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

In [2]:
```
df = pd.read_excel('QVI_transaction_data.xlsx')
```

In [3]:
```
df.head()
```

Out[3]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	

In [4]:
```
df.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
```

In [5]:
```
1 df.describe()
```

Out[5]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QT
count	264836.000000	264836.00000	2.648360e+05	2.648360e+05	264836.000000	264836.00000
mean	43464.036260	135.08011	1.355495e+05	1.351583e+05	56.583157	1.90730
std	105.389282	76.78418	8.057998e+04	7.813303e+04	32.826638	0.64365
min	43282.000000	1.00000	1.000000e+03	1.000000e+00	1.000000	1.00000
25%	43373.000000	70.00000	7.002100e+04	6.760150e+04	28.000000	2.00000
50%	43464.000000	130.00000	1.303575e+05	1.351375e+05	56.000000	2.00000
75%	43555.000000	203.00000	2.030942e+05	2.027012e+05	85.000000	2.00000
max	43646.000000	272.00000	2.373711e+06	2.415841e+06	114.000000	200.00000

In [6]:
```
1 df.isna().sum()
```

Out[6]:
```
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
```

In [7]:
```
1 df.columns
```

Out[7]:
```
Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES'],
      dtype='object')
```

In [ ]:
```
1
```

In [8]:
```
1 df.dtypes
```

Out[8]:
```
DATE              int64
STORE_NBR         int64
LYLTY_CARD_NBR    int64
TXN_ID            int64
PROD_NBR          int64
PROD_NAME        object
PROD_QTY          int64
TOT_SALES       float64
dtype: object
```

In [9]:
```
df['PROD_NAME'][5][-4:]
```

Out[9]:
```
'300g'
```

In [10]:
```
df['Amount_in_grams'] = df['PROD_NAME'].apply(lambda x: x[-4:-1])
```

In [11]:
```
df
```

Out[11]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
...	...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2
264832	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	1
264833	43410	272	272379	270187	51	Doritos Mexicana 170g	2
264834	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2
264835	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	2

264836 rows × 9 columns

In [21]:
```

```

Out[21]:
```
0    Natural Chip        Compny SeaSalt175g
1                  CCs Nacho Cheese    175g
2    Smiths Crinkle Cut  Chips Chicken 170g
3    Smiths Chip Thinly  S/Cream&Onion 175g
Name: PROD_NAME, dtype: object
```

In [31]:
```
df['Product_Name'] = df['PROD_NAME'].apply(lambda x: x[0:-4])
```

In [32]:
```
df
```

Out[32]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
...	...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2
264832	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	1
264833	43410	272	272379	270187	51	Doritos Mexicana 170g	2
264834	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2
264835	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	2

264836 rows × 10 columns

In [33]:
```
df.columns
```

Out[33]:
```
Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES', 'Amount_in_grams',
       'Product_Name'],
      dtype='object')
```

In [ ]:
```

```

In [34]:
```
cat_cols = ['STORE_NBR', 'PROD_NBR']
cont_cols = ['TOT_SALES','PROD_QTY', 'Amount_in_grams']
```

In [35]:
```
1 df['PROD_NBR'].unique()
```

Out[35]:
```
array([  5,  66,  61,  69, 108,  57,  16,  24,  42,  52, 114,  15,  92,
        44,  54,  94,  98,  93,  56,   7,  31,  32, 111,  46,  13,  99,
        26,  64,  22,  48,  37,  36,  51, 107, 106,   4, 113,  45,  39,
       102, 104,   3,  82,  88,  40,  73,  87,  84,  70,  89, 101,  63,
        25,  47,  71,  65,  33,  35,  12,   8,  75, 100,  29,  59,  30,
        81,  67, 110,  28,   2,  14,  77,  17,  83,  68,  96,  79,  23,
        50,  78,   1,  86,  53,  72,  74,  76,   9,  91, 105,  90, 109,
        27,  62, 112,  55,  18,  34,  49,  60,  38, 103,  85,  95,  97,
        20,  19,  21,   6,  80,  58,  10,  11,  43,  41], dtype=int64)
```

In [36]:
```
1 dff = pd.DataFrame()
```

In [37]:
```
1 dff = df
```

In [38]:
```
df
```

Out[38]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
...	...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2
264832	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	1
264833	43410	272	272379	270187	51	Doritos Mexicana 170g	2
264834	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2
264835	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	2

264836 rows × 10 columns

In [39]:
```
from datetime import date, timedelta
start = date(1899,12,30)
new_date_format = []

for date in df["DATE"]:
    delta = timedelta(date)
    new_date_format.append(start + delta)
```

In [40]:
```
df["DATE"] = pd.to_datetime(pd.Series(new_date_format))
print(df["DATE"].dtype)
```

```
datetime64[ns]
```

In [41]:
```
1 df
```

Out[41]:

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

264836 rows × 10 columns

In [42]:
```
df['PROD_NAME'].unique()
```

Out[42]:
```
array(['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'],
dtype=object)
```

In [43]:
```
1 split_prods = df["PROD_NAME"].str.replace(r'([0-9]+[gG])','').str.replace(r'
```

In [44]:
```
split_prods
```

Out[44]:
```
0                  [Natural, Chip, Compny, SeaSalt]
1                              [CCs, Nacho, Cheese]
2            [Smiths, Crinkle, Cut, Chips, Chicken]
3          [Smiths, Chip, Thinly, S, Cream, Onion]
4        [Kettle, Tortilla, ChpsHny, Jlpno, Chili]
                             ...
264831    [Kettle, Sweet, Chilli, And, Sour, Cream]
264832                [Tostitos, Splash, Of, Lime]
264833                         [Doritos, Mexicana]
264834     [Doritos, Corn, Chip, Mexican, Jalapeno]
264835                [Tostitos, Splash, Of, Lime]
Name: PROD_NAME, Length: 264836, dtype: object
```

In [45]:
```
word_counts = {}

def count_words(line):
    for word in line:
        if word not in word_counts:
            word_counts[word] = 1
        else:
            word_counts[word] += 1

split_prods.apply(lambda line: count_words(line))
print(pd.Series(word_counts).sort_values(ascending=False))
```

```
Chips      49770
Kettle     41288
Smiths     28860
Salt       27976
Cheese     27890
           ...
Whlegrn     1432
Pc          1431
NCC         1419
Garden      1419
Fries       1418
Length: 198, dtype: int64
```

In [46]:
```
# There are salsa products in the dataset but we are only interested in the
# category, so let's remove these.
# #### Remove salsa products
df = dff[~dff["PROD_NAME"].str.contains(r"[Ss]alsa")]
```

In [47]:
```
1 df
```

Out[47]:

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

246742 rows × 10 columns

In [48]:
```
# Next, we can use `summary()` to check summary statistics such as mean, min
# values for each feature to see if there are any obvious outliers in the da
# there are any nulls in any of the columns (`NA's : number of nulls` will a
# the output if there are any nulls).
# ```{r initial summary}
# #### Summarise the data to check for nulls and possible outliers
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 246742 entries, 0 to 264835
Data columns (total 10 columns):
 #   Column           Non-Null Count   Dtype         
---  ------           --------------   -----         
 0   DATE             246742 non-null  datetime64[ns]
 1   STORE_NBR        246742 non-null  int64         
 2   LYLTY_CARD_NBR   246742 non-null  int64         
 3   TXN_ID           246742 non-null  int64         
 4   PROD_NBR         246742 non-null  int64         
 5   PROD_NAME        246742 non-null  object        
 6   PROD_QTY         246742 non-null  int64         
 7   TOT_SALES        246742 non-null  float64       
 8   Amount_in_grams  246742 non-null  object        
 9   Product_Name     246742 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
memory usage: 20.7+ MB
```

In [49]:
```
df.describe()
```

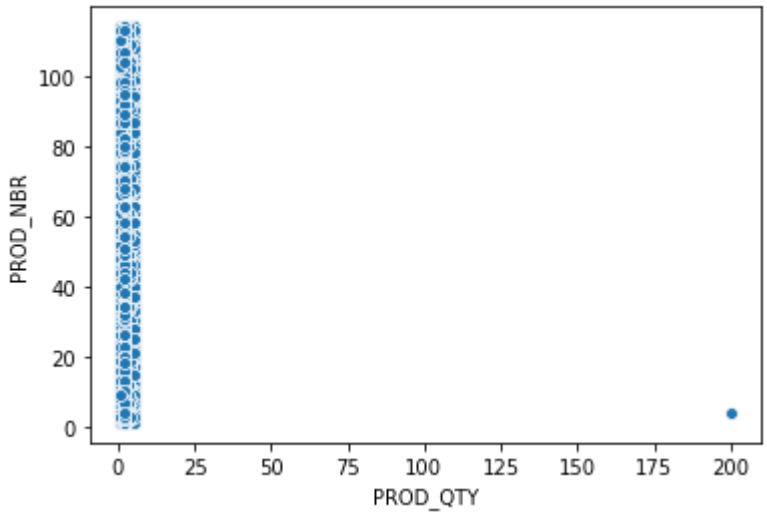
Out[49]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SAL
count	246742.000000	2.467420e+05	2.467420e+05	246742.000000	246742.000000	246742.0000
mean	135.051098	1.355310e+05	1.351311e+05	56.351789	1.908062	7.3213
std	76.787096	8.071528e+04	7.814772e+04	33.695428	0.659831	3.0778
min	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.7000
25%	70.000000	7.001500e+04	6.756925e+04	26.000000	2.000000	5.8000
50%	130.000000	1.303670e+05	1.351830e+05	53.000000	2.000000	7.4000
75%	203.000000	2.030840e+05	2.026538e+05	87.000000	2.000000	8.8000
max	272.000000	2.373711e+06	2.415841e+06	114.000000	200.000000	650.0000

In [51]:
```
import seaborn as sns
```

In [58]:
```
# #### Summarise the data to check for nulls and possible outliers
# # Over to you!
# ```
# There are no nulls in the columns but product quantity appears to have an
# which we should investigate further. Let's investigate further the case wh
# packets of chips are bought in one transaction.
sns.scatterplot(df['PROD_QTY'], df['PROD_NBR'])
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4e9b138c8>



In [55]:
```
df.sort_values(by="PROD_QTY", ascending=False).head()
```

Out[55]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	T
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	
135225	2019-05-15	46	46296	42138	81	Pringles Original Crisps 134g	5	
69523	2019-05-15	71	71142	69852	96	WW Original Stacked Chips 160g	5	
69502	2018-08-18	55	55144	49328	44	Thins Chips Light& Tangy 175g	5	

In [60]:
```
# There are two transactions where 200 packets of chips are bought in one tr
# and both of these transactions were by the same customer.
# ```{r}
# #### Let's see if the customer has had other transactions
df[df["PROD_QTY"] > 100]
```

Out[60]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TC
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	

In [62]:
```
df[df["LYLTY_CARD_NBR"] == 226000]
```

Out[62]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TC
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	

In [63]:
```
# It looks like this customer has only had the two transactions over the yea
# not an ordinary retail customer. The customer might be buying chips for co
# purposes instead. We'll remove this loyalty card number from further analy
# ```{r}
# #### Filter out the customer based on the loyalty card number
df = df[df["PROD_QTY"] < 6]
```

In [66]:
```
#### Count the number of transactions by date
df['DATE'].describe()
```

Out[66]:
```
count                   246740
unique                     364
top       2018-12-24 00:00:00
freq                       865
first     2018-07-01 00:00:00
last      2019-06-30 00:00:00
Name: DATE, dtype: object
```

In [72]:
```
# There's only 364 rows, meaning only 364 dates which indicates a missing da
# create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to
# chart of number of transactions over time to find the missing date.
# ```{r fig.align = "center"}
# #### Create a sequence of dates and join this the count of transactions by
# # Over to you - create a column of dates that includes every day from 1 Ju
# 30 Jun 2019, and join it onto the data to fill in the missing day
pd.date_range(start=df["DATE"].min(), end=df["DATE"].max()).difference(df["D
```

Out[72]:
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

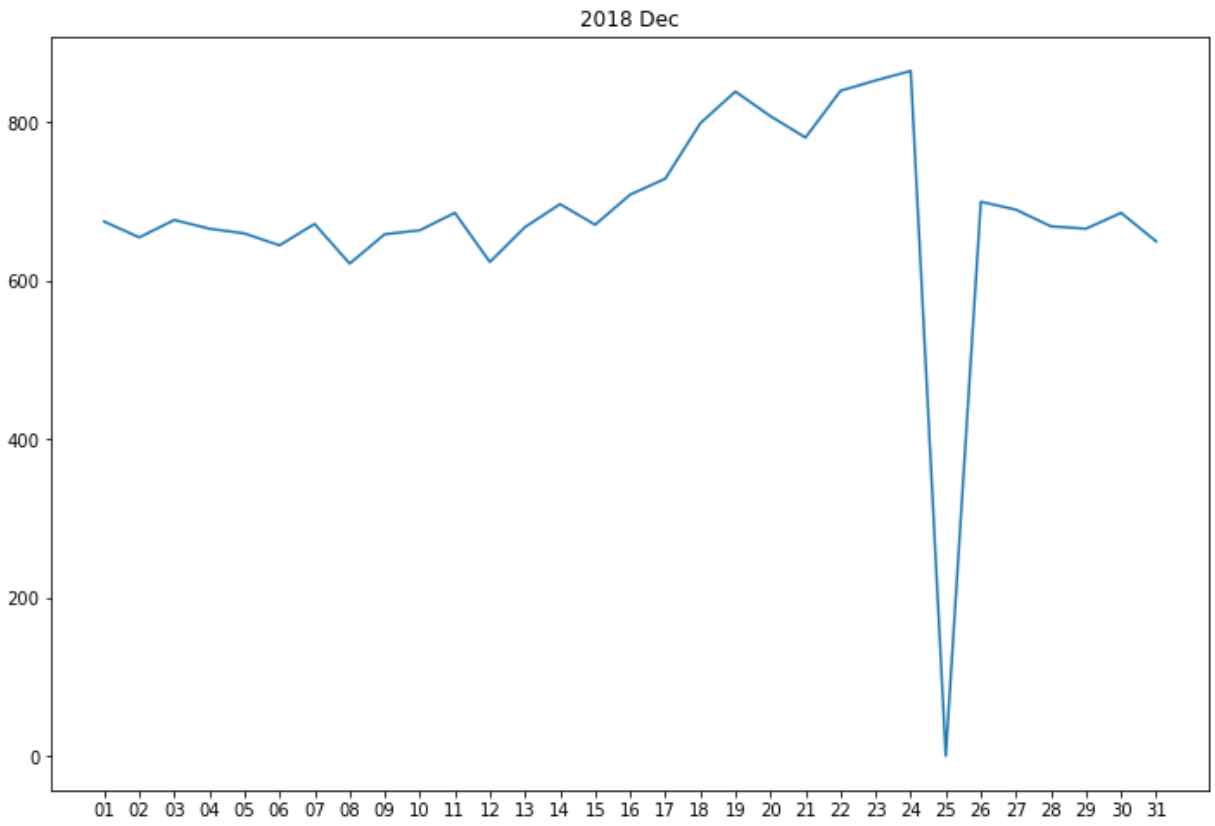
In [76]:
```
# Missing date is ['2018-12-25']
check_null_date = pd.merge(pd.Series(pd.date_range(start=df["DATE"].min(), e
```

File failed to load: /extensions/MathMenu.js

In [82]:
```
# #### Setting plot themes to format graphs
# theme_set(theme_bw())
# theme_update(plot.title = element_text(hjust = 0.5))
# #### Plot transactions over time
# ggplot(transactions_by_day, aes(x = DATE, y = N)) +
#  geom_line() +
#  labs(x = "Day", y = "Number of transactions", title = "Transactions over
#  scale_x_date(breaks = "1 month") +
#  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
trans_by_date = check_null_date["DATE"].value_counts()
dec = trans_by_date[(trans_by_date.index >= pd.datetime(2018,12,1)) & (trans
dec.index = dec.index.strftime('%d')
ax = dec.plot(figsize=(12,8))
ax.set_xticks(np.arange(len(dec)))
ax.set_xticklabels(dec.index)
plt.title("2018 Dec")
plt.show()

```

```
C:\Users\agamm\Anaconda3\lib\site-packages\ipykernel_launcher.py:11: FutureWarn
ing: The pandas.datetime class is deprecated and will be removed from pandas in
a future version. Import from datetime module instead.
  # This is added back by InteractiveShellApp.init_path()
```



File failed to load: /extensions/MathMenu.js

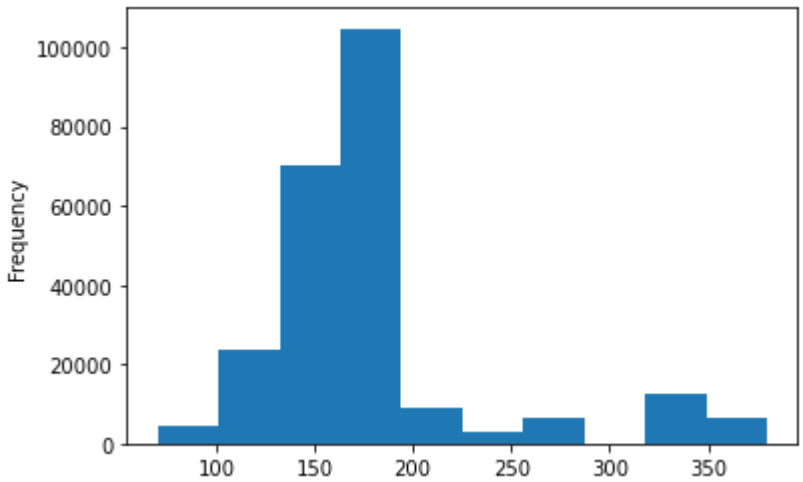
In [84]:
```
# We can see that the increase in sales occurs in the lead-up to Christmas a
# there are zero sales on Christmas day itself. This is due to shops being c
# Christmas day
```

In [86]:
```
# #### We can work this out by taking the digits that are in PROD_NAME
# transactionData[, PACK_SIZE := parse_number(PROD_NAME)]
# #### Always check your output
# #### Let's check if the pack sizes look sensible
# transactionData[, .N, PACK_SIZE][order(PACK_SIZE)]
df["PROD_NAME"] = df["PROD_NAME"].str.replace(r'[0-9]+(G)','g')
pack_sizes = df["PROD_NAME"].str.extract(r'([0-9]+[gG])')[0].str.replace("g"
pack_sizes.plot.hist()
```

```
C:\Users\agamm\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWith
CopyWarning:
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/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

Out[86]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e485153408>
```



File failed to load: /extensions/MathMenu.js

In [ ]:
```
1
```

In [104]:
```
1 # #### Brands
2 # # Over to you! Create a column which contains the brand of the product, by
3 # extracting it from the product name.
4 df["PROD_NAME"].str.split().str[0].value_counts()
```

Out[104]:
```
Kettle        41288
Smiths        27390
Pringles      25102
Doritos       22041
Thins         14075
RRD           11894
Infuzions     11057
WW            10320
Cobs           9693
Tostitos       9471
Twisties       9454
Tyrrells       6442
Grain          6272
Natural        6050
Cheezels       4603
CCs            4551
Red            4427
Dorito         3183
Infzns         3144
Smith          2963
Cheetos        2927
Snbts          1576
Burger         1564
Woolworths     1516
GrnWves        1468
Sunbites       1432
NCC            1419
French         1418
Name: PROD_NAME, dtype: int64
```

In [105]:
```
1 # Dorito Doritos
2 # Woolworths WW
3 # RRD Red
```

In [ ]:
```
1
```

In [110]:
```
1 # #### Brands
2 # # Over to you! Create a column which contains the brand of the product, by
3 # extracting it from the product name.
4
```

In [108]:
```
1 df["Brand"] = df["PROD_NAME"].str.split().str[0]
```

```
C:\Users\agamm\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWith
CopyWarning:
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/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

In [109]:
```
1 df
```

Out[109]:

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

246740 rows × 11 columns

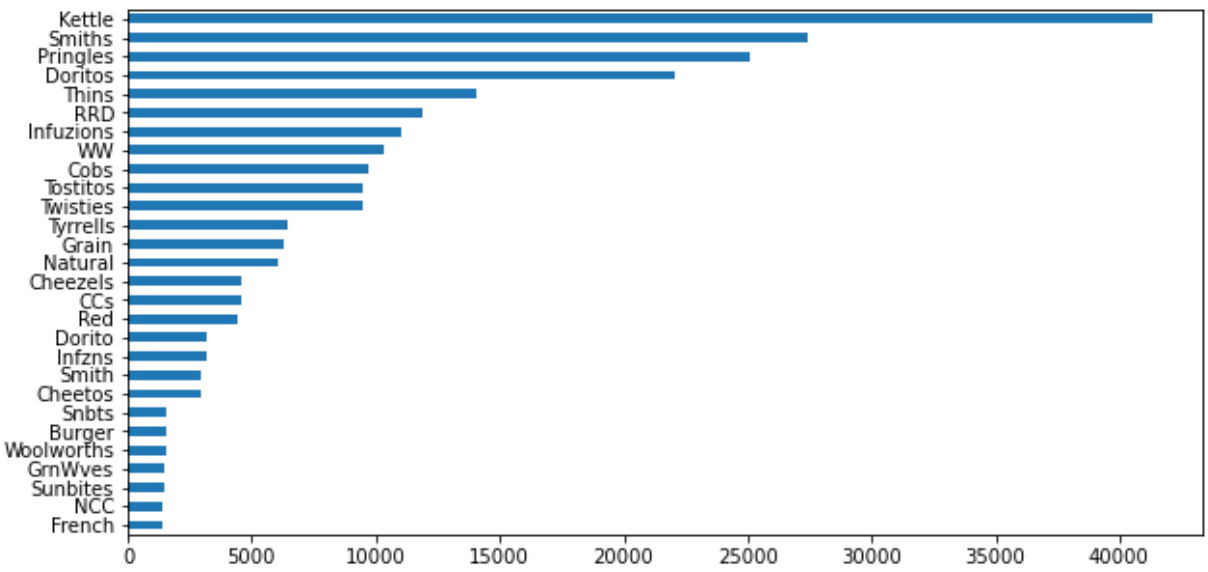
In [113]:
```
def clean(brand):
#     brand = line["Cleaned_Brand_Names"]
    if brand == "Dorito":
        return "Doritos"
    elif brand == "GrnWves" or brand == "Grain":
        return "Grain Waves"
    elif brand == "Infzns":
        return "Infuzions"
    elif brand == "Natural" or brand == "NCC":
        return "Natural Chip Co"
    elif brand == "Red":
        return "RRD"
    elif brand == "Smith":
        return "Smiths"
    elif brand == "Snbts":
        return "Sunbites"
    elif brand == "WW":
        return "Woolworths"
    else:
        return brand
```

In [114]:
```
for brand in df['Brand']:
    brand = clean(brand)
```

In [124]:
```
df["Brand"].value_counts(ascending=True).plot.barh(figsize=(10,5))

```

Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4bf177788>



In [ ]:
```

```

In [125]:
```
transact = pd.read_csv("QVI_purchase_behaviour.csv")
```

In [126]:
```
1 transact
```

Out[126]:

	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 [127]:
```
1 transact.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
```

In [129]:
```
1 transact.isna().sum()
```

Out[129]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [133]:
```
1 merged = pd.merge(df, transact, on="LYLTY_CARD_NBR", how="left")
```

In [134]:
```
1 merged.isna().sum()
```

Out[134]:
```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
Amount_in_grams   0
Product_Name      0
Brand             0
LIFESTAGE         0
PREMIUM_CUSTOMER  0
dtype: int64
```

In [135]:
```
1 merged
```

Out[135]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
...	...	...	...	...	...	...	...
246735	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2
246736	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	1
246737	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	2
246738	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2
246739	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	2

246740 rows × 13 columns

In [136]:
```
# Note that if you are continuing with Task 2, you may want to retain this d
# which you can write out as a csv
# ```{r Code to save dataset as a csv}
# fwrite(data, paste0(filePath,"QVI_data.csv"))
merged.to_csv("retain.csv", index=False)
```

In [137]:
```
# - Who spends the most on chips (total sales), describing customers by life
# how premium their general purchasing behaviour is
# - How many customers are in each segment
# - How many chips are bought per customer by segment
# - What's the average chip price by customer segment
# We could also ask our data team for more information. Examples are:
# - The customer's total spend over the period and total spend for each tran
# to understand what proportion of their grocery spend is on chips
# - Proportion of customers in each customer segment overall to compare agai
# mix of customers who purchase chips
# Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER
# plotting the split by these segments to describe which customer segment co
# most to chip sales.
# ```{r fig.width = 10, fig.align = "center"}
# #### Total sales by LIFESTAGE and PREMIUM_CUSTOMER
# # Over to you! Calculate the summary of sales by those dimensions and crea
# plot.
```

In [147]:
```
grouped_sales = pd.DataFrame(merged.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"
```

In [148]:
```
1 grouped_sales.sort_values(ascending=False, by="sum")
```

Out[148]:

		sum	mean
LIFESTAGE	PREMIUM_CUSTOMER		
OLDER FAMILIES	Budget	156863.75	7.291241
YOUNG SINGLES/COUPLES	Mainstream	147582.20	7.551279
RETIREES	Mainstream	145168.95	7.269352
YOUNG FAMILIES	Budget	129717.95	7.302705
OLDER SINGLES/COUPLES	Budget	127833.60	7.444305
	Mainstream	124648.50	7.306049
	Premium	123537.55	7.459997
RETIREES	Budget	105916.30	7.445786
OLDER FAMILIES	Mainstream	96413.55	7.281440
RETIREES	Premium	91296.65	7.461315
YOUNG FAMILIES	Mainstream	86338.25	7.226772
MIDAGE SINGLES/COUPLES	Mainstream	84734.25	7.637156
YOUNG FAMILIES	Premium	78571.70	7.285951
OLDER FAMILIES	Premium	75242.60	7.232779
YOUNG SINGLES/COUPLES	Budget	57122.10	6.663023
MIDAGE SINGLES/COUPLES	Premium	54443.85	7.152371
YOUNG SINGLES/COUPLES	Premium	39052.30	6.673325
MIDAGE SINGLES/COUPLES	Budget	33345.70	7.108442
NEW FAMILIES	Budget	20607.45	7.297256
	Mainstream	15979.70	7.313364
	Premium	10760.80	7.231720

In [155]:
```
1
2 grouped_sales['sum']
```

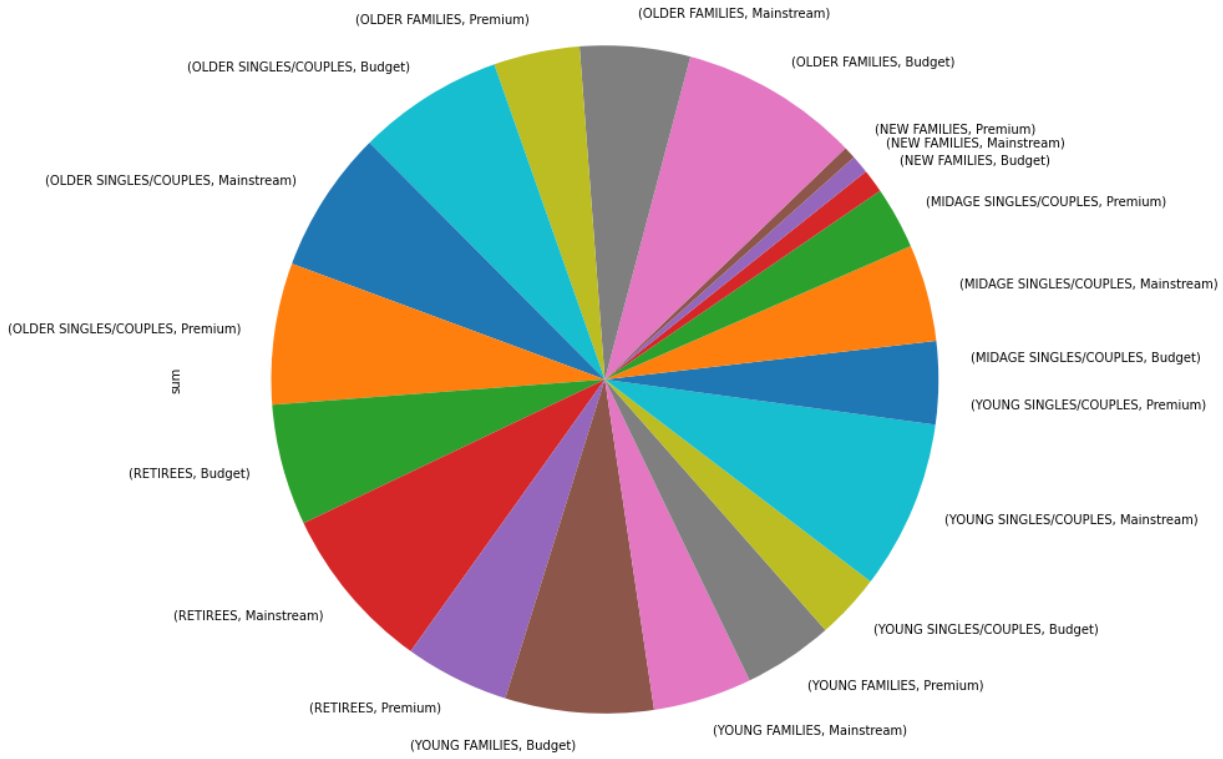
Out[155]:

LIFESTAGE	PREMIUM_CUSTOMER	
MIDAGE SINGLES/COUPLES	Budget	33345.70
	Mainstream	84734.25
	Premium	54443.85
NEW FAMILIES	Budget	20607.45
	Mainstream	15979.70
	Premium	10760.80
OLDER FAMILIES	Budget	156863.75
	Mainstream	96413.55
	Premium	75242.60
OLDER SINGLES/COUPLES	Budget	127833.60
	Mainstream	124648.50
	Premium	123537.55
RETIREES	Budget	105916.30
	Mainstream	145168.95
	Premium	91296.65
YOUNG FAMILIES	Budget	129717.95
	Mainstream	86338.25
	Premium	78571.70
YOUNG SINGLES/COUPLES	Budget	57122.10
	Mainstream	147582.20
	Premium	39052.30

Name: sum, dtype: float64

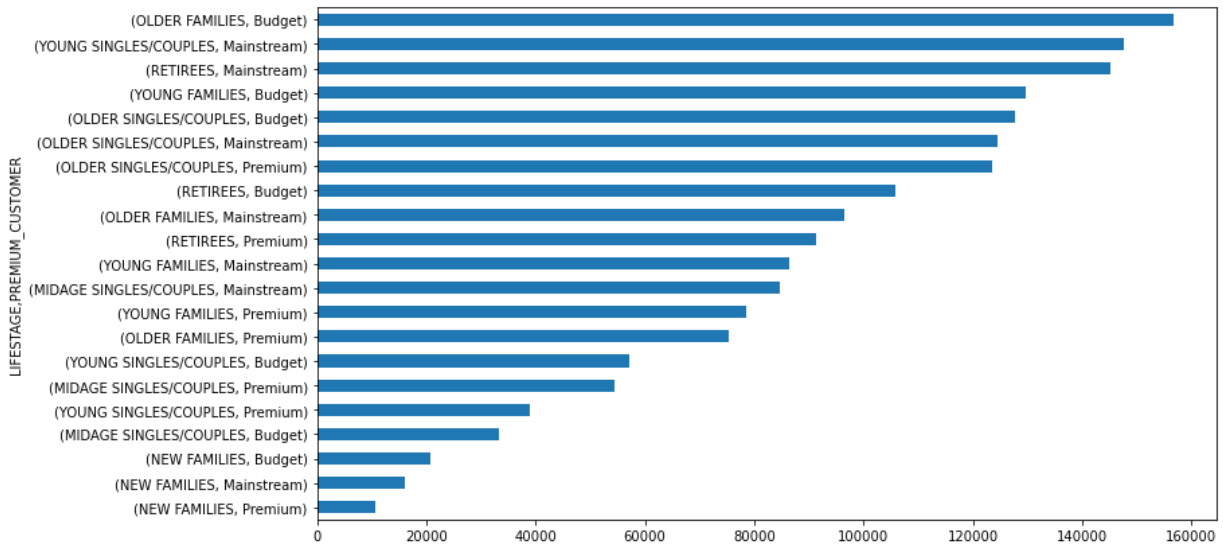
In [161]:
```
plt.figure(figsize=(12, 18))
grouped_sales['sum'].plot(kind="pie")
```

Out[161]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4bdb980c8>



In [150]:
```
grouped_sales["sum"].sort_values().plot.barh(figsize=(12,7))
```

Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4bde0e808>



In [ ]:
```

```

File failed to load: /extensions/MathMenu.js

In [169]:
```
1 merged.groupby("LIFESTAGE")["PREMIUM_CUSTOMER"].agg(pd.Series.mode).sort_val
```

Out[169]:
```
LIFESTAGE
NEW FAMILIES                  Budget
OLDER FAMILIES                Budget
OLDER SINGLES/COUPLES         Budget
YOUNG FAMILIES                Budget
MIDAGE SINGLES/COUPLES    Mainstream
RETIREES                  Mainstream
YOUNG SINGLES/COUPLES     Mainstream
Name: PREMIUM_CUSTOMER, dtype: object
```

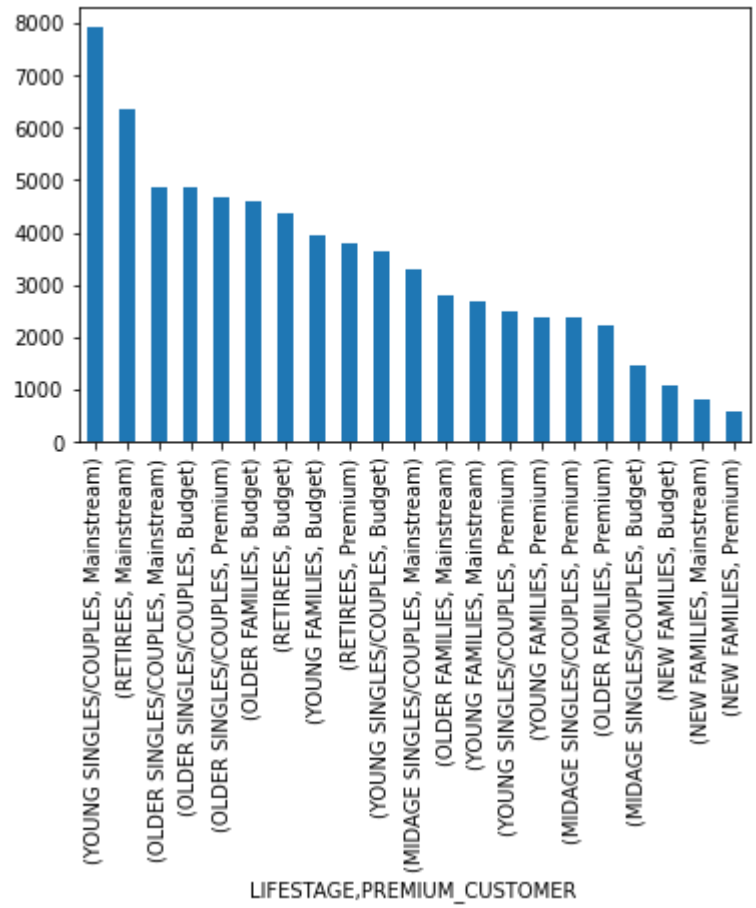
In [182]:
```
1 unique_cust = merged.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["LYLTY_CARD_
2 pd.DataFrame(unique_cust)
```

Out[182]:

		LYLTY_CARD_NBR
LIFESTAGE	**PREMIUM_CUSTOMER**	
YOUNG SINGLES/COUPLES	**Mainstream**	7917
RETIREES	**Mainstream**	6358
OLDER SINGLES/COUPLES	**Mainstream**	4858
	Budget	4849
	Premium	4682
OLDER FAMILIES	**Budget**	4611
RETIREES	**Budget**	4385
YOUNG FAMILIES	**Budget**	3953
RETIREES	**Premium**	3812
YOUNG SINGLES/COUPLES	**Budget**	3647
MIDAGE SINGLES/COUPLES	**Mainstream**	3298
OLDER FAMILIES	**Mainstream**	2788
YOUNG FAMILIES	**Mainstream**	2685
YOUNG SINGLES/COUPLES	**Premium**	2480
YOUNG FAMILIES	**Premium**	2398
MIDAGE SINGLES/COUPLES	**Premium**	2369
OLDER FAMILIES	**Premium**	2231
MIDAGE SINGLES/COUPLES	**Budget**	1474
NEW FAMILIES	**Budget**	1087
	Mainstream	830
	Premium	575

In [183]:
```
1 unique_cust.plot(kind="bar")
```

Out[183]: <matplotlib.axes._subplots.AxesSubplot at 0x1e4bd9e8f88>



In [184]:
```
freq_per_cust = merged.groupby(["LYLTY_CARD_NBR", "LIFESTAGE", "PREMIUM_CUST
freq_per_cust.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"]).agg(["mean", "count
```

Out[184]:

		mean	count
LIFESTAGE	**PREMIUM_CUSTOMER**		
OLDER FAMILIES	**Mainstream**	4.749283	2788
	Budget	4.665799	4611
	Premium	4.662931	2231
YOUNG FAMILIES	**Premium**	4.497081	2398
	Budget	4.493549	3953
	Mainstream	4.449534	2685
OLDER SINGLES/COUPLES	**Budget**	3.541349	4849
	Premium	3.536950	4682
	Mainstream	3.511939	4858
MIDAGE SINGLES/COUPLES	**Mainstream**	3.364160	3298
RETIREES	**Budget**	3.244014	4385
MIDAGE SINGLES/COUPLES	**Premium**	3.213170	2369
RETIREES	**Premium**	3.209864	3812
MIDAGE SINGLES/COUPLES	**Budget**	3.182497	1474
RETIREES	**Mainstream**	3.140925	6358
NEW FAMILIES	**Mainstream**	2.632530	830
	Budget	2.597976	1087
	Premium	2.587826	575
YOUNG SINGLES/COUPLES	**Mainstream**	2.468612	7917
	Premium	2.359677	2480
	Budget	2.350699	3647

In [185]:
```
grouped_sales.sort_values(ascending=False, by="mean")

```

Out[185]:

		sum	mean
LIFESTAGE	**PREMIUM_CUSTOMER**		
MIDAGE SINGLES/COUPLES	**Mainstream**	84734.25	7.637156
YOUNG SINGLES/COUPLES	**Mainstream**	147582.20	7.551279
RETIREES	**Premium**	91296.65	7.461315
OLDER SINGLES/COUPLES	**Premium**	123537.55	7.459997
RETIREES	**Budget**	105916.30	7.445786
OLDER SINGLES/COUPLES	**Budget**	127833.60	7.444305
NEW FAMILIES	**Mainstream**	15979.70	7.313364
OLDER SINGLES/COUPLES	**Mainstream**	124648.50	7.306049
YOUNG FAMILIES	**Budget**	129717.95	7.302705
NEW FAMILIES	**Budget**	20607.45	7.297256
OLDER FAMILIES	**Budget**	156863.75	7.291241
YOUNG FAMILIES	**Premium**	78571.70	7.285951
OLDER FAMILIES	**Mainstream**	96413.55	7.281440
RETIREES	**Mainstream**	145168.95	7.269352
OLDER FAMILIES	**Premium**	75242.60	7.232779
NEW FAMILIES	**Premium**	10760.80	7.231720
YOUNG FAMILIES	**Mainstream**	86338.25	7.226772
MIDAGE SINGLES/COUPLES	**Premium**	54443.85	7.152371
	Budget	33345.70	7.108442
YOUNG SINGLES/COUPLES	**Premium**	39052.30	6.673325
	Budget	57122.10	6.663023

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```