

Quantium Task 1

August 15, 2024

```
[1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import datetime as dt
```

```
[2]: transactions = (r'C:\Users\user\Documents\forage data analysis tasks\task_
↳1\QVI_transaction_data.xlsx')
```

```
[3]: df = pd.read_excel(transactions)
df2 = pd.read_csv(r'C:\Users\user\Documents\forage data analysis tasks\task_
↳1\QVI_purchase_behaviour.csv')
```

```
[4]: #convert the date into a proper date format
df['DATE'] = pd.to_datetime(df['DATE'], origin='1899-12-30', unit='D')
```

```
[5]: df.head()
```

```
[5]:
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	2018-10-17	1	1000	1	5	
1	2019-05-14	1	1307	348	66	
2	2019-05-20	1	1343	383	61	
3	2018-08-17	2	2373	974	69	
4	2018-08-18	2	2426	1038	108	

	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

```
[6]: df.describe()
```

```
[6]:
```

	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	1.000000	1.000000e+03
25%	2018-09-30 00:00:00	70.000000	7.002100e+04
50%	2018-12-30 00:00:00	130.000000	1.303575e+05
75%	2019-03-31 00:00:00	203.000000	2.030942e+05
max	2019-06-30 00:00:00	272.000000	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

```
[7]: #exploring the colums for missing values
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  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
```

```
[8]: #data frame 2
df2.head()
```

	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

```
[9]: df2.tail()
```

```
[9]:
```

	LYLTY_CARD_NBR		LIFESTAGE	PREMIUM_CUSTOMER
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

```
[10]: df2.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
```

```
[11]: #exploring columns and count
df2.describe()
```

```
[11]:
```

	LYLTY_CARD_NBR
count	7.263700e+04
mean	1.361859e+05
std	8.989293e+04
min	1.000000e+03
25%	6.620200e+04
50%	1.340400e+05
75%	2.033750e+05
max	2.373711e+06

```
[12]: #exploring columns and count
df2['PREMIUM_CUSTOMER'].value_counts()
```

```
[12]: PREMIUM_CUSTOMER
Mainstream    29245
Budget        24470
Premium       18922
Name: count, dtype: int64
```

```
[13]: #checking and removing outliers
merged_data = pd.merge(df,df2,on ="LYLTY_CARD_NBR",how = "right")
merged_data.head()
```

```
[13]:
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	\
0	2018-10-17	1	1000	1	5	

1	2018-09-16	1	1002	2	58
2	2019-03-07	1	1003	3	52
3	2019-03-08	1	1003	4	106
4	2018-11-02	1	1004	5	96

	PROD_NAME	PROD_QTY	TOT_SALES	\
0	Natural Chip Compny SeaSalt175g	2	6.0	
1	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	
2	Grain Waves Sour Cream&Chives 210G	1	3.6	
3	Natural ChipCo Hony Soy Chckn175g	1	3.0	
4	WW Original Stacked Chips 160g	1	1.9	

	LIFESTAGE	PREMIUM_CUSTOMER
0	YOUNG SINGLES/COUPLES	Premium
1	YOUNG SINGLES/COUPLES	Mainstream
2	YOUNG FAMILIES	Budget
3	YOUNG FAMILIES	Budget
4	OLDER SINGLES/COUPLES	Mainstream

```
[14]: len(merged_data)
len(df)
```

```
[14]: 264836
```

```
[15]: merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 10 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
8   LIFESTAGE            264836 non-null  object
9   PREMIUM_CUSTOMER     264836 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
memory usage: 20.2+ MB
```

```
[16]: #check the product name to make sure all items are chips
merged_data['PROD_NAME'].unique()
```

```
[16]: array(['Natural Chip          Compny SeaSalt175g',
'Red Rock Deli Chikn&Garlic Aioli 150g',
'Grain Waves Sour      Cream&Chives 210G',
'Natural ChipCo        Hony Soy Chckn175g',
'WW Original Stacked Chips 160g', 'Cheetos Puffs 165g',
'Infuzions SourCream&Herbs Veg Strws 110g',
'RRD SR Slow Rst       Pork Belly 150g',
'Doritos Cheese        Supreme 330g', 'Doritos Mexicana    170g',
'Old El Paso Salsa    Dip Tomato Med 300g',
'GrnWves Plus Btroot & Chilli Jam 180g',
'Smiths Crinkle Cut   Chips Barbecue 170g',
'Kettle Sensations    Camembert & Fig 150g',
'Doritos Corn Chip Southern Chicken 150g',
'CCs Tasty Cheese     175g', 'Tostitos Splash Of  Lime 175g',
'Kettle 135g Swt Pot Sea Salt', 'RRD Salt & Vinegar  165g',
'Infuzions Mango      Chutny Papadums 70g',
'Smiths Crinkle Cut   Snag&Sauce 150g',
'Smiths Crinkle       Original 330g',
'RRD Sweet Chilli &   Sour Cream 165g',
'Smiths Chip Thinly   S/Cream&Onion 175g',
'Smiths Crinkle Chips Salt & Vinegar 330g',
'Red Rock Deli SR     Salsa & Mzzrlla 150g',
'Cobs Popd Sea Salt   Chips 110g',
'Natural ChipCo Sea    Salt & Vinegr 175g',
'Natural Chip Co      Tmato Hrb&Spce 175g', 'Burger Rings 220g',
'Woolworths Cheese    Rings 190g',
'Smiths Thinly        Swt Chli&S/Cream175G',
'Thins Chips Seasonedchicken 175g',
'Smiths Thinly Cut    Roast Chicken 175g',
'Tyrrells Crisps      Ched & Chives 165g',
'Doritos Corn Chips   Cheese Supreme 170g',
'Smiths Chip Thinly   Cut Original 175g',
'Smiths Crinkle Cut   Chips Original 170g',
'Thins Chips Light&    Tangy 175g',
'Doritos Corn Chips   Original 170g',
'Kettle Sensations    Siracha Lime 150g',
'Smiths Crinkle Cut   Salt & Vinegar 170g',
'Smith Crinkle Cut    Bolognese 150g', 'Cheezels Cheese 330g',
'Kettle Chilli 175g', 'Tyrrells Crisps      Lightly Salted 165g',
'Twisties Cheese       270g', 'WW Crinkle Cut        Chicken 175g',
'RRD Chilli&          Coconut 150g',
'Infuzions BBQ Rib    Prawn Crackers 110g',
'Sunbites Whlegrn     Crisps Frch/Onin 90g',
'Doritos Salsa        Medium 300g',
'Kettle Tortilla ChpsFeta&Garlic 150g',
'Smiths Crinkle Cut   French OnionDip 150g',
'WW D/Style Chip      Sea Salt 200g',
```

'Smiths Chip Thinly CutSalt/Vinegr175g',
 'Kettle Sensations BBQ&Maple 150g',
 'Old El Paso Salsa Dip Tomato Mild 300g',
 'Tostitos Smoked Chipotle 175g', 'RRD Lime & Pepper 165g',
 'CCs Nacho Cheese 175g', 'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
 'Kettle Tortilla ChpsBtroot&Ricotta 150g',
 'Pringles Sthrn FriedChicken 134g',
 'Pringles Chicken Salt Crips 134g',
 'French Fries Potato Chips 175g',
 'Kettle Mozzarella Basil & Pesto 175g', 'CCs Original 175g',
 'Tostitos Lightly Salted 175g',
 'Smiths Crnkle Chip Orgnl Big Bag 380g',
 'Smiths Crinkle Cut Chips Chicken 170g',
 'Smiths Crinkle Cut Chips Chs&Onion170g', 'Twisties Chicken270g',
 'Woolworths Medium Salsa 300g',
 'Red Rock Deli Sp Salt & Truffle 150G',
 'RRD Pc Sea Salt 165g', 'WW Supreme Cheese Corn Chips 200g',
 'WW Original Corn Chips 200g', 'Woolworths Mild Salsa 300g',
 'Cheezels Cheese Box 125g', 'Doritos Salsa Mild 300g',
 'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
 'Infzns Crn Crnchers Tangy Gcamole 110g',
 'WW Sour Cream &OnionStacked Chips 160g',
 'Pringles Mystery Flavour 134g', 'Pringles Barbeque 134g',
 'Grain Waves Sweet Chilli 210g',
 'Pringles Sweet&Spcy BBQ 134g', 'Kettle Original 175g',
 'Infuzions Thai SweetChili PotatoMix 110g',
 'Old El Paso Salsa Dip Chnky Tom Ht300g',
 'Smiths Crinkle Cut Tomato Salsa 150g',
 'Cheetos Chs & Bacon Balls 190g',
 'Kettle Sweet Chilli And Sour Cream 175g',
 'Doritos Corn Chips Nacho Cheese 170g',
 'Cobs Popd Sour Crm &Chives Chips 110g',
 'Red Rock Deli Thai Chilli&Lime 150g',
 'Twisties Cheese Burger 250g',
 'Kettle Sea Salt And Vinegar 175g',
 'WW Crinkle Cut Original 175g',
 'Dorito Corn Chp Supreme 380g',
 'Doritos Corn Chip Mexican Jalapeno 150g',
 'Pringles SourCream Onion 134g',
 'Kettle Tortilla ChpsHny&Jlpno Chili 150g',
 'RRD Steak & Chimuchurri 150g',
 'Thins Chips Salt & Vinegar 175g',
 'Thins Chips Originl saltd 175g',
 'RRD Honey Soy Chicken 165g',
 'Kettle Honey Soy Chicken 175g',
 'NCC Sour Cream & Garden Chives 175g',
 'Pringles Original Crisps 134g',

```

'Smith Crinkle Cut   Mac N Cheese 150g',
'Thins Potato Chips Hot & Spicy 175g', 'Pringles Slt Vingar 134g'],
dtype=object)

```

```

[17]: # Step 1: Remove weight indicators like "100g" or "250G"
cleaned_names = merged_data["PROD_NAME"].str.replace(r'([0-9]+[gG])', '',
↳ regex=True)

# Step 2: Replace non-word characters with spaces
cleaned_names = cleaned_names.str.replace(r'[^\w]', ' ', regex=True)

# Step 3: Split the product names into lists of words
split_prods = cleaned_names.str.split()

```

```

[18]: from collections import Counter

# Flatten the list of lists and count the occurrences of each word
word_counts = Counter([word for line in split_prods for word in line])

# Convert the counter to a pandas Series and sort by values
word_counts_series = pd.Series(word_counts).sort_values(ascending=False)

# Print the sorted word counts
print(word_counts_series)

```

```

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

```

```

[19]: print(merged_data.describe())
print(merged_data.info())

```

	DATE	STORE_NBR	LYLTY_CARD_NBR \
count	264836	264836.00000	2.648360e+05
mean	2018-12-30 00:52:12.879215360	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
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 264836 entries, 0 to 264835
```

```
Data columns (total 10 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
8     LIFESTAGE   264836 non-null  object
9     PREMIUM_CUSTOMER  264836 non-null  object
```

```
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
```

```
memory usage: 20.2+ MB
```

```
None
```

```
[20]: merged_data["PROD_QTY"].value_counts(bins=4).sort_index()
```

```

[20]: (0.8, 50.75]      264834
      (50.75, 100.5]    0
      (100.5, 150.25]   0
      (150.25, 200.0]    2
      Name: count, dtype: int64
```

```
[21]: merged_data.sort_values(by="PROD_QTY", ascending=False).head()
```

```

[21]:      DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
221626  2019-05-20      226      226000  226210         4
221625  2018-08-19      226      226000  226201         4
145041  2019-05-20     148     148238  148046        87
89729   2018-08-16      93      93152   91631        46
76734   2018-08-15      80      80018   78356        26
```


		PROD_NAME	PROD_QTY	TOT_SALES	\
221626	Dorito Corn Chp	Supreme 380g	200	650.0	
221625	Dorito Corn Chp	Supreme 380g	200	650.0	
145041	Infuzions BBQ Rib	Prawn Crackers 110g	5	19.0	
89729		Kettle Original 175g	5	27.0	
76734	Pringles Sweet&Spcy BBQ	134g	5	18.5	

	LIFESTAGE	PREMIUM_CUSTOMER
221626	OLDER FAMILIES	Premium
221625	OLDER FAMILIES	Premium
145041	OLDER SINGLES/COUPLES	Mainstream
89729	RETIREEES	Premium
76734	MIDAGE SINGLES/COUPLES	Mainstream

```
[22]: #there is an outlier in the value of 200 in the the prod_qty.both entries are
      ↪by the same customer
merged_data = merged_data[merged_data['PROD_QTY'] < 6]
```

```
[23]: len(merged_data[merged_data['LYLTY_CARD_NBR'] == 226000])
```

```
[23]: 0
```

```
[24]: merged_data["DATE"].describe()
```

```
[24]: count                264834
mean      2018-12-30 00:52:10.292937984
min                2018-07-01 00:00:00
25%                2018-09-30 00:00:00
50%                2018-12-30 00:00:00
75%                2019-03-31 00:00:00
max                2019-06-30 00:00:00
Name: DATE, dtype: object
```

```
[25]: #there are 365 days in a year but in the DATE column there are only 364 unique
      ↪values so one is missing
pd.date_range(start=merged_data["DATE"].min(), end=merged_data["DATE"].max()).
      ↪difference(merged_data["DATE"])
```

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

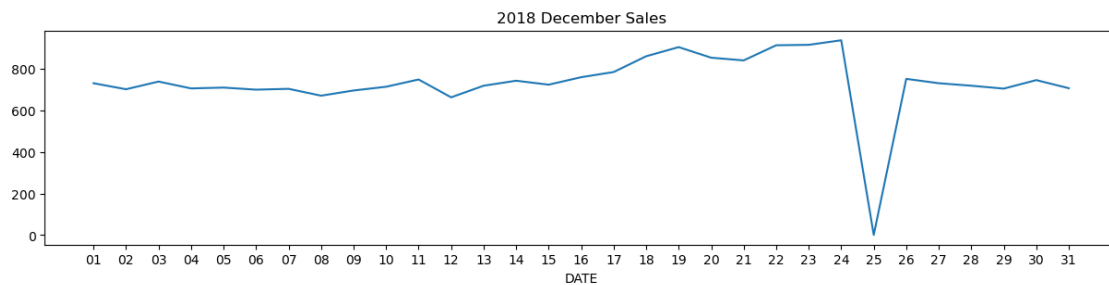
```
[26]: check_null_date = pd.merge(pd.Series(pd.date_range(start=merged_data["DATE"].
      ↪min(), end = merged_data["DATE"].max()), name="DATE"), merged_data, on =
      ↪"DATE", how = "left")
```

```
[27]: trans_by_date = check_null_date["DATE"].value_counts()
dec = trans_by_date[(trans_by_date.index >= pd.Timestamp(2018,12,1)) &
      ↪(trans_by_date.index < pd.Timestamp(2019,1,1))].sort_index()
```

```

dec.index = dec.index.strftime('%d')
ax = dec.plot(figsize=(15,3))
ax.set_xticks(np.arange(len(dec)))
ax.set_xticklabels(dec.index)
plt.title("2018 December Sales")
plt.savefig("2018 December Sales.png", bbox_inches="tight")
plt.show()

```



```
[28]: check_null_date["DATE"].value_counts().sort_values().head()
```

#The day with no transaction is a Christmas day that is when the store is closed. So there is no anomaly in this.

```
[28]: DATE
2018-12-25      1
2018-11-25     648
2018-10-18     658
2019-06-13     659
2019-06-24     662
Name: count, dtype: int64
```

```
[29]: #explore packet size
merged_data["PROD_NAME"] = merged_data["PROD_NAME"].str.
    replace(r'[0-9]+(G)', 'g')
pack_sizes = merged_data["PROD_NAME"].str.extract(r'([0-9]+[gG])')[0].str.
    replace("[gG]", "", regex=True).astype("float")
print(pack_sizes.describe())
pack_sizes.plot.hist()
```

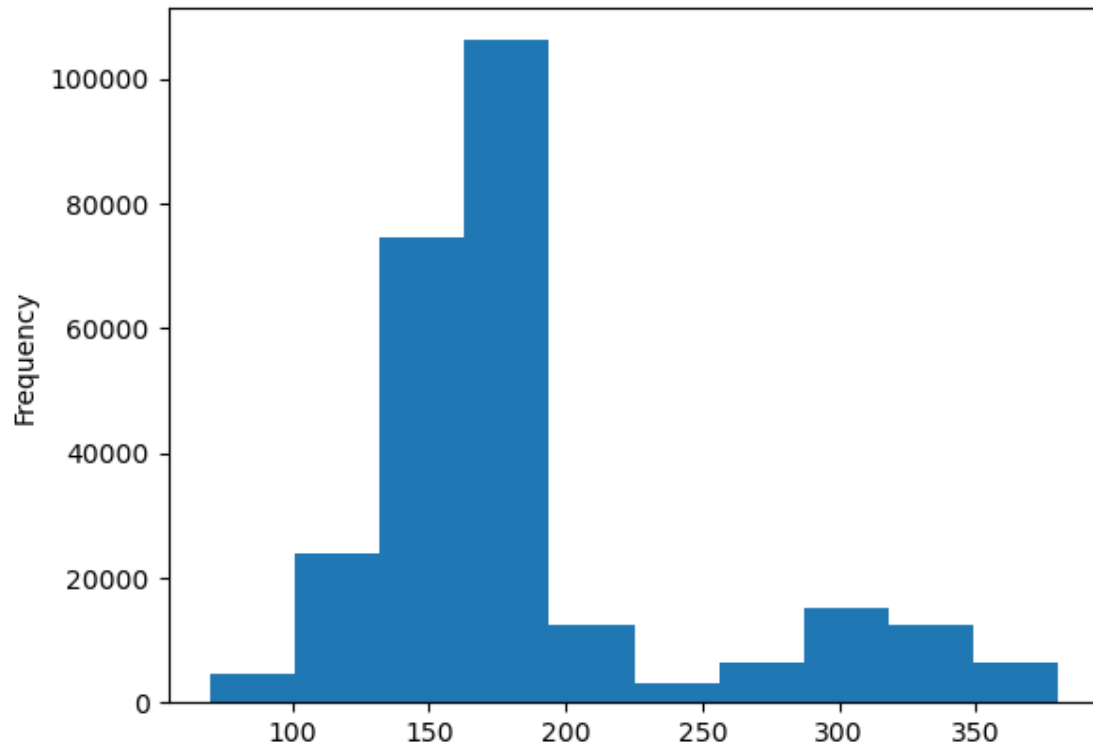
```

count    264834.000000
mean      182.425512
std       64.325148
min       70.000000
25%      150.000000
50%      170.000000
75%      175.000000

```

```
max          380.000000
Name: 0, dtype: float64
```

```
[29]: <Axes: ylabel='Frequency'>
```



```
[30]: (merged_data['PROD_NAME'].str.replace(r'[0-9]+[gG]', '', regex = True).str.  
       ↪split().str[0].value_counts().sort_index())
```

```
[30]: PROD_NAME
Burger      1564
CCs         4551
Cheetos     2927
Cheezels    4603
Cobs        9693
Dorito      3183
Doritos    24962
French      1418
Grain       6272
GrnWves     1468
Infuzions   11057
Infzns      3144
Kettle     41288
NCC         1419
```

Natural	6050
Old	9324
Pringles	25102
RRD	11894
Red	5885
Smith	2963
Smiths	28860
Snbts	1576
Sunbites	1432
Thins	14075
Tostitos	9471
Twisties	9454
Tyrrells	6442
WW	10320
Woolworths	4437

Name: count, dtype: int64

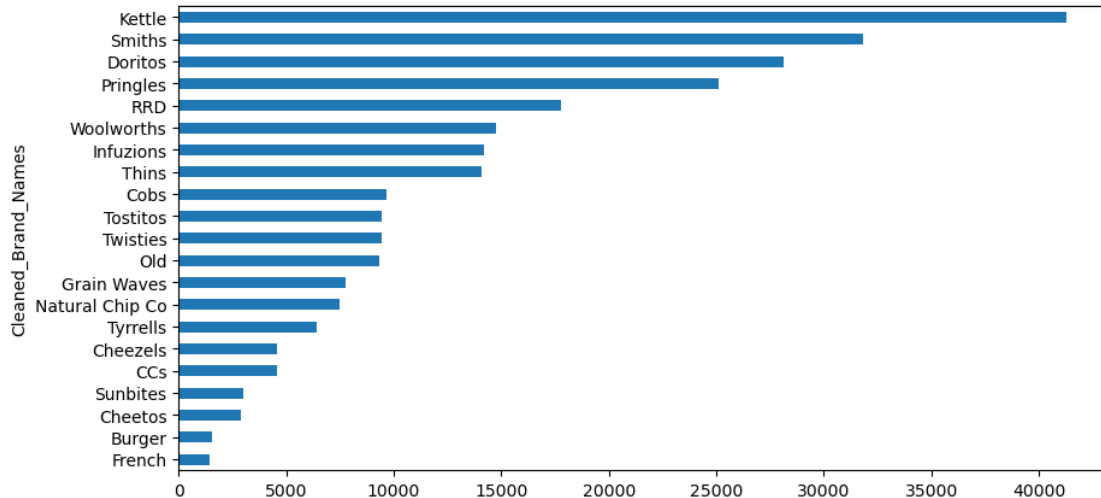
```
[31]: merged_data["Cleaned_Brand_Names"] = merged_data["PROD_NAME"].str.split().str[0]
```

```
[32]: def clean_brand_names(line):
        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
```

```
[33]: merged_data["Cleaned_Brand_Names"] = merged_data.apply(lambda line: ↵
        ↵clean_brand_names(line), axis=1)
```

```
[34]: merged_data["Cleaned_Brand_Names"].value_counts(ascending=True).plot.
        ↵barh(figsize=(10,5))
```

```
[34]: <Axes: ylabel='Cleaned_Brand_Names'>
```



```
[35]: merged_data.isnull().sum()
```

```
[35]: DATE                0
STORE_NBR                0
LYLTY_CARD_NBR          0
TXN_ID                  0
PROD_NBR                 0
PROD_NAME                0
PROD_QTY                 0
TOT_SALES                0
LIFESTAGE                0
PREMIUM_CUSTOMER         0
Cleaned_Brand_Names      0
dtype: int64
```

```
[36]: #Who spends the most on chips (total sales), describing customers by lifestage
      ↳ and 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

grouped_sales = pd.DataFrame(merged_data.groupby(["LIFESTAGE",
      ↳ "PREMIUM_CUSTOMER"])["TOT_SALES"].agg(["sum", "mean"]))
grouped_sales.sort_values(ascending=False, by="sum")
```

```
[36]:
```

LIFESTAGE	PREMIUM_CUSTOMER	sum	mean
OLDER FAMILIES	Budget	168363.25	7.269570
YOUNG SINGLES/COUPLES	Mainstream	157621.60	7.558339
RETIREEES	Mainstream	155677.05	7.252262

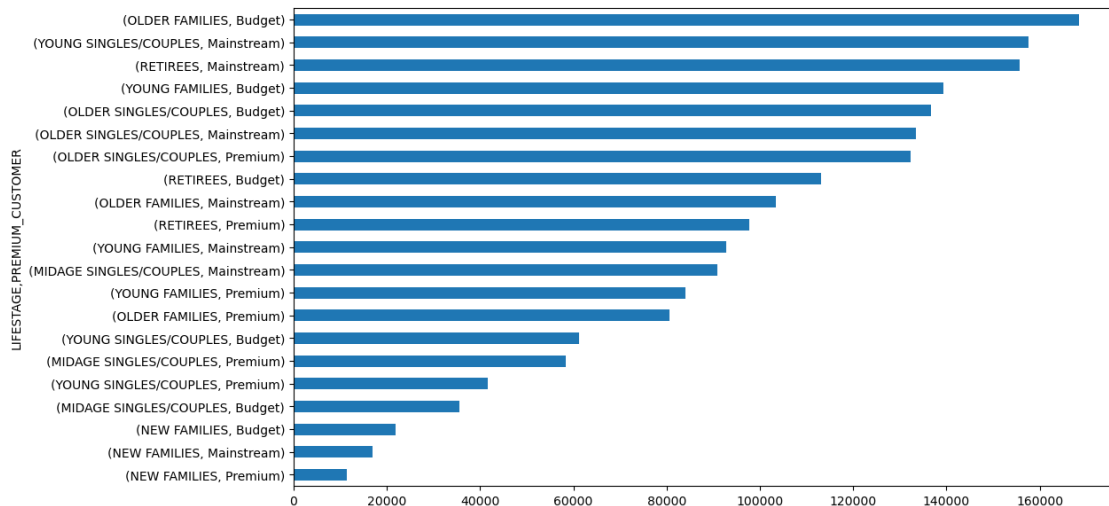
YOUNG FAMILIES	Budget	139345.85	7.287201
OLDER SINGLES/COUPLES	Budget	136769.80	7.430315
	Mainstream	133393.80	7.282116
	Premium	132263.15	7.449766
RETIREEES	Budget	113147.80	7.443445
OLDER FAMILIES	Mainstream	103445.55	7.262395
RETIREEES	Premium	97646.05	7.456174
YOUNG FAMILIES	Mainstream	92788.75	7.189025
MIDAGE SINGLES/COUPLES	Mainstream	90803.85	7.647284
YOUNG FAMILIES	Premium	84025.50	7.266756
OLDER FAMILIES	Premium	80658.40	7.208079
YOUNG SINGLES/COUPLES	Budget	61141.60	6.615624
MIDAGE SINGLES/COUPLES	Premium	58432.65	7.112056
YOUNG SINGLES/COUPLES	Premium	41642.10	6.629852
MIDAGE SINGLES/COUPLES	Budget	35514.80	7.074661
NEW FAMILIES	Budget	21928.45	7.297321
	Mainstream	17013.90	7.317806
	Premium	11491.10	7.231655

```
[37]: grouped_sales["sum"].sum()
```

```
[37]: 1933115.0000000002
```

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

```
[38]: <Axes: ylabel='LIFESTAGE,PREMIUM_CUSTOMER'>
```



```
[39]: # Values of each group
bars1 = grouped_sales[grouped_sales.index.get_level_values("PREMIUM_CUSTOMER")_
↳ == "Budget"] ["sum"]
```

```

bars2 = grouped_sales[grouped_sales.index.get_level_values("PREMIUM_CUSTOMER")
↳ == "Mainstream"] ["sum"]
bars3 = grouped_sales[grouped_sales.index.get_level_values("PREMIUM_CUSTOMER")
↳ == "Premium"] ["sum"]

bars1_text = (bars1 / sum(grouped_sales["sum"])).apply("{:.1%}".format)
bars2_text = (bars2 / sum(grouped_sales["sum"])).apply("{:.1%}".format)
bars3_text = (bars3 / sum(grouped_sales["sum"])).apply("{:.1%}".format)

# Names of group and bar width
names = grouped_sales.index.get_level_values("LIFESTAGE").unique()

# The position of the bars on the x-axis
r = np.arange(len(names))

plt.figure(figsize=(13,5))

# Create brown bars
budget_bar = plt.barh(r, bars1, edgecolor='grey', height=1, label="Budget")
# Create green bars (middle), on top of the first ones
mains_bar = plt.barh(r, bars2, left=bars1, edgecolor='grey', height=1,
↳ label="Mainstream")
# Create green bars (top)
tmp_bar = np.add(bars1, bars2)
prem_bar = plt.barh(r, bars3, left=bars2, edgecolor='grey', height=1,
↳ label="Premium")

# ... (previous code remains the same)

for i in range(7):
    budget_width = budget_bar[i].get_width()
    budget_main_width = budget_width + mains_bar[i].get_width()
    plt.text(budget_width/2, i, bars1_text.iloc[i], va='center', ha='center',
↳ size=8)
    plt.text(budget_width + mains_bar[i].get_width()/2, i, bars2_text.iloc[i],
↳ va='center', ha='center', size=8)
    plt.text(budget_main_width + prem_bar[i].get_width()/2, i, bars3_text.
↳ iloc[i], va='center', ha='center', size=8)

# ... (rest of the code remains the same)

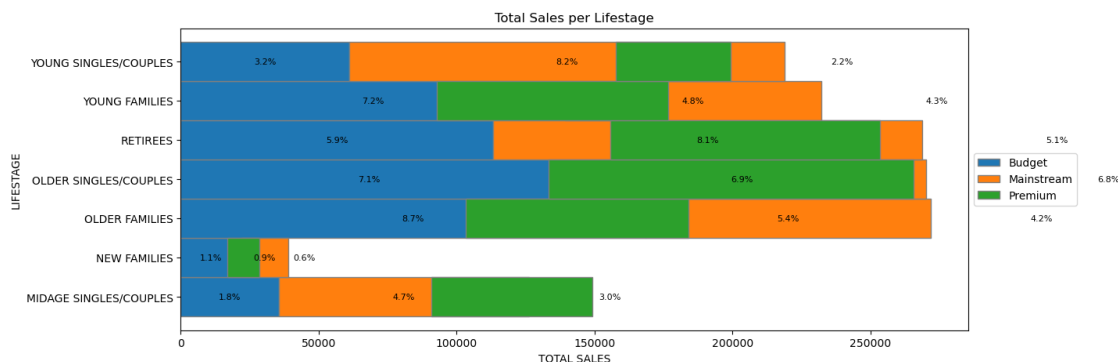
# Custom X axis
plt.yticks(r, names)
plt.ylabel("LIFESTAGE")
plt.xlabel("TOTAL SALES")
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))

```

```
plt.title("Total Sales per Lifestage")

plt.savefig("lifestage_sales.png", bbox_inches="tight")

# Show graphic
plt.show()
```



```
[40]: stage_agg_prem = merged_data.groupby("LIFESTAGE")["PREMIUM_CUSTOMER"].agg(pd.
      ↳Series.mode).sort_values()
print("Top contributor per LIFESTAGE by PREMIUM category")
print(stage_agg_prem)
```

Top contributor per LIFESTAGE by PREMIUM category

LIFESTAGE

NEW FAMILIES	Budget
OLDER FAMILIES	Budget
OLDER SINGLES/COUPLES	Budget
YOUNG FAMILIES	Budget
MIDAGE SINGLES/COUPLES	Mainstream
RETIREEES	Mainstream
YOUNG SINGLES/COUPLES	Mainstream

Name: PREMIUM_CUSTOMER, dtype: object

```
[41]: unique_cust = merged_data.groupby(["LIFESTAGE",
      ↳ "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique().sort_values(ascending=False)
pd.DataFrame(unique_cust)
```

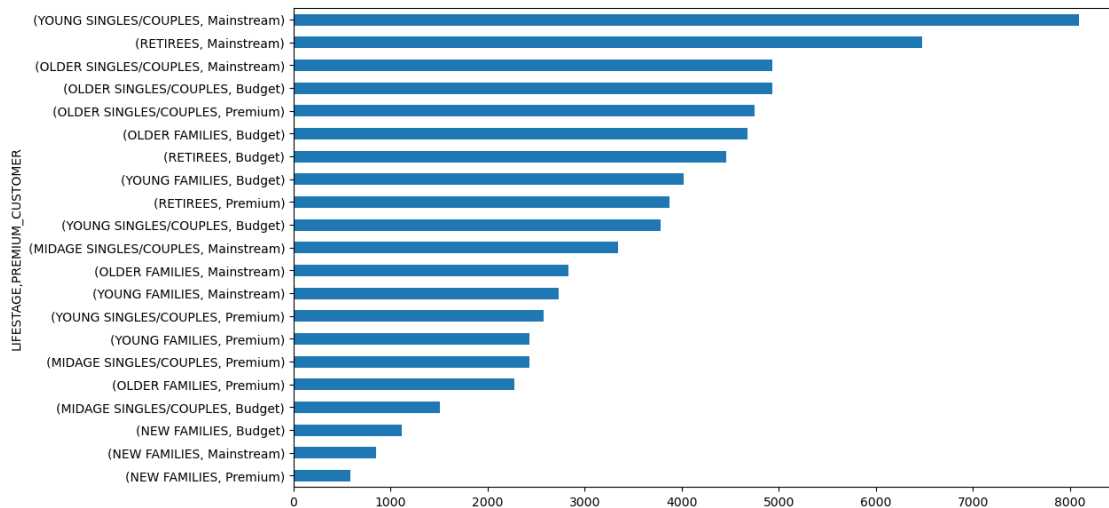
```
[41]:
```

LIFESTAGE	PREMIUM_CUSTOMER	LYLTY_CARD_NBR
YOUNG SINGLES/COUPLES	Mainstream	8088
RETIREEES	Mainstream	6479
OLDER SINGLES/COUPLES	Mainstream	4930
	Budget	4929

	Premium	4750
OLDER FAMILIES	Budget	4675
RETIREEES	Budget	4454
YOUNG FAMILIES	Budget	4017
RETIREEES	Premium	3872
YOUNG SINGLES/COUPLES	Budget	3779
MIDAGE SINGLES/COUPLES	Mainstream	3340
OLDER FAMILIES	Mainstream	2831
YOUNG FAMILIES	Mainstream	2728
YOUNG SINGLES/COUPLES	Premium	2574
YOUNG FAMILIES	Premium	2433
MIDAGE SINGLES/COUPLES	Premium	2431
OLDER FAMILIES	Premium	2273
MIDAGE SINGLES/COUPLES	Budget	1504
NEW FAMILIES	Budget	1112
	Mainstream	849
	Premium	588

```
[42]: unique_cust.sort_values().plot.barh(figsize=(12,7))
```

```
[42]: <Axes: ylabel='LIFESTAGE,PREMIUM_CUSTOMER'>
```



```
[43]: # Values of each group
ncust_bars1 = unique_cust[unique_cust.index.
    ↳get_level_values("PREMIUM_CUSTOMER") == "Budget"]
ncust_bars2 = unique_cust[unique_cust.index.
    ↳get_level_values("PREMIUM_CUSTOMER") == "Mainstream"]
ncust_bars3 = unique_cust[unique_cust.index.
    ↳get_level_values("PREMIUM_CUSTOMER") == "Premium"]
```

```

ncustBars1_text = (ncustBars1 / sum(unique_cust)).apply("{:.1%}".format)
ncustBars2_text = (ncustBars2 / sum(unique_cust)).apply("{:.1%}".format)
ncustBars3_text = (ncustBars3 / sum(unique_cust)).apply("{:.1%}".format)

# # Names of group and bar width
#names = unique_cust.index.get_level_values("LIFESTAGE").unique()

# # The position of the bars on the x-axis
#r = np.arange(len(names))

plt.figure(figsize=(13,5))

# # Create brown bars
budget_bar = plt.barh(r, ncustBars1, edgecolor='grey', height=1,
    ↳label="Budget")
# # Create green bars (middle), on top of the first ones
mains_bar = plt.barh(r, ncustBars2, left=ncustBars1, edgecolor='grey',
    ↳height=1, label="Mainstream")
# # Create green bars (top)
prem_bar = plt.barh(r, ncustBars3, left=ncustBars2, edgecolor='grey',
    ↳height=1, label="Premium")

for i in range(7):
    budget_width = budget_bar[i].get_width()
    budget_main_width = budget_width + mains_bar[i].get_width()
    plt.text(budget_width/2, i, ncustBars1_text.iloc[i], va='center',
    ↳ha='center', size=8)
    plt.text(budget_width + mains_bar[i].get_width()/2, i, ncustBars2_text.
    ↳iloc[i], va='center', ha='center', size=8)
    plt.text(budget_main_width + prem_bar[i].get_width()/2, i, ncustBars3_text.
    ↳iloc[i], va='center', ha='center', size=8)
# Custom X axis
plt.yticks(r, names)
plt.ylabel("LIFESTAGE")
plt.xlabel("UNIQUE CUSTOMERS")
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))

plt.title("Unique Customers per Lifestage")

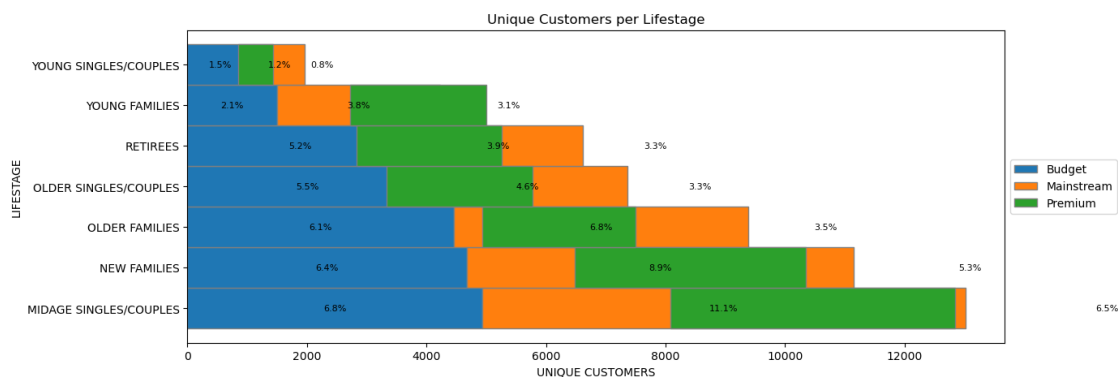
plt.savefig("lifestage_customers.png", bbox_inches="tight")

# # Show graphic
plt.show()

```

#The high sales amount by segment "Young Singles/Couples - Mainstream" and
 ↳ "Retirees - Mainstream" are due to their large number of unique customers,
 ↳ but not for the "Older - Budget" segment. Next we'll explore if the "Older -
 ↳ Budget" segment has:

#High Frequency of Purchase and, Average Sales per Customer compared to the
 ↳ other segment.



```
[44]: freq_per_cust = merged_data.groupby(["LYLTY_CARD_NBR", "LIFESTAGE",  

  ↳ "PREMIUM_CUSTOMER"]).count()["DATE"]  

freq_per_cust.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"]).agg(["mean", "count"]).  

  ↳ sort_values(ascending=False, by="mean")
```

#The above table describes the "Average frequency of Purchase per segment" and
 ↳ "Unique customer per segment". The top three most frequent purchase is
 ↳ contributed by the "Older Families" lifestage segment. We can see now that
 ↳ the "Older - Budget" segment contributes to high sales partly because of the
 ↳ combination of:

#High Frequency of Purchase and, Fairly high unique number of customer in the
 ↳ segment

```
[44]:
```

LIFESTAGE	PREMIUM_CUSTOMER	mean	count
OLDER FAMILIES	Mainstream	5.031438	2831
	Budget	4.954011	4675
	Premium	4.923009	2273
YOUNG FAMILIES	Budget	4.760269	4017
	Premium	4.752569	2433
	Mainstream	4.731305	2728
OLDER SINGLES/COUPLES	Premium	3.737684	4750
	Budget	3.734429	4929
	Mainstream	3.715619	4930

MIDAGE SINGLES/COUPLES	Mainstream	3.555090	3340
RETIREES	Budget	3.412887	4454
	Premium	3.382231	3872
MIDAGE SINGLES/COUPLES	Premium	3.379679	2431
	Budget	3.337766	1504
RETIREES	Mainstream	3.313166	6479
NEW FAMILIES	Mainstream	2.738516	849
	Premium	2.702381	588
	Budget	2.702338	1112
YOUNG SINGLES/COUPLES	Mainstream	2.578388	8088
	Budget	2.445621	3779
	Premium	2.440171	2574

```
[45]: grouped_sales.sort_values(ascending=False, by="mean")
#Highest average spending per purchase are contributed by the Midage and Young
↳ "Singles/Couples".
#The difference between their Mainstream and Non-Mainstream group might seem
↳ insignificant (7.6 vs 6.6), but we'll find out by examining if the
↳ difference is statistically significant.
```

```
[45]:
```

		sum	mean
LIFESTAGE	PREMIUM_CUSTOMER		
MIDAGE SINGLES/COUPLES	Mainstream	90803.85	7.647284
YOUNG SINGLES/COUPLES	Mainstream	157621.60	7.558339
RETIREES	Premium	97646.05	7.456174
OLDER SINGLES/COUPLES	Premium	132263.15	7.449766
RETIREES	Budget	113147.80	7.443445
OLDER SINGLES/COUPLES	Budget	136769.80	7.430315
NEW FAMILIES	Mainstream	17013.90	7.317806
	Budget	21928.45	7.297321
YOUNG FAMILIES	Budget	139345.85	7.287201
OLDER SINGLES/COUPLES	Mainstream	133393.80	7.282116
OLDER FAMILIES	Budget	168363.25	7.269570
YOUNG FAMILIES	Premium	84025.50	7.266756
OLDER FAMILIES	Mainstream	103445.55	7.262395
RETIREES	Mainstream	155677.05	7.252262
NEW FAMILIES	Premium	11491.10	7.231655
OLDER FAMILIES	Premium	80658.40	7.208079
YOUNG FAMILIES	Mainstream	92788.75	7.189025
MIDAGE SINGLES/COUPLES	Premium	58432.65	7.112056
	Budget	35514.80	7.074661
YOUNG SINGLES/COUPLES	Premium	41642.10	6.629852
	Budget	61141.60	6.615624

```
[46]: from scipy.stats import ttest_ind
mainstream = merged_data["PREMIUM_CUSTOMER"] == "Mainstream"
```

```

young_midage = (merged_data["LIFESTAGE"] == "MIDAGE SINGLES/COUPLES") |
↳(merged_data["LIFESTAGE"] == "YOUNG SINGLES/COUPLES")

budget_premium = (merged_data["PREMIUM_CUSTOMER"] == "Budget") |
↳(merged_data["PREMIUM_CUSTOMER"] == "Premium")

a = merged_data[young_midage & mainstream]["TOT_SALES"]
b = merged_data[young_midage & budget_premium]["TOT_SALES"]
stat, pval = ttest_ind(a.values, b.values, equal_var=False)

print(pval)
pval < 0.0000001

#P-Value is close to 0. There is a statistically significant difference to the
↳Total Sales between the "Mainstream Young Midage" segment to the "Budget and
↳Premium Young Midage" segment.

#Next, let's look examine what brand of chips the top 3 segments contributing
↳to Total Sales are buying.

```

1.8542040107536954e-281

[46]: True

```

[47]: merged_data.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["Cleaned_Brand_Names"].
↳agg(pd.Series.mode).sort_values()

```

```

[47]: LIFESTAGE      PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES Budget      Kettle
YOUNG FAMILIES      Premium      Kettle
                   Mainstream     Kettle
                   Budget      Kettle
RETIREES            Premium      Kettle
                   Mainstream     Kettle
                   Budget      Kettle
OLDER SINGLES/COUPLES Premium      Kettle
YOUNG SINGLES/COUPLES Mainstream     Kettle
OLDER SINGLES/COUPLES Mainstream     Kettle
OLDER FAMILIES      Mainstream     Kettle
                   Budget      Kettle
NEW FAMILIES        Premium      Kettle
                   Mainstream     Kettle
                   Budget      Kettle
MIDAGE SINGLES/COUPLES Premium      Kettle
                   Mainstream     Kettle
OLDER SINGLES/COUPLES Budget      Kettle
YOUNG SINGLES/COUPLES Premium      Kettle

```

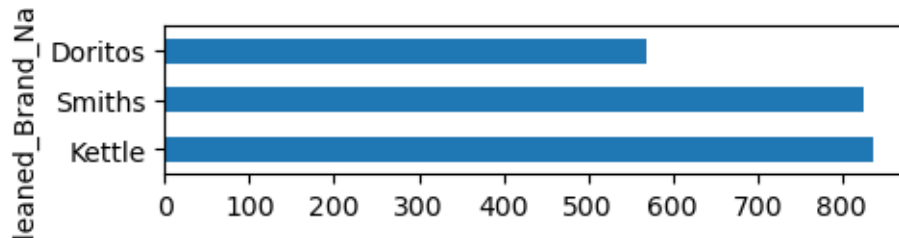
OLDER FAMILIES	Premium	Smiths
YOUNG SINGLES/COUPLES	Budget	Smiths

Name: Cleaned_Brand_Names, dtype: object

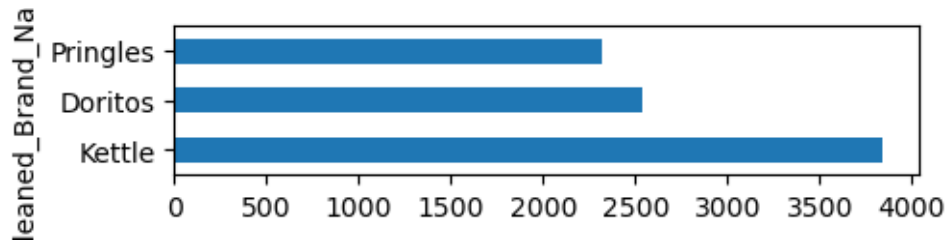
```
[48]: for stage in merged_data["LIFESTAGE"].unique():
      for prem in merged_data["PREMIUM_CUSTOMER"].unique():
          print('=====', stage, '-', prem, '=====')
          summary = merged_data[(merged_data["LIFESTAGE"] == stage) &
          ↪ (merged_data["PREMIUM_CUSTOMER"] == prem)]["Cleaned_Brand_Names"].
          ↪ value_counts().head(3)
          print(summary)
          plt.figure()
          summary.plot.barh(figsize=(5,1))
          plt.show()

#Every segment had Kettle as the most purchased brand. Every segment except
↪ "YOUNG SINGLES/COUPLES Mainstream" had Smiths as their second most purchased
↪ brand.
#"YOUNG SINGLES/COUPLES Mainstream" had Doritos as their second most purchased
↪ brand.
```

```
===== YOUNG SINGLES/COUPLES - Premium =====
Cleaned_Brand_Names
Kettle      838
Smiths      826
Doritos     570
Name: count, dtype: int64
```



```
===== YOUNG SINGLES/COUPLES - Mainstream =====
Cleaned_Brand_Names
Kettle      3844
Doritos     2541
Pringles    2315
Name: count, dtype: int64
```



===== YOUNG SINGLES/COUPLES - Budget =====

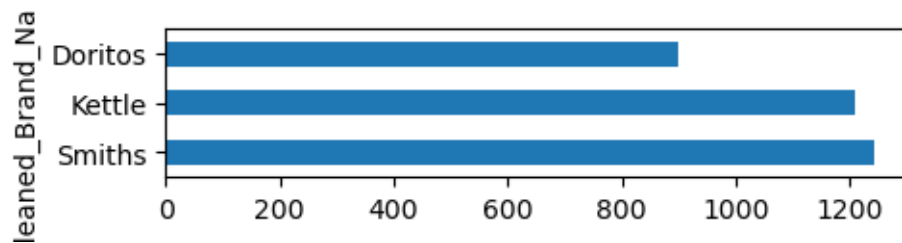
Cleaned_Brand_Names

Smiths 1245

Kettle 1211

Doritos 899

Name: count, dtype: int64



===== YOUNG FAMILIES - Premium =====

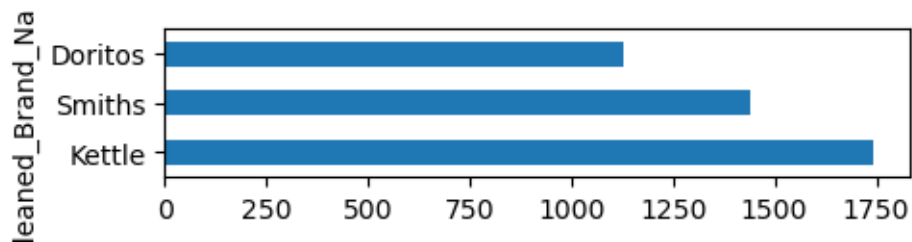
Cleaned_Brand_Names

Kettle 1745

Smiths 1442

Doritos 1129

Name: count, dtype: int64



===== YOUNG FAMILIES - Mainstream =====

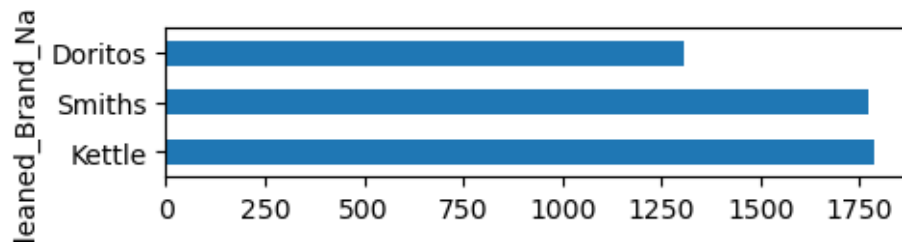
Cleaned_Brand_Names

Kettle 1789

Smiths 1772

Doritos 1309

Name: count, dtype: int64



===== YOUNG FAMILIES - Budget =====

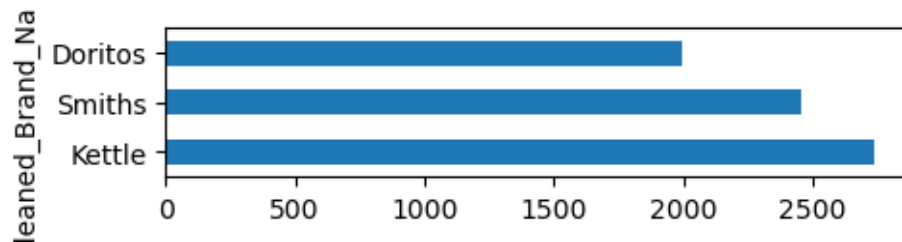
Cleaned_Brand_Names

Kettle 2743

Smiths 2459

Doritos 1996

Name: count, dtype: int64



===== OLDER SINGLES/COUPLES - Premium =====

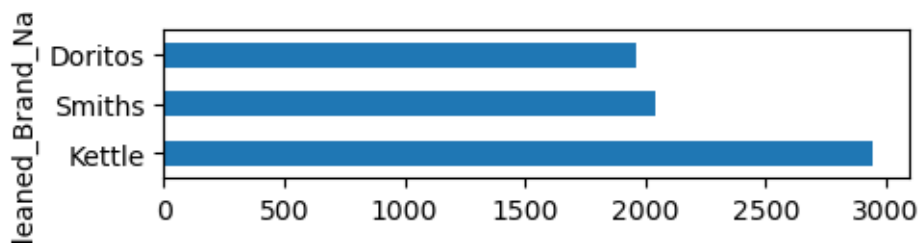
Cleaned_Brand_Names

Kettle 2947

Smiths 2042

Doritos 1958

Name: count, dtype: int64



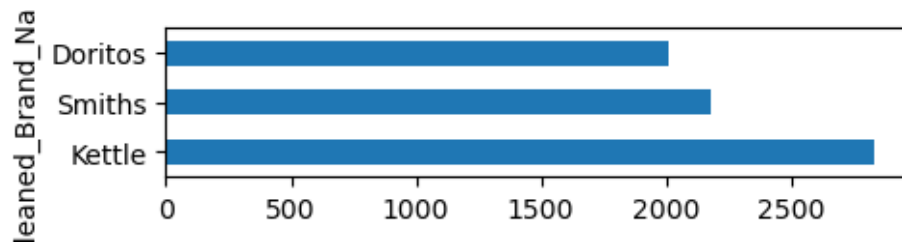
===== OLDER SINGLES/COUPLES - Mainstream =====

Cleaned_Brand_Names

Kettle 2835

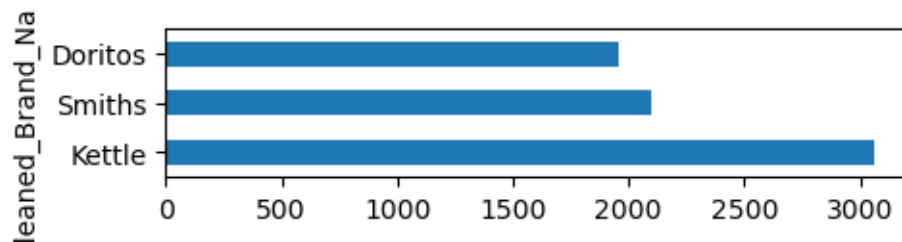
Smiths 2180

Doritos 2008
Name: count, dtype: int64



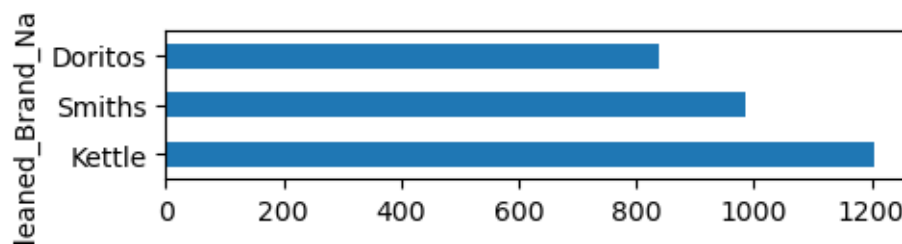
===== OLDER SINGLES/COUPLES - Budget =====

Cleaned_Brand_Names
Kettle 3065
Smiths 2098
Doritos 1954
Name: count, dtype: int64



===== MIDAGE SINGLES/COUPLES - Premium =====

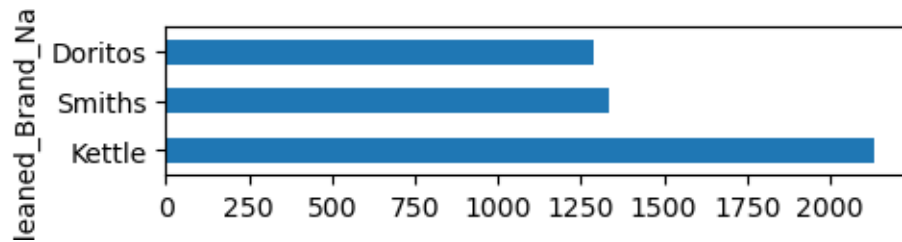
Cleaned_Brand_Names
Kettle 1206
Smiths 986
Doritos 837
Name: count, dtype: int64



===== MIDAGE SINGLES/COUPLES - Mainstream =====

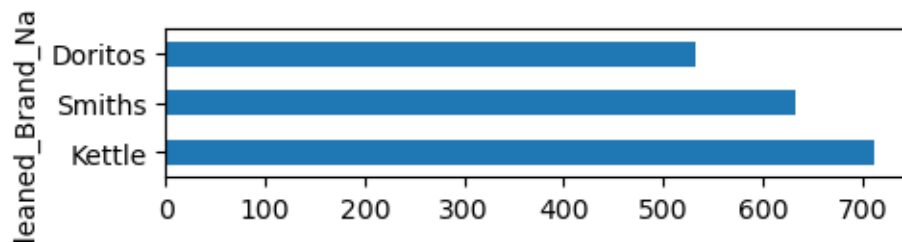
Cleaned_Brand_Names
Kettle 2136

```
Smiths      1337
Doritos     1291
Name: count, dtype: int64
```



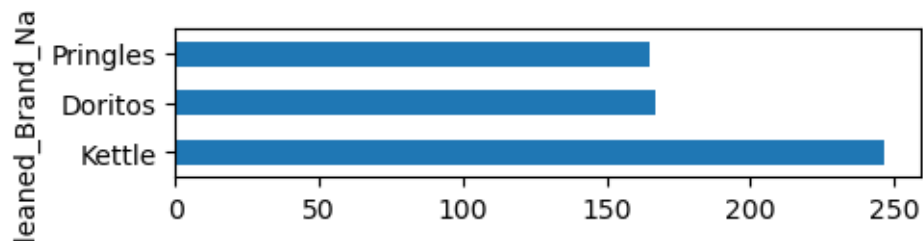
===== MIDAGE SINGLES/COUPLES - Budget =====

```
Cleaned_Brand_Names
Kettle      713
Smiths      633
Doritos     533
Name: count, dtype: int64
```



===== NEW FAMILIES - Premium =====

```
Cleaned_Brand_Names
Kettle      247
Doritos     167
Pringles    165
Name: count, dtype: int64
```



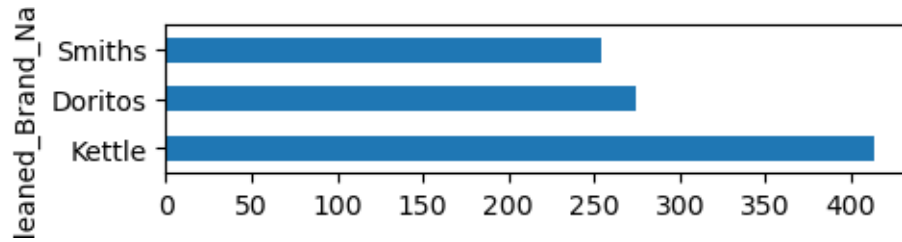
===== NEW FAMILIES - Mainstream =====

```
Cleaned_Brand_Names
```

```

Kettle      414
Doritos     274
Smiths      254
Name: count, dtype: int64

```

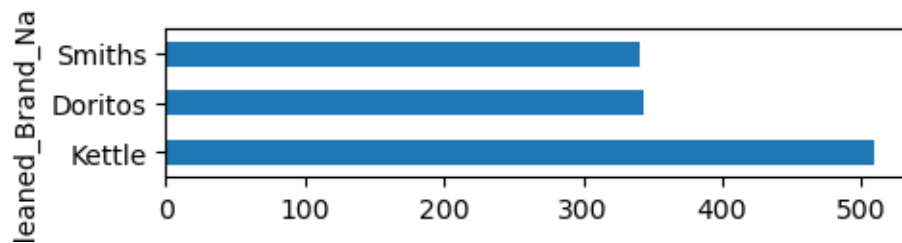


===== NEW FAMILIES - Budget =====

```

Cleaned_Brand_Names
Kettle      510
Doritos     343
Smiths      341
Name: count, dtype: int64

```

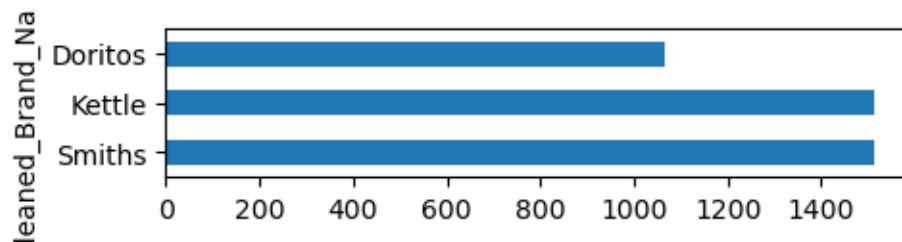


===== OLDER FAMILIES - Premium =====

```

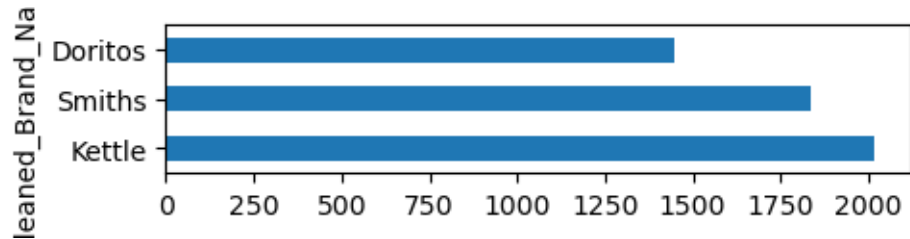
Cleaned_Brand_Names
Smiths      1515
Kettle      1512
Doritos     1065
Name: count, dtype: int64

```



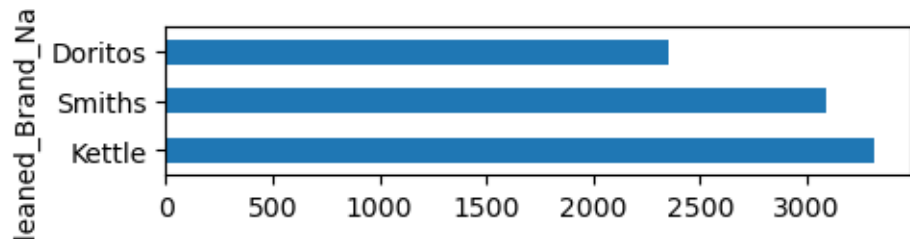
===== OLDER FAMILIES - Mainstream =====

```
Cleaned_Brand_Names
Kettle      2019
Smiths      1835
Doritos     1449
Name: count, dtype: int64
```



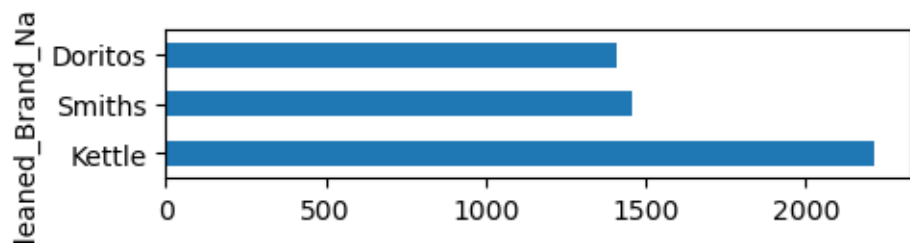
===== OLDER FAMILIES - Budget =====

```
Cleaned_Brand_Names
Kettle      3320
Smiths      3093
Doritos     2351
Name: count, dtype: int64
```



===== RETIREES - Premium =====

```
Cleaned_Brand_Names
Kettle      2216
Smiths      1458
Doritos     1409
Name: count, dtype: int64
```



===== RETIREES - Mainstream =====

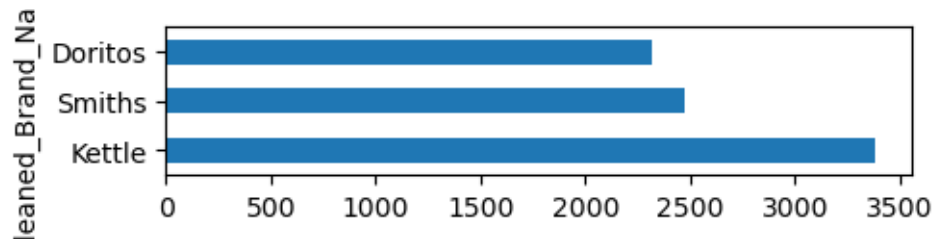
Cleaned_Brand_Names

Kettle 3386

Smiths 2476

Doritos 2320

Name: count, dtype: int64



===== RETIREES - Budget =====

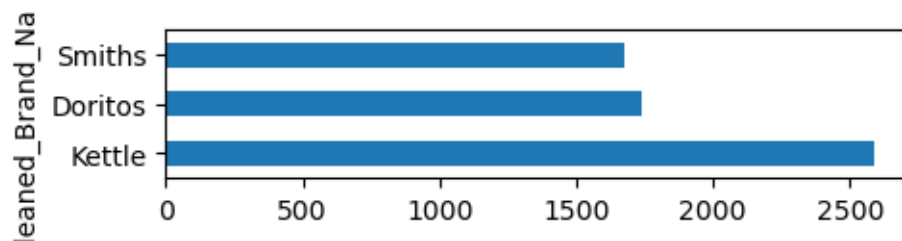
Cleaned_Brand_Names

Kettle 2592

Doritos 1742

Smiths 1679

Name: count, dtype: int64



```
[50]: from mlxtend.frequent_patterns import apriori
      from mlxtend.frequent_patterns import association_rules

temp = merged_data.reset_index().rename(columns = {"index": "transaction"})
temp["Segment"] = temp["LIFESTAGE"] + ' - ' + temp['PREMIUM_CUSTOMER']
segment_brand_encode = pd.concat([pd.get_dummies(temp["Segment"]), pd.
    ↳ get_dummies(temp["Cleaned_Brand_Names"])], axis=1)

frequent_sets = apriori(segment_brand_encode, min_support=0.01,
    ↳ use_colnames=True)
rules = association_rules(frequent_sets, metric="lift", min_threshold=1)

set_temp = temp["Segment"].unique()
```

```
rules[rules["antecedents"].apply(lambda x: list(x)).apply(lambda x: x in
↳set_temp)]

#By looking at our a-priori analysis, we can conclude that Kettle is the brand
↳of choice for most segment.

#Next, we'll find out the pack size preferences of different segments
```

```
[50]:
```

	antecedents	consequents	antecedent support	\
0	(OLDER FAMILIES - Budget)	(Smiths)	0.087451	
3	(OLDER SINGLES/COUPLES - Budget)	(Kettle)	0.069504	
4	(OLDER SINGLES/COUPLES - Premium)	(Kettle)	0.067038	
6	(RETIREEES - Mainstream)	(Kettle)	0.081055	
8	(YOUNG SINGLES/COUPLES - Mainstream)	(Kettle)	0.078744	

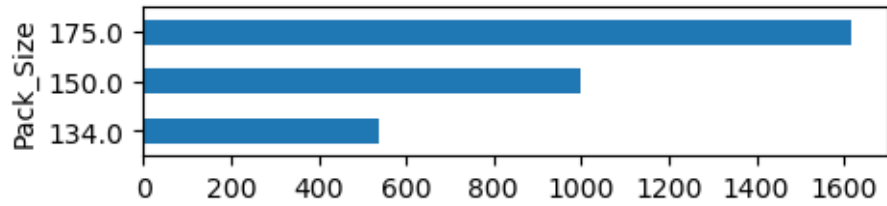
	consequent support	support	confidence	lift	leverage	conviction	\
0	0.120162	0.011679	0.133549	1.111409	0.001171	1.015451	
3	0.155901	0.011573	0.166513	1.068064	0.000738	1.012731	
4	0.155901	0.011128	0.165991	1.064716	0.000676	1.012097	
6	0.155901	0.012785	0.157738	1.011779	0.000149	1.002180	
8	0.155901	0.014515	0.184329	1.182344	0.002239	1.034852	

	zhangs_metric
0	0.109848
3	0.068487
4	0.065150
6	0.012669
8	0.167405

```
[51]: merged_pack = pd.concat([merged_data, pack_sizes.rename("Pack_Size")], axis=1)

for stage in merged_data["LIFESTAGE"].unique():
    for prem in merged_data["PREMIUM_CUSTOMER"].unique():
        print('=====', stage, '-', prem, '=====' )
        summary = merged_pack[(merged_pack["LIFESTAGE"] == stage) &
↳(merged_pack["PREMIUM_CUSTOMER"] == prem)]["Pack_Size"].value_counts().
↳head(3).sort_index()
        print(summary)
        plt.figure()
        summary.plot.barh(figsize=(5,1))
        plt.show()
```

```
===== YOUNG SINGLES/COUPLES - Premium =====
Pack_Size
134.0      537
150.0      998
175.0     1618
Name: count, dtype: int64
```



===== YOUNG SINGLES/COUPLES - Mainstream =====

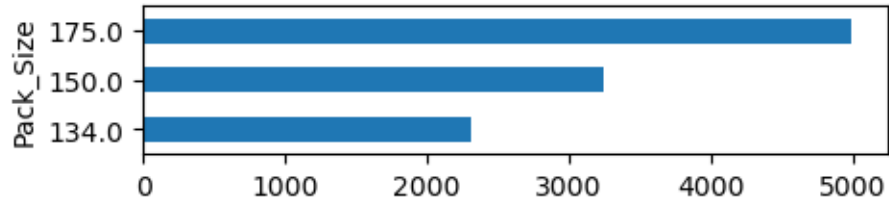
Pack_Size

134.0 2315

150.0 3241

175.0 4997

Name: count, dtype: int64



===== YOUNG SINGLES/COUPLES - Budget =====

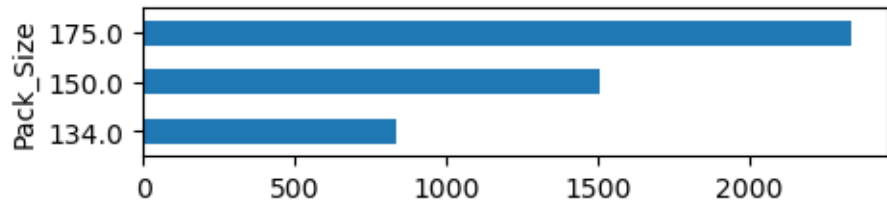
Pack_Size

134.0 832

150.0 1504

175.0 2338

Name: count, dtype: int64



===== YOUNG FAMILIES - Premium =====

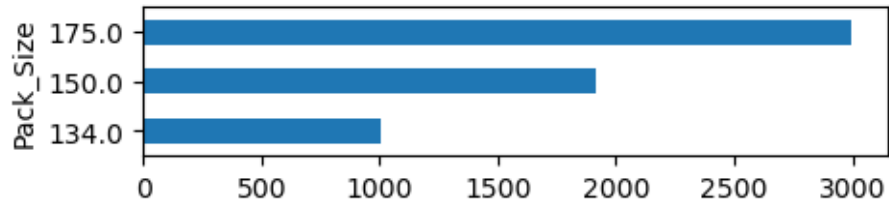
Pack_Size

134.0 1007

150.0 1913

175.0 2998

Name: count, dtype: int64



===== YOUNG FAMILIES - Mainstream =====

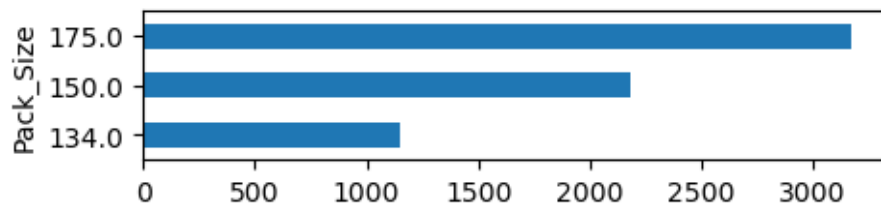
Pack_Size

134.0 1148

150.0 2178

175.0 3174

Name: count, dtype: int64



===== YOUNG FAMILIES - Budget =====

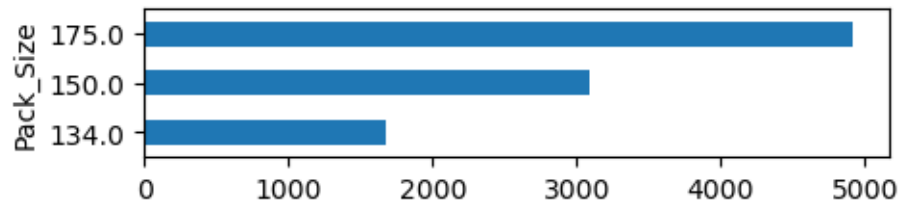
Pack_Size

134.0 1674

150.0 3094

175.0 4921

Name: count, dtype: int64



===== OLDER SINGLES/COUPLES - Premium =====

Pack_Size

134.0 1744

150.0 2950


```
175.0    4458
Name: count, dtype: int64
```



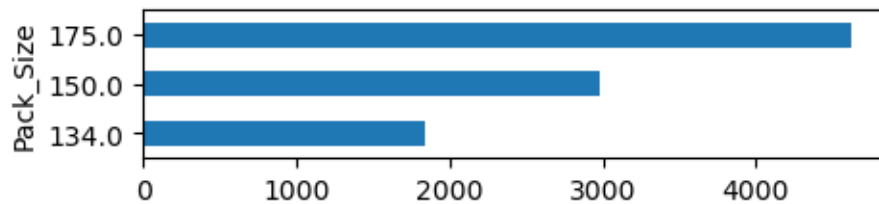
===== OLDER SINGLES/COUPLES - Mainstream =====

```
Pack_Size
134.0    1720
150.0    2988
175.0    4525
Name: count, dtype: int64
```



===== OLDER SINGLES/COUPLES - Budget =====

```
Pack_Size
134.0    1843
150.0    2984
175.0    4625
Name: count, dtype: int64
```



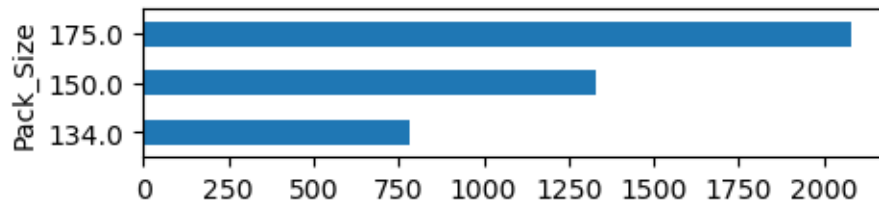
===== MIDAGE SINGLES/COUPLES - Premium =====

```
Pack_Size
134.0    781
```

```

150.0    1329
175.0    2082
Name: count, dtype: int64

```



```

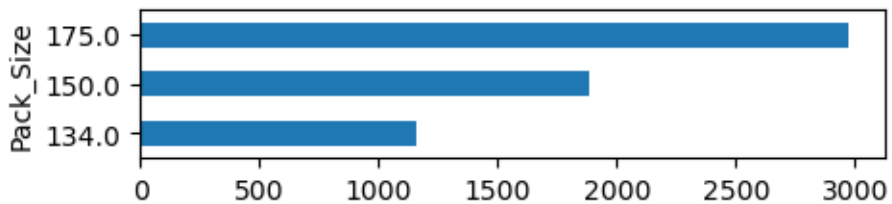
===== MIDAGE SINGLES/COUPLES - Mainstream =====

```

```

Pack_Size
134.0    1159
150.0    1882
175.0    2975
Name: count, dtype: int64

```



```

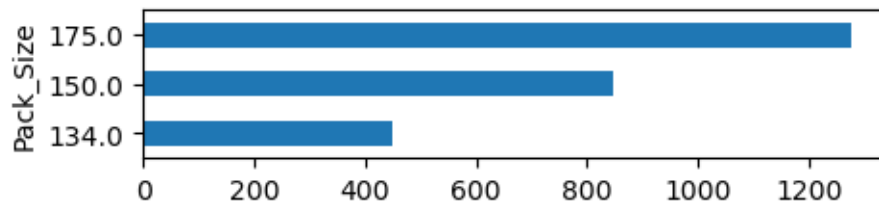
===== MIDAGE SINGLES/COUPLES - Budget =====

```

```

Pack_Size
134.0    449
150.0    846
175.0    1277
Name: count, dtype: int64

```



```

===== NEW FAMILIES - Premium =====

```

```

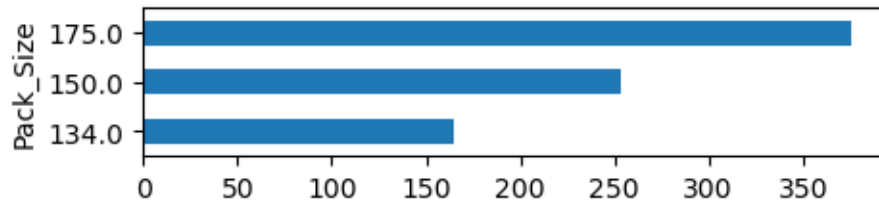
Pack_Size

```

```

134.0    165
150.0    253
175.0    376
Name: count, dtype: int64

```



```

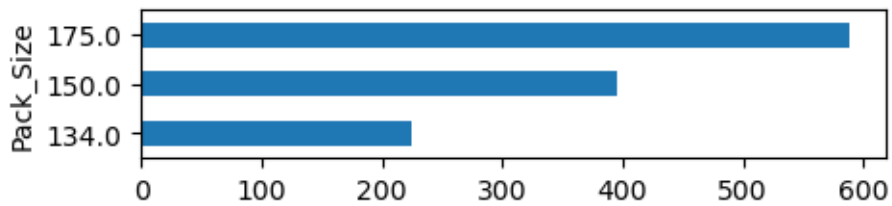
===== NEW FAMILIES - Mainstream =====

```

```

Pack_Size
134.0    224
150.0    396
175.0    589
Name: count, dtype: int64

```



```

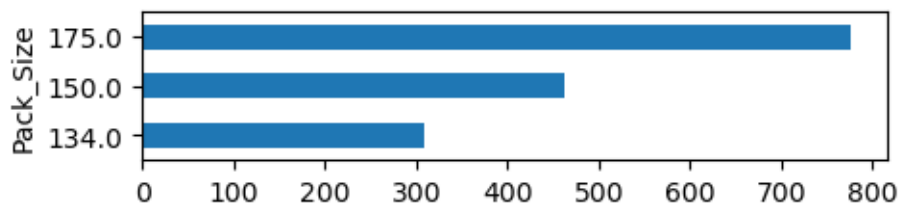
===== NEW FAMILIES - Budget =====

```

```

Pack_Size
134.0    309
150.0    463
175.0    777
Name: count, dtype: int64

```

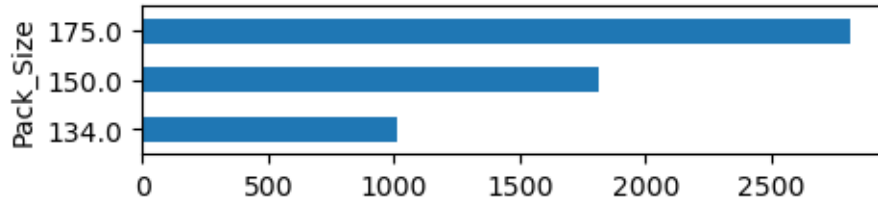


```

===== OLDER FAMILIES - Premium =====

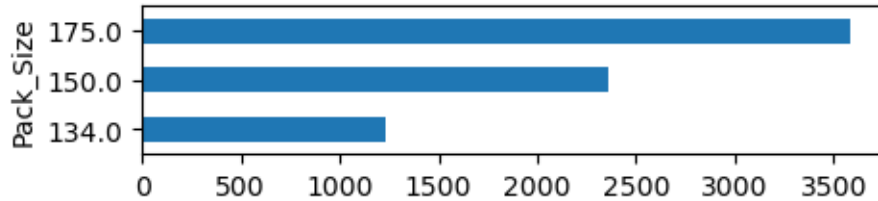
```

```
Pack_Size
134.0    1014
150.0    1816
175.0    2816
Name: count, dtype: int64
```



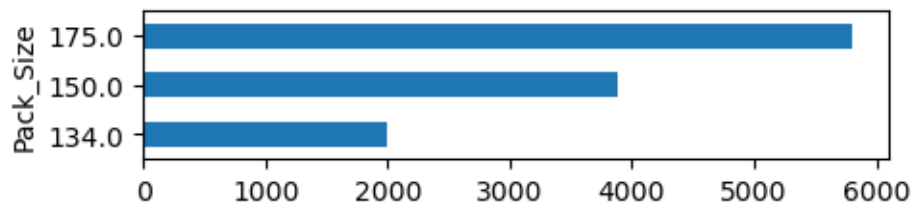
```
===== OLDER FAMILIES - Mainstream =====
```

```
Pack_Size
134.0    1234
150.0    2359
175.0    3588
Name: count, dtype: int64
```



```
===== OLDER FAMILIES - Budget =====
```

```
Pack_Size
134.0    1996
150.0    3882
175.0    5808
Name: count, dtype: int64
```



===== RETIREES - Premium =====

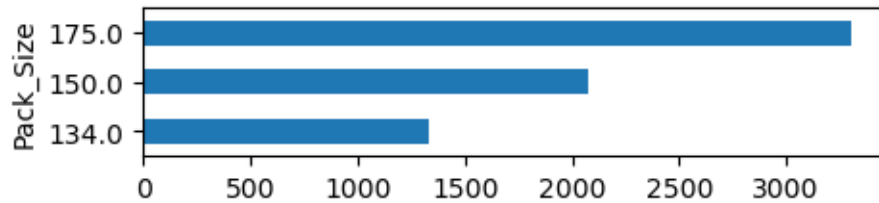
Pack_Size

134.0 1331

150.0 2075

175.0 3306

Name: count, dtype: int64



===== RETIREES - Mainstream =====

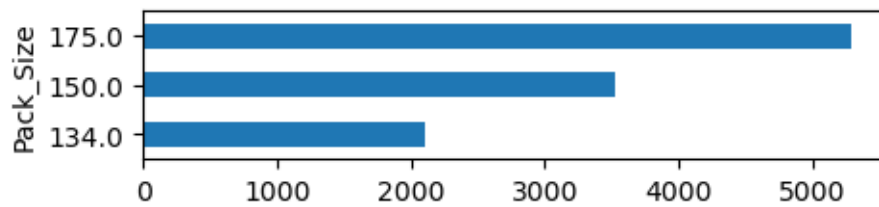
Pack_Size

134.0 2103

150.0 3522

175.0 5295

Name: count, dtype: int64



===== RETIREES - Budget =====

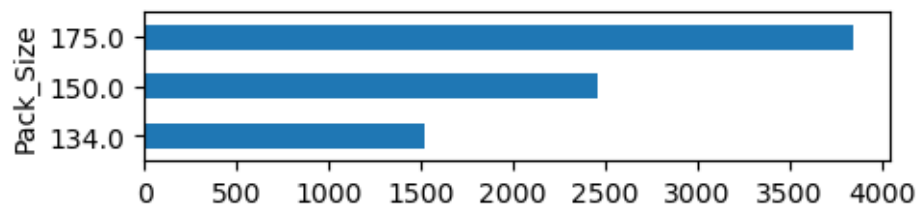
Pack_Size

134.0 1517

150.0 2458

175.0 3847

Name: count, dtype: int64

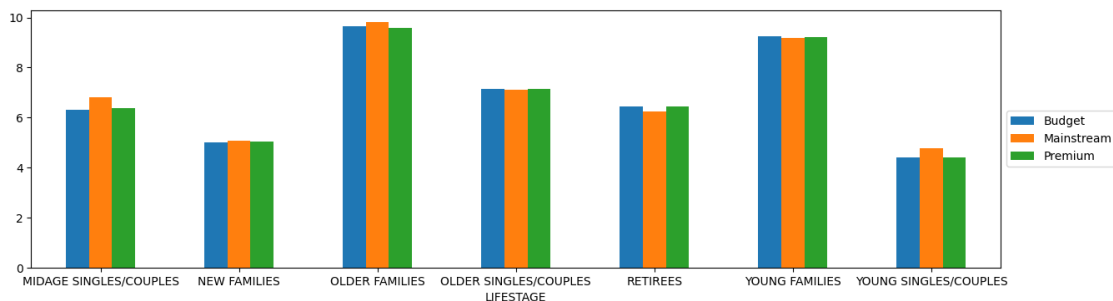


```
[52]: (temp.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["PROD_QTY"].sum() / temp.
      ↳groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique()).
      ↳sort_values(ascending=False)
```

```
[52]: LIFESTAGE      PREMIUM_CUSTOMER
      OLDER FAMILIES  Mainstream      9.804309
      Budget        9.639572
      Premium        9.578091
      YOUNG FAMILIES Budget        9.238486
      Premium        9.209207
      Mainstream     9.180352
      OLDER SINGLES/COUPLES Premium    7.154947
      Budget        7.145466
      Mainstream     7.098783
      MIDAGE SINGLES/COUPLES Mainstream 6.796108
      RETIREES      Budget        6.458015
      Premium        6.426653
      MIDAGE SINGLES/COUPLES Premium    6.386672
      Budget        6.313830
      RETIREES      Mainstream     6.253743
      NEW FAMILIES  Mainstream     5.087161
      Premium        5.028912
      Budget        5.009892
      YOUNG SINGLES/COUPLES Mainstream 4.776459
      Budget        4.411485
      Premium        4.402098

dtype: float64
```

```
[53]: (temp.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["PROD_QTY"].sum() / temp.
      ↳groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique()).
      ↳unstack().plot.bar(figsize=(15,4), rot=0)
plt.legend(loc="center left", bbox_to_anchor=(1.0, 0.5))
plt.savefig("Average purchase quantity per segment.png", bbox_inches="tight")
```



[]: Trends and Insights :

Top 3 total sales contributor segment are

Older families (Budget) \$156,864

Young Singles/Couples (Mainstream) \$147,582

Retirees (Mainstream) \$145,169

Young Singles/Couples (Mainstream) has the highest population, followed by
↳ Retirees (Mainstream). Which explains their high total sales.

Despite Older Families not having the highest population, they have the highest
↳ frequency of purchase, which contributes to their high total sales.

Older Families followed by Young Families has the highest average quantity of
↳ chips bought per purchase.

The Mainstream category of the "Young and Midage Singles/Couples" have the
↳ highest spending of chips per purchase. And the difference to the
↳ non-Mainstream "Young and Midage Singles/Couples" are statistically
↳ significant.

Chips brand Kettle is dominating every segment as the most purchased brand.

Observing the 2nd most purchased brand, "Young and Midage Singles/Couples" is
↳ the only segment with a different preference (Doritos) as compared to
↳ others' (Smiths).

Most frequent chip size purchased is 175gr followed by the 150gr chip size for
↳ all segments.

Views and Recommendations:

Older Families: Focus on the Budget segment. Strength: Frequent purchase. We
↳ can give promotions that encourages more frequency of purchase. Strength:
↳ High quantity of chips purchased per visit. We can give promotions that
↳ encourage them to buy more quantity of chips per purchase.

Young Singles/Couples: Focus on the Mainstream segment. This segment is the
↳ only segment that had Doritos as their 2nd most purchased brand (after
↳ Kettle). To specifically target this segment it might be a good idea to
↳ collaborate with Doritos merchant to do some branding promotion catered to
↳ "Young Singles/Couples - Mainstream" segment. Strength: Population quantity.
↳ We can spend more effort on making sure our promotions reach them, and it
↳ reaches them frequently.

Retirees: Focus on the Mainstream segment. Strength: Population quantity.□

→Again, since their population quantity **is** the contributor to the high total□
→sales, we should spend more effort on making sure our promotions reaches **as**□
→many of them **as** possible **and** frequent.

General: All segments has Kettle **as** the most frequently purchased brand, **and**□

→175gr (regardless of brand) followed by 150gr **as** the preferred chip size.□
→When promoting chips **in** general to **all** segments it **is** good to take advantage□
→of these two points.

[]: