Main goals of this task are :

Examine transaction data - check for missing data, anomalies, outliers and clean them

Examine customer data - similar to above transaction data

Data analysis and customer segments - create charts and graphs, note trends and insights

Deep dive into customer segments - determine which segments should be targetted

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

```
trans_data = pd.read_excel("/content/QVI_transaction_data.xlsx")
```

```
trans_data
```

```
         DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR                                 PROD_NAME  PROD_QTY  TOT_SALES
     0  43390          1            1000       1         5          Natural Chip Compny SeaSalt175g         2        6.0
     1  43599          1            1307     348        66                  CCs Nacho Cheese 175g         3        6.3
     2  43605          1            1343     383        61       Smiths Crinkle Cut Chips Chicken 170g         2        2.9
     3  43329          2            2373     974        69     Smiths Chip Thinly S/Cream&Onion 175g         5       15.0
     4  43330          2            2426    1038       108     Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
   ...    ...        ...             ...     ...       ...                                       ...       ...        ...
264831  43533        272          272319  270088        89    Kettle Sweet Chilli And Sour Cream 175g         2       10.8
264832  43325        272          272358  270154        74               Tostitos Splash Of Lime 175g         1        4.4
264833  43410        272          272379  270187        51                   Doritos Mexicana 170g         2        8.8
264834  43461        272          272379  270188        42    Doritos Corn Chip Mexican Jalapeno 150g         2        7.8
264835  43365        272          272380  270189        74               Tostitos Splash Of Lime 175g         2        8.8
264836 rows × 8 columns
```

```
trans_data.describe()
```

```
               DATE     STORE_NBR  LYLTY_CARD_NBR        TXN_ID       PROD_NBR       PROD_QTY      TOT_SALES
count  264836.000000  264836.00000    2.648360e+05  2.648360e+05  264836.000000  264836.000000  264836.000000
mean    43464.036260     135.08011    1.355495e+05  1.351583e+05      56.583157       1.907309       7.304200
std       105.389282      76.78418    8.057998e+04  7.813303e+04      32.826638       0.643654       3.083226
min     43282.000000       1.00000    1.000000e+03  1.000000e+00       1.000000       1.000000       1.500000
25%     43373.000000      70.00000    7.002100e+04  6.760150e+04      28.000000       2.000000       5.400000
50%     43464.000000     130.00000    1.303575e+05  1.351375e+05      56.000000       2.000000       7.400000
75%     43555.000000     203.00000    2.030942e+05  2.027012e+05      85.000000       2.000000       9.200000
max     43646.000000     272.00000    2.373711e+06  2.415841e+06     114.000000     200.000000     650.000000
```

```
purch_behav = pd.read_csv("/content/QVI_purchase_behaviour.csv")
```

```
purch_behav.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
```

Next steps: Generate code with purch_behav | View recommended plots | New interactive sheet

```
purch_behav.describe()
```

	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

```
trans_data.isnull().sum()
```

	0
DATE	0
STORE_NBR	0
LYLTY_CARD_NBR	0
TXN_ID	0
PROD_NBR	0
PROD_NAME	0
PROD_QTY	0
TOT_SALES	0

```
purch_behav.isnull().sum()
```

	0
LYLTY_CARD_NBR	0
LIFESTAGE	0
PREMIUM_CUSTOMER	0

```
merged_data = pd.merge(trans_data, purch_behav, on='LYLTY_CARD_NBR' , how = 'right')
merged_data.head()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Premium
1	43359	1	1002	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	YOUNG SINGLES/COUPLES	Mainstream

```
print(len(merged_data))
```

```
264836
```

```
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  int64
 1   STORE_NBR         264836 non-null  int64
 2   LYLTY_CARD_NBR    264836 non-null  int64
 3   TXN_ID            264836 non-null  int64
 4   PROD_NBR          264836 non-null  int64
 5   PROD_NAME         264836 non-null  object
 6   PROD_QTY          264836 non-null  int64
 7   TOT_SALES         264836 non-null  float64
 8   LIFESTAGE         264836 non-null  object
 9   PREMIUM_CUSTOMER  264836 non-null  object
dtypes: float64(1), int64(6), object(3)
memory usage: 20.2+ MB
```

Date Should be in datetime format

```
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)
```

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

```
datetime64[ns]
```

```
merged_data["PROD_NAME"].unique()
```

```
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',
split_prods = merged_data["PROD_NAME"].str.replace(r'(\d+[gG])', '', regex=True).str.replace(r'[^\w]', ' ').str.split()
```

```
split_prods
```

	PROD_NAME
0	[Natural, Chip, Compny, SeaSalt]
1	[Red, Rock, Deli, Chikn&Garlic, Aioli]
2	[Grain, Waves, Sour, Cream&Chives]
3	[Natural, ChipCo, Hony, Soy, Chckn]
4	[WW, Original, Stacked, Chips]
...	...
264831	[Grain, Waves, Sweet, Chilli]
264832	[Kettle, Tortilla, ChpsFeta&Garlic]
264833	[Tyrrells, Crisps, Lightly, Salted]
264834	[Old, El, Paso, Salsa, Dip, Chnky, Tom, Ht]
264835	[Smiths, Crinkle, Chips, Salt, &, Vinegar]

264836 rows × 1 columns

```
word_count = {}
def count_word(line):
  for word in line:
    if word in word_count:
      word_count[word] += 1
    else:
      word_count[word] = 1
split_prods.apply(lambda line: count_word(line))
print(pd.Series(word_count).sort_values(ascending = False))
```

```
Chips       49770
Kettle      41288
&           35565
Smiths      28860
Salt        27976
            ...
Sunbites     1432
Pc           1431
NCC          1419
Garden       1419
Fries        1418
Length: 196, dtype: int64
```

Performing Binning

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

	count
(0.8, 50.75]	264834
(50.75, 100.5]	0
(100.5, 150.25]	0
(150.25, 200.0]	2

```
merged_data.sort_values(by = "PROD_QTY" , ascending = False ).head(10)
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUSTOMER
221626	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0	OLDER FAMILIES	Premium
221625	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0	OLDER FAMILIES	Premium
145041	2019-05-20	148	148238	148046	87	Infuzions BBQ Rib Prawn Crackers 110g	5	19.0	OLDER SINGLES/COUPLES	Mainstream
89729	2018-	93	93152	91631	46	Kettle Original	5	27.0	RETIREES	Premium

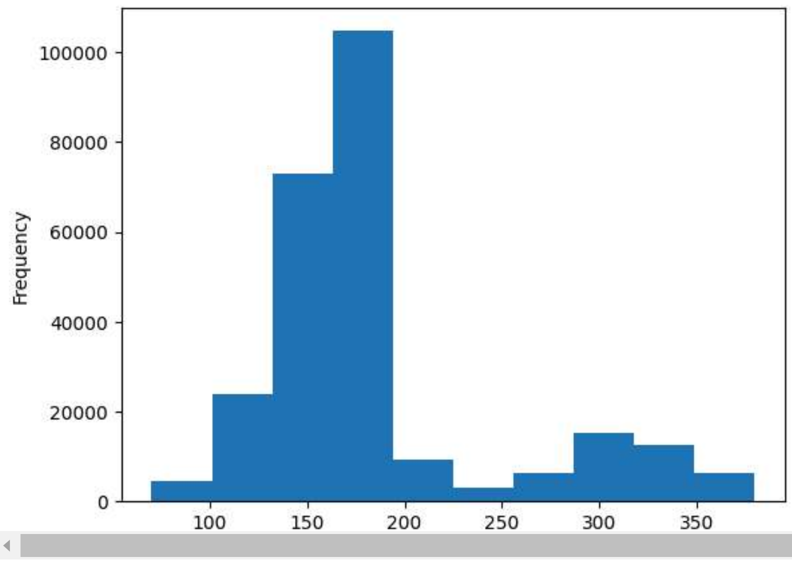
First 2 Rows are outliers in this and they are also the same entry

```
merged_data = merged_data[merged_data["PROD_QTY"] < 6]
```

```
# Step 1: Standardize 'G' and 'g' to lowercase
merged_data["PROD_NAME"] = merged_data["PROD_NAME"].str.replace(r'[0-9]+(G)', 'g', regex=True)

# Step 2: Extract the numeric values followed by 'g' or 'G'
pack_sizes = merged_data["PROD_NAME"].str.extract(r'([0-9]+[gG])')[0]

# Step 3: Remove 'g' and convert the result to a float
pack_sizes = pack_sizes.str.replace('g', '', case=False).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
<Axes: ylabel='Frequency'>
```



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

PROD_NAME	count
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

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.

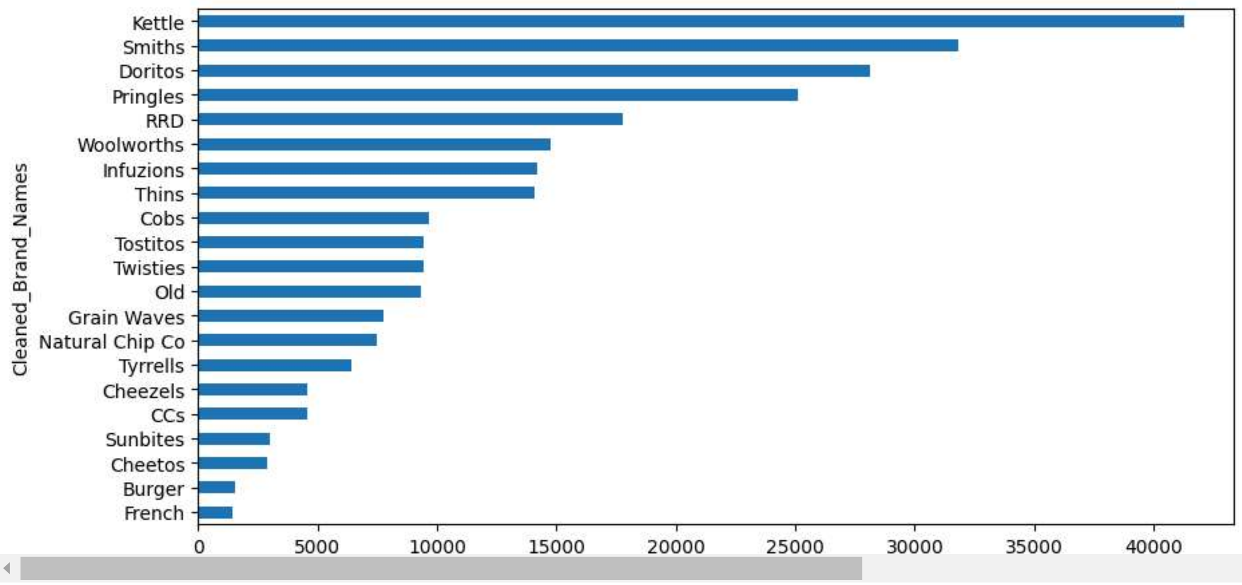
```
merged_data["Cleaned_Brand_Names"] = merged_data["PROD_NAME"].str.split().str[0]
```

```
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
```

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

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

```
<Axes: ylabel='Cleaned_Brand_Names'>
```



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

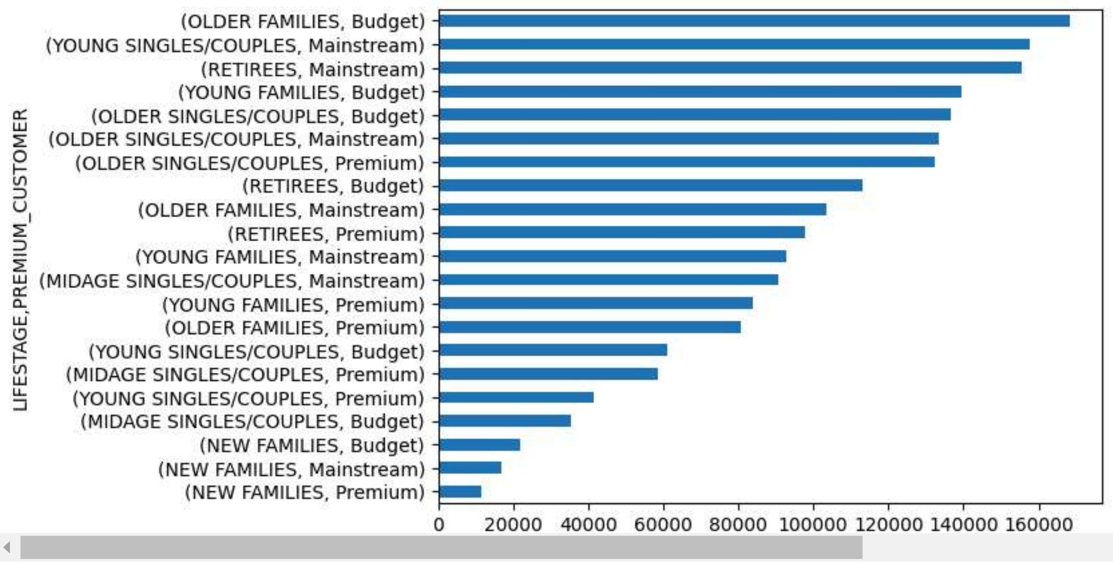
```
grouped_sales = pd.DataFrame(merged_data.groupby(["LIFESTAGE" , "PREMIUM_CUSTOMER"])["TOT_SALES"].agg(["sum", "mean"]))
```

```
grouped_sales.sort_values(ascending=False, by="sum")
```

		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

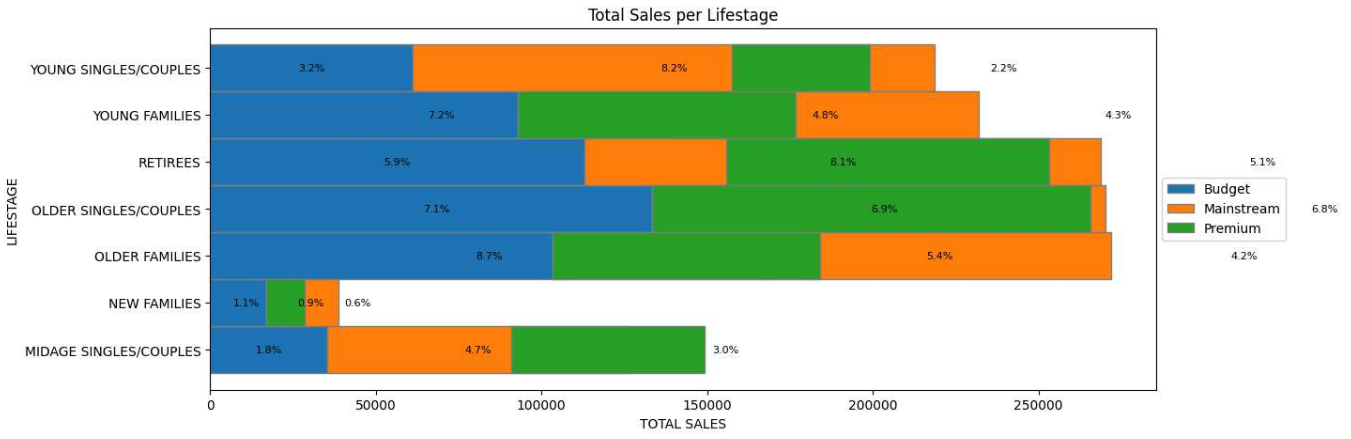
```
grouped_sales["sum"].sort_values().plot.barh()
```

```
<Axes: ylabel='LIFESTAGE,PREMIUM_CUSTOMER'>
```



```
# 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 firs 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")

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', size=8)
    plt.text(budget_width + mains_bar[i].get_width()/2, i, bars2_text[i], va='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()
```

```
<ipython-input-59-3a4a1365980e>:29: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version,
  plt.text(budget_width/2, i, bars1_text[i], va='center', ha='center', size=8)
<ipython-input-59-3a4a1365980e>:30: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version,
  plt.text(budget_width + mains_bar[i].get_width()/2, i, bars2_text[i], va='center', ha='center', size=8)
<ipython-input-59-3a4a1365980e>:31: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version,
  plt.text(budget_main_width + prem_bar[i].get_width()/2, i, bars3_text[i], va='center', ha='center', size=8)
```



```
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):

Older families (Budget) $156,864

Young Singles/Couples (Mainstream) $147,582

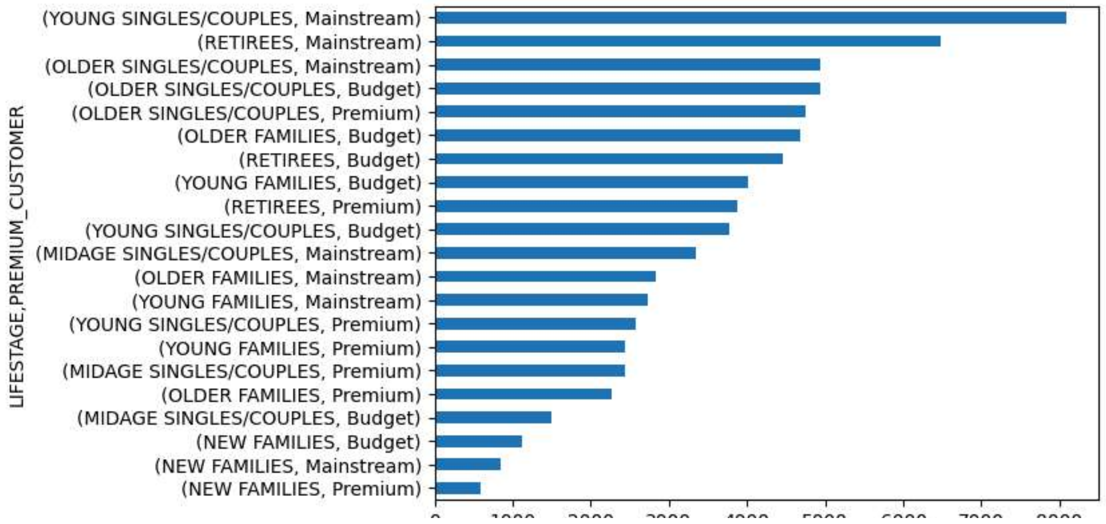
Retirees (Mainstream) $145,169

```
uniq_cust = merged_data.groupby(["LIFESTAGE" , "PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique().sort_values(ascending = False)
print("Number of unique customers per segment")
print(uniq_cust)
```

```
Number of unique customers per segment
LIFESTAGE               PREMIUM_CUSTOMER
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
Name: LYLTY_CARD_NBR, dtype: int64
```

```
uniq_cust.sort_values().plot.barh()
```

```
<Axes: ylabel='LIFESTAGE,PREMIUM_CUSTOMER'>
```


LIFESTAGE,PREMIUM_CUSTOMER
(YOUNG SINGLES/COUPLES, Mainstream)
(RETIREES, Mainstream)
(OLDER SINGLES/COUPLES, Mainstream)
(OLDER SINGLES/COUPLES, Budget)
(OLDER SINGLES/COUPLES, Premium)
(OLDER FAMILIES, Budget)
(RETIREES, Budget)
(YOUNG FAMILIES, Budget)
(RETIREES, Premium)
(YOUNG SINGLES/COUPLES, Budget)
(MIDAGE SINGLES/COUPLES, Mainstream)
(OLDER FAMILIES, Mainstream)
(YOUNG FAMILIES, Mainstream)
(YOUNG SINGLES/COUPLES, Premium)
(YOUNG FAMILIES, Premium)
(MIDAGE SINGLES/COUPLES, Premium)
(OLDER FAMILIES, Premium)
(MIDAGE SINGLES/COUPLES, Budget)
(NEW FAMILIES, Budget)
(NEW FAMILIES, Mainstream)
(NEW FAMILIES, Premium)