Task 1-Data Preparation and Customer Analytics

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

Background information for the task

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

Main goals of this task are :

1. Examine transaction data - check for missing data, anomalies, outliers and clean them
2. Examine customer data - similar to above transaction data
3. Data analysis and customer segments - create charts and graphs, note trends and insights
4. Deep dive into customer segments - determine which segments should be targetted

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

In [10]:
```
tran_data = pd.read_excel("/content/QVI_transaction_data.xlsx")
```

In [12]:
```
tran_data.head()
```

Out[12]:

	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

In [13]:
```
tran_data.describe()
```

Out[13]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PRO
count	264836.000000	264836.00000	2.648360e+05	2.648360e+05	264836.000000	264836.
mean	43464.036260	135.08011	1.355495e+05	1.351583e+05	56.583157	1.
std	105.389282	76.78418	8.057998e+04	7.813303e+04	32.826638	0.
min	43282.000000	1.00000	1.000000e+03	1.000000e+00	1.000000	1.
25%	43373.000000	70.00000	7.002100e+04	6.760150e+04	28.000000	2.
50%	43464.000000	130.00000	1.303575e+05	1.351375e+05	56.000000	2.
75%	43555.000000	203.00000	2.030942e+05	2.027012e+05	85.000000	2.
max	43646.000000	272.00000	2.373711e+06	2.415841e+06	114.000000	200.

In [14]:
```
pur_bvr = pd.read_csv("/content/QVI_purchase_behaviour.csv")
```

In [15]:
```
pur_bvr.head()
```

Out[15]:

	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

In [16]:
```
pur_bvr.describe()
```

Out[16]:

	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

In [17]:
```
tran_data.isnull().sum()
```

Out[17]:
```
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 [18]:
```
pur_bvr.isnull().sum()
```

Out[18]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

Checking and Removing Outliers

In [25]:
```
merged_data = pd.merge(pur_bvr, tran_data, on = 'LYLTY_CARD_NBR', how = 'ri
ght')
merged_data.head()
```

Out[25]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID
0	1000	YOUNG SINGLES/COUPLES	Premium	43390	1	
1	1307	MIDAGE SINGLES/COUPLES	Budget	43599	1	34
2	1307	MIDAGE SINGLES/COUPLES	Budget	43414	1	34
3	1307	MIDAGE SINGLES/COUPLES	Budget	43533	1	34
4	1343	MIDAGE SINGLES/COUPLES	Budget	43605	1	38

In [26]:
```
print(len(merged_data))
print(len(tran_data))
```

```
264836
264836
```

In [27]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 264836 entries, 0 to 264835
Data columns (total 10 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   LYLTY_CARD_NBR    264836 non-null  int64  
 1   LIFESTAGE         264836 non-null  object 
 2   PREMIUM_CUSTOMER  264836 non-null  object 
 3   DATE              264836 non-null  int64  
 4   STORE_NBR         264836 non-null  int64  
 5   TXN_ID            264836 non-null  int64  
 6   PROD_NBR          264836 non-null  int64  
 7   PROD_NAME         264836 non-null  object 
 8   PROD_QTY          264836 non-null  int64  
 9   TOT_SALES         264836 non-null  float64
dtypes: float64(1), int64(6), object(3)
memory usage: 22.2+ MB
```

Date column should be data time format

In [29]:
```
from datetime import date, timedelta
start = date(1899, 12, 30)
new_date_format = []
for date in merged_data["DATE"]:
  delta = timedelta(date)
  new_date_format.append(start + delta)
```

In [30]:
```
merged_data["DATE"] = pd.to_datetime(pd.Series(new_date_format))
print(merged_data["DATE"].dtype)
```

```
datetime64[ns]
```

Checking the product name column to make sure all items are chips

In [31]:
```
merged_data["PROD_NAME"].unique()
```

Out[31]:
```
array(['Natural Chip        Compny SeaSalt175g',
       'CCs Nacho Cheese    175g', 'WW Original Stacked Chips 160g',
       'CCs Original 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',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Tortilla ChpsFeta&Garlic 150g',
       'Dorito Corn Chp     Supreme 380g', 'Doritos Mexicana    170g',
       'Smiths Crinkle Chips Salt & Vinegar 330g', 'Kettle Original 175g',
       'Tyrrells Crisps     Ched & Chives 165g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'Grain Waves         Sweet Chilli 210g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
       'Doritos Corn Chip Southern Chicken 150g',
       'Thins Potato Chips  Hot & Spicy 175g',
       'Doritos Corn Chip Mexican Jalapeno 150g',
       'Red Rock Deli Thai  Chilli&Lime 150g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'WW D/Style Chip     Sea Salt 200g',
       'Thins Chips Light&  Tangy 175g',
       'Grain Waves Sour    Cream&Chives 210G',
       'Tostitos Smoked     Chipotle 175g',
       'Infuzions Thai SweetChili PotatoMix 110g',
       'Kettle Sensations   Siracha Lime 150g',
       'Pringles Chicken    Salt Crips 134g',
       'Thins Chips Salt &  Vinegar 175g',
       'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
       'Twisties Cheese     270g', 'WW Crinkle Cut      Chicken 175g',
       'RRD Sweet Chilli &  Sour Cream 165g',
       'Doritos Corn Chips  Cheese Supreme 170g',
       'RRD Salt & Vinegar  165g', 'Doritos Corn Chips  Original 170g',
       'Smiths Crinkle      Original 330g',
       'Infzns Crn Crnchers Tangy Gcamole 110g',
       'Kettle Sea Salt     And Vinegar 175g',
       'Red Rock Deli Chikn&Garlic Aioli 150g',
       'RRD SR Slow Rst     Pork Belly 150g', 'Burger Rings 220g',
       'NCC Sour Cream &    Garden Chives 175g',
       'Smiths Crinkle Cut  French OnionDip 150g',
       'Natural ChipCo Sea  Salt & Vinegr 175g',
       'Cheezels Cheese Box 125g', 'CCs Tasty Cheese    175g',
       'Smith Crinkle Cut   Bolognese 150g', 'Pringles Slt Vingar 134g',
       'WW Sour Cream &OnionStacked Chips 160g',
       'Doritos Salsa Mild  300g', 'Pringles Original   Crisps 134g',
       'Pringles Sthrn FriedChicken 134g',
       'Pringles SourCream  Onion 134g',
       'Smiths Crinkle Cut  Chips Barbecue 170g',
       'Infuzions Mango     Chutny Papadums 70g',
       'RRD Pc Sea Salt     165g', 'Doritos Salsa       Medium 300g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Smiths Chip Thinly  Cut Original 175g', 'Twisties Chicken270g',
       'Smiths Crinkle Cut  Tomato Salsa 150g',
       'Kettle 135g Swt Pot Sea Salt',
       'Natural ChipCo      Hony Soy Chckn175g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'WW Supreme Cheese   Corn Chips 200g',
       'WW Original Corn    Chips 200g',
       'Cobs Popd Sour Crm  &Chives Chips 110g',
       'Pringles Sweet&Spcy BBQ 134g', 'Doritos Cheese      Supreme 330g',
       'Red Rock Deli SR    Salsa & Mzzrlla 150g',
       'Pringles Mystery    Flavour 134g',
```

```
'Thins Chips         Originl saltd 175g',
'Smiths Thinly Cut   Roast Chicken 175g',
'Kettle Mozzarella   Basil & Pesto 175g',
'Smiths Crinkle Cut  Salt & Vinegar 170g',
'Red Rock Deli Sp    Salt & Truffle 150G',
'Smiths Thinly       Swt Chli&S/Cream175G', 'Kettle Chilli 175g',
'Kettle Honey Soy    Chicken 175g', 'Pringles Barbeque   134g',
'Kettle Sensations   BBQ&Maple 150g',
'RRD Steak &         Chimuchurri 150g',
'Doritos Corn Chips  Nacho Cheese 170g',
'Tostitos Splash Of  Lime 175g',
'Kettle Sensations   Camembert & Fig 150g',
'Cobs Popd Sea Salt  Chips 110g',
'Smith Crinkle Cut   Mac N Cheese 150g',
'Woolworths Mild     Salsa 300g',
'Smiths Crinkle Cut  Snag&Sauce 150g',
'Thins Chips Seasonedchicken 175g',
'Woolworths Medium   Salsa 300g',
'Kettle Tortilla ChpsBtroot&Ricotta 150g',
'Infuzions SourCream&Herbs Veg Strws 110g', 'Cheezels Cheese 330g',
'RRD Chilli&         Coconut 150g',
'Smiths Crinkle Cut  Chips Chs&Onion170g',
'Smiths Chip Thinly  CutSalt/Vinegr175g',
'Twisties Cheese     Burger 250g', 'RRD Lime & Pepper   165g',
'RRD Honey Soy       Chicken 165g',
'French Fries Potato Chips 175g',
'Natural Chip Co     Tmato Hrb&Spce 175g',
'Sunbites Whlegrn    Crisps Frch/Onin 90g', 'Cheetos Puffs 165g',
'Smiths Crinkle Cut  Chips Original 170g',
'Tostitos Lightly    Salted 175g',
'Woolworths Cheese   Rings 190g',
'Smiths Crnkle Chip  Orgnl Big Bag 380g',
'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
'WW Crinkle Cut      Original 175g',
'Cheetos Chs & Bacon Balls 190g'], dtype=object)
```

In [32]:
```
split_prods = merged_data["PROD_NAME"].str.replace(r'([0-9]+[gG])','').str.
replace(r'[^\w]',' ').str.split()
```

In [34]:
```
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
          ...
Onin       1432
Pc         1431
Garden     1419
NCC        1419
Fries      1418
Length: 198, dtype: int64
```

In [35]:
```
print(merged_data.describe(), '\n')
print(merged_data.info())
```

```
       LYLTY_CARD_NBR     STORE_NBR  ...       PROD_QTY      TOT_SALES
count    2.648360e+05  264836.00000  ...  264836.000000  264836.000000
mean     1.355495e+05     135.08011  ...       1.907309       7.304200
std      8.057998e+04      76.78418  ...       0.643654       3.083226
min      1.000000e+03       1.00000  ...       1.000000       1.500000
25%      7.002100e+04      70.00000  ...       2.000000       5.400000
50%      1.303575e+05     130.00000  ...       2.000000       7.400000
75%      2.030942e+05     203.00000  ...       2.000000       9.200000
max      2.373711e+06     272.00000  ...     200.000000     650.000000

[8 rows x 6 columns]

<class 'pandas.core.frame.DataFrame'>
Int64Index: 264836 entries, 0 to 264835
Data columns (total 10 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264836 non-null  int64
 1   LIFESTAGE         264836 non-null  object
 2   PREMIUM_CUSTOMER  264836 non-null  object
 3   DATE              264836 non-null  datetime64[ns]
 4   STORE_NBR         264836 non-null  int64
 5   TXN_ID            264836 non-null  int64
 6   PROD_NBR          264836 non-null  int64
 7   PROD_NAME         264836 non-null  object
 8   PROD_QTY          264836 non-null  int64
 9   TOT_SALES         264836 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
memory usage: 22.2+ MB
None
```

In [36]:
```
merged_data["PROD_QTY"].value_counts(bins=4).sort_index()
```

Out[36]:
```
(0.8, 50.75]       264834
(50.75, 100.5]          0
(100.5, 150.25]         0
(150.25, 200.0]         2
Name: PROD_QTY, dtype: int64
```

From above binning we see that PROD_QTY values above 50.75

In [37]:
```
merged_data.sort_values(by="PROD_QTY", ascending=False).head()
```

Out[37]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	T
71456	226000	OLDER FAMILIES	Premium	2018-08-19	226	2
71457	226000	OLDER FAMILIES	Premium	2019-05-20	226	2
34105	97211	OLDER FAMILIES	Budget	2018-08-15	97	
102451	235164	OLDER SINGLES/COUPLES	Mainstream	2018-08-17	235	2
259068	183032	YOUNG SINGLES/COUPLES	Premium	2019-05-14	183	1

Two outliers of value 200 in PROD_QTY will be removed. Both entries are by the same customer and will be examined by this customer's transactions

In [38]:
```
merged_data = merged_data[merged_data["PROD_QTY"] < 6]
```

In [39]:
```
len(merged_data[merged_data["LYLTY_CARD_NBR"]==226000])
```

Out[39]:
```
0
```

In [40]:
```
merged_data["DATE"].describe()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarni
ng: Treating datetime data as categorical rather than numeric in `.describe
` is deprecated and will be removed in a future version of pandas. Specify
`datetime_is_numeric=True` to silence this warning and adopt the future beh
avior now.
  """Entry point for launching an IPython kernel.
```

Out[40]:
```
count                  264834
unique                    364
top       2018-12-24 00:00:00
freq                      939
first     2018-07-01 00:00:00
last      2019-06-30 00:00:00
Name: DATE, dtype: object
```

There are 365 days in a year but in the DATE column there are only 364 unique values so one is missing

In [41]:
```
pd.date_range(start=merged_data["DATE"].min(), end=merged_data["DATE"].max
()).difference(merged_data["DATE"])
```

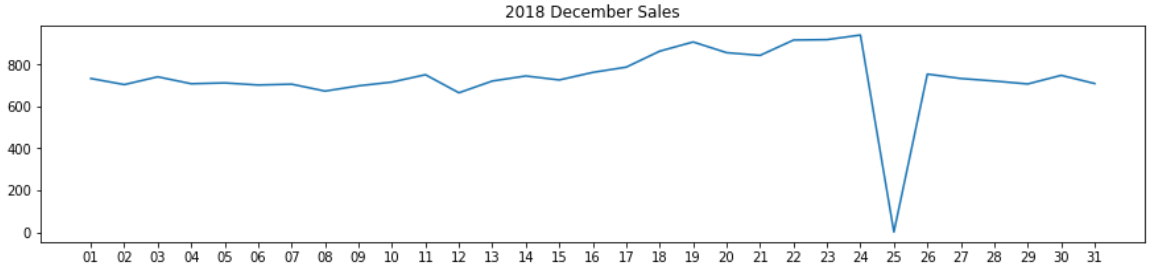
Out[41]:
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

Using the difference method we see that 2018-12-25 was a missing date

In [42]:
```
check_null_date = pd.merge(pd.Series(pd.date_range(start=merged_data["DAT
E"].min(), end = merged_data["DATE"].max()), name="DATE"), merged_data, on
= "DATE", how = "left")
```

In [43]:
```
trans_by_date = check_null_date["DATE"].value_counts()
dec = trans_by_date[(trans_by_date.index >= pd.datetime(2018,12,1)) & (tran
s_by_date.index < pd.datetime(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()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarni
ng: The pandas.datetime class is deprecated and will be removed from pandas
in a future version. Import from datetime instead.
```



In [44]:
```
check_null_date["DATE"].value_counts().sort_values().head()
```

Out[44]:
```
2018-12-25      1
2018-11-25    648
2018-10-18    658
2019-06-13    659
2019-06-24    662
Name: DATE, dtype: int64
```

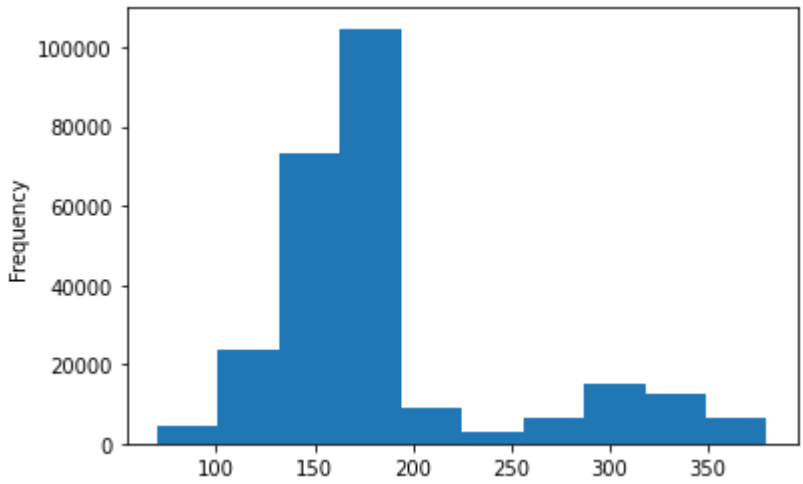
The day with no transaction is a Christmas day that is when the store is closed. So there is no anomaly in this.

Explore Packet sizes

In [45]:
```
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("g","").astype("float")
print(pack_sizes.describe())
pack_sizes.plot.hist()
```

```
count    258770.000000
mean        182.324276
std          64.955035
min          70.000000
25%         150.000000
50%         170.000000
75%         175.000000
max         380.000000
Name: 0, dtype: float64
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe9b157b00>
```



In [46]:
```
merged_data["PROD_NAME"].str.split().str[0].value_counts().sort_index()
```

Out[46]:
```
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: PROD_NAME, dtype: int64
```

Some product names are written in more than one way. Example : Dorito and Doritos, Grains and GrnWves, Infusions and Ifzns, Natural and NCC, Red and RRD, Smith and Smiths and Snbts and Sunbites.

In [47]:
```
merged_data["PROD_NAME"].str.split()[merged_data["PROD_NAME"].str.split().s
tr[0] == "Red"].value_counts()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectH
ashTable.map_locations()

TypeError: unhashable type: 'list'

Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas/_libs/hashtable_class_helper.pxi", line 1709, in pandas._lib
s.hashtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'list'
```

Out[47]:
```
[Red, Rock, Deli, Sp, Salt, &, Truffle, g]        1498
[Red, Rock, Deli, Thai, Chilli&Lime, 150g]        1495
[Red, Rock, Deli, SR, Salsa, &, Mzzrlla, 150g]    1458
[Red, Rock, Deli, Chikn&Garlic, Aioli, 150g]      1434
Name: PROD_NAME, dtype: int64
```

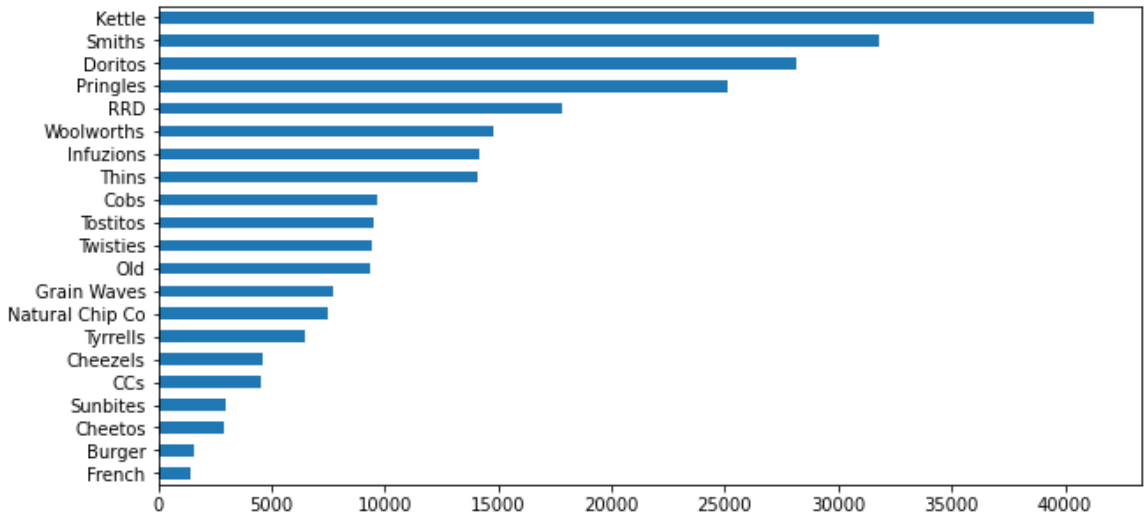
In [48]:
```
merged_data["Cleaned_Brand_Names"] = merged_data["PROD_NAME"].str.split().str[0]
```

In [49]:
```
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
```

In [50]:
```
merged_data["Cleaned_Brand_Names"] = merged_data.apply(lambda line: clean_brand_names(line), axis=1)
```

In [51]:
```
merged_data["Cleaned_Brand_Names"].value_counts(ascending=True).plot.barh(figsize=(10,5))
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x7efe8d7b9c18>



In [52]:
```
merged_data.isnull().sum()
```

Out[52]:
```
LYLTY_CARD_NBR         0
LIFESTAGE              0
PREMIUM_CUSTOMER       0
DATE                   0
STORE_NBR              0
TXN_ID                 0
PROD_NBR               0
PROD_NAME              0
PROD_QTY               0
TOT_SALES              0
Cleaned_Brand_Names    0
dtype: int64
```

1. Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is 2.How many customers are in each segment
2. How many chips are bought per customer by segment
3. What's the average chip price by customer segment

In [54]:
```
grouped_sales = pd.DataFrame(merged_data.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["TOT_SALES"].agg(["sum", "mean"]))
grouped_sales.sort_values(ascending=False, by="sum")
```

Out[54]:

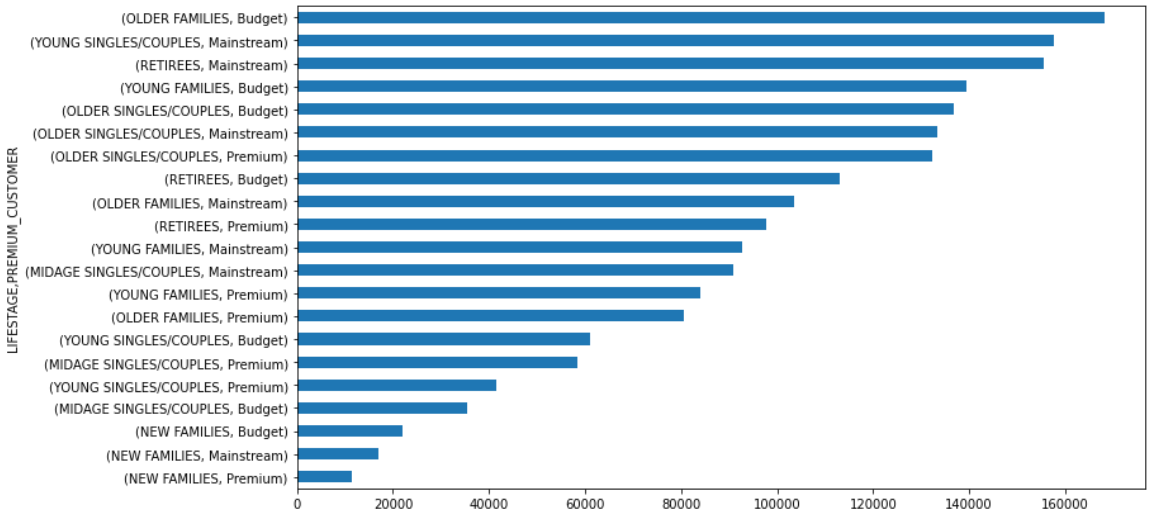
		sum	mean
LIFESTAGE	PREMIUM_CUSTOMER		
OLDER FAMILIES	Budget	168363.25	7.269570
YOUNG SINGLES/COUPLES	Mainstream	157621.60	7.558339
RETIREES	Mainstream	155677.05	7.252262
YOUNG FAMILIES	Budget	139345.85	7.287201
OLDER SINGLES/COUPLES	Budget	136769.80	7.430315
	Mainstream	133393.80	7.282116
	Premium	132263.15	7.449766
RETIREES	Budget	113147.80	7.443445
OLDER FAMILIES	Mainstream	103445.55	7.262395
RETIREES	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

In [55]:
```
grouped_sales["sum"].sum()
```

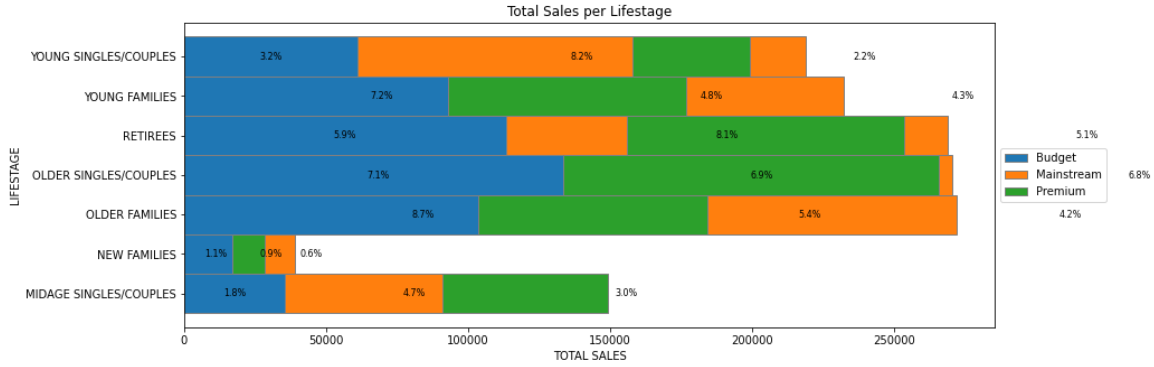
Out[55]: 1933114.9999999765

In [56]:
```
grouped_sales["sum"].sort_values().plot.barh(figsize=(12,7))
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe933a5438>
```



In [62]:
```
# Values of each group
bars1 = grouped_sales[grouped_sales.index.get_level_values("PREMIUM_CUSTOME
R") == "Budget"]["sum"]
bars2 = grouped_sales[grouped_sales.index.get_level_values("PREMIUM_CUSTOME
R") == "Mainstream"]["sum"]
bars3 = grouped_sales[grouped_sales.index.get_level_values("PREMIUM_CUSTOME
R") == "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 firs ones
mains_bar = plt.barh(r, bars2, left=bars1, edgecolor='grey', height=1, labe
l="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")

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[i], va='center', ha='center', si
ze=8)
    plt.text(budget_width + mains_bar[i].get_width()/2, i, bars2_text[i], v
a='center', ha='center', size=8)
    plt.text(budget_main_width + prem_bar[i].get_width()/2, i, bars3_text
[i], va='center', ha='center', size=8)

# 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()
```



In [63]:
```
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
RETIREES                  Mainstream
YOUNG SINGLES/COUPLES     Mainstream
Name: PREMIUM_CUSTOMER, dtype: object
```

The top 3 total sales contributor segment are (in order):

1. Older families (Budget) $156,864
2. Young Singles/Couples (Mainstream) $147,582
3. Retirees (Mainstream) $145,169

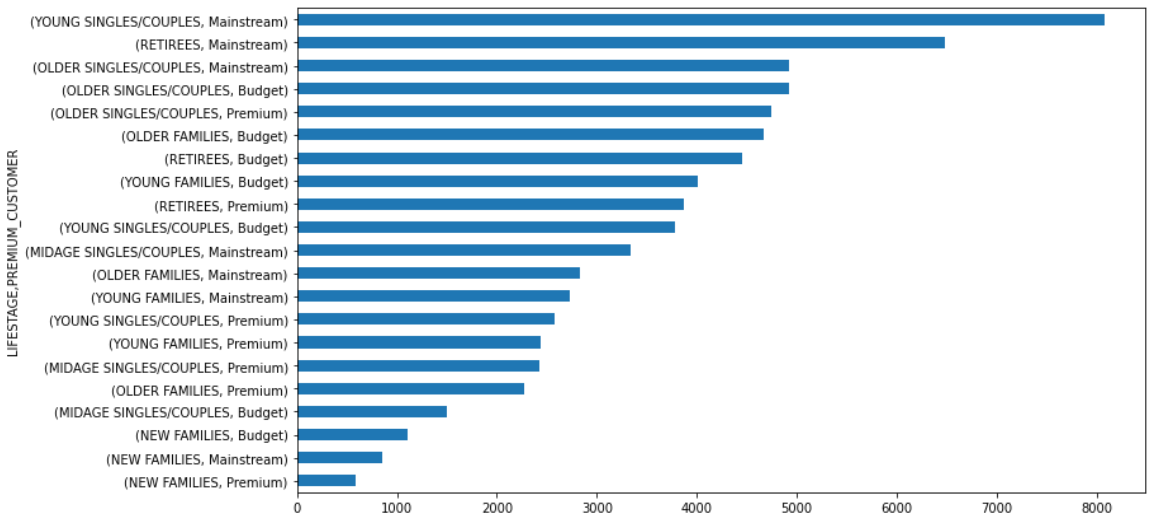
In [64]:
```
unique_cust = merged_data.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique().sort_values(ascending=False)
pd.DataFrame(unique_cust)
```

Out[64]:

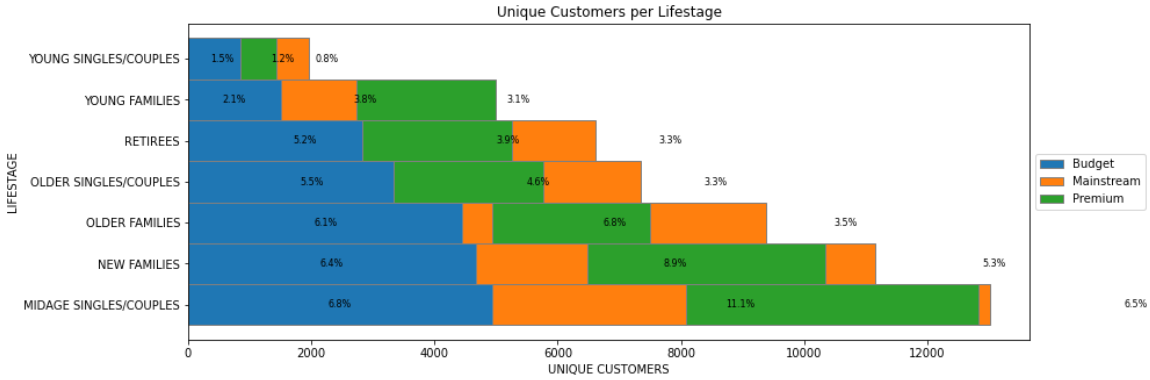
LIFESTAGE	PREMIUM_CUSTOMER	LYLTY_CARD_NBR
YOUNG SINGLES/COUPLES	Mainstream	8088
RETIREES	Mainstream	6479
OLDER SINGLES/COUPLES	Mainstream	4930
	Budget	4929
	Premium	4750
OLDER FAMILIES	Budget	4675
RETIREES	Budget	4454
YOUNG FAMILIES	Budget	4017
RETIREES	Premium	3872
YOUNG SINGLES/COUPLES	Budget	3779
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

In [65]:
```
unique_cust.sort_values().plot.barh(figsize=(12,7))
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x7efe8d90ea58>



In [67]:
```
# Values of each group
ncust_bars1 = unique_cust[unique_cust.index.get_level_values("PREMIUM_CUSTO
MER") == "Budget"]
ncust_bars2 = unique_cust[unique_cust.index.get_level_values("PREMIUM_CUSTO
MER") == "Mainstream"]
ncust_bars3 = unique_cust[unique_cust.index.get_level_values("PREMIUM_CUSTO
MER") == "Premium"]

ncust_bars1_text = (ncust_bars1 / sum(unique_cust)).apply("{:.1%}".format)
ncust_bars2_text = (ncust_bars2 / sum(unique_cust)).apply("{:.1%}".format)
ncust_bars3_text = (ncust_bars3 / 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, ncust_bars1, edgecolor='grey', height=1, label="Bu
dget")
# # Create green bars (middle), on top of the firs ones
mains_bar = plt.barh(r, ncust_bars2, left=ncust_bars1, edgecolor='grey', he
ight=1, label="Mainstream")
# # Create green bars (top)
prem_bar = plt.barh(r, ncust_bars3, left=ncust_bars2, edgecolor='grey', hei
ght=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, ncust_bars1_text[i], va='center', ha='cente
r', size=8)
    plt.text(budget_width + mains_bar[i].get_width()/2, i, ncust_bars2_text
[i], va='center', ha='center', size=8)
    plt.text(budget_main_width + prem_bar[i].get_width()/2, i, ncust_bars3_
text[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.

In [68]:
```
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")
```

Out[68]:

		mean	count
LIFESTAGE	**PREMIUM_CUSTOMER**		
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

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

In [69]:
```
grouped_sales.sort_values(ascending=False, by="mean")
```

Out[69]:

		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

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.

In [70]:
```
from scipy.stats import ttest_ind
mainstream = merged_data["PREMIUM_CUSTOMER"] == "Mainstream"
young_midage = (merged_data["LIFESTAGE"] == "MIDAGE SINGLES/COUPLES") | (me
rged_data["LIFESTAGE"] == "YOUNG SINGLES/COUPLES")

budget_premium = (merged_data["PREMIUM_CUSTOMER"] == "Budget") | (merged_da
ta["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
```

```
1.854204010750742e-281
```

Out[70]:
```
True
```

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.

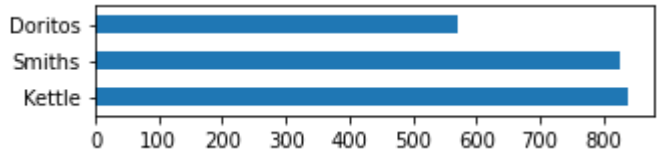
Next, let's look examine what brand of chips the top 3 segments contributing to Total Sales are buying.

In [71]:
```
merged_data.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["Cleaned_Brand_Name
s"].agg(pd.Series.mode).sort_values()
```

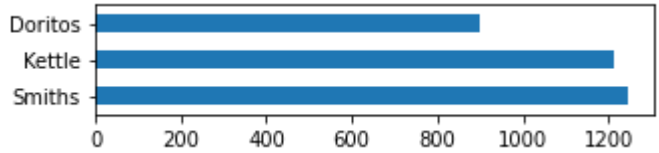
Out[71]:
```
LIFESTAGE               PREMIUM_CUSTOMER
MIDAGE SINGLES/COUPLES  Budget              Kettle
YOUNG FAMILIES          Premium             Kettle
                        Mainstream          Kettle
                        Budget              Kettle
RETIREES                Premium             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
```

In [72]:
```
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().h
ead(3)
        print(summary)
        plt.figure()
        summary.plot.barh(figsize=(5,1))
        plt.show()
```

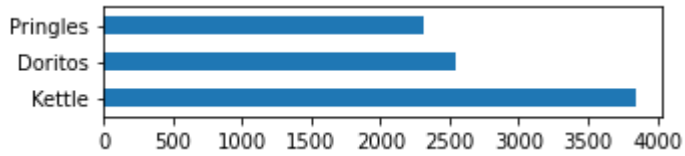
```
========== YOUNG SINGLES/COUPLES - Premium ==========
Kettle     838
Smiths     826
Doritos    570
Name: Cleaned_Brand_Names, dtype: int64
```



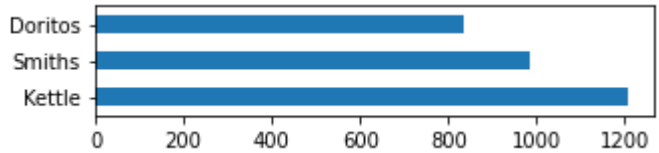

```
========== YOUNG SINGLES/COUPLES - Budget ==========
Smiths     1245
Kettle     1211
Doritos     899
Name: Cleaned_Brand_Names, dtype: int64
```



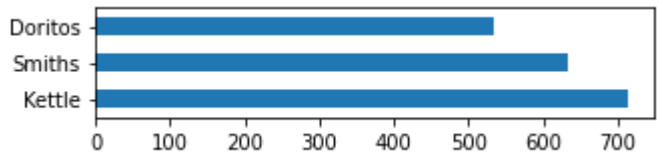

```
========== YOUNG SINGLES/COUPLES - Mainstream ==========
Kettle      3844
Doritos     2541
Pringles    2315
Name: Cleaned_Brand_Names, dtype: int64
```



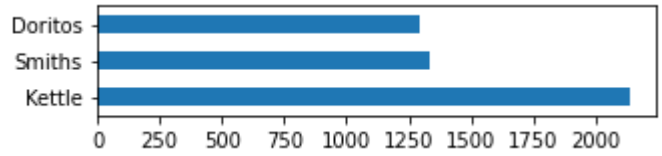

```
========== MIDAGE SINGLES/COUPLES - Premium ==========
Kettle     1206
Smiths      986
Doritos     837
Name: Cleaned_Brand_Names, dtype: int64
```



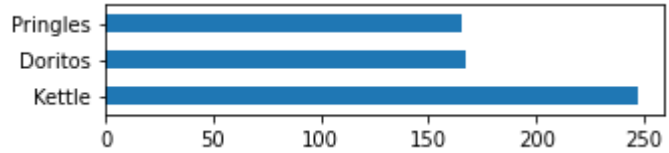

```
========== MIDAGE SINGLES/COUPLES - Budget ==========
Kettle     713
Smiths     633
Doritos    533
Name: Cleaned_Brand_Names, dtype: int64
```



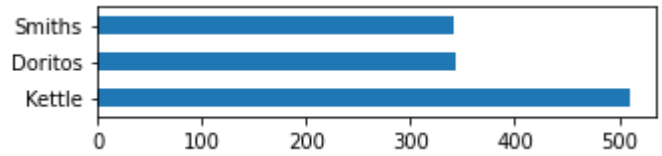
```
========== MIDAGE SINGLES/COUPLES - Mainstream ==========
Kettle     2136
Smiths     1337
Doritos    1291
Name: Cleaned_Brand_Names, dtype: int64
```



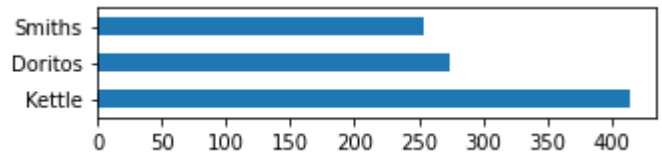

```
========== NEW FAMILIES - Premium ==========
Kettle      247
Doritos     167
Pringles    165
Name: Cleaned_Brand_Names, dtype: int64
```



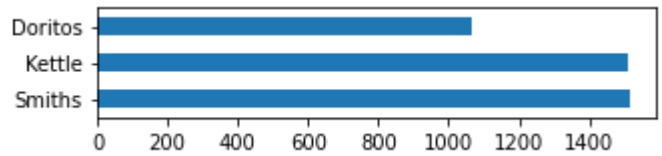

```
========== NEW FAMILIES - Budget ==========
Kettle     510
Doritos    343
Smiths     341
Name: Cleaned_Brand_Names, dtype: int64
```



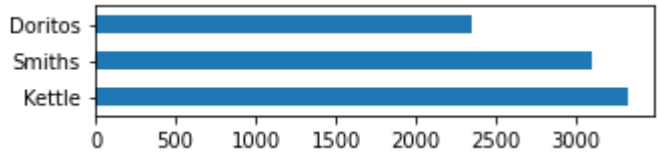

```
========== NEW FAMILIES - Mainstream ==========
Kettle     414
Doritos    274
Smiths     254
Name: Cleaned_Brand_Names, dtype: int64
```



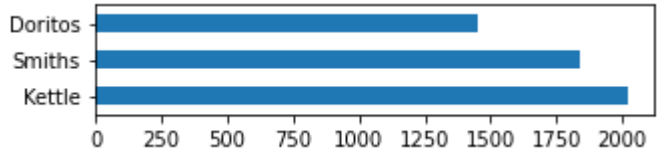

```
========== OLDER FAMILIES - Premium ==========
Smiths     1515
Kettle     1512
Doritos    1065
Name: Cleaned_Brand_Names, dtype: int64
```



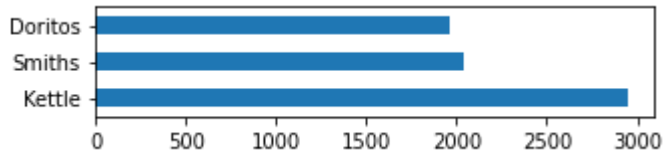
```
========== OLDER FAMILIES - Budget ==========
Kettle     3320
Smiths     3093
Doritos    2351
Name: Cleaned_Brand_Names, dtype: int64
```



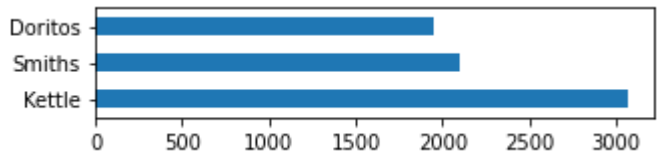

```
========== OLDER FAMILIES - Mainstream ==========
Kettle     2019
Smiths     1835
Doritos    1449
Name: Cleaned_Brand_Names, dtype: int64
```



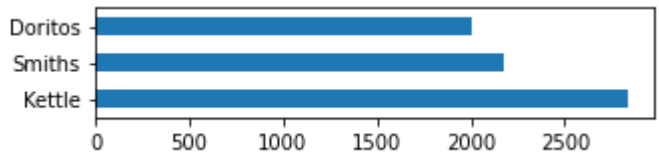

```
========== OLDER SINGLES/COUPLES - Premium ==========
Kettle     2947
Smiths     2042
Doritos    1958
Name: Cleaned_Brand_Names, dtype: int64
```



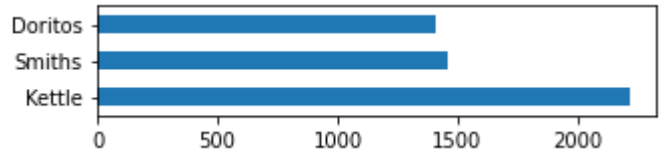

```
========== OLDER SINGLES/COUPLES - Budget ==========
Kettle     3065
Smiths     2098
Doritos    1954
Name: Cleaned_Brand_Names, dtype: int64
```



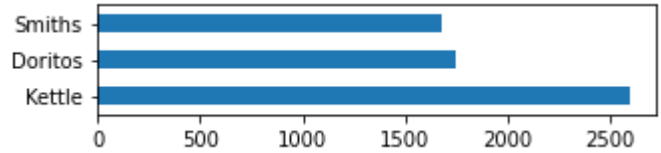

```
========== OLDER SINGLES/COUPLES - Mainstream ==========
Kettle     2835
Smiths     2180
Doritos    2008
Name: Cleaned_Brand_Names, dtype: int64
```



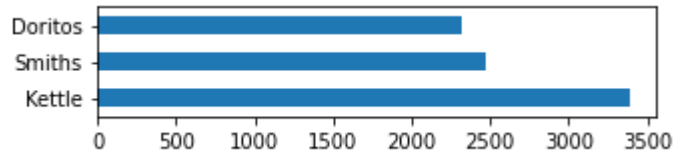
```
========== RETIREES - Premium ==========
Kettle     2216
Smiths     1458
Doritos    1409
Name: Cleaned_Brand_Names, dtype: int64
```



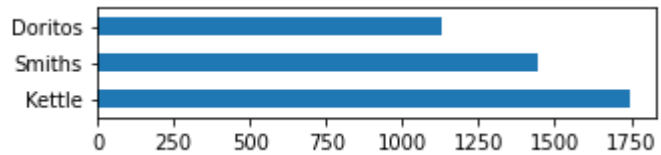

```
========== RETIREES - Budget ==========
Kettle     2592
Doritos    1742
Smiths     1679
Name: Cleaned_Brand_Names, dtype: int64
```



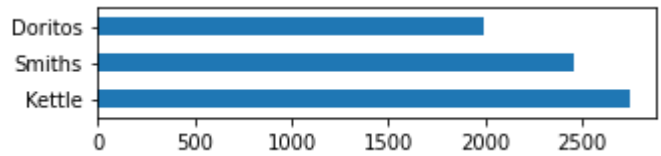

```
========== RETIREES - Mainstream ==========
Kettle     3386
Smiths     2476
Doritos    2320
Name: Cleaned_Brand_Names, dtype: int64
```



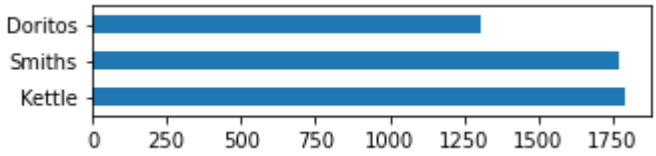

```
========== YOUNG FAMILIES - Premium ==========
Kettle     1745
Smiths     1442
Doritos    1129
Name: Cleaned_Brand_Names, dtype: int64
```




```
========== YOUNG FAMILIES - Budget ==========
Kettle     2743
Smiths     2459
Doritos    1996
Name: Cleaned_Brand_Names, dtype: int64
```



```
========== YOUNG FAMILIES - Mainstream ==========
Kettle     1789
Smiths     1772
Doritos    1309
Name: Cleaned_Brand_Names, dtype: int64
```



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.

In [73]:
```
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_d
ummies(temp["Cleaned_Brand_Names"])], axis=1)

frequent_sets = apriori(segment_brand_encode, min_support=0.01, use_colname
s=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 se
t_temp)]
```

Out[73]:

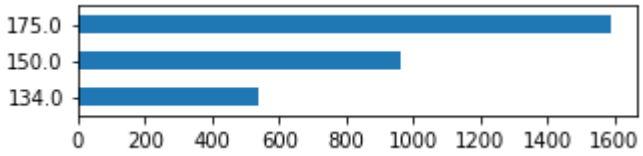
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(OLDER FAMILIES - Budget)	(Smiths)	0.087451	0.120162	0.011679	0.133549	1.111409
2	(OLDER SINGLES/COUPLES - Budget)	(Kettle)	0.069504	0.155901	0.011573	0.166513	1.068064
4	(OLDER SINGLES/COUPLES - Premium)	(Kettle)	0.067038	0.155901	0.011128	0.165991	1.064716
6	(RETIREES - Mainstream)	(Kettle)	0.081055	0.155901	0.012785	0.157738	1.011779
8	(YOUNG SINGLES/COUPLES - Mainstream)	(Kettle)	0.078744	0.155901	0.014515	0.184329	1.182344

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

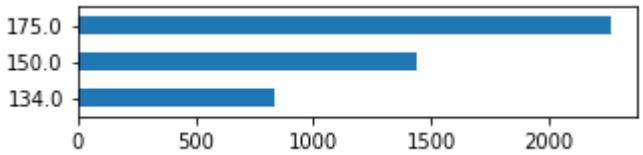
Next, we'll find out the pack size preferences of different segments

In [74]:
```
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).sor
t_index()
        print(summary)
        plt.figure()
        summary.plot.barh(figsize=(5,1))
        plt.show()
```

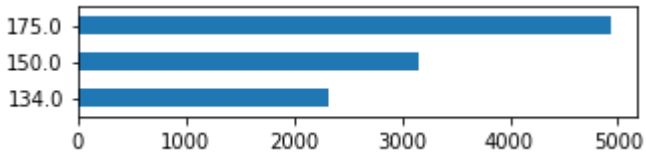
```
========== YOUNG SINGLES/COUPLES - Premium ==========
134.0     537
150.0     961
175.0    1587
Name: Pack_Size, dtype: int64
```



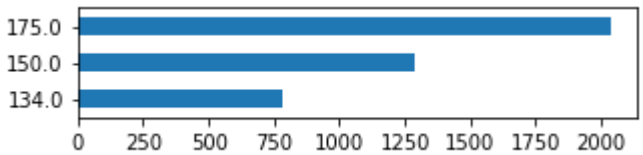

```
========== YOUNG SINGLES/COUPLES - Budget ==========
134.0     832
150.0    1439
175.0    2262
Name: Pack_Size, dtype: int64
```



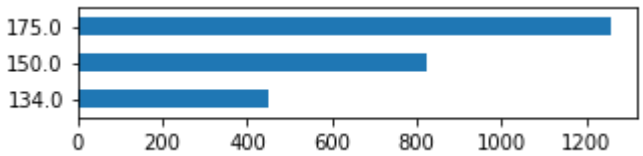

```
========== YOUNG SINGLES/COUPLES - Mainstream ==========
134.0    2315
150.0    3159
175.0    4928
Name: Pack_Size, dtype: int64
```



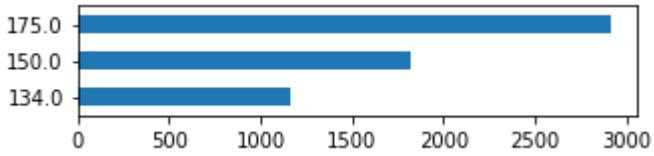

```
========== MIDAGE SINGLES/COUPLES - Premium ==========
134.0     781
150.0    1285
175.0    2034
Name: Pack_Size, dtype: int64
```



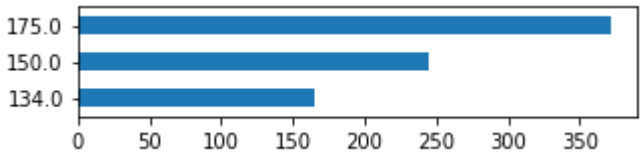

```
========== MIDAGE SINGLES/COUPLES - Budget ==========
134.0     449
150.0     821
175.0    1256
Name: Pack_Size, dtype: int64
```



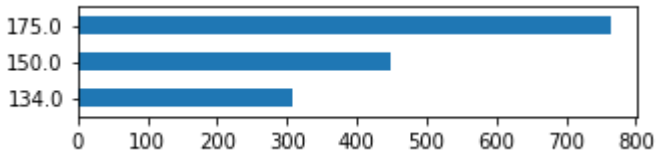
```
========== MIDAGE SINGLES/COUPLES - Mainstream ==========
134.0    1159
150.0    1819
175.0    2912
Name: Pack_Size, dtype: int64
```



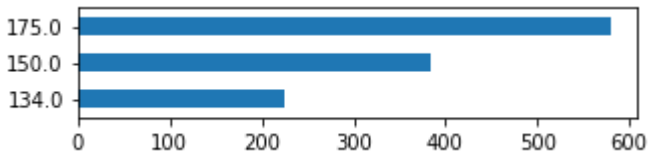

```
========== NEW FAMILIES - Premium ==========
134.0    165
150.0    245
175.0    371
Name: Pack_Size, dtype: int64
```



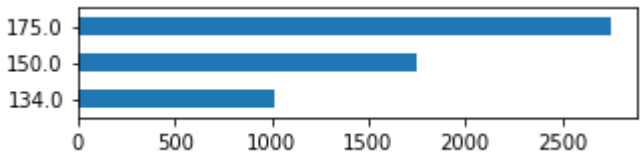

```
========== NEW FAMILIES - Budget ==========
134.0    309
150.0    448
175.0    763
Name: Pack_Size, dtype: int64
```



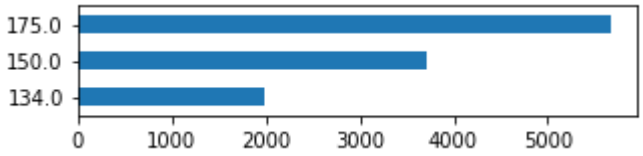

```
========== NEW FAMILIES - Mainstream ==========
134.0    224
150.0    384
175.0    579
Name: Pack_Size, dtype: int64
```



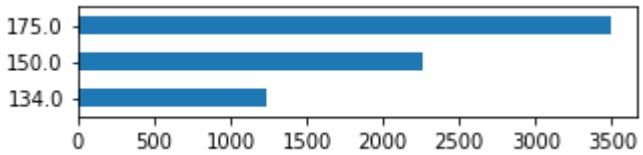

```
========== OLDER FAMILIES - Premium ==========
134.0    1014
150.0    1750
175.0    2747
Name: Pack_Size, dtype: int64
```



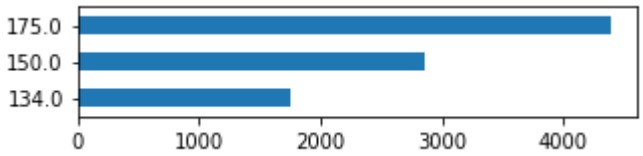
```
========== OLDER FAMILIES - Budget ==========
134.0    1996
150.0    3708
175.0    5662
Name: Pack_Size, dtype: int64
```



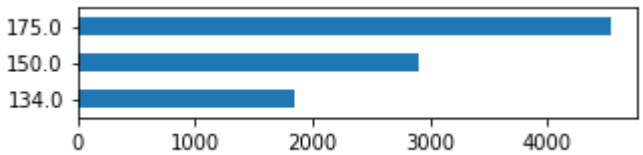

```
========== OLDER FAMILIES - Mainstream ==========
134.0    1234
150.0    2261
175.0    3489
Name: Pack_Size, dtype: int64
```



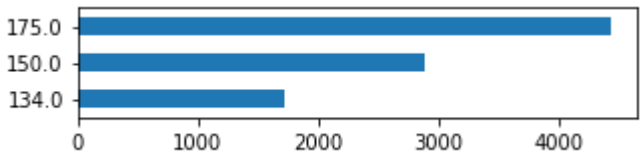

```
========== OLDER SINGLES/COUPLES - Premium ==========
134.0    1744
150.0    2854
175.0    4382
Name: Pack_Size, dtype: int64
```



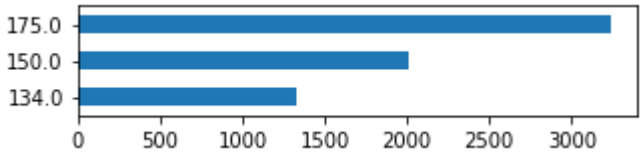

```
========== OLDER SINGLES/COUPLES - Budget ==========
134.0    1843
150.0    2899
175.0    4535
Name: Pack_Size, dtype: int64
```



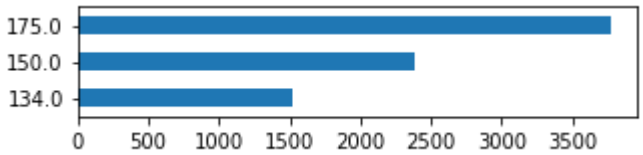

```
========== OLDER SINGLES/COUPLES - Mainstream ==========
134.0    1720
150.0    2875
175.0    4422
Name: Pack_Size, dtype: int64
```



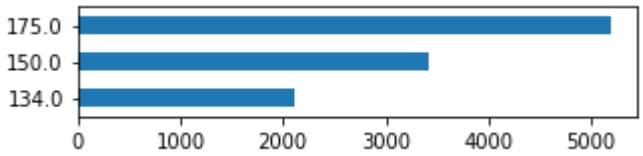
```
========== RETIREES - Premium ==========
134.0    1331
150.0    2015
175.0    3232
Name: Pack_Size, dtype: int64
```



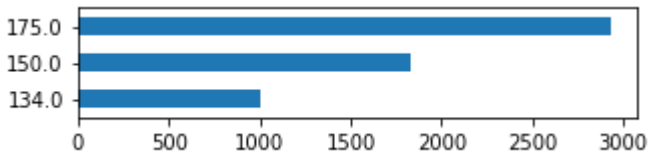

```
========== RETIREES - Budget ==========
134.0    1517
150.0    2381
175.0    3768
Name: Pack_Size, dtype: int64
```



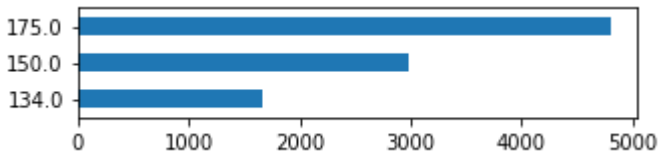

```
========== RETIREES - Mainstream ==========
134.0    2103
150.0    3415
175.0    5187
Name: Pack_Size, dtype: int64
```



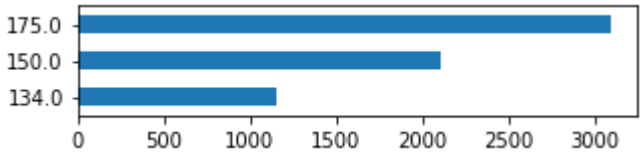

```
========== YOUNG FAMILIES - Premium ==========
134.0    1007
150.0    1832
175.0    2926
Name: Pack_Size, dtype: int64
```




```
========== YOUNG FAMILIES - Budget ==========
134.0    1674
150.0    2981
175.0    4800
Name: Pack_Size, dtype: int64
```



```
========== YOUNG FAMILIES - Mainstream ==========
134.0    1148
150.0    2101
175.0    3087
Name: Pack_Size, dtype: int64
```



In [75]:
```
(temp.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["PROD_QTY"].sum() / temp.g
roupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique()).sort
_values(ascending=False)
```

Out[75]:
```
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
```

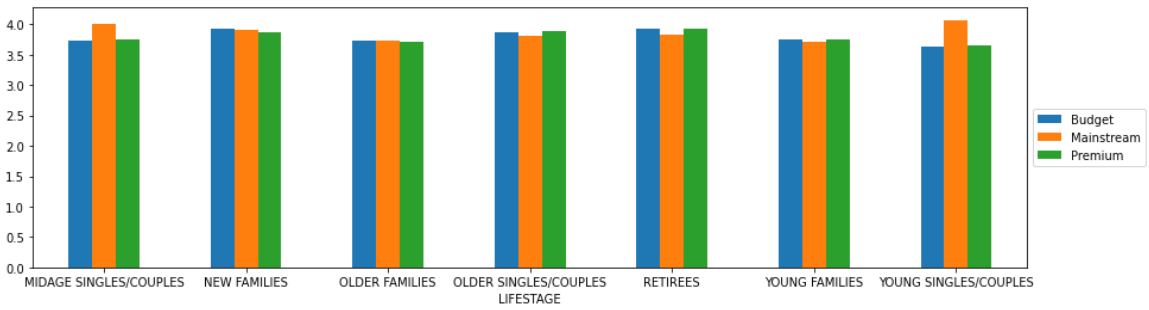
In [76]:
```
(temp.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["PROD_QTY"].sum() / temp.g
roupby(["LIFESTAGE", "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique()).unst
ack().plot.bar(figsize=(15,4), rot=0)
plt.legend(loc="center left", bbox_to_anchor=(1.0, 0.5))
plt.savefig("Average purchase quantity per segment.png", bbox_inches="tigh
t")
```

In [77]:
```
#Average chips price per transaction by segments
temp["Unit_Price"] = temp["TOT_SALES"] / temp["PROD_QTY"]
temp.groupby(["Segment"]).mean()["Unit_Price"].sort_values(ascending=False)
```

Out[77]:
```
Segment
YOUNG SINGLES/COUPLES - Mainstream     4.071485
MIDAGE SINGLES/COUPLES - Mainstream    4.000101
RETIREES - Budget                      3.924883
RETIREES - Premium                     3.921323
NEW FAMILIES - Budget                  3.919251
NEW FAMILIES - Mainstream              3.916581
OLDER SINGLES/COUPLES - Premium        3.887220
OLDER SINGLES/COUPLES - Budget         3.877022
NEW FAMILIES - Premium                 3.871743
RETIREES - Mainstream                  3.833343
OLDER SINGLES/COUPLES - Mainstream     3.803800
YOUNG FAMILIES - Budget                3.753659
MIDAGE SINGLES/COUPLES - Premium       3.752915
YOUNG FAMILIES - Premium               3.752402
OLDER FAMILIES - Budget                3.733344
MIDAGE SINGLES/COUPLES - Budget        3.728496
OLDER FAMILIES - Mainstream            3.727383
YOUNG FAMILIES - Mainstream            3.707097
OLDER FAMILIES - Premium               3.704625
YOUNG SINGLES/COUPLES - Premium        3.645518
YOUNG SINGLES/COUPLES - Budget         3.637681
Name: Unit_Price, dtype: float64
```

In [78]:
```
temp.groupby(["LIFESTAGE", "PREMIUM_CUSTOMER"]).mean()["Unit_Price"].unstac
k().plot.bar(figsize=(15,4), rot=0)
plt.legend(loc="center left", bbox_to_anchor=(1,0.5))
```

Out[78]:
```
<matplotlib.legend.Legend at 0x7efe8dd11630>
```



In [79]:
```
z = temp.groupby(["Segment", "Cleaned_Brand_Names"]).sum()["TOT_SALES"].sort_values(ascending=False).reset_index()
z[z["Segment"] == "YOUNG SINGLES/COUPLES - Mainstream"]
```

Out[79]:

	Segment	Cleaned_Brand_Names	TOT_SALES
0	YOUNG SINGLES/COUPLES - Mainstream	Kettle	35423.6
8	YOUNG SINGLES/COUPLES - Mainstream	Doritos	21705.9
23	YOUNG SINGLES/COUPLES - Mainstream	Pringles	16006.2
24	YOUNG SINGLES/COUPLES - Mainstream	Smiths	15265.7
55	YOUNG SINGLES/COUPLES - Mainstream	Infuzions	8749.4
59	YOUNG SINGLES/COUPLES - Mainstream	Old	8180.4
65	YOUNG SINGLES/COUPLES - Mainstream	Twisties	7539.8
73	YOUNG SINGLES/COUPLES - Mainstream	Tostitos	7238.0
74	YOUNG SINGLES/COUPLES - Mainstream	Thins	7217.1
92	YOUNG SINGLES/COUPLES - Mainstream	Cobs	6144.6
124	YOUNG SINGLES/COUPLES - Mainstream	RRD	4958.1
129	YOUNG SINGLES/COUPLES - Mainstream	Tyrrells	4800.6
148	YOUNG SINGLES/COUPLES - Mainstream	Grain Waves	4201.0
189	YOUNG SINGLES/COUPLES - Mainstream	Cheezels	3318.3
246	YOUNG SINGLES/COUPLES - Mainstream	Natural Chip Co	2130.0
258	YOUNG SINGLES/COUPLES - Mainstream	Woolworths	1929.8
318	YOUNG SINGLES/COUPLES - Mainstream	Cheetos	898.8
327	YOUNG SINGLES/COUPLES - Mainstream	CCs	850.5
383	YOUNG SINGLES/COUPLES - Mainstream	French	429.0
393	YOUNG SINGLES/COUPLES - Mainstream	Sunbites	391.0
415	YOUNG SINGLES/COUPLES - Mainstream	Burger	243.8

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

1. Young Singles/Couples (Mainstream) has the highest population, followed by Retirees (Mainstream). Which explains their high total sales.
2. Despite Older Families not having the highest population, they have the highest frequency of purchase, which contributes to their high total sales.
3. Older Families followed by Young Families has the highest average quantity of chips bought per purchase.
4. 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.
5. Chips brand Kettle is dominating every segment as the most purchased brand.
6. 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).
7. Most frequent chip size purchased is 175gr followed by the 150gr chip size for all segments.

Views and Recommendations:

1. 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.
2. 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.
3. 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.
4. 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.