

Quantum virtual internship task 1

Deploying the six phases of data analytics; Ask, Prepare, Process, Analyze, Visualize and Act.

Ask phase

The aim and objective of this analysis is to understand the type of customers who purchase chips and their purchasing behaviour within the region and our stakeholder is Mrs Julia the chips category manager.

Prepare process

In [1]: 1 # Let us import some necessary Libraries and read our datasets

In [2]: 1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from datetime import datetime
5 import seaborn as sns
6 sns.set()
7 %matplotlib inline

In [3]: 1 trans_df = pd.read_excel("QVI_transaction_data.xlsx")
2 behave_df = pd.read_csv("QVI_purchase_behaviour.csv")

In [4]: 1 # Let us see how our datasets are organised

In [5]: 1 trans_df.head(10)

Out[5]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TO
0	43390		1	1000	1	5	Natural Chip Compy 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
5	43604		4	4074	2982	57	Old El Paso Salsa Dip Tomato Mild 300g	1
6	43601		4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g	1
7	43601		4	4196	3539	24	Grain Waves Sweet Chilli 210g	1
8	43332		5	5026	4525	42	Doritos Corn Chip Mexican Jalapeno 150g	1
9	43330		7	7150	6900	52	Grain Waves Sour Cream&Chives 210G	2

In [6]: 1 trans_df.TOT_SALES.sort_values(ascending = False)

Out[6]:

69762	650.0
69763	650.0
69496	29.5
55558	29.5
171815	29.5
	...
259695	1.5
259707	1.5
197005	1.5
216449	1.5
150019	1.5

Name: TOT_SALES, Length: 264836, dtype: float64

In [7]: 1 trans_df.TOT_SALES.describe()

Out[7]: count 264836.000000
mean 7.304200
std 3.083226
min 1.500000
25% 5.400000
50% 7.400000
75% 9.200000
max 650.000000
Name: TOT_SALES, dtype: float64

In [8]: 1 trans_df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   DATE             264836 non-null   int64  
 1   STORE_NBR        264836 non-null   int64  
 2   LYLTY_CARD_NBR  264836 non-null   int64  
 3   TXN_ID           264836 non-null   int64  
 4   PROD_NBR         264836 non-null   int64  
 5   PROD_NAME        264836 non-null   object  
 6   PROD_QTY         264836 non-null   int64  
 7   TOT_SALES        264836 non-null   float64 
dtypes: float64(1), int64(6), object(1)
memory usage: 16.2+ MB
```

In [9]: 1 trans_df.describe()

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY
count	264836.000000	264836.000000	2.648360e+05	2.648360e+05	264836.000000	264836.000000
mean	43464.036260	135.08011	1.355495e+05	1.351583e+05	56.583157	1.907
std	105.389282	76.78418	8.057998e+04	7.813303e+04	32.826638	0.643
min	43282.000000	1.00000	1.000000e+03	1.000000e+00	1.000000	1.000
25%	43373.000000	70.00000	7.002100e+04	6.760150e+04	28.000000	2.000
50%	43464.000000	130.00000	1.303575e+05	1.351375e+05	56.000000	2.000
75%	43555.000000	203.00000	2.030942e+05	2.027012e+05	85.000000	2.000
max	43646.000000	272.00000	2.373711e+06	2.415841e+06	114.000000	200.000

```
In [10]: 1 trans_df.dtypes
```

```
Out[10]: DATE          int64
STORE_NBR      int64
LYLTY_CARD_NBR  int64
TXN_ID         int64
PROD_NBR       int64
PROD_NAME      object
PROD_QTY        int64
TOT_SALES      float64
dtype: object
```

```
In [11]: 1 trans_df.isna().sum()
```

```
Out[11]: 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 [12]: 1 # from the transaction dataset we see that "DATE" is in integer datatype
```

```
In [13]: 1 behave_df.head(10)
```

	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
5	1007	YOUNG SINGLES/COUPLES	Budget
6	1009	NEW FAMILIES	Premium
7	1010	YOUNG SINGLES/COUPLES	Mainstream
8	1011	OLDER SINGLES/COUPLES	Mainstream
9	1012	OLDER FAMILIES	Mainstream

```
In [14]: 1 behave_df.PREMIUM_CUSTOMER.unique()
```

```
Out[14]: array(['Premium', 'Mainstream', 'Budget'], dtype=object)
```

```
In [15]: 1 behave_df.PREMIUM_CUSTOMER.value_counts().sort_values(ascending = False)
```

```
Out[15]: Mainstream    29245
          Budget        24470
          Premium       18922
          Name: PREMIUM_CUSTOMER, dtype: int64
```

```
In [16]: 1 behave_df.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]: 1 behave_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72637 entries, 0 to 72636
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   LYLTY_CARD_NBR    72637 non-null   int64  
 1   LIFESTAGE         72637 non-null   object  
 2   PREMIUM_CUSTOMER  72637 non-null   object  
dtypes: int64(1), object(2)
memory usage: 1.7+ MB
```

```
In [18]: 1 behave_df.isna().sum()
```

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

```
In [19]: 1 # From the customer behaviour dataset we can see there no missing and t
```

The two datasets given align with the findings as they are reliable, original, comprehensive and cited though they are not current.

Process phase

In [20]: 1 # Let us duplicate our datasets before cleaning

In [21]: 1 trans_df

Out[21]:

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

264836 rows × 8 columns



In [22]: 1 # Let us convert "DATE" datatype to datetime

In [23]: 1 trans_df["DATE"] = pd.to_datetime(trans_df["DATE"])

```
In [24]: 1 trans_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
```

```
In [25]: 1 # we now have our DATE in datetime. Let us create DAY column
```

```
In [26]: 1 trans_df["DAY"] = pd.DatetimeIndex(trans_df["DATE"]).day_name()
```

In [27]: 1 trans_df

Out[27]:

		DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0		1970-01-01 00:00:00.000043390		1	1000	1	5 Natural Chip Compani SeaSalt175g
1		1970-01-01 00:00:00.000043599		1	1307	348	66 CCs Nacho Cheese 175g
2		1970-01-01 00:00:00.000043605		1	1343	383	61 Smiths Crinkle Cut Chips Chicken 170g
3		1970-01-01 00:00:00.000043329		2	2373	974	69 Smiths Chilli Thin S/Cream&Onions 175g
4		1970-01-01 00:00:00.000043330		2	2426	1038	108 Kettle Tortilla ChpsHny&Jlpnchili 150g
...
264831		1970-01-01 00:00:00.000043533		272	272319	270088	89 Kettle Sweet Chilli And Sou Cream 175g
264832		1970-01-01 00:00:00.000043325		272	272358	270154	74 Tostitos Splashed Of Lime 175g
264833		1970-01-01 00:00:00.000043410		272	272379	270187	51 Doritos Mexicana 170g
264834		1970-01-01 00:00:00.000043461		272	272379	270188	42 Doritos Cori Chip Mexican Jalapeno 150g
264835		1970-01-01 00:00:00.000043365		272	272380	270189	74 Tostitos Splashed Of Lime 175g

264836 rows × 9 columns



In [28]: 1 # Let us create month column

In [29]: 1 trans_df["MONTH"] = pd.DatetimeIndex(trans_df["DATE"]).month_name()

In [30]: 1 trans_df

Out[30]:

		DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0		1970-01-01 00:00:00.000043390		1	1000	1	5 Natural Chip Compan SeaSalt175g
1		1970-01-01 00:00:00.000043599		1	1307	348	66 CCs Nacho Cheese 175g
2		1970-01-01 00:00:00.000043605		1	1343	383	61 Smiths Crinkle Cut Chip Chicken 170g
3		1970-01-01 00:00:00.000043329		2	2373	974	69 Smiths Chip Thin S/Cream&Onion 175g
4		1970-01-01 00:00:00.000043330		2	2426	1038	108 Kettle Tortilla ChpsHny&Jlpn Chili 150g
...
264831		1970-01-01 00:00:00.000043533		272	272319	270088	89 Kettle Swee Chilli And Sou Cream 175g
264832		1970-01-01 00:00:00.000043325		272	272358	270154	74 Tostitos Splas Of Lime 175g
264833		1970-01-01 00:00:00.000043410		272	272379	270187	51 Dorito Mexicana 170g
264834		1970-01-01 00:00:00.000043461		272	272379	270188	42 Doritos Cor Chip Mexican Jalapeno 150g
264835		1970-01-01 00:00:00.000043365		272	272380	270189	74 Tostitos Splas Of Lime 175g

264836 rows × 10 columns



In [31]: 1 # Let us create YEAR column

In [32]: 1 trans_df["YEAR"] = pd.DatetimeIndex(trans_df["DATE"]).year

In [33]: 1 trans_df

Out[33]:

		DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0		1970-01-01 00:00:00.000043390	1		1000	1	5 Natural Chip Compani SeaSalt175g
1		1970-01-01 00:00:00.000043599	1		1307	348	66 CCs Nacho Cheese 175g
2		1970-01-01 00:00:00.000043605	1		1343	383	61 Smiths Crinkle Cut Chip Chicken 170g
3		1970-01-01 00:00:00.000043329	2		2373	974	69 Smiths Chip Thinn S/Cream&Onion 175g
4		1970-01-01 00:00:00.000043330	2		2426	1038	108 Kettle Tortilla ChpsHny&JlpnC Chili 150g
...
264831		1970-01-01 00:00:00.000043533	272		272319	270088	89 Kettle Swee Chilli And Sou Cream 175g
264832		1970-01-01 00:00:00.000043325	272		272358	270154	74 Tostitos Splas Of Lime 175g
264833		1970-01-01 00:00:00.000043410	272		272379	270187	51 Dorito Mexicana 170g
264834		1970-01-01 00:00:00.000043461	272		272379	270188	42 Doritos Cor Chip Mexican Jalapeno 150g
264835		1970-01-01 00:00:00.000043365	272		272380	270189	74 Tostitos Splas Of Lime 175g

264836 rows × 11 columns



In [34]: 1 # we have successfully added DAY, MONTH and YEAR column to our trans da
2 # Let us work on the inconsistency in PROD_NAME column

In [35]: 1 def remove_prod(df):
2 unwanted = "salsa"
3 if unwanted in df["PROD_NAME"].lower().split():
4 return df.name

In [36]: 1 drop_index = list(trans_df.apply(remove_prod, axis = 1))

In [37]: 1 old_df = trans_df.copy()

```
In [38]: 1 # Making a copy of all transaction and updating the new transactions wi
          2 old_df = trans_df.copy()
          3 trans_df = trans_df.drop([i for i in drop_index if ~np.isnan(i)])
```

```
In [39]: 1 print("Total Trans: "+str(len(old_df)))
          2 print("Total Salsa Products: "+str(len(old_df)-len(trans_df)))
          3 print("Trans without Salsa: "+str(len(trans_df)))
```

Total Trans: 264836
 Total Salsa Products: 18094
 Trans without Salsa: 246742

```
In [40]: 1 # Let us remove outliers that may skew our analysis
```

```
In [41]: 1 trans_df[trans_df.PROD_QTY > 100]
```

Out[41]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
69762	1970-01-01 00:00:00.000043331	226	226000	226201	4	Dorito Corn Chp Supreme 380g
69763	1970-01-01 00:00:00.000043605	226	226000	226210	4	Dorito Corn Chp Supreme 380g

```
In [42]: 1 trans_df = trans_df[trans_df['PROD_QTY'] < 200].reset_index(drop=True)
          2 trans_df.describe()
```

Out[42]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SA
count	246740.000000	2.467400e+05	2.467400e+05	246740.000000	246740.000000	246740.00
mean	135.050361	1.355303e+05	1.351304e+05	56.352213	1.906456	7.31
std	76.786971	8.071520e+04	7.814760e+04	33.695235	0.342499	2.47
min	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.70
25%	70.000000	7.001500e+04	6.756875e+04	26.000000	2.000000	5.80
50%	130.000000	1.303670e+05	1.351815e+05	53.000000	2.000000	7.40
75%	203.000000	2.030832e+05	2.026522e+05	87.000000	2.000000	8.80
max	272.000000	2.373711e+06	2.415841e+06	114.000000	5.000000	29.50

In [43]:

```

1 def packet_size(grp):
2     string = grp["PROD_NAME"]
3     num = []
4     for i in string:
5         if i.isdigit():
6             num.append(i)
7     number = "".join(num)
8     return int(number)

```

In [44]:

```

1 trans_df["PACKET_SIZE"] = trans_df.apply(packet_size, axis=1)
2 trans_df.head()

```

Out[44]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
0	1970-01-01 00:00:00.000043390		1	1000	1	5	Natural Chip Comppny SeaSalt175g
1	1970-01-01 00:00:00.000043599		1	1307	348	66	CCs Nacho Cheese 175g
2	1970-01-01 00:00:00.000043605		1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
3	1970-01-01 00:00:00.000043329		2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g
4	1970-01-01 00:00:00.000043330		2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g



In [45]:

```

1 print("Largest Packet Size: "+str(max(trans_df["PACKET_SIZE"])))+"g")
2 print("Smallest Packet Size: "+str(min(trans_df["PACKET_SIZE"])))+"g")

```

Largest Packet Size: 380g
Smallest Packet Size: 70g

In [46]:

```
1 # let us create some columns for better analysis
```

In [47]:

```

1 def Product_Company(grp):
2     return grp["PROD_NAME"].split()[0]

```

```
In [48]: 1 trans_df["BRAND"] = trans_df.apply(Product_Company, axis=1)
2 trans_df.head()
```

Out[48]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
0	1970-01-01 00:00:00.000043390	1	1000	1	5	Natural Chip Comnpy SeaSalt175g	
1	1970-01-01 00:00:00.000043599	1	1307	348	66	CCs Nacho Cheese 175g	
2	1970-01-01 00:00:00.000043605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	1970-01-01 00:00:00.000043329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	
4	1970-01-01 00:00:00.000043330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlno Chili 150g	

```
In [49]: 1 def Product_Company(grp):
2     return grp["PROD_NAME"].split()[-1]
```

```
In [50]: 1 trans_df["value"] = trans_df.apply(Product_Company, axis=1)
2 trans_df.head()
```

Out[50]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
0	1970-01-01 00:00:00.000043390	1	1000	1	5	Natural Chip Comnpy SeaSalt175g	
1	1970-01-01 00:00:00.000043599	1	1307	348	66	CCs Nacho Cheese 175g	
2	1970-01-01 00:00:00.000043605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	1970-01-01 00:00:00.000043329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	
4	1970-01-01 00:00:00.000043330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlno Chili 150g	

```
In [51]: 1 trans_df["value"].unique()
```

Out[51]: array(['SeaSalt175g', '175g', '170g', '150g', '330g', '210g', '210G',
 '270g', '220g', '125g', '110g', '134g', '150G', 'Chli&S/Cream175G',
 'Chckn175g', '380g', 'Chicken270g', '180g', '165g', 'Salt', '250g',
 'Chs&Onion170g', '200g', '160g', 'CutSalt/Vinegr175g', '190g',
 '90g', '70g'], dtype=object)

```
In [52]: 1 def Product_Company(grp):  
2     return grp["PROD_NAME"].split()[-1][-4:]
```

```
In [53]: 1 trans_df["o_value"] = trans_df.apply(Product_Company, axis=1)  
2 trans_df.head()
```

Out[53]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
0	1970-01-01 00:00:00.000043390	1		1000	1	5	Natural Chip Compy SeaSalt175g
1	1970-01-01 00:00:00.000043599	1		1307	348	66	CCs Nacho Cheese 175g
2	1970-01-01 00:00:00.000043605	1		1343	383	61	Smiths Crinkle Cut Chips Chicken 170g
3	1970-01-01 00:00:00.000043329	2		2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g
4	1970-01-01 00:00:00.000043330	2		2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g



```
In [54]: 1 trans_df["PACKET_SIZE"] = trans_df["o_value"]
```

In [55]: 1 trans_df

Out[55]:

		DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME
0		1970-01-01 00:00:00.000043390		1	1000	1	5 Natural Chip Compani SeaSalt175g
1		1970-01-01 00:00:00.000043599		1	1307	348	66 CCs Nacho Cheese 175g
2		1970-01-01 00:00:00.000043605		1	1343	383	61 Smiths Crinkle Cut Chips Chicken 170g
3		1970-01-01 00:00:00.000043329		2	2373	974	69 Smiths Chip Thinn S/Cream&Onion 175g
4		1970-01-01 00:00:00.000043330		2	2426	1038	108 Kettle Tortilla ChpsHny&Jlpnch Chili 150g
...
246735		1970-01-01 00:00:00.000043533		272	272319	270088	89 Kettle Swee Chilli And Sou Cream 175g
246736		1970-01-01 00:00:00.000043325		272	272358	270154	74 Tostitos Splas Of Lime 175g
246737		1970-01-01 00:00:00.000043410		272	272379	270187	51 Dorito Mexicana 170g
246738		1970-01-01 00:00:00.000043461		272	272379	270188	42 Doritos Cor Chip Mexican Jalapeno 150g
246739		1970-01-01 00:00:00.000043365		272	272380	270189	74 Tostitos Splas Of Lime 175g

246740 rows × 15 columns



In [56]: 1 trans_df = trans_df.drop(["o_value", "value", "DATE"], axis = 1)

In [57]:

```

1 d = {'red':'RRD', 'ww':'WOOLWORTHS', 'ncc':'NATURAL','snbts':'SUNBITES','inf'
2 trans_df['BRAND'] = trans_df['BRAND'].str.lower().replace(d).str.upper()
3 trans_df.head(10)

```

Out[57]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALE
0	1		1000	1	Natural Chip Comnpy SeaSalt175g	2	6
1	1		1307	348	CCs Nacho Cheese 175g	3	6
2	1		1343	383	Smiths Crinkle Cut Chips Chicken 170g	2	2
3	2		2373	974	Smiths Chip Thinly S/Cream&Onion 175g	5	15
4	2		2426	1038	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13
5	4		4149	3333	Smiths Crinkle Chips Salt & Vinegar 330g	1	5
6	4		4196	3539	Grain Waves Sweet Chilli 210g	1	3
7	5		5026	4525	Doritos Corn Chip Mexican Jalapeno 150g	1	3
8	7		7150	6900	Grain Waves Sour Cream&Chives 210G	2	7
9	7		7215	7176	Smiths Crinkle Chips Salt & Vinegar 330g	1	5

```
In [58]: 1 trans_df.PACKET_SIZE.value_counts().sort_values()
```

```
Out[58]: 125g    1454
175G    1461
180g    1468
150G    1498
70g     1507
220g    1564
160g    2970
190g    2995
90g     3008
210G    3105
210g    3167
250g    3169
Salt    3257
200g    4473
270g    6285
380g    6416
330g    12540
165g    15297
170g    19983
110g    22387
134g    25102
150g    38705
175g    64929
Name: PACKET_SIZE, dtype: int64
```

```
In [59]: 1 # you can see out PACKET_SIZE column is still not yet consistent. let u
```

```
In [60]: 1 b = {'G':'g','salt':'175g'}
```

```
In [61]: 1 trans_df['PACKET_SIZE'] = trans_df['PACKET_SIZE'].str.lower().replace(b)
```

```
In [62]: 1 trans_df.PACKET_SIZE.value_counts().sort_values()
```

```
Out[62]: 125g    1454
180g    1468
70g     1507
220g    1564
160g    2970
190g    2995
90g     3008
250g    3169
200g    4473
210g    6272
270g    6285
380g    6416
330g    12540
165g    15297
170g    19983
110g    22387
134g    25102
150g    40203
175g    69647
Name: PACKET_SIZE, dtype: int64
```

```
In [63]: 1 behave_df.head()
```

```
Out[63]:   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
```

Analyze phase

```
In [64]: 1 # Let us combined our datasets
```

```
In [65]: 1 combined_data = trans_df.join(behave_df.set_index('LYLTY_CARD_NBR'), on =
2 combined_data.head()
```

Out[65]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALE
0	1	1000	1	5	Natural Chip Comnpy SeaSalt175g	2	6
1	1	1307	348	66	CCs Nacho Cheese 175g	3	6
2	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2
3	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15
4	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13



```
In [66]: 1 combined_data.isna().sum()
```

Out[66]:

STORE_NBR	0
LYLTY_CARD_NBR	0
TXN_ID	0
PROD_NBR	0
PROD_NAME	0
PROD_QTY	0
TOT_SALES	0
DAY	0
MONTH	0
YEAR	0
PACKET_SIZE	0
BRAND	0
LIFESTAGE	0
PREMIUM_CUSTOMER	0
dtype: int64	

In [67]: 1 combined_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246740 entries, 0 to 246739
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   STORE_NBR        246740 non-null   int64  
 1   LYLTY_CARD_NBR  246740 non-null   int64  
 2   TXN_ID           246740 non-null   int64  
 3   PROD_NBR         246740 non-null   int64  
 4   PROD_NAME        246740 non-null   object  
 5   PROD_QTY         246740 non-null   int64  
 6   TOT_SALES        246740 non-null   float64 
 7   DAY              246740 non-null   object  
 8   MONTH             246740 non-null   object  
 9   YEAR              246740 non-null   int64  
 10  PACKET_SIZE      246740 non-null   object  
 11  BRAND             246740 non-null   object  
 12  LIFESTAGE         246740 non-null   object  
 13  PREMIUM_CUSTOMER 246740 non-null   object  
dtypes: float64(1), int64(6), object(7)
memory usage: 26.4+ MB
```

In [68]: 1 trans_df["YEAR"].nunique()

Out[68]: 1

In [69]: 1 trans_df["YEAR"].unique()

Out[69]: array([1970], dtype=int64)

In [70]: 1 trans_df["MONTH"].nunique()

Out[70]: 1

In [71]: 1 trans_df["MONTH"].unique()

Out[71]: array(['January'], dtype=object)

In [72]: 1 trans_df["DAY"].nunique()

Out[72]: 1

In [73]: 1 trans_df["DAY"].unique()

Out[73]: array(['Thursday'], dtype=object)

In [74]: 1 # It was a dataset collected on Thursday January 1970

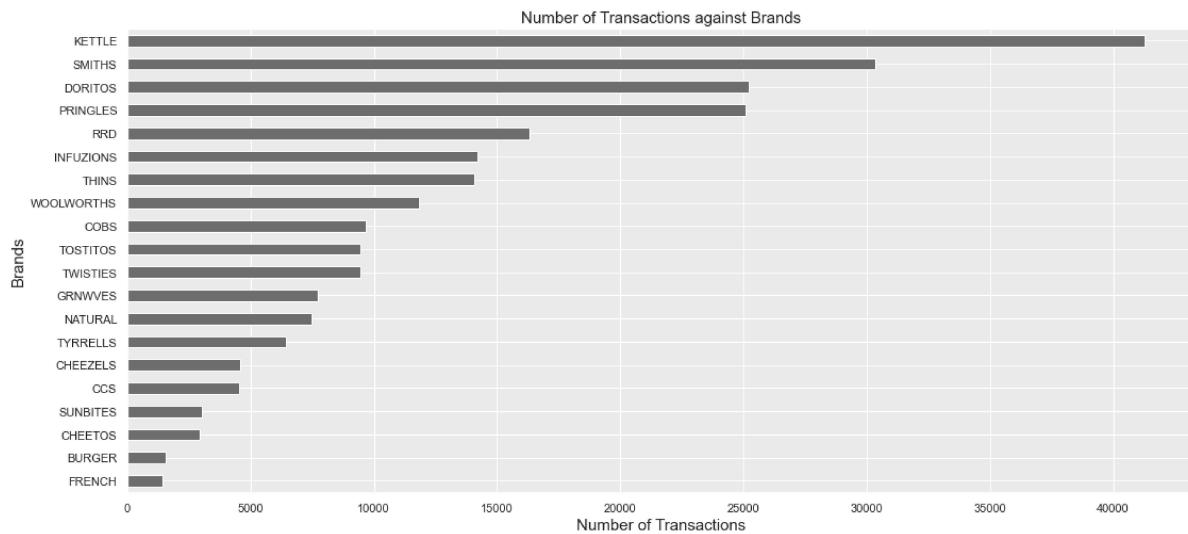
Share phase

In [75]:

```

1 plt.xlabel('Number of Transactions', {'fontsize':15})
2 plt.ylabel('Brands', {'fontsize':15})
3 trans_df.BRAND.value_counts().sort_values().plot(kind='barh', figsize=(18,8)
4 print("figure 1.0")
5 plt.title("Number of Transactions against Brands", {'fontsize':15});
```

figure 1.0



In [76]:

```

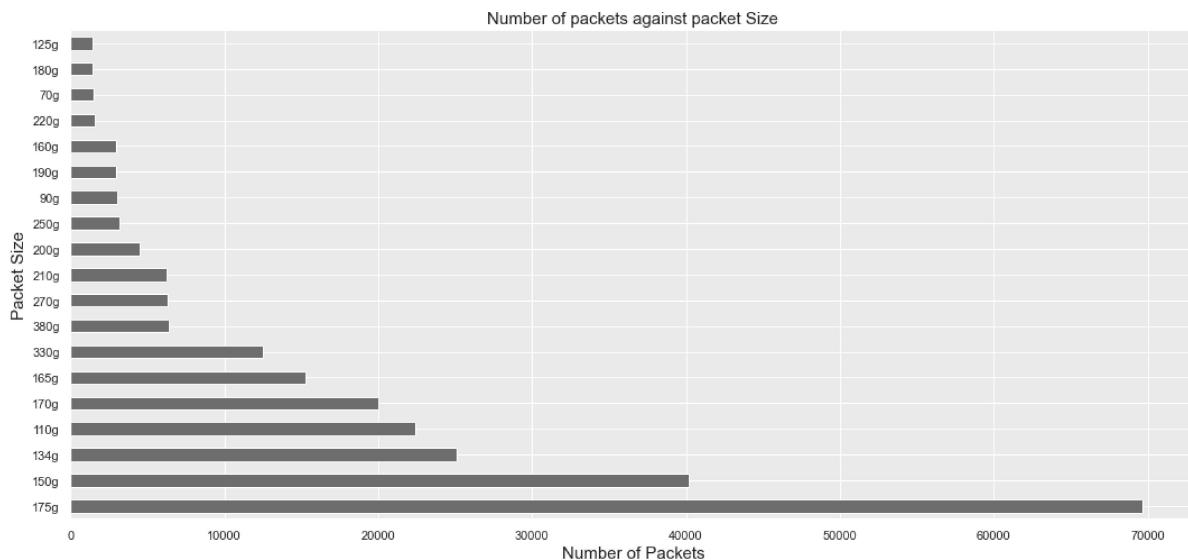
1 # people purchase more product KETTLE and SMITHS while less FRENCH and L
```

In [77]:

```

1 plt.xlabel('Number of Packets', {'fontsize':15})
2 plt.ylabel('Packet Size', {'fontsize':15})
3 trans_df.PACKET_SIZE.value_counts().plot(kind='barh', figsize=(18,8))
4 plt.xticks(rotation='horizontal')
5 print("figure 2.0")
6 plt.title("Number of packets against packet Size", {'fontsize':15});
```

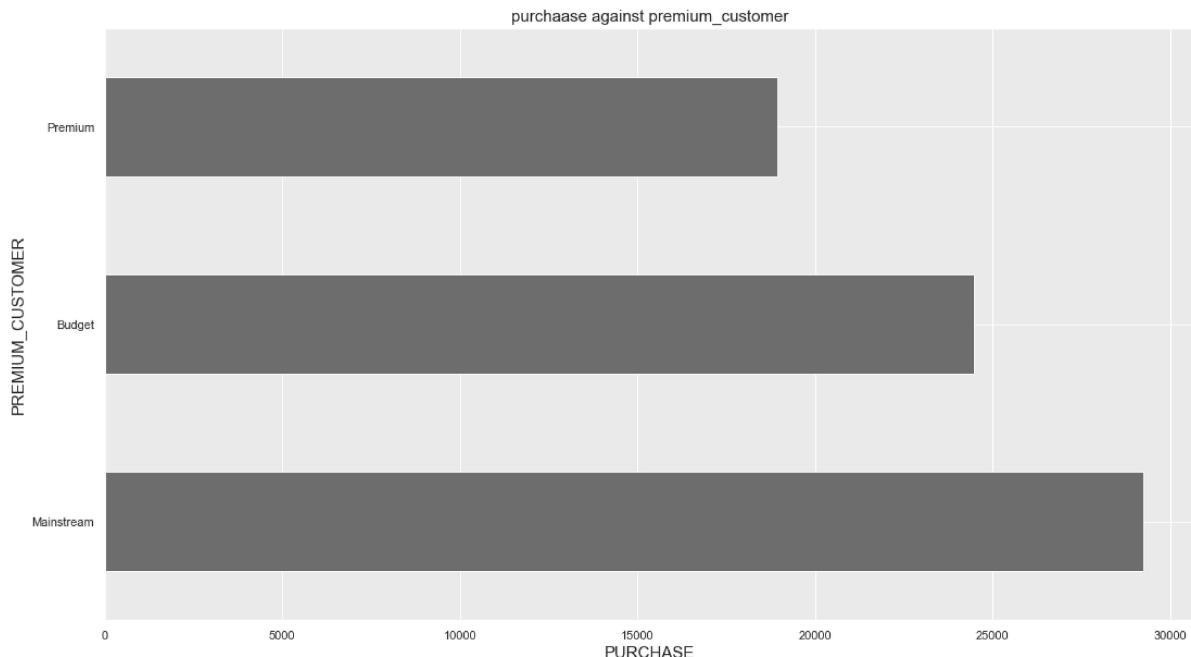
figure 2.0



In []: 1

```
In [78]: 1 behave_df["PREMIUM_CUSTOMER"].value_counts().plot(kind = "barh",figsize =
2 plt.xlabel('PURCHASE',{'fontsize':15})
3 plt.ylabel('PREMIUM_CUSTOMER',{'fontsize':15})
4 print("figure 3.0")
5 plt.title("purchaase against premium_customer",{"fontsize":15});
```

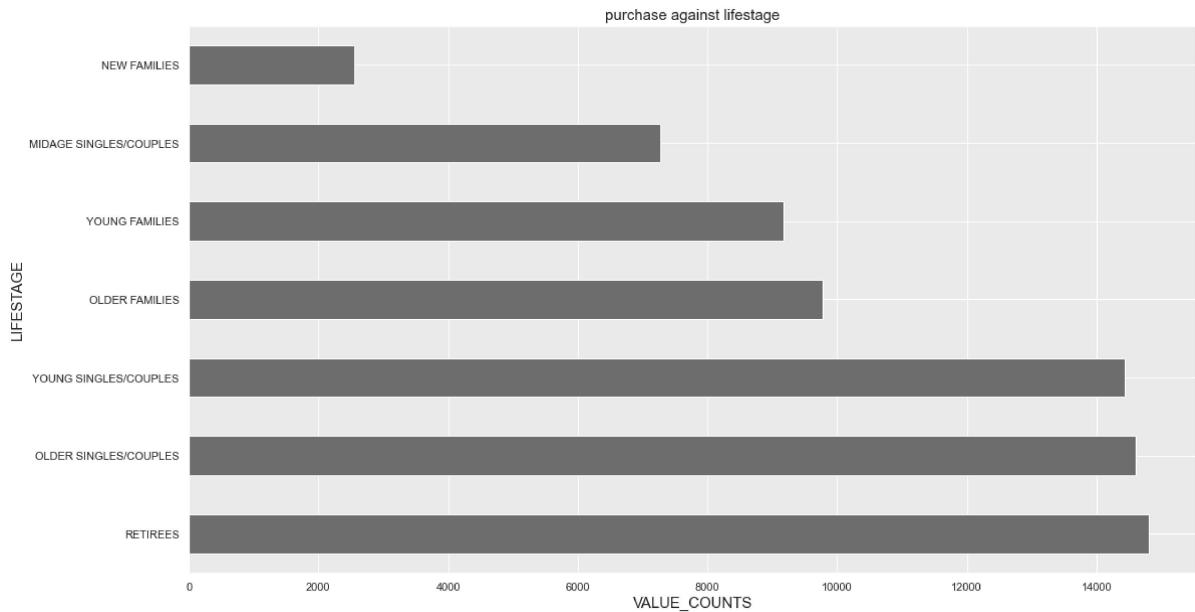
figure 3.0



```
In [79]: 1 #      mainstream purchase most while premium purchase Least
```

```
In [80]: 1 behave_df["LIFESTAGE"].value_counts().plot(kind = "barh",figsize = (18,10))
2 plt.xlabel("VALUE_COUNTS",{'fontsize':15})
3 plt.ylabel('LIFESTAGE',{'fontsize':15});
4 print("figure 4.0")
5 plt.title("purchase against lifestage",{"fontsize":15});
```

figure 4.0



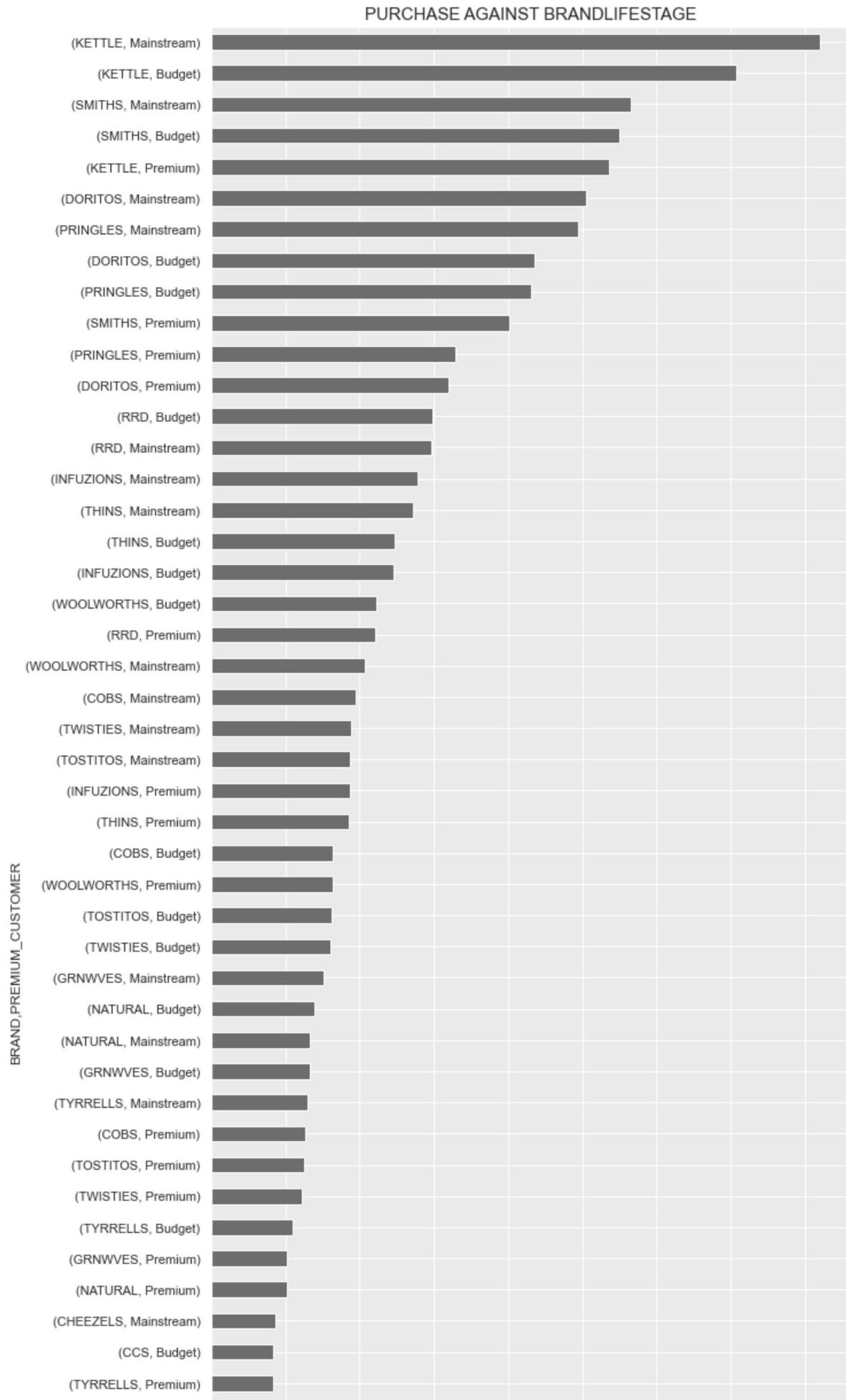
```
In [81]: 1 # from the chat above you can see that the working class LIFESTAGE purchas
```

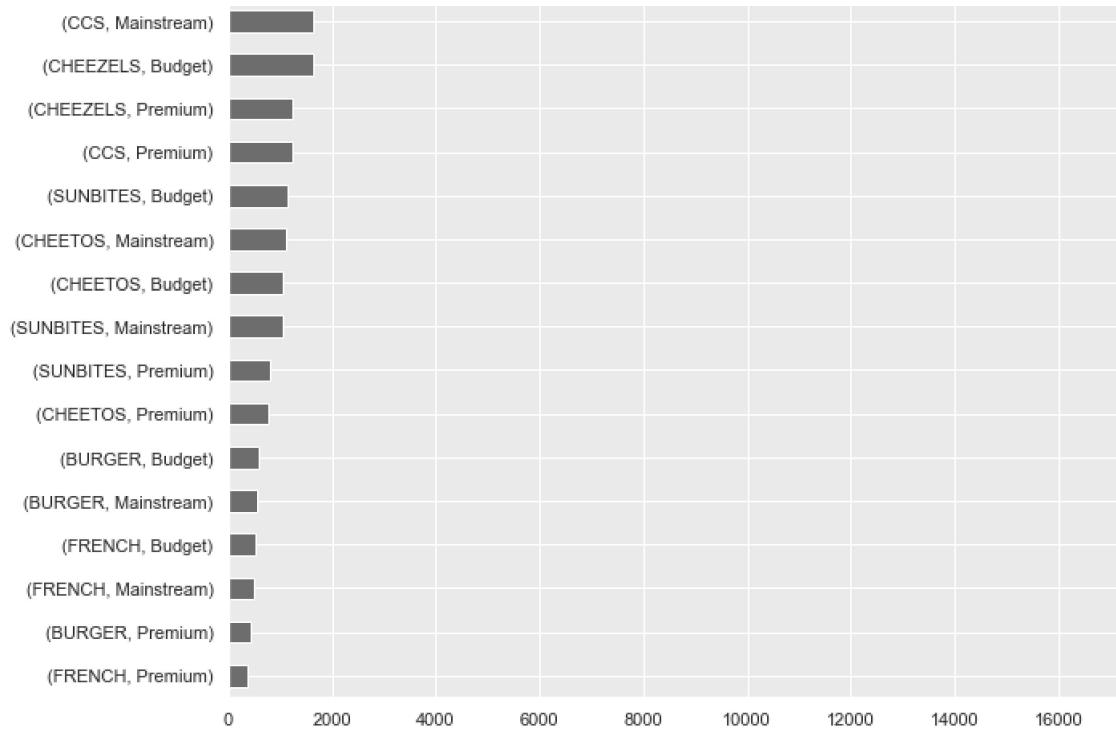
```
In [82]: 1 BRANDgrp = combined_data["PREMIUM_CUSTOMER"].groupby(combined_data["BRAND"])
```

In [83]:

```
1 BRANDgrp.plot(kind = "barh",figsize = (10,30));
2 print("figure 5.0")
3 plt.title("PURCHASE AGAINST BRANDLIFESTAGE",{"fontsize":15});
```

figure 5.0





```
In [84]: 1 BRAND_LIFESTAGEgrp = combined_data["LIFESTAGE"].groupby(combined_data["BRA
◀ ━━━━━━ 100 ━━━━━━ ▶
```

```
In [85]: 1 BRAND_LIFESTAGEgrp.plot(kind = "barh", figsize = (10,60))
2 print("figure 6.0")
3 plt.title("purchase against brandlifestage", {"fontsize":15});
```

figure 6.0



Act phase

CONCLUSION

1 The dataset was collected thursday january 1907

2 Kettle and smiths brands are most demanded

3 French and Burger brands are least demanded

4 Mainstream premium customer, purchase more

5 Premium purchase least

6 Working class lifestage, purchase least

ASSUMPTIONS

I assumed the product_name variable comprises of only chips and salsa during my analysis

```
In [86]: 1 combined_data.to_csv("task1.csv")
```

```
In [ ]: 1
```