Task - 1

The scope of task-1 is to analyze chip for a Qunatium client. They want to gain insights on the sales data.

```
#import initial libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#import transaction data file
from google.colab import files
data=files.upload()
```

Browse... No files selected. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving QVI_transaction_data.xlsx to QVI_transaction_data.xlsx
```

```
#import purchase behaviour file
from google.colab import files
datas=files.upload()
```

Browse... No files selected. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving QVI_purchase_behaviour.csv to QVI_purchase_behaviour.csv
```

Creating and interpreting high level summaries of the data

```
#read ths transaction data file using a variable chips
chips=pd.read_excel(data['QVI_transaction_data.xlsx'])

#exploring the dataset
chips
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
						Natural Chip	

0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	43329	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	5
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
...	...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour	2

```
#exploring the column types
chips.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
```

```
#read the purchase behaviour data file using a variable chips_beh
chips_beh=pd.read_csv('QVI_purchase_behaviour.csv')
```

```
chips_beh
```

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream

4	1005	MIDAGE SINGLES/COUPLES	Mainstream
...	...	...	...
72632	2370651	MIDAGE SINGLES/COUPLES	Mainstream
72633	2370701	YOUNG FAMILIES	Mainstream
72634	2370751	YOUNG FAMILIES	Premium
72635	2370961	OLDER FAMILIES	Budget
72636	2373711	YOUNG SINGLES/COUPLES	Mainstream

72637 rows × 3 columns

```
chips_beh.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
```

```
chips_beh['LIFESTAGE'].value_counts()
```

```
RETIREES                  14805
OLDER SINGLES/COUPLES     14609
YOUNG SINGLES/COUPLES     14441
OLDER FAMILIES             9780
YOUNG FAMILIES             9178
MIDAGE SINGLES/COUPLES     7275
NEW FAMILIES               2549
Name: LIFESTAGE, dtype: int64
```

```
chips_beh['PREMIUM_CUSTOMER'].value_counts()
```

```
Mainstream    29245
Budget        24470
Premium       18922
Name: PREMIUM_CUSTOMER, dtype: int64
```

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

```
array(['Natural Chip        Compny SeaSalt175g',
       'CCs Nacho Cheese    175g',
       'Smiths Crinkle Cut  Chips Chicken 170g',
       'Smiths Chip Thinly  S/Cream&Onion 175g',
       'Kettle Tortilla ChpsHny&Jlpno Chili 150g',
       'Old El Paso Salsa   Dip Tomato Mild 300g',
       'Smiths Crinkle Chips Salt & Vinegar 330g',
       'Grain Waves         Sweet Chilli 210g',
       'Doritos Corn Chip Mexican Jalapeno 150g',
       'Grain Waves Sour    Cream&Chives 210G',
       'Kettle Sensations   Siracha Lime 150g',
       'Twisties Cheese     270g', 'WW Crinkle Cut      Chicken 175g',
       'Thins Chips Light&  Tangy 175g', 'CCs Original 175g',
       'Burger Rings 220g', 'NCC Sour Cream &    Garden Chives 175g',
       'Doritos Corn Chip Southern Chicken 150g',
       'Cheezels Cheese Box 125g', 'Smiths Crinkle      Original 330g',
       'Infzns Crn Crnchers Tangy Gcamole 110g',
       'Kettle Sea Salt     And Vinegar 175g',
       'Smiths Chip Thinly  Cut Original 175g', 'Kettle Original 175g',
       'Red Rock Deli Thai  Chilli&Lime 150g',
       'Pringles Sthrn FriedChicken 134g', 'Pringles Sweet&Spcy BBQ 134g',
       'Red Rock Deli SR    Salsa & Mzzrlla 150g',
       'Thins Chips         Originl saltd 175g',
       'Red Rock Deli Sp    Salt & Truffle 150G',
       'Smiths Thinly       Swt Chli&S/Cream175G', 'Kettle Chilli 175g',
       'Doritos Mexicana    170g',
       'Smiths Crinkle Cut  French OnionDip 150g',
       'Natural ChipCo      Hony Soy Chckn175g',
       'Dorito Corn Chp     Supreme 380g', 'Twisties Chicken270g',
       'Smiths Thinly Cut   Roast Chicken 175g',
       'Smiths Crinkle Cut  Tomato Salsa 150g',
       'Kettle Mozzarella   Basil & Pesto 175g',
       'Infuzions Thai SweetChili PotatoMix 110g',
       'Kettle Sensations   Camembert & Fig 150g',
       'Smith Crinkle Cut   Mac N Cheese 150g',
       'Kettle Honey Soy    Chicken 175g',
       'Thins Chips Seasonedchicken 175g',
       'Smiths Crinkle Cut  Salt & Vinegar 170g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'Doritos Salsa       Medium 300g', 'Kettle 135g Swt Pot Sea Salt',
       'Pringles SourCream  Onion 134g',
       'Doritos Corn Chips  Original 170g',
       'Twisties Cheese     Burger 250g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
       'Woolworths Mild     Salsa 300g',
       'Natural Chip Co     Tmato Hrb&Spce 175g',
       'Smiths Crinkle Cut  Chips Original 170g',
       'Cobs Popd Sea Salt  Chips 110g',
       'Smiths Crinkle Cut  Chips Chs&Onion170g',
       'French Fries Potato Chips 175g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
       'Doritos Corn Chips  Cheese Supreme 170g',
       'Pringles Original   Crisps 134g',
       'RRD Chilli&         Coconut 150g',

#separating chips weight
```

```
#seperating chips weight
chips['WEIGHT']=chips['PROD_NAME'].str[-4:]
chips['WEIGHT']
```

```
0         175g
1         175g
2         170g
3         175g
4         150g
          ...
264831    175g
264832    175g
264833    170g
264834    150g
264835    175g
Name: WEIGHT, Length: 264836, dtype: object
```

```
#salt having weight is 135g
chips['WEIGHT'].value_counts()
```

```
175g    64929
150g    41633
134g    25102
110g    22387
170g    19983
165g    15297
300g    15166
330g    12540
380g     6418
270g     6285
200g     4473
Salt     3257
250g     3169
210g     3167
210G     3105
 90g     3008
190g     2995
160g     2970
220g     1564
 70g     1507
150G     1498
180g     1468
175G     1461
125g     1454
Name: WEIGHT, dtype: int64
```

```
#correcting the data
chips['WEIGHT']=chips['WEIGHT'].replace({'Salt':'135g','210G':'210g','150G':'150g','175G'
```

```
chips['WEIGHT'].value_counts()
```

```
175g    66390
150g    43131
134g    25102
110g    22387
```

```
     170g    19983
     165g    15297
     300g    15166
     330g    12540
     380g     6418
     270g     6285
     210g     6272
     200g     4473
     135g     3257
     250g     3169
      90g     3008
     190g     2995
     160g     2970
     220g     1564
      70g     1507
     180g     1468
     125g     1454
     Name: WEIGHT, dtype: int64
```

```
chips['PROD_NAME']=chips['PROD_NAME'].str.strip()
```

```
#since "salsa" is not a chip
index_drop=chips[chips['PROD_NAME']=="Old belongs to 1 Paso Salsa"].index
chips=chips.drop(index_drop)
```

```
chips[chips['PROD_NAME']=="Old belongs to 1 Paso Salsa"].count()
```

```
     DATE              0
     STORE_NBR         0
     LYLTY_CARD_NBR    0
     TXN_ID            0
     PROD_NBR          0
     PROD_NAME         0
     PROD_QTY          0
     TOT_SALES         0
     WEIGHT            0
     dtype: int64
```

```
#for brand name
chips['BRAND']=chips['PROD_NAME'].str.split().str.get(0)
chips['BRAND'].value_counts()
```

```
     Kettle        41288
     Smiths        28860
     Pringles      25102
     Doritos       24962
     Thins         14075
     RRD           11894
     Infuzions     11057
     WW            10320
     Cobs           9693
     Tostitos       9471
```

```
Twisties       9454
Old            9324
Tyrrells       6442
Grain          6272
Natural        6050
Red            5885
Cheezels       4603
CCs            4551
Woolworths     4437
Dorito         3185
Infzns         3144
Smith          2963
Cheetos        2927
Snbts          1576
Burger         1564
GrnWves        1468
Sunbites       1432
NCC            1419
French         1418
Name: BRAND, dtype: int64
```

```
chips['DATE'].unique()
```

```
array([43390, 43599, 43605, 43329, 43330, 43604, 43601, 43332, 43602,
       43603, 43600, 43326, 43328, 43331, 43327, 43633, 43348, 43370,
       43523, 43543, 43632, 43429, 43414, 43533, 43405, 43537, 43561,
       43311, 43391, 43409, 43528, 43535, 43576, 43468, 43366, 43624,
       43313, 43448, 43587, 43474, 43566, 43529, 43369, 43554, 43519,
       43451, 43564, 43563, 43382, 43407, 43489, 43510, 43627, 43284,
       43299, 43309, 43552, 43593, 43611, 43526, 43578, 43584, 43357,
       43494, 43547, 43550, 43318, 43467, 43555, 43294, 43386, 43496,
       43504, 43520, 43583, 43321, 43446, 43568, 43643, 43646, 43287,
       43562, 43423, 43434, 43479, 43333, 43508, 43634, 43436, 43458,
       43463, 43579, 43620, 43622, 43345, 43361, 43482, 43617, 43625,
       43288, 43352, 43360, 43404, 43290, 43323, 43363, 43399, 43402,
       43551, 43556, 43365, 43367, 43439, 43518, 43539, 43639, 43292,
       43395, 43450, 43462, 43503, 43336, 43375, 43381, 43534, 43644,
       43295, 43509, 43325, 43400, 43443, 43502, 43607, 43302, 43387,
       43412, 43536, 43424, 43499, 43515, 43608, 43308, 43428, 43484,
       43316, 43470, 43572, 43355, 43481, 43571, 43344, 43590, 43475,
       43507, 43641, 43531, 43430, 43389, 43438, 43306, 43408, 43419,
       43615, 43301, 43317, 43452, 43606, 43454, 43485, 43637, 43319,
       43349, 43350, 43445, 43582, 43298, 43351, 43457, 43532, 43283,
       43340, 43455, 43497, 43540, 43631, 43437, 43444, 43303, 43374,
       43589, 43591, 43456, 43538, 43285, 43435, 43477, 43619, 43322,
       43376, 43358, 43464, 43418, 43417, 43553, 43569, 43394, 43420,
       43293, 43384, 43506, 43613, 43616, 43315, 43460, 43565, 43645,
       43483, 43609, 43567, 43337, 43422, 43635, 43542, 43594, 43433,
       43426, 43525, 43585, 43573, 43597, 43286, 43385, 43413, 43546,
       43441, 43478, 43342, 43364, 43472, 43618, 43373, 43449, 43415,
       43396, 43416, 43410, 43498, 43548, 43398, 43480, 43339, 43378,
       43388, 43500, 43610, 43588, 43530, 43621, 43491, 43307, 43300,
       43581, 43392, 43406, 43596, 43490, 43353, 43431, 43522, 43312,
       43427, 43487, 43559, 43383, 43466, 43453, 43516, 43359, 43521,
       43628, 43393, 43324, 43501, 43544, 43421, 43403, 43304, 43347,
       [illegible]
       43541, 43447, 43343, 43397, 43297, 43368, 43461, 43545, 43469,
       43305, 43575, 43334, 43314, 43524, 43514, 43356, 43574, 43411,
       43626, 43549, 43640, 43440, 43362, 43492, 43517, 43570, 43310,
       43558, 43623, 43282, 43341, 43335, 43401, 43289, 43379, 43513,
       43371, 43471, 43638, 43486, 43614, 43629, 43557, 43592, 43338,
       43511, 43432, 43493, 43598, 43380, 43642, 43473, 43291, 43612,
       43320, 43586, 43354, 43527, 43580, 43296, 43636, 43512, 43377,
       43372, 43560, 43465, 43488, 43577, 43630, 43476, 43346, 43425,
       43595, 43442, 43495, 43505])
```

```
chips_salesdate=chips.sort_values(by='DATE')
chips_salesdate['DATE'].unique()
```

```
array([43282, 43283, 43284, 43285, 43286, 43287, 43288, 43289, 43290,
       43291, 43292, 43293, 43294, 43295, 43296, 43297, 43298, 43299,
       43300, 43301, 43302, 43303, 43304, 43305, 43306, 43307, 43308,
       43309, 43310, 43311, 43312, 43313, 43314, 43315, 43316, 43317,
       43318, 43319, 43320, 43321, 43322, 43323, 43324, 43325, 43326,
       43327, 43328, 43329, 43330, 43331, 43332, 43333, 43334, 43335,
       43336, 43337, 43338, 43339, 43340, 43341, 43342, 43343, 43344,
       43345, 43346, 43347, 43348, 43349, 43350, 43351, 43352, 43353,
       43354, 43355, 43356, 43357, 43358, 43359, 43360, 43361, 43362,
       43363, 43364, 43365, 43366, 43367, 43368, 43369, 43370, 43371,
       43372, 43373, 43374, 43375, 43376, 43377, 43378, 43379, 43380,
       43381, 43382, 43383, 43384, 43385, 43386, 43387, 43388, 43389,
       43390, 43391, 43392, 43393, 43394, 43395, 43396, 43397, 43398,
       43399, 43400, 43401, 43402, 43403, 43404, 43405, 43406, 43407,
       43408, 43409, 43410, 43411, 43412, 43413, 43414, 43415, 43416,
       43417, 43418, 43419, 43420, 43421, 43422, 43423, 43424, 43425,
       43426, 43427, 43428, 43429, 43430, 43431, 43432, 43433, 43434,
       43435, 43436, 43437, 43438, 43439, 43440, 43441, 43442, 43443,
       43444, 43445, 43446, 43447, 43448, 43449, 43450, 43451, 43452,
       43453, 43454, 43455, 43456, 43457, 43458, 43460, 43461, 43462,
       43463, 43464, 43465, 43466, 43467, 43468, 43469, 43470, 43471,
       43472, 43473, 43474, 43475, 43476, 43477, 43478, 43479, 43480,
       43481, 43482, 43483, 43484, 43485, 43486, 43487, 43488, 43489,
       43490, 43491, 43492, 43493, 43494, 43495, 43496, 43497, 43498,
       43499, 43500, 43501, 43502, 43503, 43504, 43505, 43506, 43507,
       43508, 43509, 43510, 43511, 43512, 43513, 43514, 43515, 43516,
       43517, 43518, 43519, 43520, 43521, 43522, 43523, 43524, 43525,
       43526, 43527, 43528, 43529, 43530, 43531, 43532, 43533, 43534,
       43535, 43536, 43537, 43538, 43539, 43540, 43541, 43542, 43543,
       43544, 43545, 43546, 43547, 43548, 43549, 43550, 43551, 43552,
       43553, 43554, 43555, 43556, 43557, 43558, 43559, 43560, 43561,
       43562, 43563, 43564, 43565, 43566, 43567, 43568, 43569, 43570,
       43571, 43572, 43573, 43574, 43575, 43576, 43577, 43578, 43579,
       43580, 43581, 43582, 43583, 43584, 43585, 43586, 43587, 43588,
       43589, 43590, 43591, 43592, 43593, 43594, 43595, 43596, 43597,
       43598, 43599, 43600, 43601, 43602, 43603, 43604, 43605, 43606,
       43607, 43608, 43609, 43610, 43611, 43612, 43613, 43614, 43615,
       43616, 43617, 43618, 43619, 43620, 43621, 43622, 43623, 43624,
       43625, 43626, 43627, 43628, 43629, 43630, 43631, 43632, 43633,
       43634, 43635, 43636, 43637, 43638, 43639, 43640, 43641, 43642,
       43643, 43644, 43645, 43646])
```

```
#rrd brand is red brand
#merging two
chips["BRAND"]=chips["BRAND"].replace({'RRD':'Red'})
chips["BRAND"].value_counts()
```

```
Kettle        41288
Smiths        28860
Pringles      25102
Doritos       24962
Red           17779
Thins         14075
Infuzions     11057
WW            10320
Cobs           9693
Tostitos       9471
Twisties       9454
Old            9324
Tyrrells       6442
Grain          6272
Natural        6050
Cheezels       4603
CCs            4551
Woolworths     4437
Dorito         3185
Infzns         3144
Smith          2963
Cheetos        2927
Snbts          1576
Burger         1564
GrnWves        1468
Sunbites       1432
NCC            1419
French         1418
Name: BRAND, dtype: int64
```

```
chips["BRAND"]=chips["BRAND"].replace({'Dorito':'Doritos','Smith':'Smiths','Infzns':'Infu
chips["BRAND"].value_counts()
```

```
Kettle        41288
Smiths        31823
Doritos       28147
Pringles      25102
Red           17779
Infuzions     14201
Thins         14075
WW            10320
Cobs           9693
Tostitos       9471
Twisties       9454
Old            9324
Tyrrells       6442
Grain          6272
Natural        6050
Cheezels       4603
```

```
CCs           4551
Woolworths    4437
Cheetos       2927
Snbts         1576
Burger        1564
GrnWves       1468
Sunbites      1432
NCC           1419
French        1418
Name: BRAND, dtype: int64
```

```
chips
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	43329	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	5
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
...	...	...	...	...	...	...	...
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour	2

```
chips.describe()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	P
count	264836.000000	264836.00000	2.648360e+05	2.648360e+05	264836.000000	26483
mean	43464.036260	135.08011	1.355495e+05	1.351583e+05	56.583157	[illegible]
std	105.389282	76.78418	8.057998e+04	7.813303e+04	32.826638	[illegible]
min	43282.000000	1.00000	1.000000e+03	1.000000e+00	1.000000	[illegible]
25%	43373.000000	70.00000	7.002100e+04	6.760150e+04	28.000000	[illegible]
50%	43464.000000	130.00000	1.303575e+05	1.351375e+05	56.000000	[illegible]
75%	43555.000000	203.00000	2.030942e+05	2.027012e+05	85.000000	[illegible]
max	43646.000000	272.00000	2.373711e+06	2.415841e+06	114.000000	20

max	43646.000000	272.00000	2.373711e+06	2.415841e+06	114.000000	20

```
chips.isnull().sum()
```

```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
WEIGHT            0
BRAND             0
dtype: int64
```

Checking and removing for outliers

```
chips[chips['LYLTY_CARD_NBR']==226000]
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_S
69762	43331	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	6

```
chips=chips.drop([0,1])
chips=chips.reset_index(drop=True)
```

```
chips
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
1	43329	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	5
2	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
3	43604	4	4074	2982	57	Old El Paso Salsa Dip Tomato Mild 300g	1

4	43601	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g	1
...	...	...	...	...	...	...	...

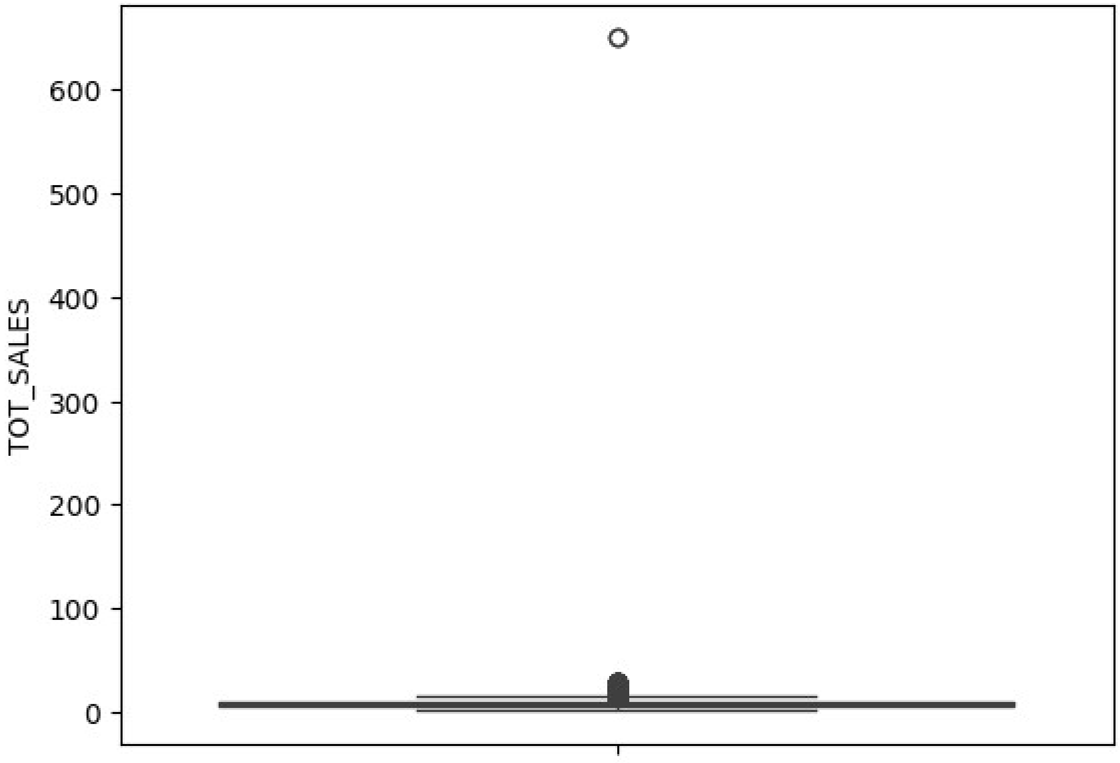
```
chips=pd.merge(chips,chips_beh,on='LYLTY_CARD_NBR',how='left')
chips
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
1	43329	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	5
2	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
3	43604	4	4074	2982	57	Old El Paso Salsa Dip Tomato Mild 300g	1
4	43601	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g	1
...	...	...	...	...	...	...	...
264829	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2
264830	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	1
264831	43410	272	272379	270187	51	Doritos Mexicana 170g	2
264832	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2
264833	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	2

264834 rows × 14 columns

```
sns.boxplot(chips.TOT_SALES)
```

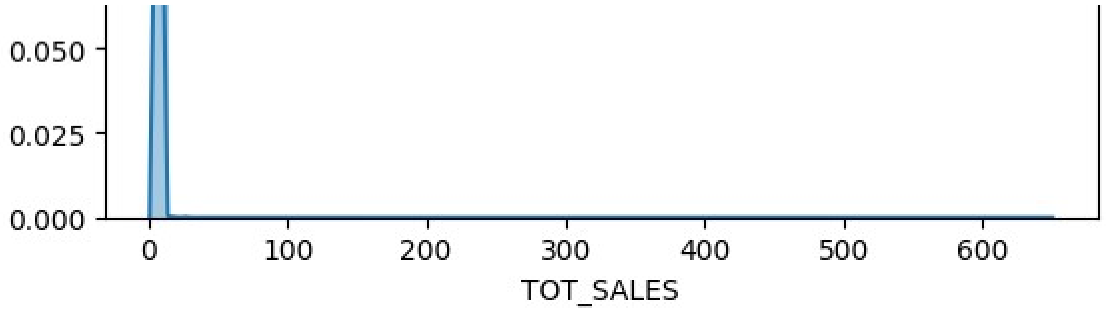
```
<Axes: ylabel='TOT_SALES'>
```



```
sns.distplot(chips.TOT_SALES,kde=True)
```

```
<ipython-input-35-a25589c44840>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(chips.TOT_SALES,kde=True)
<Axes: xlabel='TOT_SALES', ylabel='Density'>
```

Density
0.200
0.175
0.150
0.125
0.100
0.075



```
ndata=chips.select_dtypes(['float','int'])
```

```
ndata
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
0	43605	1	1343	383	61	2	2.9
1	43329	2	2373	974	69	5	15.0
2	43330	2	2426	1038	108	3	13.8
3	43604	4	4074	2982	57	1	5.1
4	43601	4	4149	3333	16	1	5.7
...	...	...	...	...	...	...	...
264829	43533	272	272319	270088	89	2	10.8
264830	43325	272	272358	270154	74	1	4.4
264831	43410	272	272379	270187	51	2	8.8
264832	43461	272	272379	270188	42	2	7.8
264833	43365	272	272380	270189	74	2	8.8

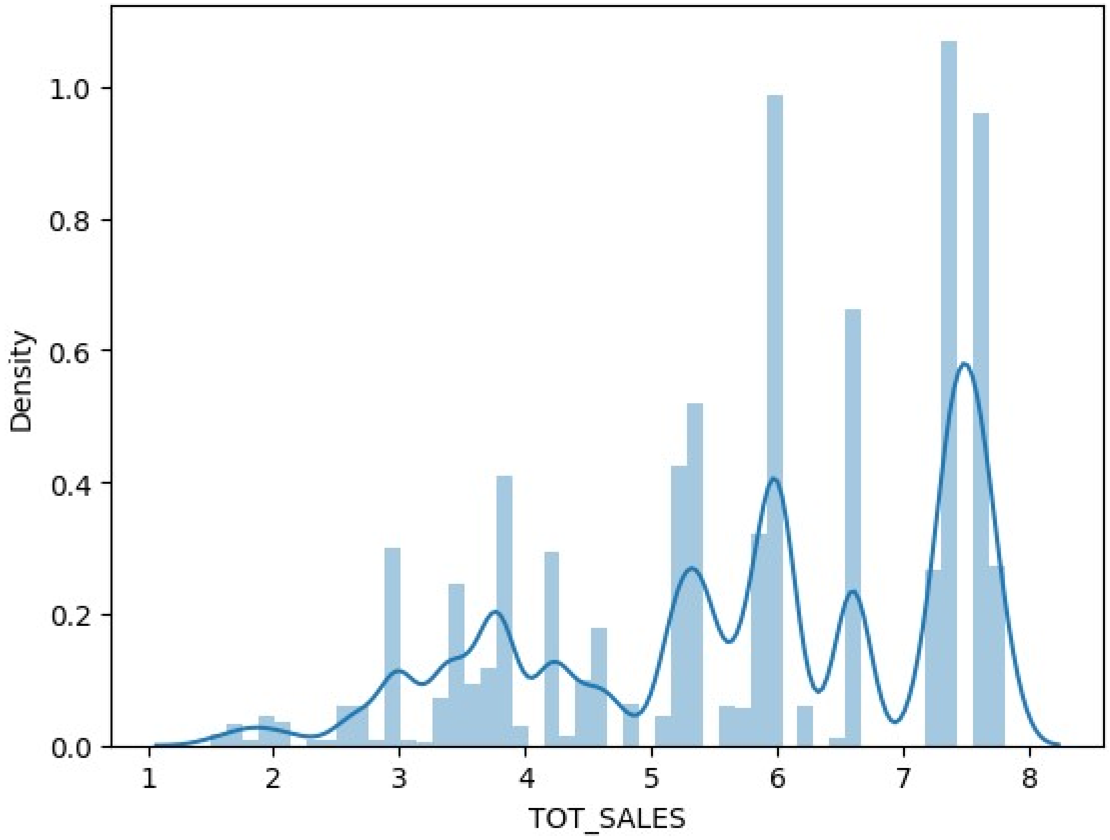
264834 rows × 7 columns

```
fil=ndata[ndata['TOT_SALES']<8.00]
```

```
sns.distplot(fil.TOT_SALES,kde=True)
```

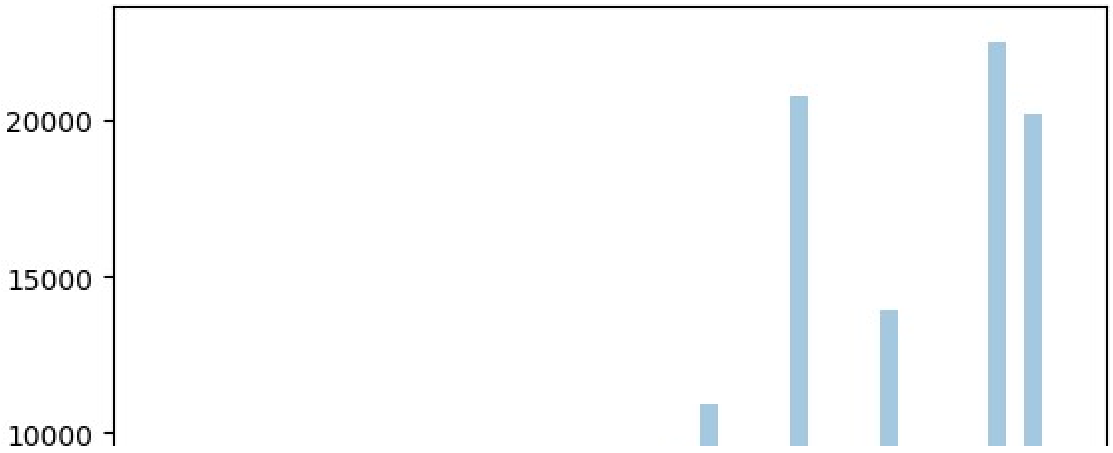
```
<ipython-input-39-3e087f3fc1c0>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

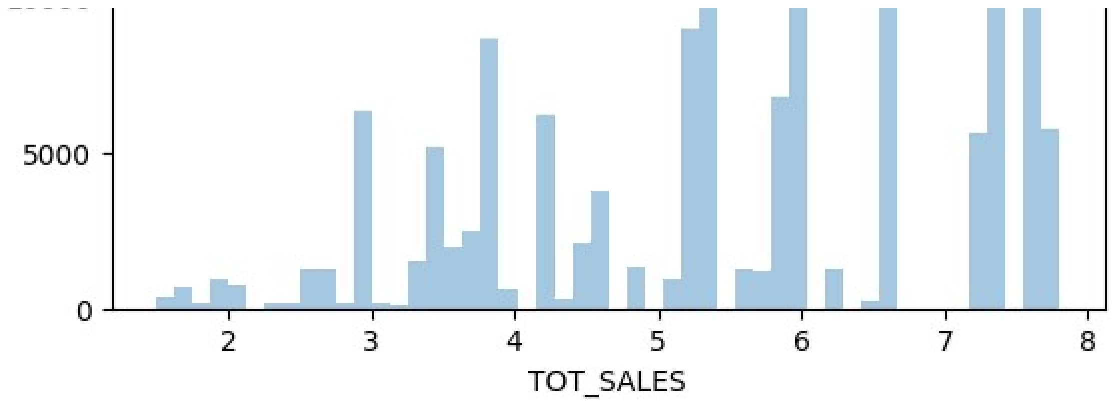
```
  sns.distplot(fil.TOT_SALES,kde=True)
<Axes: xlabel='TOT_SALES', ylabel='Density'>
```



```
sns.distplot(fil.TOT_SALES,kde=False)
```

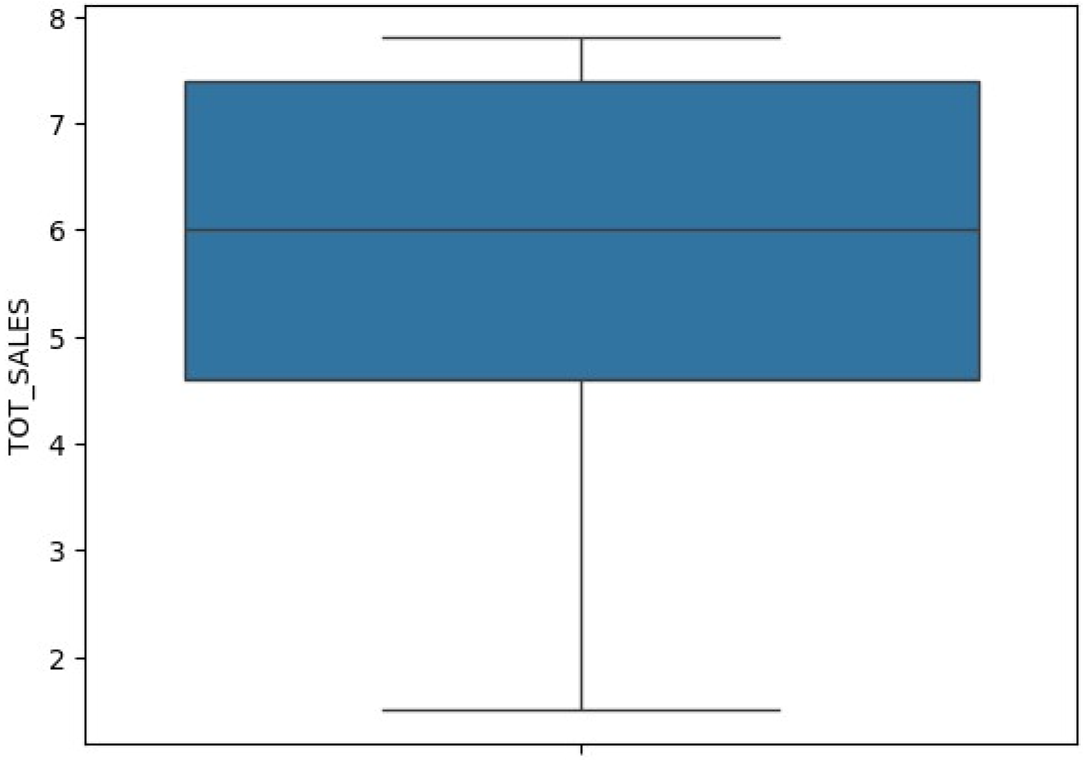
```
<ipython-input-40-208400af1635>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(fil.TOT_SALES,kde=False)
<Axes: xlabel='TOT_SALES'>
```





```
sns.boxplot(fil.TOT_SALES)
```

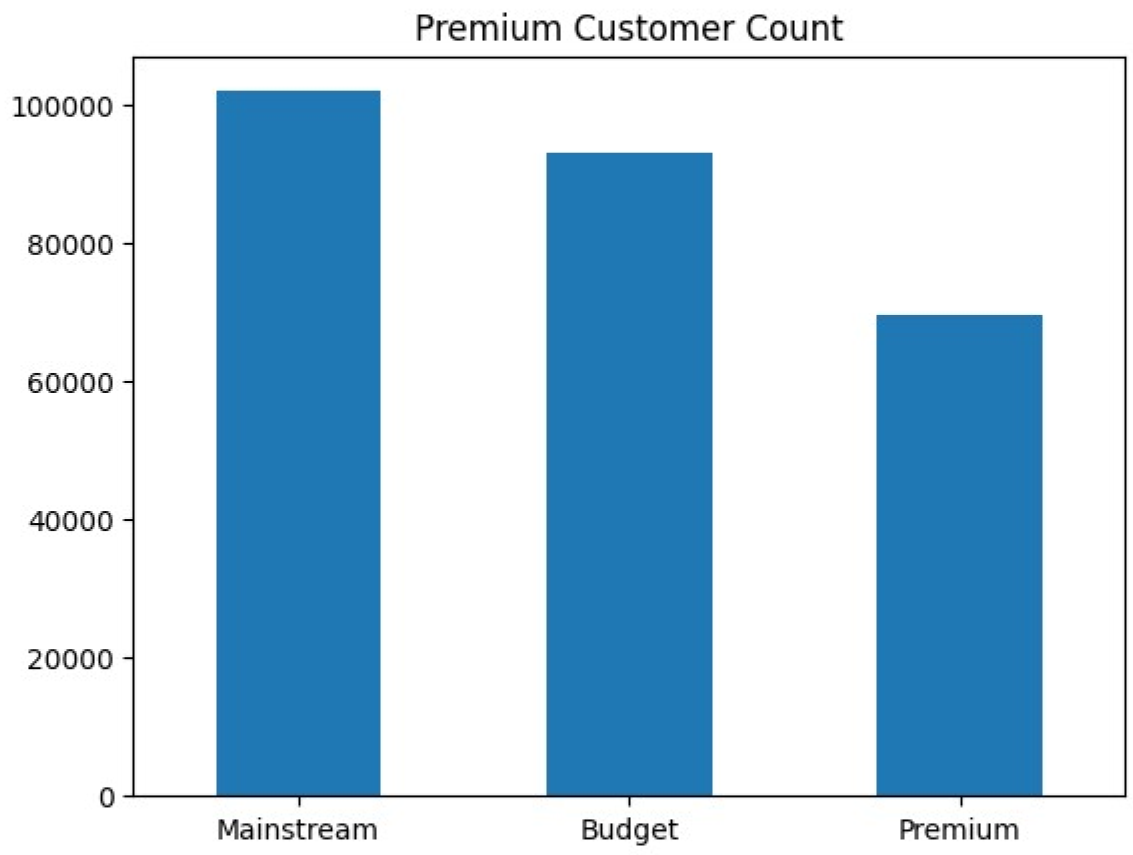
```
<Axes: ylabel='TOT_SALES'>
```



Data set is ready

Gathering some insights

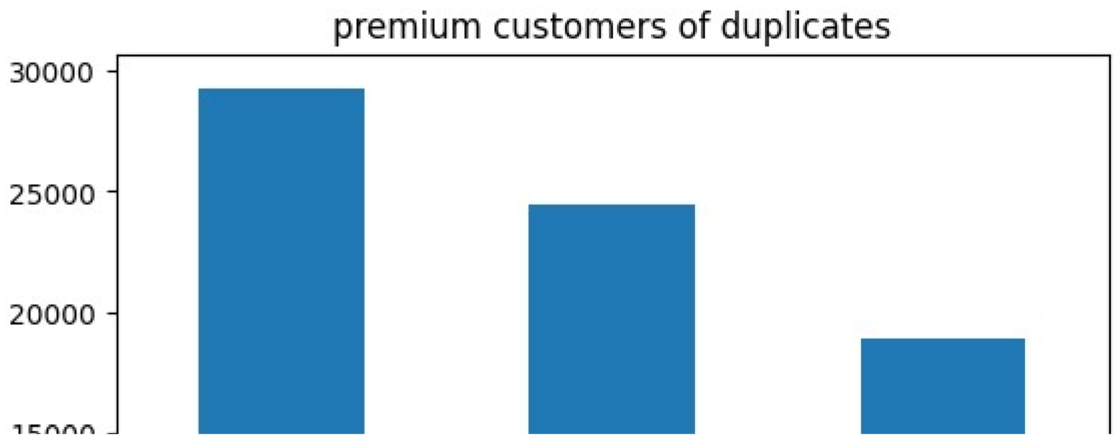
```
#premium coustomers
pcvc=chips['PREMIUM_CUSTOMER_x'].value_counts()
pcvc.plot(kind='bar')
plt.xticks(rotation=360)
plt.title('Premium Customer Count')
plt.show()
```

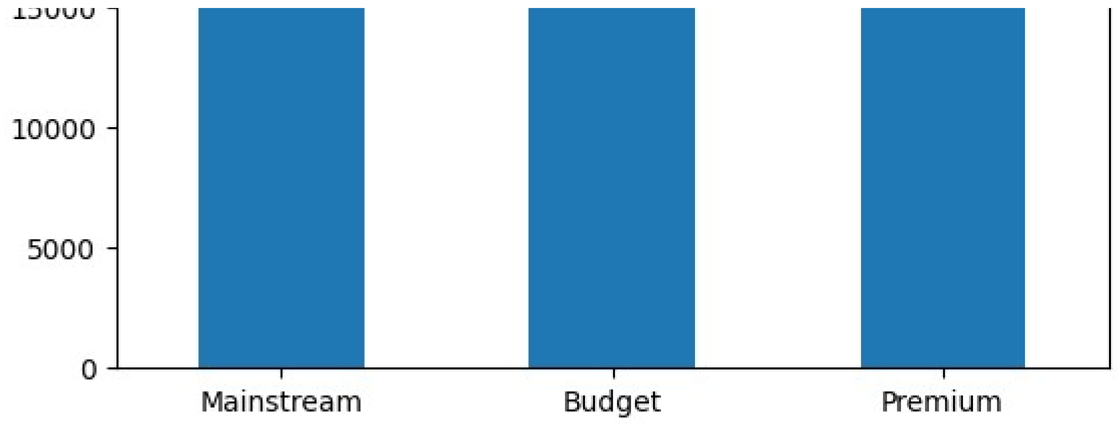


```
#unique members in premium customer type
unique=chips.drop_duplicates(subset='LYLTY_CARD_NBR')
unique['PREMIUM_CUSTOMER_x'].value_counts()
```

```
Mainstream    29245
Budget        24470
Premium       18921
Name: PREMIUM_CUSTOMER_x, dtype: int64
```

```
unpc=unique['PREMIUM_CUSTOMER_x'].value_counts()
unpc.plot(kind='bar')
plt.xticks(rotation=360)
plt.title('premium customers of duplicates')
plt.show()
```

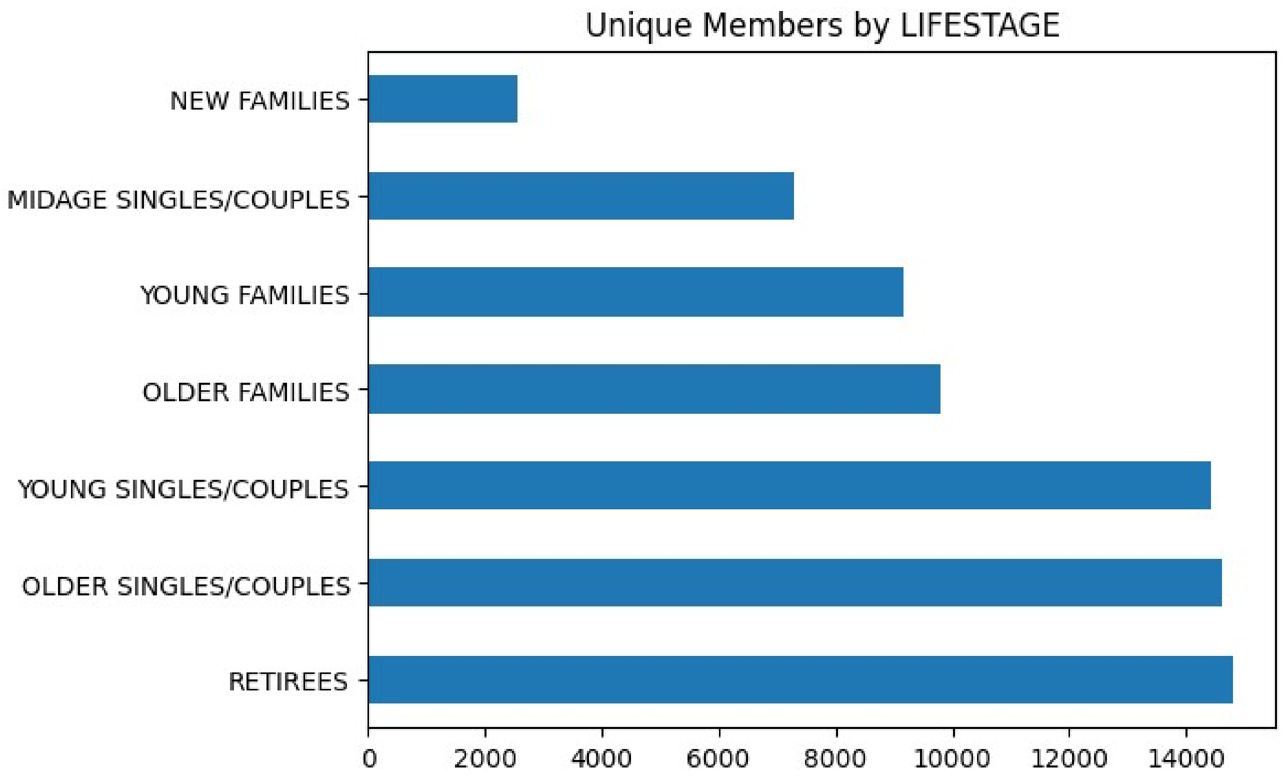




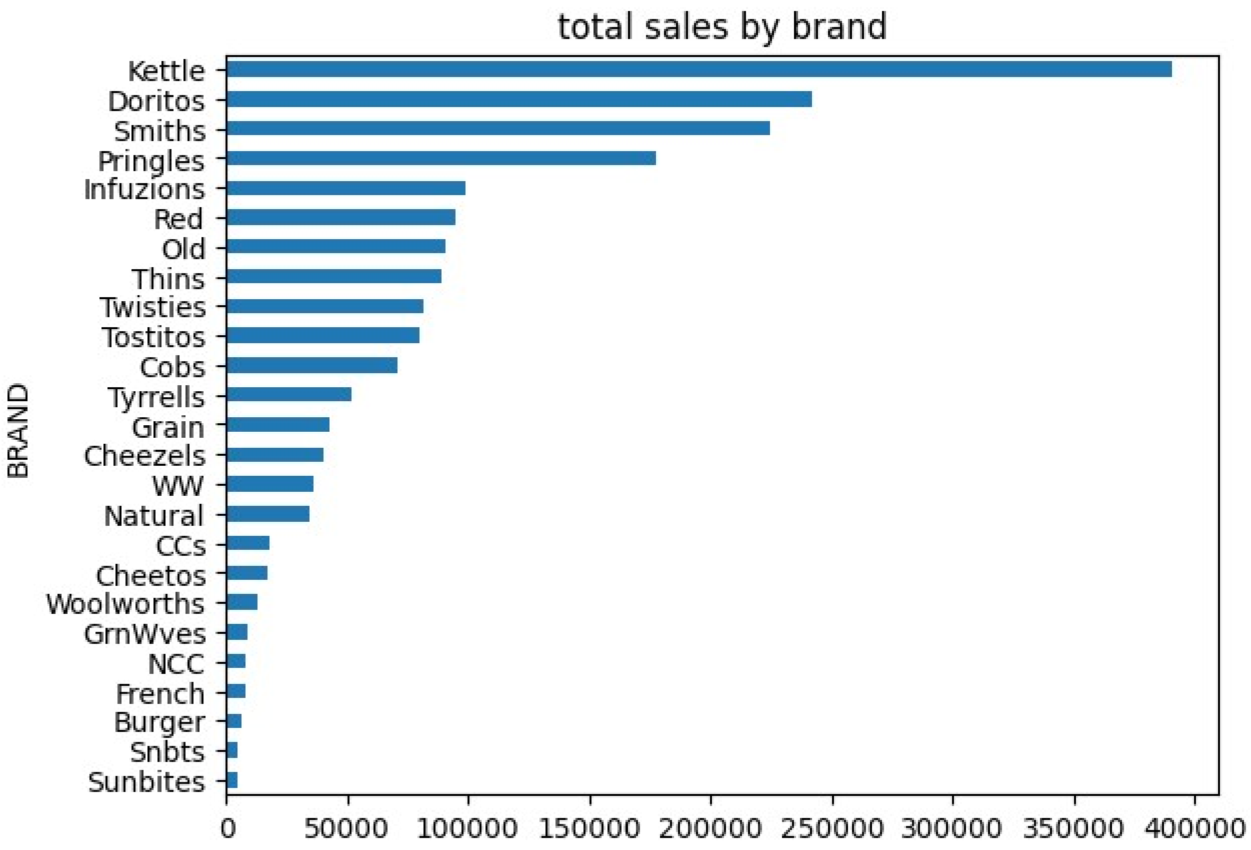
```
#lifestage of customers
unique['LIFESTAGE_x'].value_counts()
```

```
RETIREES                  14805
OLDER SINGLES/COUPLES     14609
YOUNG SINGLES/COUPLES     14440
OLDER FAMILIES             9780
YOUNG FAMILIES             9178
MIDAGE SINGLES/COUPLES     7275
NEW FAMILIES               2549
Name: LIFESTAGE_x, dtype: int64
```

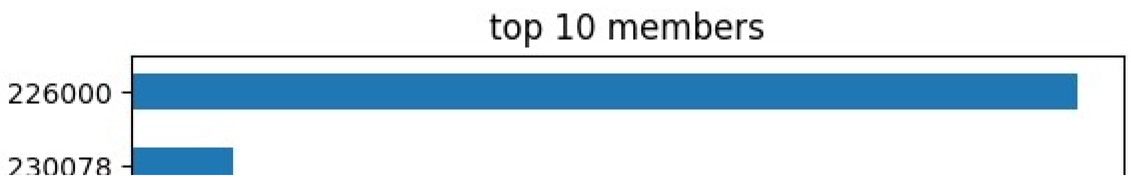
```
unls=unique['LIFESTAGE_x'].value_counts()
unls.plot(kind='barh')
plt.xticks(rotation=360)
plt.title('Unique Members by LIFESTAGE')
plt.show()
```

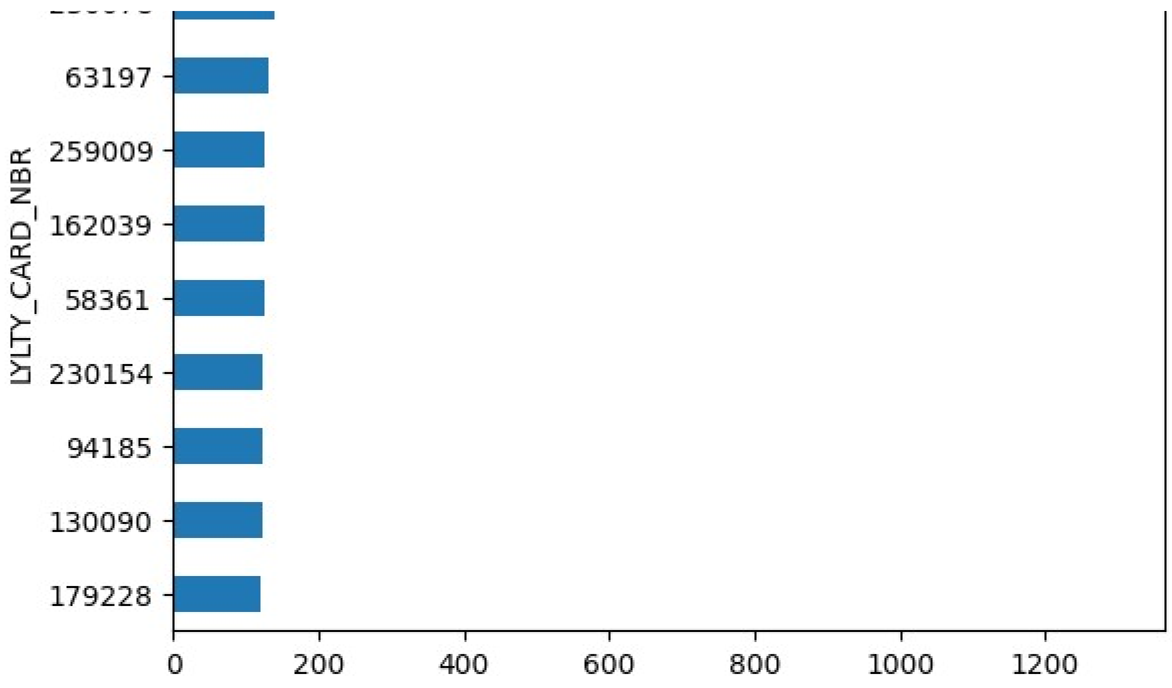


```
#gruping by brand
chips_br=chips.groupby('BRAND')

#total sale for each brand
chips_salesbrand=chips_br['TOT_SALES'].sum()

chips_salesbrand.sort_values().plot(kind='barh')
plt.title('total sales by brand')
plt.show()
```



```
chips_mbr=chips.groupby('LYLTY_CARD_NBR')
chips_salesmbr=chips_mbr['TOT_SALES'].sum()

chips_sorted=chips_salesmbr.sort_values()
chips_sorted.tail(10).plot(kind='barh')
plt.title('top 10 members')
plt.show()
```





```
top_ten=[226000,230078,63197,259009,162039,58361,230154,94185,130090,179228]
top_tenmbr=chips[chips['LYLTY_CARD_NBR'].isin(top_ten)]
top_tengrp=top_tenmbr.groupby('LYLTY_CARD_NBR')
top_tengrp['LIFESTAGE_x'].value_counts()
```

```
LYLTY_CARD_NBR  LIFESTAGE_x
58361           YOUNG FAMILIES            14
63197           OLDER FAMILIES            15
94185           YOUNG FAMILIES            16
130090          YOUNG FAMILIES            14
162039          OLDER FAMILIES            18
179228          YOUNG FAMILIES            16
226000          OLDER FAMILIES             2
230078          OLDER FAMILIES            17
230154          OLDER FAMILIES            14
259009          OLDER SINGLES/COUPLES     15
Name: LIFESTAGE_x, dtype: int64
```

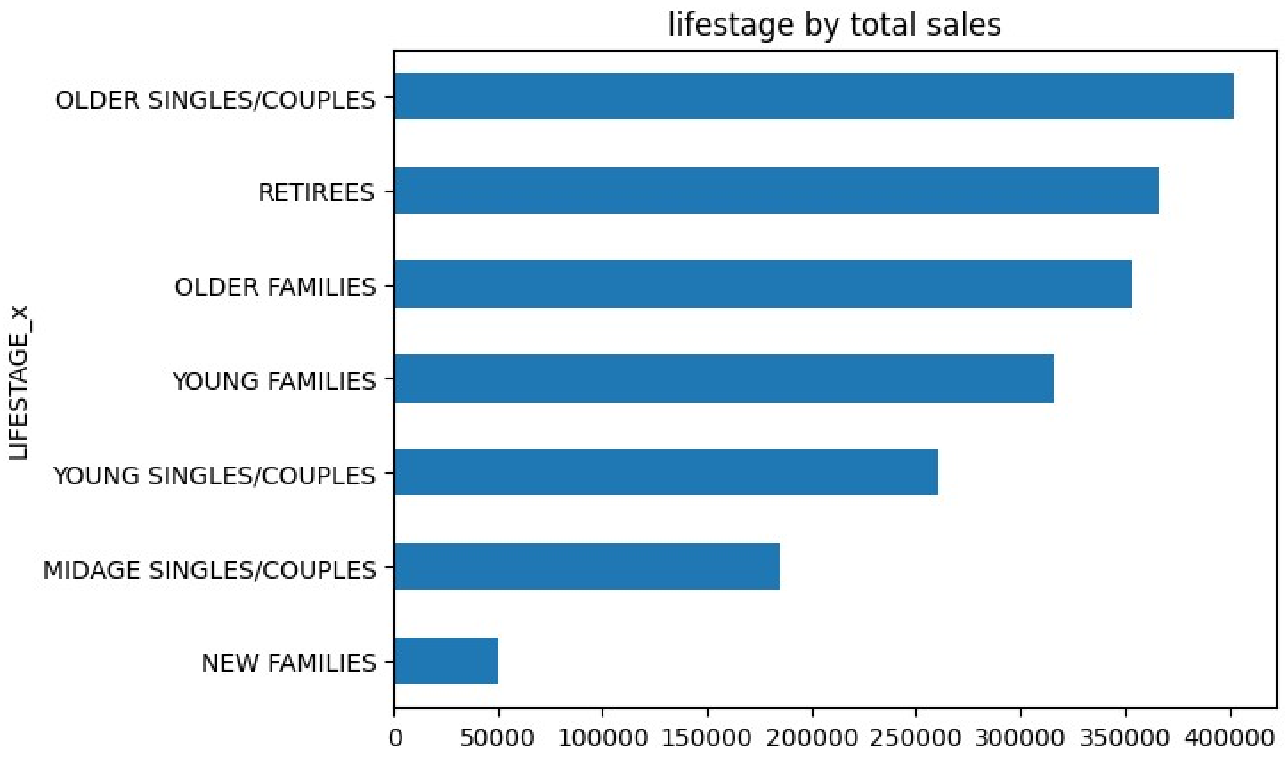
```
top_tengrp['PREMIUM_CUSTOMER_x'].value_counts()
```

```
LYLTY_CARD_NBR  PREMIUM_CUSTOMER_x
58361           Budget                14
63197           Budget                15
94185           Premium               16
130090          Budget                14
162039          Mainstream            18
179228          Budget                16
226000          Premium                2
230078          Budget                17
230154          Budget                14
259009          Mainstream            15
Name: PREMIUM_CUSTOMER_x, dtype: int64
```

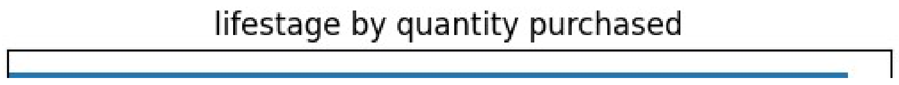
```
chips_sorted.describe()
```

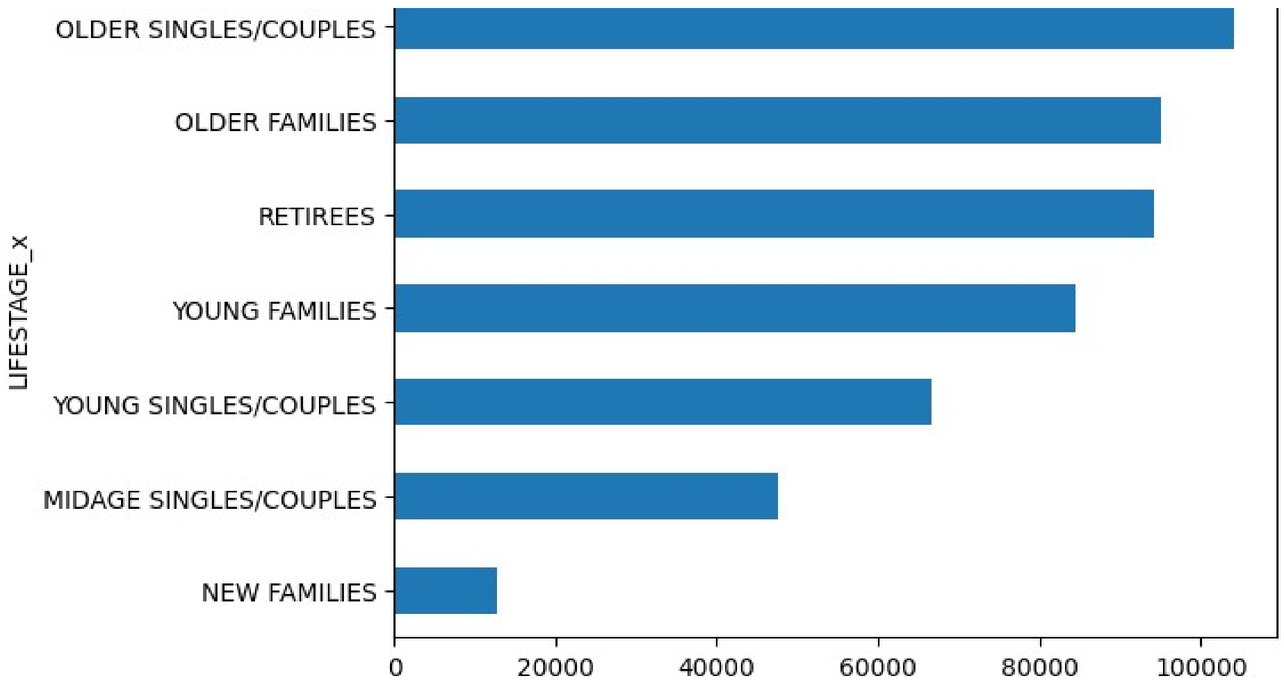
```
count    72636.00000
mean        26.63146
std         20.81440
min          1.50000
25%          9.10000
50%         21.70000
75%         40.00000
max       1300.00000
Name: TOT_SALES, dtype: float64
```

```
#grouping lifestage and finding total sales
chips_ls=chips.groupby('LIFESTAGE_x')
chips_ls_sales=chips_ls['TOT_SALES'].sum()

chips_ls_sales.sort_values().plot(kind='barh')
plt.title('lifestage by total sales')
plt.show()
```

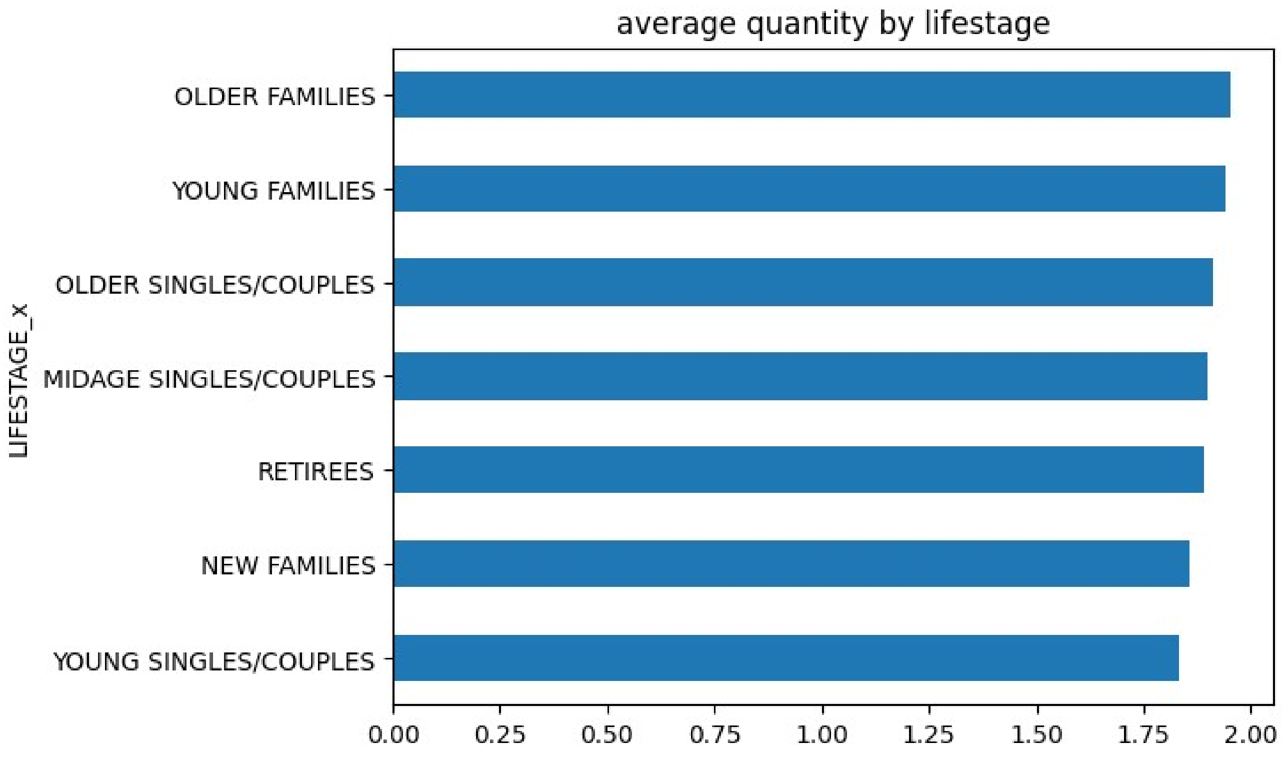


```
chips_ls_qty=chips_ls['PROD_QTY'].sum()
chips_ls_qty.sort_values().plot(kind='barh')
plt.title('lifestage by quantity purchased')
plt.show()
```

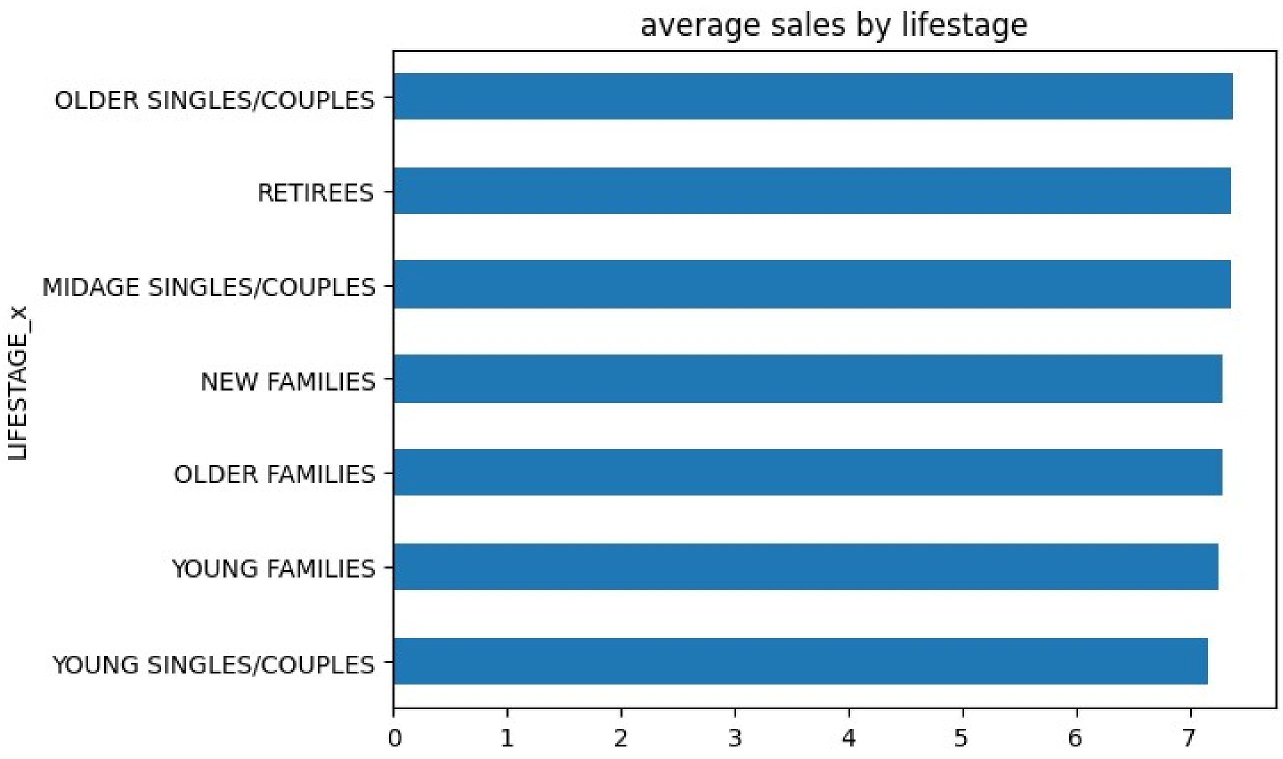




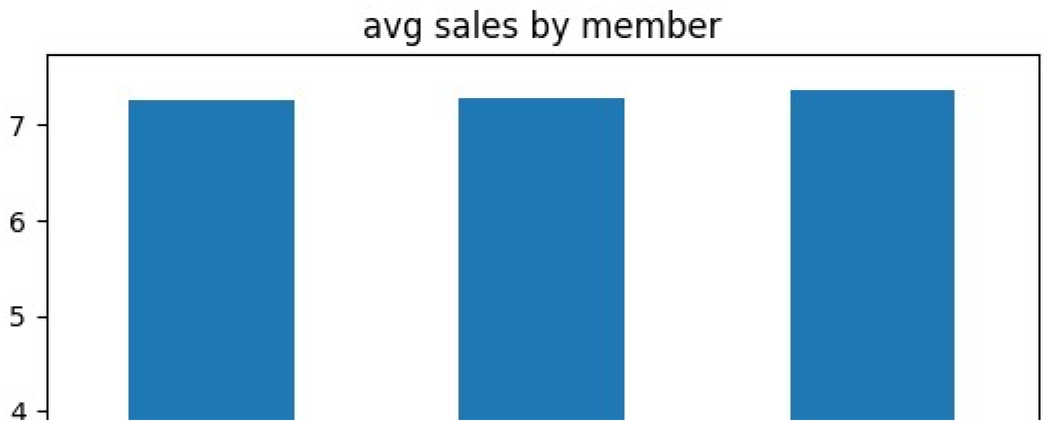
```
#average quantity by lifestage
chips_ls_avg_qty=chips_ls['PROD_QTY'].mean()
chips_ls_avg_sales=chips_ls['TOT_SALES'].mean()
chips_ls_avg_qty.sort_values().plot(kind='barh')
plt.title('average quantity by lifestage')
plt.show()
```

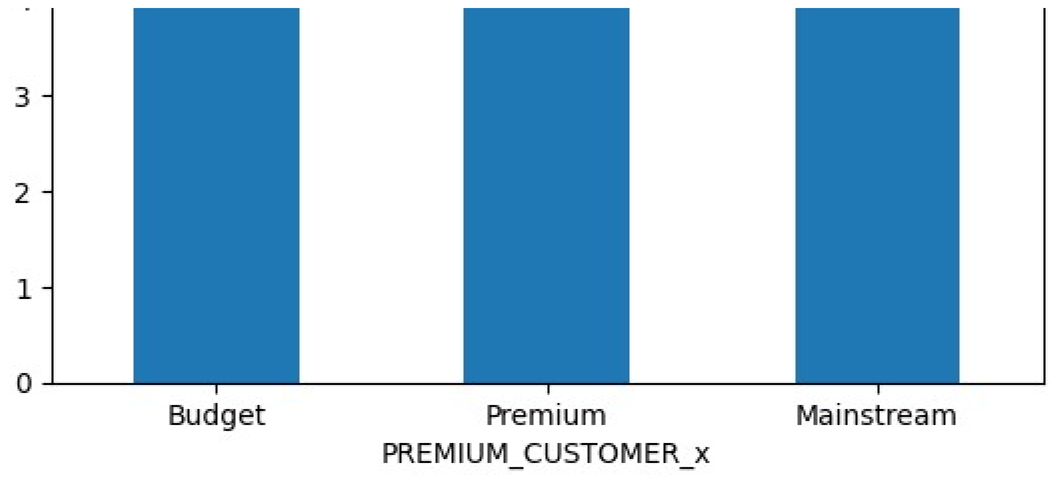


```
chips_ls_avg_sales.sort_values().plot(kind='barh')
plt.title('average sales by lifestage')
plt.show()
```



```
#membership
chips_pt=chips.groupby('PREMIUM_CUSTOMER_x')
chips_pt_avg_qty=chips_pt['PROD_QTY'].mean()
chips_pt_avg_sales=chips_pt['TOT_SALES'].mean()
chips_pt_avg_sales.sort_values().plot(kind='bar')
plt.xticks(rotation=360)
plt.title('avg sales by member')
plt.show()
```

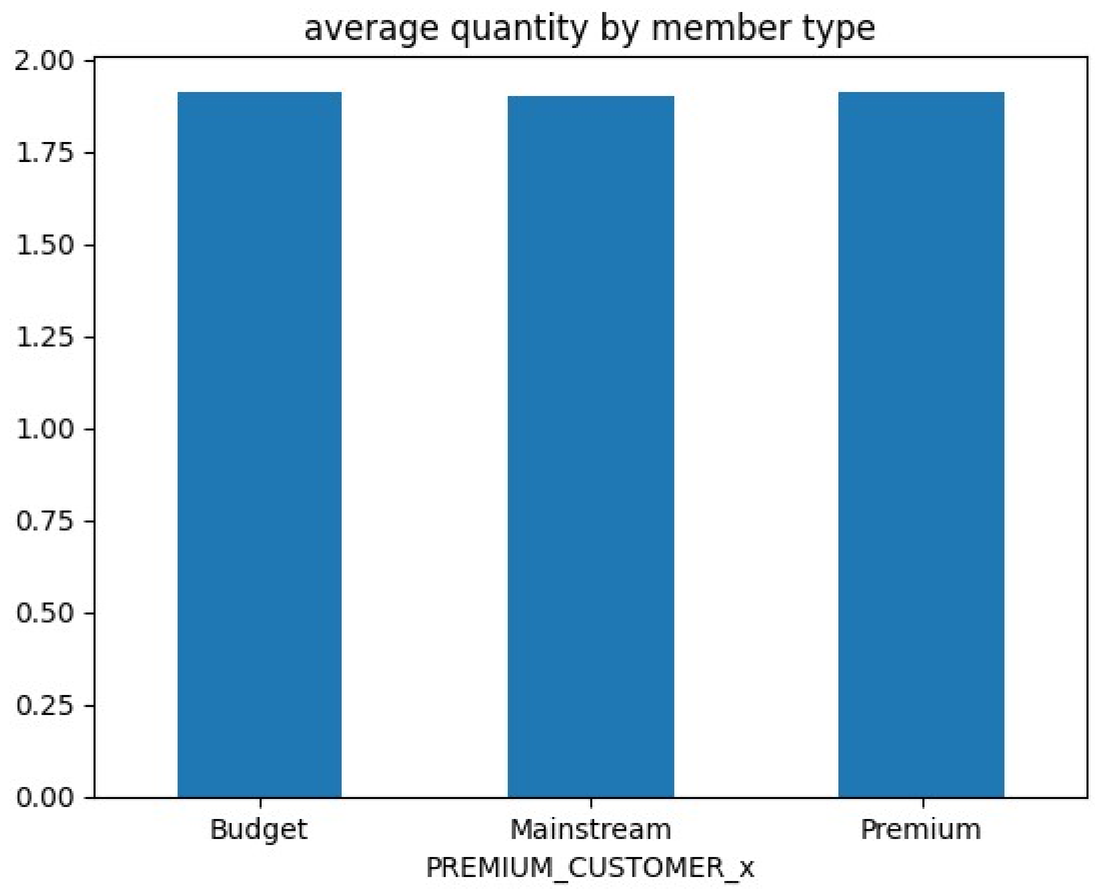




```
chips_pt_avg_sales.round(3)
```

```
PREMIUM_CUSTOMER_x
Budget        7.259
Mainstream    7.361
Premium       7.282
Name: TOT_SALES, dtype: float64
```

```
chips_pt_avg_qty.sort_index().plot(kind='bar')
plt.xticks(rotation=360)
plt.title('average quantity by member type')
plt.show()
```



```
chips_pt_avg_qty.round(3)
```

```
PREMIUM_CUSTOMER_x
Budget        1.910
Mainstream    1.902
Premium       1.912
Name: PROD_QTY, dtype: float64
```

```
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
chips_pt['BRAND'].value_counts()
```

```
PREMIUM_CUSTOMER_x  BRAND
Budget              Kettle        14154
                    Smiths        11548
                    Doritos        9818
                    Pringles       8620
                    Red            6480
                    Thins          4931
                    Infuzions      4922
                    WW             3881
                    Cobs           3274
                    Tostitos       3236
                    Twisties       3229
                    Old            3203
                    Natural        2246
                    Tyrrells       2195
                    Grain          2114
                    CCs            1678
                    Cheezels       1626
                    Woolworths     1605
                    Cheetos        1051
                    Snbts           610
                    Burger          579
                    GrnWves         542
                    French          539
                    NCC             539
                    Sunbites        536
Mainstream          Kettle        16423
                    Smiths        11842
                    Doritos       11192
                    Pringles       9903
                    Red            6462
                    Infuzions      5550
                    Thins          5436
                    Cobs           3889
                    Twisties       3785
                    Tostitos       3737
                    Old            3725
                    WW             3586
                    Tyrrells       2583
                    Grain          2516
                    Natural        2162
                    Cheezels       1735
                    CCs            1631
                    Woolworths     1607
                    Cheetos        1111
                    Burger          548
                    Snbts           544
                    GrnWves         521
                    French          507
                    Sunbites        498
                    NCC             495
Premium             Kettle        10711
                    Smiths         8433
                    Doritos        7137
                    Pringles       6579
                    Red            4837
                    Infuzions      3729
                    Thins          3708
```

```
customer_type_counts=chips['PREMIUM_CUSTOMER_x'].value_counts()
pivot_table=chips.pivot_table(index='PREMIUM_CUSTOMER_x',columns='BRAND',aggfunc='size',f
percentage_difference=(pivot_table/customer_type_counts[:,np.newaxis])*100
percentage_difference
```

```
<ipython-input-71-71a690a1bdd4>:3: FutureWarning: Support for multi-dimensional index
  percentage_difference=(pivot_table/customer_type_counts[:,np.newaxis])*100
```

BRAND	Burger	CCs	Cheetos	Cheezels	Cobs	Doritos	Frenc
PREMIUM_CUSTOMER_x							
Budget	0.567714	1.645292	1.030513	1.594305	3.210182	9.626623	0.52849
Mainstream	0.588261	1.750827	1.192623	1.862467	4.174718	12.014256	0.54424
Premium	0.627063	1.780743	1.097718	1.782178	3.630363	10.241068	0.53379

There is not much difference between lifestage and member, When it comes to average price and quantity purchased.

Deeper insights

```
pd.reset_option('display.max_rows')
pd.reset_option('display.max_columns')
import scipy.stats as stats
chips_ls.describe()
```

```
              DATE
              count    mean         std         min       25%       50%       75%       max
```

LIFESTAGE_x								
MIDAGE SINGLES/ COUPLES	25109.0	43464.125851	105.823780	43282.0	43372.0	43465.0	43557.0	4364
NEW FAMILIES	6919.0	43466.116202	105.514786	43282.0	43375.0	43469.0	43557.0	4364
OLDER FAMILIES	48596.0	43463.975842	105.048459	43282.0	43374.0	43464.0	43554.0	4364
OLDER SINGLES/ COUPLES	54479.0	43463.347583	105.739590	43282.0	43372.0	43462.0	43555.0	4364
RETIREES	49763.0	43464.106726	105.027814	43282.0	43373.0	43463.0	43555.0	4364
YOUNG FAMILIES	43592.0	43464.414204	105.431715	43282.0	43373.0	43465.0	43556.0	4364
YOUNG SINGLES/	36376.0	43464.139927	105.436989	43282.0	43373.0	43463.0	43554.0	4364

```
#grouping by lifestage
chips_youngfam=chips[chips['LIFESTAGE_x']=='YOUNG FAMILIES']
chips_young=chips[chips['LIFESTAGE_x']=='YOUNG SINGLES/COUPLES']

t_statistic,p_value=stats.ttest_ind(chips_youngfam['TOT_SALES'],chips_young['TOT_SALES'])
print('T STAT',t_statistic)
print('P VALUE',p_value)
```

```
T STAT 5.196463492534499
P VALUE 2.0361280135116597e-07
```

There is no difference between young families and young singles.

```
chips_prem=chips[chips['PREMIUM_CUSTOMER_x']=='premium']
chips_bud=chips[chips['PREMIUM_CUSTOMER_x']=='budget']
t_statistic,p_value=stats.ttest_ind(chips_prem['TOT_SALES'],chips_bud['TOT_SALES'])
print('T STAT',t_statistic)
print('P VALUE',p_value)
```

```
T STAT nan
P VALUE nan
```

This is also rejected.

```
chips['WEIGHT'].value_counts()
```

```
175g    66388
150g    43131
134g    25102
110g    22387
```

```
     170g    19983
     165g    15297
     300g    15166
     330g    12540
     380g     6418
     270g     6285
     210g     6272
     200g     4473
     135g     3257
     250g     3169
      90g     3008
     190g     2995
     160g     2970
     220g     1564
      70g     1507
     180g     1468
     125g     1454
     Name: WEIGHT, dtype: int64
```

```
chips['WEIGHT']=chips['WEIGHT'].astype(str)
```

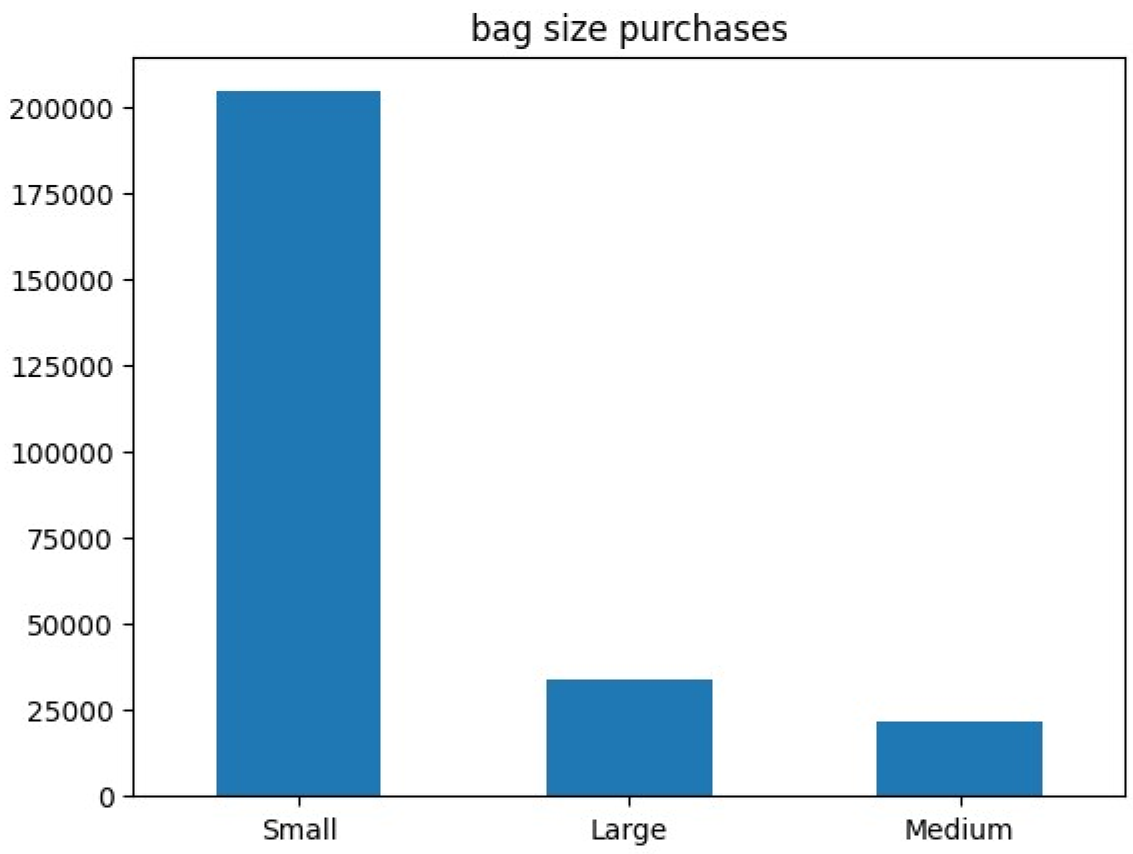
```
weight_category_map={
    '70g':'Small',
    '90g':'Small',
    '110g':'Small',
    '125g':'Small',
    '134g':'Small',
    '135g':'Small',
    '150g':'Small',
    '160g':'Small',
    '165g':'Small',
    '170g':'Small',
    '175g':'Small',
    '180g':'Small',
    '190g':'Small',
    '200g':'Medium',
    '210g':'Medium',
    '220g':'Medium',
    '250g':'Medium',
    '270g':'Medium',
    '300g':'Large',
    '330g':'Large',
    '380g':'Large',
    'nan':np.nan
}
chips['BAG_SIZE']=chips['WEIGHT'].map(weight_category_map)
```

```
chips['BAG_SIZE'].value_counts()
```

```
     Small     204432
     Large      34124
```

```
Medium     21763
Name: BAG_SIZE, dtype: int64
```

```
chips_bs=chips['BAG_SIZE'].value_counts()
chips_bs.plot(kind='bar')
plt.xticks(rotation=360)
plt.title('bag size purchases')
plt.show()
```



```
chips
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
1	43329	2	2373	974	69	Smiths Chip Thinly S/ Cream&Onion 175g	5
2	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3
3	43604	4	4074	2982	57	Old El Paso Salsa Dip	1

						Tomato Mild 300g	
4	43601	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g	1
...	...	...	...	...	...	...	...
264829	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2
264830	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	1
264831	43410	272	272379	270187	51	Doritos Mexicana 170g	2
264832	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2
264833	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	2

264834 rows × 15 columns

Checking data formats

```
#checking data types for transaction data
chips.dtypes
```

```
DATE                    int64
STORE_NBR               int64
LYLTY_CARD_NBR          int64
TXN_ID                  int64
PROD_NBR                int64
PROD_NAME              object
PROD_QTY                int64
TOT_SALES             float64
WEIGHT                 object
BRAND                  object
LIFESTAGE_x            object
PREMIUM_CUSTOMER_x     object
LIFESTAGE_y            object
PREMIUM_CUSTOMER_y     object
BAG_SIZE               object
dtype: object
```

```
#checking data types for purchase behaviour
chips_beh.dtypes
```

```
chips_ben.dtypes
```

```
LYLTY_CARD_NBR       int64
LIFESTAGE           object
PREMIUM_CUSTOMER    object
dtype: object
```

Here there is no change in data formats, as every thing is clear.

```
chips.to_csv('chips.csv')
```

Conclusion:

- Largest coustomer type is the mainstream group.
- Largest membership group is the older population.
- Top 10 members spent over 120 dollars on chips within a year.
- Top 4 brands sold are: Doritos, Smiths, Pringles and Kettle.
- Older individuals purchased the most chips includes single individuals and families.
- New families purchased the least on chips.
- There doesn't appear to be any statistical difference with purchase prices with customer in either life stage or membership type.
- The most purchased sized chips were the small bags and then large bags and medium sized bags were sold the least.