In [1]:

```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from mpl_toolkits.mplot3d import axis3d

plt.rcParams['figure.figsize'] = (16, 12)
```

In [2]:

```python
transact = pd.read_csv('QVI_transaction_data.xlsx; filename*.csv', parse_dates=['DATE'])
```

In [3]:

```python
transact
```

Out[3]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8
...	...	...	...	...	...	...	...	...
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2	10.8
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	1	4.4
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	2	8.8
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2	7.8
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	2	8.8

264836 rows × 8 columns

In [4]:

```python
cust = pd.read_csv('QVI_purchase_behaviour.csv; filename*.csv')
```

In [5]:

```python
cust
```

Out[5]:

	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 × 3 columns

In [6]:

```
transact.info()
```

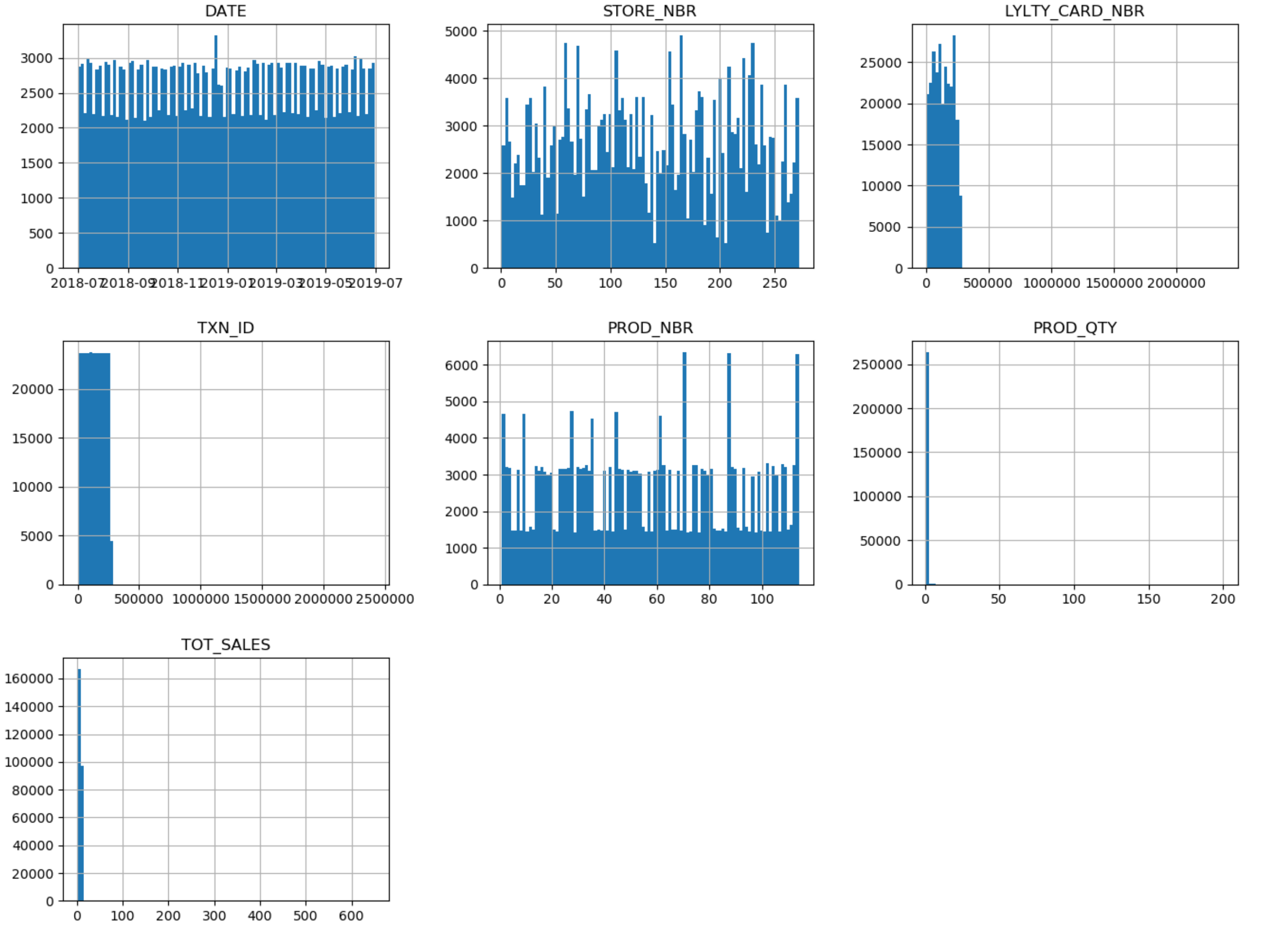
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            264836 non-null  datetime64[ns]
 1   STORE_NBR       264836 non-null  int64
 2   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
```

In [7]:

```
transact.hist(bins=100)
```

Out[7]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f19543f9b20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f195234f400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f19522fbc70>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f19522b24f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f19522dcd00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1952290580>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f195229e490>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f19521ff5b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f19521ff700>]],
      dtype=object)
```



I see obvious outliers in the data, we can see this from boxpot figure.

```
In [8]: transact.LYLTY_CARD_NBR.sort_values(ascending=False).head(50)
```

```
Out[8]: 256040    2373711
        53107     2370961
        53106     2370961
        227371    2370751
        215522    2370701
        15676     2370651
        97172     2370581
        97173     2370581
        97171     2370361
        255925    2370181
        133253    2370001
        96939     2330501
        32030     2330461
        104927    2330431
        135105    2330331
        244444    2330321
        228460    2330311
        115267    2330291
        99033     2330291
        99034     2330291
        215358    2330271
        135106    2330251
        1440      2330211
        80809     2330191
        169026    2330171
        135734    2330121
        169023    2330081
        172311    2330051
        238493    2330041
        19463     2330031
        19455     2330031
        105898     883791
        105899     883791
        105897     883791
        39795      880711
        39794      880711
        123510     880551
        39777      880171
        39778      880171
        191118     862501
        191117     862501
        191105     861961
        140568     861951
        123456     861921
        25107      272392
        25108      272392
        25109      272392
        25110      272392
        258555     272391
        28115      272390
        Name: LYLTY_CARD_NBR, dtype: int64
```

```
In [9]: transact.TXN_ID.sort_values(ascending=False).head(50)
```

Out[9]:

```
15726     2415841
25110      270209
25109      270208
25108      270207
25107      270206
258555     270205
28115      270204
258554     270202
258553     270201
258552     270200
135104     270199
135103     270198
99032      270197
99031      270196
118034     270195
135102     270194
135101     270193
135100     270192
135099     270191
99030      270190
264835     270189
264834     270188
264833     270187
258551     270186
258550     270185
258549     270184
135098     270183
135097     270182
135096     270181
99029      270180
171771     270179
171770     270178
171769     270177
228459     270176
150305     270175
150304     270174
150303     270173
258548     270172
258547     270171
99028      270170
99027      270169
99026      270168
216896     270166
216895     270165
216894     270164
216893     270163
135095     270162
117350     270161
117349     270160
117348     270159
Name: TXN_ID, dtype: int64
```

In [10]:

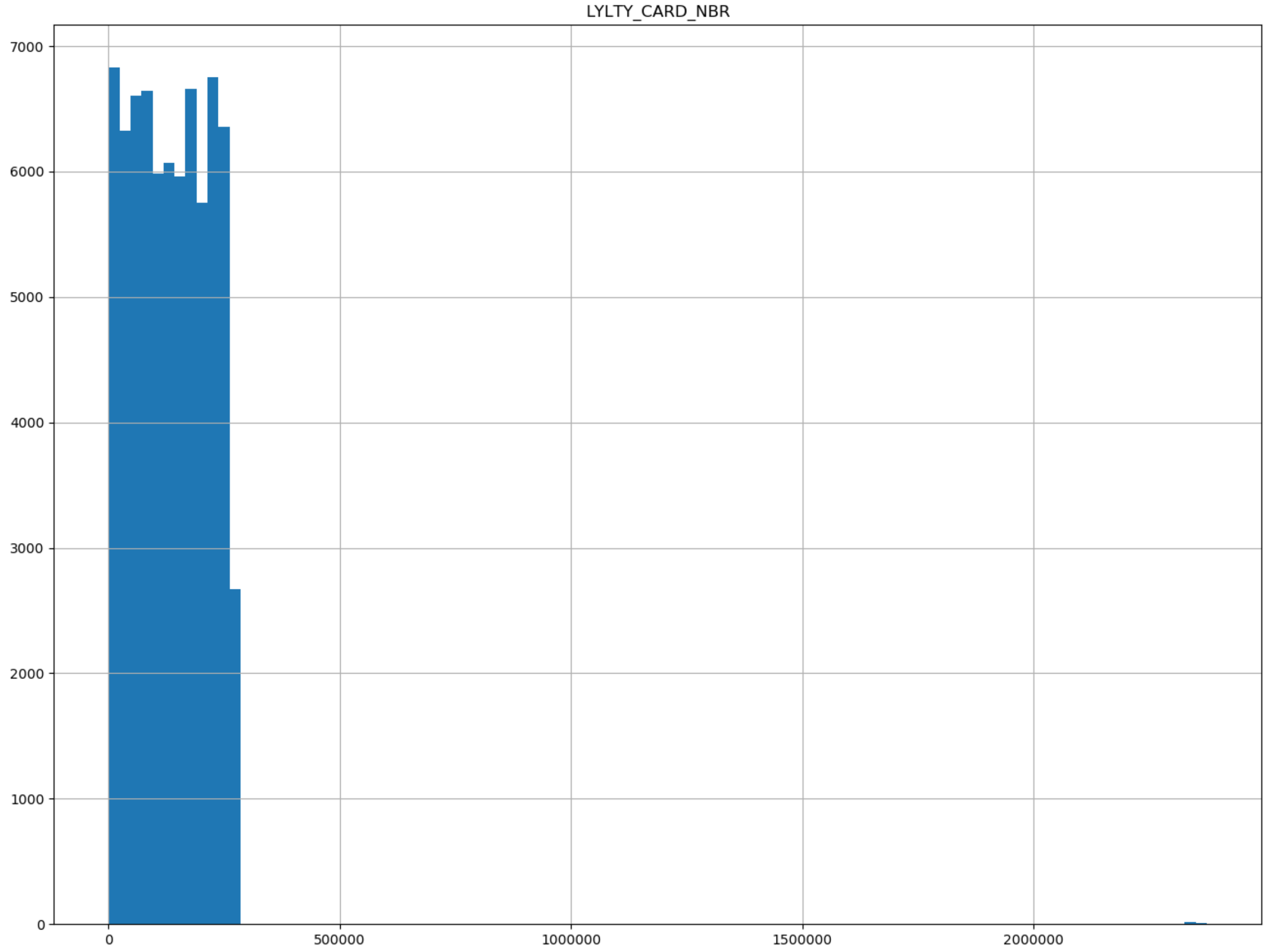
```
index_outliers_TXN_ID = transact.TXN_ID.idxmax()
```

In [11]:

```
cust.hist(bins=100)
```

Out[11]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f1954465a00>]],
      dtype=object)
```



I see that numbers of outliers in Loyalty_cards match in both datasets, so they are not outliers, and should be taken into consideration.

```
In [12]: transact.PROD_QTY.sort_values(ascending=False)
```

```
Out[12]: 69762     200
         69763     200
         217237      5
         238333      5
         238471      5
                  ... 
         82354       1
         82357       1
         172438      1
         82358       1
         32479       1
         Name: PROD_QTY, Length: 264836, dtype: int64
```

```
In [13]: transact.PROD_QTY.value_counts()
```

```
Out[13]: 2      236039
         1       27518
         5         450
         3         430
         4         397
         200         2
         Name: PROD_QTY, dtype: int64
```

Quantity of products equal 200 strongly looks like outliers, there should be number 2 as a volume.

```
In [14]: transact.PROD_QTY.replace(to_replace=200, value=2,  inplace=True)
```

```
In [15]: transact.PROD_QTY.value_counts()
```

```
Out[15]: 2    236041
         1     27518
         5       450
         3       430
         4       397
         Name: PROD_QTY, dtype: int64
```

```
In [16]: transact.columns
```

```
Out[16]: Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
                'PROD_NAME', 'PROD_QTY', 'TOT_SALES'],
               dtype='object')
```

```
In [17]: transact.TOT_SALES.sort_values(ascending=False)
```

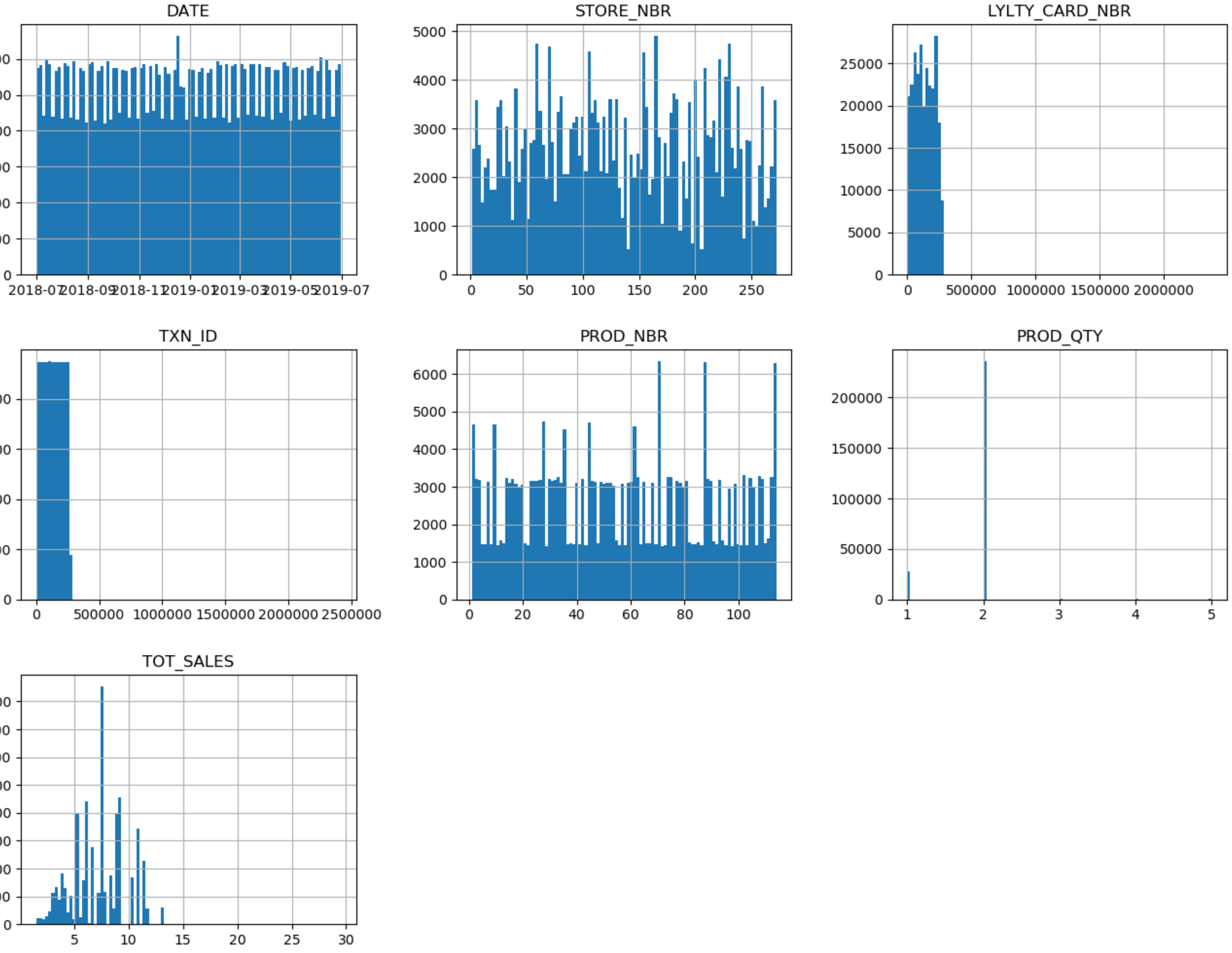
```
Out[17]: 69762     650.0
         69763     650.0
         69496      29.5
         55558      29.5
         171815     29.5
                   ...
         259695      1.5
         259707      1.5
         197005      1.5
         216449      1.5
         150019      1.5
         Name: TOT_SALES, Length: 264836, dtype: float64
```

I see strange recordings with ID 69762 and 69763, there are mistakes in three columns. It will be better drop them, as there are sufficient number of samples in datasets.

```
In [18]: transact.drop(index=[69762, 69763], axis=0, inplace=True)
```

```
In [19]: transact.hist(bins=100)
```

```
Out[19]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f198c3dff40>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7f1950544f40>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7f195189ba00>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x7f19518d0280>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7f195187aac0>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7f1951830340>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x7f195183f250>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7f19517a0340>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x7f19517a0490>]],
               dtype=object)
```



In [20]:
```
transact.columns
```

Out[20]:
```
Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES'],
      dtype='object')
```

In [21]:
```
renaming_columns = {'DATE': 'transaction_date',
                    'STORE_NBR': 'store_number',
                    'LYLTY_CARD_NBR': 'loyalty_card_number',
                    'TXN_ID': 'tax_ID',
                    'PROD_NBR': 'product_number',
                    'PROD_NAME': 'product_name',
                    'PROD_QTY': 'product_quantity',
                    'TOT_SALES': 'total_sales'}
```

In [22]:
```
transact.rename(renaming_columns, axis=1, inplace=True)
```

In [23]:
```
cust.columns
```

Out[23]:
```
Index(['LYLTY_CARD_NBR', 'LIFESTAGE', 'PREMIUM_CUSTOMER'], dtype='object')
```

In [24]:
```
renaming_columns_cust = {'LYLTY_CARD_NBR': 'loyalty_card_number',
                         'LIFESTAGE': 'lifestage',
                         'PREMIUM_CUSTOMER': 'premium_customer'}
```

In [25]:
```
cust.rename(renaming_columns_cust, axis=1, inplace=True)
```

In [26]:
```
transact
```

Out[26]:

	transaction_date	store_number	loyalty_card_number	tax_ID	product_number	product_name	produc
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
...	...	...	...	...	...	...	
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

264834 rows × 8 columns

In [27]:

```
(list(transact.product_name.unique()))
```

```
Out[27]: ['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']
```

Let's create new columns with weights_of_chips and company_names.

In [28]:

```python
import re
def abstract_weights(string):
    prog = re.compile('[0-9]')
    res = ''.join(prog.findall(string))
    return res
```

In [29]:
```
transact['weights_of_chips']  = transact.product_name.apply(abstract_weights)
```

In [30]:
```
transact.weights_of_chips.unique()
```

Out[30]:
```
array(['175', '170', '150', '300', '330', '210', '270', '220', '125',
       '110', '134', '380', '180', '165', '135', '250', '200', '160',
       '190', '90', '70'], dtype=object)
```

In [31]:
```
def company_names(string):
    temp = string.split()
    return ' '.join(temp[0:1])
```

In [32]:
```
transact['company_name'] = transact.product_name.apply(company_names)
```

In [33]:
```
list(transact.company_name.unique())
```

Out[33]:
```
['Natural',
 'CCs',
 'Smiths',
 'Kettle',
 'Old',
 'Grain',
 'Doritos',
 'Twisties',
 'WW',
 'Thins',
 'Burger',
 'NCC',
 'Cheezels',
 'Infzns',
 'Red',
 'Pringles',
 'Dorito',
 'Infuzions',
 'Smith',
 'GrnWves',
 'Tyrrells',
 'Cobs',
 'Woolworths',
 'French',
 'RRD',
 'Tostitos',
 'Cheetos',
 'Snbts',
 'Sunbites']
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Let's combine these together.

In [34]:
```
transact.company_name = transact.company_name.replace(to_replace='RRD', value='Red')
transact.company_name = transact.company_name.replace(to_replace="SNBTS", value="SUNBITE
transact.company_name = transact.company_name.replace(to_replace="INFZNS", value="INFUZI
transact.company_name = transact.company_name.replace(to_replace="WW", value="WOOLWORTHS
transact.company_name = transact.company_name.replace(to_replace="SMITH", value="SMITHS"
transact.company_name = transact.company_name.replace(to_replace="NCC", value="NATURAL")
transact.company_name = transact.company_name.replace(to_replace="DORITO", value="DORITO
transact.company_name = transact.company_name.replace(to_replace="GRAIN", value="GRNWVES
```

In [35]:
```
transact.company_name.unique()
```

Out[35]:
```
array(['Natural', 'CCs', 'Smiths', 'Kettle', 'Old', 'Grain', 'Doritos',
       'Twisties', 'WOOLWORTHS', 'Thins', 'Burger', 'NATURAL', 'Cheezels',
       'Infzns', 'Red', 'Pringles', 'Dorito', 'Infuzions', 'Smith',
       'GrnWves', 'Tyrrells', 'Cobs', 'Woolworths', 'French', 'Tostitos',
       'Cheetos', 'Snbts', 'Sunbites'], dtype=object)
```

We should use only chips product - thus separate chips from occasionally appeared there other products.¶

In [36]:
```
transact.head(50)
```

Out[36]:

	transaction_date	store_number	loyalty_card_number	tax_ID	product_number	product_name	product_
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
5	2019-05-19	4	4074	2982	57	Old El Paso Salsa Dip Tomato Mild 300g	
6	2019-05-16	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g	
7	2019-05-16	4	4196	3539	24	Grain Waves Sweet Chilli 210g	
8	2018-08-20	5	5026	4525	42	Doritos Corn Chip Mexican Jalapeno 150g	
9	2018-08-18	7	7150	6900	52	Grain Waves Sour Cream&Chives 210G	
10	2019-05-17	7	7215	7176	16	Smiths Crinkle Chips Salt & Vinegar 330g	
11	2018-08-20	8	8294	8221	114	Kettle Sensations Siracha Lime 150g	
12	2019-05-18	9	9208	8634	15	Twisties Cheese 270g	
13	2018-08-17	13	13213	12447	92	WW Crinkle Cut Chicken 175g	
14	2019-05-15	19	19272	16686	44	Thins Chips Light& Tangy 175g	
15	2019-05-19	20	20164	17136	54	CCs Original 175g	
16	2018-08-18	20	20418	17413	94	Burger Rings 220g	
17	2018-08-14	22	22411	18646	98	NCC Sour Cream & Garden Chives 175g	
18	2018-08-17	22	22456	18696	93	Doritos Corn Chip Southern Chicken 150g	
19	2019-05-16	23	23067	19162	56	Cheezels Cheese Box 125g	
20	2019-05-19	25	25105	21815	7	Smiths Crinkle Original 330g	
21	2018-08-16	33	33081	29949	98	NCC Sour Cream & Garden Chives 175g	

	transaction_date	store_number	loyalty_card_number	tax_ID	product_number	product_name	product_
22	2018-08-16	36	36012	32077	31	Infzns Crn Crnchers Tangy Gcamole 110g	
23	2018-08-19	36	36302	33188	32	Kettle Sea Salt And Vinegar 175g	
24	2018-08-15	38	38142	34181	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
25	2019-05-15	39	39144	35506	57	Old El Paso Salsa Dip Tomato Mild 300g	
26	2018-08-19	39	39167	35638	111	Smiths Chip Thinly Cut Original 175g	
27	2019-05-15	41	41423	38393	46	Kettle Original 175g	
28	2018-08-15	41	41486	38472	13	Red Rock Deli Thai Chilli&Lime 150g	
29	2019-05-20	43	43110	39342	31	Infzns Crn Crnchers Tangy Gcamole 110g	
30	2019-05-16	43	43147	39608	99	Pringles Sthrn FriedChicken 134g	
31	2019-05-15	43	43227	40186	26	Pringles Sweet&Spcy BBQ 134g	
32	2019-05-20	45	45127	41122	64	Red Rock Deli SR Salsa & Mzzrlla 150g	
33	2019-05-18	45	45220	41651	22	Thins Chips Originl saltd 175g	
34	2018-08-16	51	51100	46802	48	Red Rock Deli Sp Salt & Truffle 150G	
35	2018-08-19	51	51100	46803	37	Smiths Thinly Swt Chli&S/Cream175G	
36	2018-08-18	51	51113	46828	36	Kettle Chilli 175g	
37	2018-08-17	54	54226	48173	51	Doritos Mexicana 170g	
38	2018-08-18	54	54305	48301	44	Thins Chips Light& Tangy 175g	
39	2018-08-18	55	55072	48878	107	Smiths Crinkle Cut French OnionDip 150g	
40	2019-05-14	55	55072	48883	106	Natural ChipCo Hony Soy Chckn175g	
41	2019-05-20	55	55073	48887	4	Dorito Corn Chp Supreme 380g	
42	2019-05-20	55	55073	48887	113	Twisties Chicken270g	
43	2019-05-16	55	55202	49690	45	Smiths Thinly Cut Roast Chicken 175g	
44	2018-08-18	56	56013	50090	39	Smiths Crinkle Cut Tomato Salsa 150g	

	transaction_date	store_number	loyalty_card_number	tax_ID	product_number	product_name	product_
45	2019-05-16	58	58324	54252	102	Kettle Mozzarella Basil & Pesto 175g	
46	2019-05-16	59	59344	56007	31	Infzns Crn Crnchers Tangy Gcamole 110g	
47	2018-08-16	60	60038	56304	104	Infuzions Thai SweetChili PotatoMix 110g	
48	2019-05-15	60	60162	56825	7	Smiths Crinkle Original 330g	
49	2019-05-14	62	62177	58848	3	Kettle Sensations Camembert & Fig 150g	

In [37]:
```
def remove_not_useful(string):
    pat = re.compile('[0-9]{0,3}[gG]{1}')
    string1 = re.sub(pat, '', string)
    string2 = string1.replace('&', '')
    return string2.rstrip()
```

In [38]:
```
transact.product_name = transact.product_name.apply(remove_not_useful)
```

In [39]:
```
#let's create text file for tokenizing and frequency analysis
text = ' '.join(transact.product_name.unique()).lower()
text
```

Out[39]:

```
'natural chip        compny seasalt ccs nacho cheese smiths crinkle cut  chips chicken s
miths chip thinly  s/creamonion kettle tortilla chpshnyjlpno chili old el paso salsa   d
ip tomato mild smiths crinkle chips salt  vinear rain waves         sweet chilli doritos
corn chip mexican jalapeno rain waves sour    creamchives kettle sensations   siracha li
me twisties cheese ww crinkle cut      chicken thins chips liht  tany ccs oriinal burer
rins ncc sour cream     arden chives doritos corn chip southern chicken cheezels cheese
box smiths crinkle      oriinal infzns crn crnchers tany camole kettle sea salt     and
vinear smiths chip thinly  cut oriinal kettle oriinal red rock deli thai  chillilime pri
nles sthrn friedchicken prinles sweetspcy bbq red rock deli sr    salsa  mzzrlla thins c
hips         oriinl saltd red rock deli sp    salt  truffle smiths thinly       swt chli
s/cream kettle chilli doritos mexicana smiths crinkle cut  french oniondip natural chipc
o      hony soy chckn dorito corn chp     supreme twisties chicken smiths thinly cut   r
oast chicken smiths crinkle cut  tomato salsa kettle mozzarella   basil  pesto infuzions
thai sweetchili potatomix kettle sensations   camembert  fi smith crinkle cut   mac n ch
eese kettle honey soy    chicken thins chips seasonedchicken smiths crinkle cut  salt  v
inear infuzions bbq rib   prawn crackers rnwves plus btroot  chilli jam tyrrells crisps
    lihtly salted kettle sweet chilli and sour cream doritos salsa       medium kettle
swt pot sea salt prinles sourcream  onion doritos corn chips  oriinal twisties cheese
  burer old el paso salsa   dip chnky tom ht cobs popd swt/chlli sr/cream chips woolwort
hs mild     salsa natural chip co     tmato hrbspce smiths crinkle cut  chips oriinal co
bs popd sea salt  chips smiths crinkle cut  chips chsonion french fries potato chips old
el paso salsa   dip tomato med doritos corn chips  cheese supreme prinles oriinal   cris
ps rrd chilli         coconut ww oriinal corn    chips thins potato chips  hot  spicy co
bs popd sour crm  chives chips smiths crnkle chip  ornl bi ba doritos corn chips  nacho
cheese kettle sensations   bbqmaple ww d/style chip     sea salt prinles chicken    salt
crips ww oriinal stacked chips smiths chip thinly  cutsalt/viner cheezels cheese tostito
s lihtly    salted thins chips salt   vinear smiths crinkle cut  chips barbecue cheetos
puffs rrd sweet chilli   sour cream ww crinkle cut      oriinal tostitos splash of  lime
woolworths medium   salsa kettle tortilla chpsbtrootricotta ccs tasty cheese woolworths
cheese   rins tostitos smoked     chipotle prinles barbeque ww supreme cheese   corn chi
ps prinles mystery    flavour tyrrells crisps     ched  chives snbts whlrn crisps cheddr
mstrd cheetos chs  bacon balls prinles slt vinar infuzions sourcreamherbs ve strws kettl
e tortilla chpsfetaarlic infuzions mano     chutny papadums rrd steak          chimuchur
ri rrd honey soy       chicken sunbites whlern    crisps frch/onin rrd salt  vinear dori
tos cheese      supreme smiths crinkle cut  snasauce ww sour cream onionstacked chips rr
d lime  pepper natural chipco sea  salt  viner red rock deli chiknarlic aioli rrd sr slo
w rst     pork belly rrd pc sea salt smith crinkle cut   bolonese doritos salsa mild'
```

In [40]:

```python
regex = re.compile('[A-Za-z]+')
def words_only(text, regex=regex):
    try:
        return " ".join(regex.findall(text))
    except:
        return ""
```

In [41]:

```python
tokenized_text = words_only(text)
tokenized_text
```

Out[41]:

```
'natural chip compny seasalt ccs nacho cheese smiths crinkle cut chips chicken smiths ch
ip thinly s creamonion kettle tortilla chpshnyjlpno chili old el paso salsa dip tomato m
ild smiths crinkle chips salt vinear rain waves sweet chilli doritos corn chip mexican j
alapeno rain waves sour creamchives kettle sensations siracha lime twisties cheese ww cr
inkle cut chicken thins chips liht tany ccs oriinal burer rins ncc sour cream arden chiv
es doritos corn chip southern chicken cheezels cheese box smiths crinkle oriinal infzns 
crn crnchers tany camole kettle sea salt and vinear smiths chip thinly cut oriinal kettl
e oriinal red rock deli thai chillilime prinles sthrn friedchicken prinles sweetspcy bbq
red rock deli sr salsa mzzrlla thins chips oriinl saltd red rock deli sp salt truffle sm
iths thinly swt chlis cream kettle chilli doritos mexicana smiths crinkle cut french oni
ondip natural chipco hony soy chckn dorito corn chp supreme twisties chicken smiths thin
ly cut roast chicken smiths crinkle cut tomato salsa kettle mozzarella basil pesto infuz
ions thai sweetchili potatomix kettle sensations camembert fi smith crinkle cut mac n ch
eese kettle honey soy chicken thins chips seasonedchicken smiths crinkle cut salt vinear
infuzions bbq rib prawn crackers rnwves plus btroot chilli jam tyrrells crisps lihtly sa
lted kettle sweet chilli and sour cream doritos salsa medium kettle swt pot sea salt pri
nles sourcream onion doritos corn chips oriinal twisties cheese burer old el paso salsa 
dip chnky tom ht cobs popd swt chlli sr cream chips woolworths mild salsa natural chip c
o tmato hrbspce smiths crinkle cut chips oriinal cobs popd sea salt chips smiths crinkle
cut chips chsonion french fries potato chips old el paso salsa dip tomato med doritos co
rn chips cheese supreme prinles oriinal crisps rrd chilli coconut ww oriinal corn chips 
thins potato chips hot spicy cobs popd sour crm chives chips smiths crnkle chip ornl bi
ba doritos corn chips nacho cheese kettle sensations bbqmaple ww d style chip sea salt p
rinles chicken salt crips ww oriinal stacked chips smiths chip thinly cutsalt viner chee
zels cheese tostitos lihtly salted thins chips salt vinear smiths crinkle cut chips barb
ecue cheetos puffs rrd sweet chilli sour cream ww crinkle cut oriinal tostitos splash of
lime woolworths medium salsa kettle tortilla chpsbtrootricotta ccs tasty cheese woolwort
hs cheese rins tostitos smoked chipotle prinles barbeque ww supreme cheese corn chips pr
inles mystery flavour tyrrells crisps ched chives snbts whlrn crisps cheddrmstrd cheetos
chs bacon balls prinles slt vinar infuzions sourcreamherbs ve strws kettle tortilla chps
fetaarlic infuzions mano chutny papadums rrd steak chimuchurri rrd honey soy chicken sun
bites whlern crisps frch onin rrd salt vinear doritos cheese supreme smiths crinkle cut 
snasauce ww sour cream onionstacked chips rrd lime pepper natural chipco sea salt viner 
red rock deli chiknarlic aioli rrd sr slow rst pork belly rrd pc sea salt smith crinkle 
cut bolonese doritos salsa mild'
```

In [42]:

```python
from nltk import FreqDist
fd = FreqDist(tokenized_text.split())
for i in fd.most_common(30):
    print(i)
```

```
('chips', 21)
('smiths', 16)
('crinkle', 14)
('cut', 14)
('kettle', 13)
('cheese', 12)
('salt', 12)
('oriinal', 10)
('chip', 9)
('salsa', 9)
('doritos', 9)
('chicken', 8)
('corn', 8)
('prinles', 8)
('rrd', 8)
('ww', 7)
('chilli', 6)
('sour', 6)
('cream', 6)
('sea', 6)
('thinly', 5)
('vinear', 5)
('thins', 5)
('crisps', 5)
('natural', 4)
('red', 4)
('rock', 4)
('deli', 4)
('supreme', 4)
('infuzions', 4)
```

There are salsa products in the dataset but we are only interested in the chips category, so let's remove these.

In [43]:

```python
def clear_salsa(string):
    if ('Salsa' or 'salsa') in string:
        return ''
    else:
        return string
```

In [44]:

```python
transact.product_name = transact.product_name.apply(clear_salsa)
```

In [45]:

```python
index_salsa = transact.query('product_name==""').index
```

In [46]:

```python
transact_chips = transact.drop(index=index_salsa)
```

In [47]:

```python
transact_chips.describe()
```

Out[47]:

	store_number	loyalty_card_number	tax_ID	product_number	product_quantity	total_sales
count	246740.000000	2.467400e+05	2.467400e+05	246740.000000	246740.000000	246740.000000
mean	135.050361	1.355303e+05	1.351304e+05	56.352213	1.906456	7.316113
std	76.786971	8.071520e+04	7.814760e+04	33.695235	0.342499	2.474897
min	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.700000
25%	70.000000	7.001500e+04	6.756875e+04	26.000000	2.000000	5.800000
50%	130.000000	1.303670e+05	1.351815e+05	53.000000	2.000000	7.400000
75%	203.000000	2.030832e+05	2.026522e+05	87.000000	2.000000	8.800000
max	272.000000	2.373711e+06	2.415841e+06	114.000000	5.000000	29.500000

In [48]:

```
transact_chips.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 246740 entries, 0 to 264835
Data columns (total 10 columns):
 #   Column               Non-Null Count   Dtype         
---  ------               --------------   -----         
 0   transaction_date     246740 non-null  datetime64[ns]
 1   store_number         246740 non-null  int64         
 2   loyalty_card_number  246740 non-null  int64         
 3   tax_ID               246740 non-null  int64         
 4   product_number       246740 non-null  int64         
 5   product_name         246740 non-null  object        
 6   product_quantity     246740 non-null  int64         
 7   total_sales          246740 non-null  float64       
 8   weights_of_chips     246740 non-null  object        
 9   company_name         246740 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
memory usage: 20.7+ MB
```

In [49]:

```
from IPython.display import FileLink
```

In [50]:

```
common_df = transact_chips.merge(cust, on='loyalty_card_number', how='outer')
```

In [51]:

```
common_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 248090 entries, 0 to 248089
Data columns (total 12 columns):
 #   Column               Non-Null Count   Dtype         
---  ------               --------------   -----         
 0   transaction_date     246740 non-null  datetime64[ns]
 1   store_number         246740 non-null  float64       
 2   loyalty_card_number  248090 non-null  int64         
 3   tax_ID               246740 non-null  float64       
 4   product_number       246740 non-null  float64       
 5   product_name         246740 non-null  object        
 6   product_quantity     246740 non-null  float64       
 7   total_sales          246740 non-null  float64       
 8   weights_of_chips     246740 non-null  object        
 9   company_name         246740 non-null  object        
 10  lifestage            248090 non-null  object        
 11  premium_customer     248090 non-null  object        
dtypes: datetime64[ns](1), float64(5), int64(1), object(5)
memory usage: 24.6+ MB
```

After merging two tables together we see, that there is one loyalty_card_number that does not correspond with transactions data, we will remove it. As this is just one sample, we may apply dropna to whole dataset.

In [52]:

```
cons_df = common_df.dropna()
```

In [53]:

```
cons_df
```

Out[53]:

	transaction_date	store_number	loyalty_card_number	tax_ID	product_number	product_name	produ
0	2018-10-17	1.0	1000	1.0	5.0	Natural Chip Compny SeaSalt	
1	2019-05-14	1.0	1307	348.0	66.0	CCs Nacho Cheese	
2	2018-11-10	1.0	1307	346.0	96.0	WW Oriinal Stacked Chips	
3	2019-03-09	1.0	1307	347.0	54.0	CCs Oriinal	
4	2019-05-20	1.0	1343	383.0	61.0	Smiths Crinkle Cut Chips Chicken	
...	...	...	...	...	...	...	
246735	2019-03-09	272.0	272319	270088.0	89.0	Kettle Sweet Chilli And Sour Cream	
246736	2018-08-13	272.0	272358	270154.0	74.0	Tostitos Splash Of Lime	
246737	2018-11-06	272.0	272379	270187.0	51.0	Doritos Mexicana	
246738	2018-12-27	272.0	272379	270188.0	42.0	Doritos Corn Chip Mexican Jalapeno	
246739	2018-09-22	272.0	272380	270189.0	74.0	Tostitos Splash Of Lime	

246740 rows × 12 columns

In [54]:

```
cons_df['day_of_week'] = cons_df.transaction_date.dt.day_name()
```

```
/tmp/ipykernel_6325/3752925889.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  cons_df['day_of_week'] = cons_df.transaction_date.dt.day_name()
```

Now, let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data.

There's only 364 rows, meaning only 364 dates which indicates a missing date. Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

In [55]:

```
dates_range = pd.date_range(start='1 Jul 2018', end='30 Jun 2019')
dates_range
```

Out[55]:

```
DatetimeIndex(['2018-07-01', '2018-07-02', '2018-07-03', '2018-07-04',
               '2018-07-05', '2018-07-06', '2018-07-07', '2018-07-08',
               '2018-07-09', '2018-07-10',
               ...
               '2019-06-21', '2019-06-22', '2019-06-23', '2019-06-24',
               '2019-06-25', '2019-06-26', '2019-06-27', '2019-06-28',
               '2019-06-29', '2019-06-30'],
              dtype='datetime64[ns]', length=365, freq='D')
```

In [56]:

```
all_dates = pd.DataFrame(data=dates_range, columns=['all_dates'])
all_dates
```

Out[56]:

	all_dates
0	2018-07-01
1	2018-07-02
2	2018-07-03
3	2018-07-04
4	2018-07-05
...	...
360	2019-06-26
361	2019-06-27
362	2019-06-28
363	2019-06-29
364	2019-06-30

365 rows × 1 columns

In [57]:

```
with_dates = pd.concat([cons_df, all_dates], axis=1)
```

In [58]:

```
with_dates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 246740 entries, 0 to 246739
Data columns (total 14 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   transaction_date     246740 non-null  datetime64[ns]
 1   store_number         246740 non-null  float64
 2   loyalty_card_number  246740 non-null  int64
 3   tax_ID               246740 non-null  float64
 4   product_number       246740 non-null  float64
 5   product_name         246740 non-null  object
 6   product_quantity     246740 non-null  float64
 7   total_sales          246740 non-null  float64
 8   weights_of_chips     246740 non-null  object
 9   company_name         246740 non-null  object
 10  lifestage            246740 non-null  object
 11  premium_customer     246740 non-null  object
 12  day_of_week          246740 non-null  object
 13  all_dates            365 non-null     datetime64[ns]
dtypes: datetime64[ns](2), float64(5), int64(1), object(6)
memory usage: 28.2+ MB
```

In [59]:
```
with_dates.all_dates = with_dates.all_dates.fillna(with_dates.transaction_date)
```

In [60]:
```
sales_per_day = with_dates.groupby('all_dates', as_index=False).agg({'total_sales': 'sum
sales_per_day
```

Out[60]:

	all_dates	total_sales
0	2018-07-01	4905.3
1	2018-07-02	4883.3
2	2018-07-03	4958.5
3	2018-07-04	4970.2
4	2018-07-05	4668.7
...	...	...
360	2019-06-26	4840.5
361	2019-06-27	4929.5
362	2019-06-28	4872.8
363	2019-06-29	5163.9
364	2019-06-30	5099.0

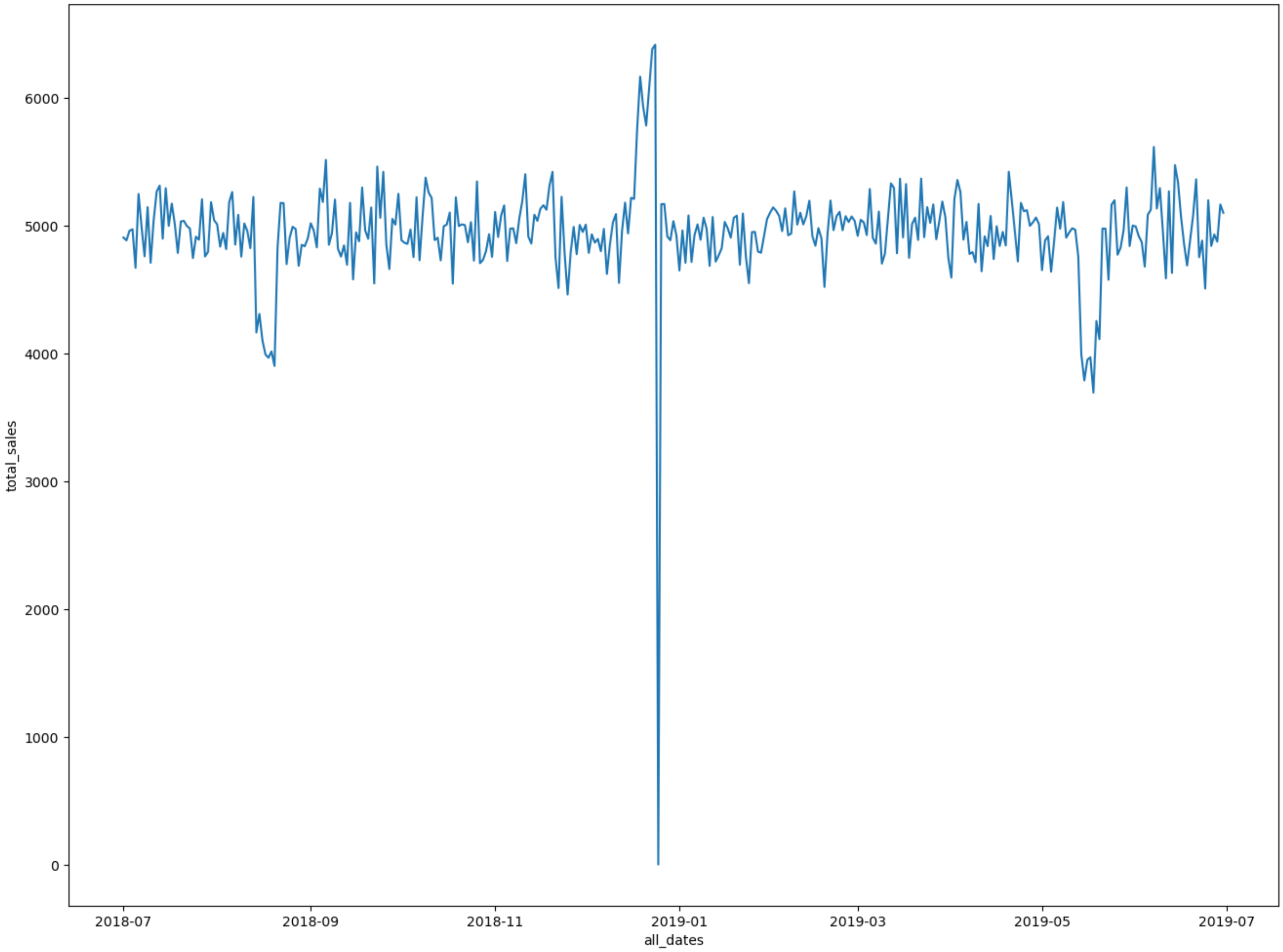
365 rows × 2 columns

Plot transactions over time

In [61]:
```
sns.lineplot(x="all_dates", y="total_sales",
             data=sales_per_day)
```

```
/usr/lib/python3/dist-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support
for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed i
n a future version.  Convert to a numpy array before indexing instead.
  ndim = x[:, None].ndim
/usr/lib/python3/dist-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for
multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a
future version.  Convert to a numpy array before indexing instead.
  x = x[:, np.newaxis]
/usr/lib/python3/dist-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for
multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a
future version.  Convert to a numpy array before indexing instead.
  y = y[:, np.newaxis]
```

```
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1944ec1d60>
```



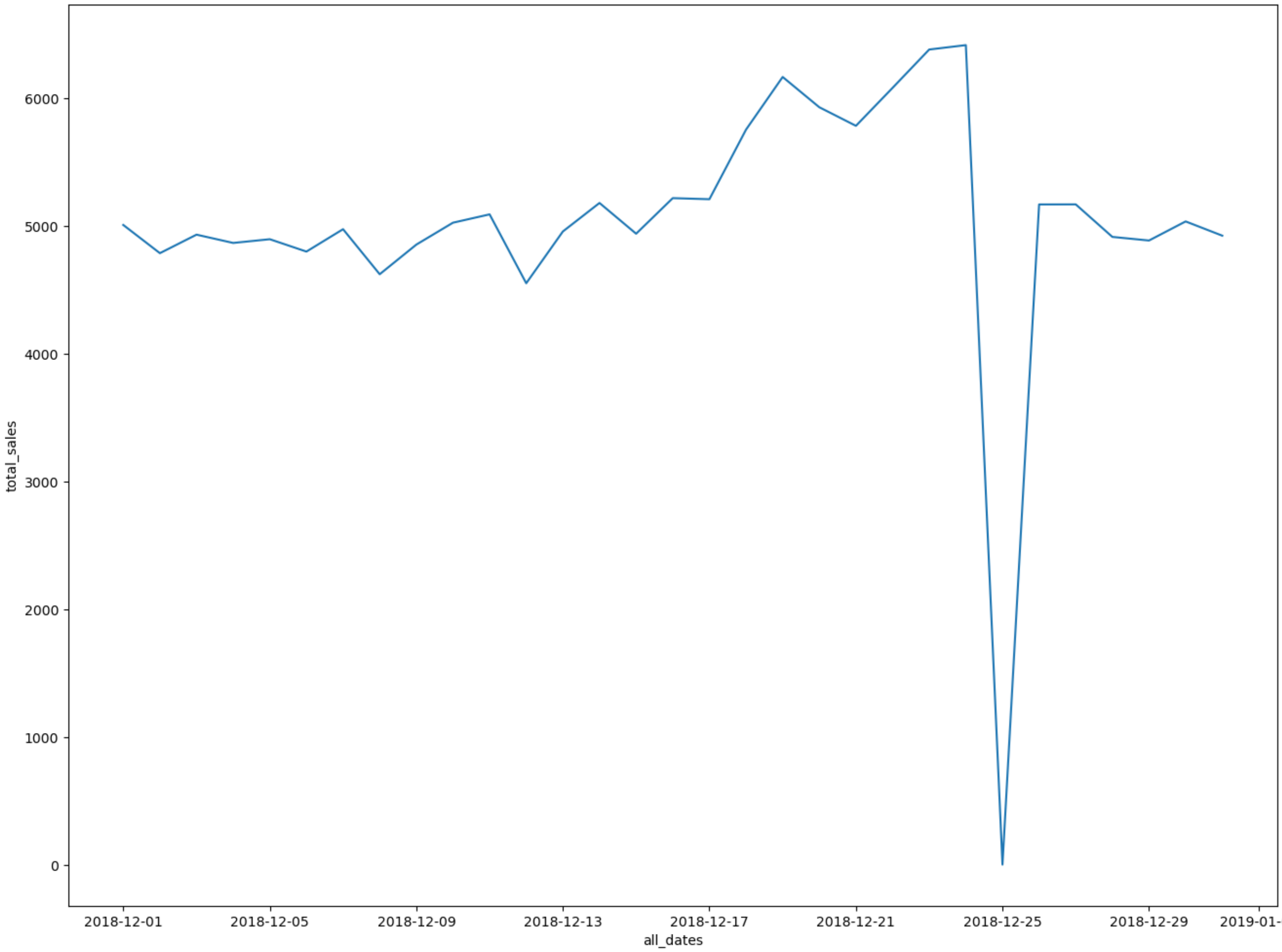
```
In [62]: x = with_dates.query('"2018-11-30" <all_dates < "2019-01-01"')\
                     .groupby('all_dates', as_index=False)\
                     .agg({'total_sales': 'sum'})

         sns.lineplot(x="all_dates", y="total_sales", data=x)
```

```
/usr/lib/python3/dist-packages/matplotlib/cbook/__init__.py:1402: FutureWarning: Support
for multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed i
n a future version.  Convert to a numpy array before indexing instead.
  ndim = x[:, None].ndim
/usr/lib/python3/dist-packages/matplotlib/axes/_base.py:276: FutureWarning: Support for
multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a
future version.  Convert to a numpy array before indexing instead.
  x = x[:, np.newaxis]
/usr/lib/python3/dist-packages/matplotlib/axes/_base.py:278: FutureWarning: Support for
multi-dimensional indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a
future version.  Convert to a numpy array before indexing instead.
  y = y[:, np.newaxis]
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1944e1be80>
```



We can see that there is an increase in purchases in December and a break in late December. We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day.

In [63]:

```
cons_df = with_dates
```

In [64]:

```
cons_df.weights_of_chips.astype('int64').unique()
```

Out[64]:

```
array([175, 160, 170, 150, 330, 165, 110, 210, 180, 200, 134, 270, 220,
       125, 135, 380, 250,  90, 190,  70])
```

Let's download the whole dataset for future analysis

In [ ]:

```
cons_df.to_csv(r'outliers.csv')
```

Activity of customers depending on their lifestage and let's see if the higher sales are due to there being more customers who buy chips. How many customers are in each segment?

In [ ]:

```
FileLink(r'activity_on_lifestage.png')
active_stage = cons_df.groupby(['lifestage', 'premium_customer'], as_index=False)\
                .agg({'total_sales': 'sum'})\
                .sort_values(by='total_sales', ascending=False)
activity_on_lifestage = sns.catplot(
                        data=active_stage, kind="bar",
                        x="lifestage", y="total_sales", hue="premium_customer",
                        errorbar="sd", palette="inferno")
plt.xticks(rotation=0)
plt.savefig('activity_on_lifestage.png')
FileLink(r'cust_per_segment.png')
cust_per_segment = cons_df.groupby(['lifestage','premium_customer'], as_index=False)\
                .agg({'loyalty_card_number': 'nunique'})\
                .sort_values(by='loyalty_card_number', ascending=False)

cust_per_segment_plot = sns.catplot(
                        data=cust_per_segment, kind="bar",
                        x="lifestage", y="loyalty_card_number", hue="premium_customer",
                        errorbar="sd", palette="inferno")

plt.xticks(rotation=0)
plt.savefig('cust_per_segment_plot.png')
```

There are more Mainstream - young singles/couples and Mainstream - retirees who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Budget - Older families segment.

Most popular size of pack.

In [ ]:

```
FileLink(r'popular_package.png')
popular_package = cons_df.groupby('weights_of_chips', as_index=False)\
                .agg({'total_sales': 'sum'})\
                .sort_values(by='total_sales', ascending=False)\
                .head(20)
popular_package = sns.barplot(data = popular_package, x='weights_of_chips', y='total_sal
plt.savefig('popular_package.png')
```

Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER

In [ ]:

```
FileLink(r'average_units.png')
average_units = cons_df.groupby(['lifestage','premium_customer'], as_index=False)\
                .agg({'product_quantity': 'mean'})
```

```
average_units = sns.catplot(
                        data=average_units, kind="bar",
                        x="lifestage", y="product_quantity", hue="premium_customer",
                        errorbar="sd", palette="twilight")

plt.xticks(rotation=70)
plt.savefig('average_units.png')
```

Older families and young families in general buy more chips per customer

Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

In [ ]:

```
cons_df['average_price'] = cons_df.total_sales/cons_df.product_quantity
```

In [ ]:

```
cons_df.average_price.describe()
```

In [ ]:

```
FileLink(r'average_unit_price.png')
average_unit_price = cons_df.groupby(['lifestage','premium_customer'], as_index=False)\
                .agg({'average_price': 'mean'})

average_unit_price_plot = sns.catplot(
                        data=average_unit_price, kind="bar",
                        x="lifestage", y="average_price", hue="premium_customer",
                        errorbar="sd", palette="twilight")

plt.xticks(rotation=70)
plt.savefig('average_unit_price_plot.png')
```

"Mainstream midage" and "mainstream young singles and couples" are more willing to pay more per packet of chips compared to their budget and premium counterparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consumption. This is also supported by there being fewer premium midage and young singles and couples buying chips compared to their mainstream counterparts.

As the difference in average price per unit isn't large, we can check if this difference is statistically different. Let's perform a t-test to see if the difference is significant. Perform an independent t-test between mainstream vs premium and budget midage and young singles and couples.

In [ ]:

```
from scipy.stats import ttest_ind
```

In [ ]:

```
t_test = ttest_ind(cons_df.query('(lifestage == "MIDAGE SINGLES/COUPLES" or lifestage ==
                            and premium_customer == "Mainstream"').average_price,
        cons_df.query('(lifestage == "MIDAGE SINGLES/COUPLES" or lifestage =="YOUNG SING
                            and premium_customer != "Mainstream"').average_price)
t_test
```

In [ ]:

```
cons_df.query('(lifestage == "MIDAGE SINGLES/COUPLES" or lifestage =="YOUNG SINGLES/COUP
                            and premium_customer == "Mainstream"').average_price.mean()
```

In [ ]:

```
cons_df.query('(lifestage == "MIDAGE SINGLES/COUPLES" or lifestage =="YOUNG SINGLES/COUP
                            and premium_customer != "Mainstream"').average_price.mean()
```

In [ ]:

```

```

The t-test results in a p-value of 2.235645611540966e-309 i.e. the unit price for mainstream young and mainstream mid-age singles and couples ARE statistically significantly higher than that of budget or premium, young and midage singles and couples.

We might want to target customer segments that contribute the most to sales to retain them or further increase sales. Let's look at Mainstream - young singles/couples. For instance, let's find out if they tend to buy a particular brand of chips.

In [ ]:

```
target_group = cons_df\
                .query('lifestage == "YOUNG SINGLES/COUPLES" and premium_customer == "Ma
```

In [ ]:

```
other_groups = cons_df\
                    .query('lifestage != "YOUNG SINGLES/COUPLES" and premium_customer !=
```

In [ ]:

```
target_group_quantity = target_group.product_quantity.sum()
other_groups_quantity = other_groups.product_quantity.sum()
```

In [ ]:

```
affinity_target = target_group.groupby('company_name', as_index=False).agg({'product_qua
affinity_target['affinity_target'] = affinity_target.product_quantity/target_group_quant
```

In [ ]:

```
affinity_other = other_groups.groupby('company_name', as_index=False).agg({'product_quan
```

In [ ]:

```
affinity_other['affinity_other'] = affinity_other.product_quantity/other_groups_quantity
```

In [ ]:

```
merged_df = affinity_target.merge(affinity_other, on='company_name')
```

In [ ]:

```
merged_df['affinitytobrand'] = merged_df.affinity_target/merged_df.affinity_other
merged_df.sort_values(by='affinitytobrand', ascending=False)
```

We can see that : • Mainstream young singles/couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population • Mainstream young singles/couples are 56% less likely to purchase Burger Rings compared to the rest of the population

Let’s also find out if our target segment tends to buy larger

packs of chips.

In [ ]:

```
affinity_target_weights = target_group.groupby('weights_of_chips', as_index=False)\
                          .agg({'product_quantity': 'sum'})
affinity_target_weights['affinity_target_weights'] = affinity_target_weights.\
                          product_quantity/target_group_quantity
```

In [ ]:

```
affinity_other_weights = other_groups.groupby('weights_of_chips', as_index=False)\
                          .agg({'product_quantity': 'sum'})
affinity_other_weights['affinity_other_weights'] = affinity_other_weights.\
                          product_quantity/other_groups_quantity
```

In [ ]:

```
merged_df_weights = affinity_target_weights.merge(affinity_other_weights, on='weights_of
merged_df_weights['affinitytosize'] = merged_df_weights\
                                        .affinity_target_weights/merged_df_weights.affinity_
merged_df_weights.sort_values(by='affinitytosize', ascending=False)
```

It looks like Mainstream young singles/couples are 27% more likely to purchase a 270g pack of chips compared to the rest of the population but let’s dive into what brands sell this pack size.

In [ ]:

```
cons_df.query('weights_of_chips == "270"')
```

Twisties are the only brand offering 270g packs and so this may instead be reflecting a higher likelihood of purchasing Twisties.

Conclusion Let’s recap what we’ve found! Sales have mainly been due to Budget - older families, Mainstream - young singles/couples, and Mainstream

- retirees shoppers. We found that the high spend in chips for mainstream young singles/couples and retirees is due to there being more of them than other buyers. Mainstream, midage and young singles and

couples are also more likely to pay more per packet of chips. This is indicative of impulse buying behaviour. We’ve also found that Mainstream young singles and couples are 23% more likely to purchase Tyrrells chips compared to the rest of the population. The Category Manager may want to increase the category’s performance by off-locating some Tyrrells and smaller packs of chips in discretionary space near segments where young singles and couples frequent more often to increase visibilty and impulse behaviour. Quantium can help the Category Manager with recommendations of where these segments are and further help them with measuring the impact of the changed placement. We’ll work on measuring the impact of trials in the next task and putting all these together in the third task.

In [ ]:

```
#!pip install nbconvert[webpdf]
```

In [ ]:

```
#!jupyter nbconvert --to webpdf --allow-chromium-download your-notebook-file.ipynb
```

In [ ]:

```
#!jupyter nbconvert --to webpdf --allow-chromium-download quantium.ipynb
```

In [ ]: