kettle … $390239.8
        Doritos … $240590.9
```

```
Smiths … $224660.2
Pringles … $177655.5
Infuzions … $99047.6
RedRockDeli … $95046.0
Old El Paso … $90785.1
Thins … $88852.5
Twisties … $81522.1
Tostitos … $79789.6
Cobs … $70569.8
Tyrrells … $51647.4
GrainWaves … $51617.2
Woolworths … $49343.6
NaturalChipCo … $42318.0
Cheezels … $40029.9
CCs … $18078.9
Cheetos … $16884.5
SunBites … $9676.4
French … $7929.0
Burger … $6831.0
```

Analysis : the brand of chips seems to have a strong effect on sales. The four most popular chip brands include Kettle, Doritos, Smiths, and Pringles with all four of them crossing over $100,000 in total sales. Meanwhile, SunBites, French Fries, and Burger Rings are three brands that failed to cross the $10,000 mark in sales indicating a poor response to consumer demand.

5 Task 2 : Evaluating performance of store trials : 77, 86, 88

[23]:
```
combined_dataset.to_csv('combined_dataset.csv')
```

[31]:
```
combined_dataset.head()
```

[31]:
```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2019-05-18        201          201060  200202        26
1 2018-12-11        201          201060  200200        33
2 2019-05-26        201          201060  200203        90
3 2019-01-14        201          201060  200201       108
4 2018-08-14        219          219004  218018        25

                                  PROD_NAME  PROD_QTY  TOT_SALES  \
0              Pringles Sweet&Spcy BBQ 134g         5       18.5
1  Cobs Popd Swt/Chlli &Sr/Cream Chips 110g         2        7.6
2         Tostitos Smoked     Chipotle 175g         2        8.8
3  Kettle Tortilla ChpsHny&Jlpno Chili 150g         1        4.6
4            Pringles SourCream  Onion 134g         5       18.5
```

```
               LIFESTAGE PREMIUM_CUSTOMER  PROD_WEIGHT PROD_BRAND
0         YOUNG FAMILIES          Premium        134.0   Pringles
1         YOUNG FAMILIES          Premium        110.0       Cobs
2         YOUNG FAMILIES          Premium        175.0   Tostitos
3         YOUNG FAMILIES          Premium        150.0     Kettle
4  YOUNG SINGLES/COUPLES       Mainstream        134.0   Pringles
```

[85]:

```
#Find the mean product quantity sold and sales of all records excluding those␣
 ↪from the store trials : 77, 86 , 88
#this will be used to compare the performance of the trial stores with the new␣
 ↪store layout vs current store layout.

exclude_trial_stores_df = combined_dataset[~combined_dataset['STORE_NBR'].
 ↪isin([77, 86, 88])]

total_prod_qty = exclude_trial_stores_df['PROD_QTY'].sum()
total_sales = exclude_trial_stores_df['TOT_SALES'].sum()

total_rows = exclude_trial_stores_df.shape[0]

#get averages
average_prod_qty = total_prod_qty / total_rows
average_sales = total_sales / total_rows

#get totals per store
average_total_prod_qty_per_store = exclude_trial_stores_df.
 ↪groupby('STORE_NBR')['PROD_QTY'].sum().mean()
average_total_sales_per_store = exclude_trial_stores_df.
 ↪groupby('STORE_NBR')['TOT_SALES'].sum().mean()




print(f'Current Store Layout')
print(f'Average Product Quantity Sold : {round(average_prod_qty , 2)}')
print(f'average sales : ${round(average_sales ,2)} \n')

print(f'Average Total Product Quantity Sold Per Store:␣
 ↪{round(average_total_prod_qty_per_store, 2)}')
print(f'Average Total Sales per Store: $ {round(average_total_sales_per_store,␣
 ↪2)}')
```

```
Current Store Layout
Average Product Quantity Sold : 1.91
average sales : $7.3

Average Total Product Quantity Sold Per Store: 1847.84
Average Total Sales per Store: $ 7074.74
```

[93]:

```
#create a function to test all 3 store trials
def analyze_store_sales(store_number, df):
    filtered_df = df[df['STORE_NBR'] == store_number]

    total_store_prod_qty = filtered_df['PROD_QTY'].sum()
    total_store_sales = filtered_df['TOT_SALES'].sum()

    total_store_rows = filtered_df.shape[0]

    if total_rows > 0:
        #Get averages
        average_store_prod_qty = total_store_prod_qty / total_store_rows
        average_total_store_sales = total_store_sales / total_store_rows


        print(f'Store Number: {store_number}')

        print(f'Average Product Quantity... {round(average_store_prod_qty, 2)}')
        print(f'Average Sales... $ {round(average_total_store_sales , 2)}')

        print(f'Total Product Quantity Sold... {round(total_store_prod_qty,␣
 ↪2)}')
        print(f'Total Store Sales... $ {round(total_store_sales, 2)} \n')




    else:
        print(f'Invalid Store Number')
```

[98]:

```
analyze_store_sales(77, combined_dataset)
analyze_store_sales(86, combined_dataset)
analyze_store_sales(88, combined_dataset)

print("----------------------------------------------------------------")

print(f'Current Store Layout')
print(f'Average Product Quantity Sold : {round(average_prod_qty , 2)}')
print(f'average sales : ${round(average_sales ,2)} \n')

print(f'Average Total Product Quantity Sold Per Store:␣
 ↪{round(average_total_prod_qty_per_store, 2)}')
print(f'Average Total Sales per Store: $ {round(average_total_sales_per_store,␣
 ↪2)}')
```

```
Store Number: 77
```

```
Average Product Quantity… 1.55
Average Sales… $ 5.4
Total Product Quantity Sold… 872
Total Store Sales… $ 3040.0

Store Number: 86
Average Product Quantity… 1.99
Average Sales… $ 6.92
Total Product Quantity Sold… 3066
Total Store Sales… $ 10635.35

Store Number: 88
Average Product Quantity… 1.99
Average Sales… $ 8.72
Total Product Quantity Sold… 3718
Total Store Sales… $ 16333.25

----------------------------------------------------------------
Current Store Layout
Average Product Quantity Sold : 1.91
average sales : $7.3

Average Total Product Quantity Sold Per Store: 1847.84
Average Total Sales per Store: $ 7074.74
```

Analysis: Two out of the three trial stores with the new chip layout is performing better in terms of qantity sold and thus leading to higher sales. For further analysis, we should dive deeper into understanding why the new chips layout for Store #77 resulted in a decrease in performance. Many factors may contribute to this decrease such as customers in Store #77 specficially having trouble finding the newly placed chips isle which could explain the decrease in quantity sold. Another factor may be that for customers in Store #77, the changed layout may influence them to purchase other snacks instead. In that case, we would want to dive deeper to understand the impact of changing the chips layout on other goods in the store and see if theres a correlation between chip purchases and other purchases.