In [142]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns
from scipy import stats
```

Quantium Data Analytics Virtual Experience Program

Part 1 Data Preparation and customer analytics

by Murong (Sophie) Cui

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

Table of Context

Introduction

I am part of Quantium's retail analytics team and have been approached by your client, the Category Manager for Chips, who wants to better understand the types of customers who purchase Chips and their purchasing behaviour within the region. The insights from my analysis will feed into the supermarket's strategic plan for the chip category in the next half year.

Examine transaction data – look for inconsistencies, missing data across the data set, outliers, correctly identified category items, numeric data across all tables. Determining any anomalies make the necessary changes in the dataset and save it. Having clean data will help when it comes to the analysis.

Examine customer data – check for similar issues in the customer data, look for nulls and when I merge the transaction and customer data together so it's ready for the analysis.

Data analysis and customer segments – in the analysis make sure to define the metrics – look at total sales, drivers of sales, where the highest sales are coming from etc. Explore the data, create charts and graphs as well as noting any interesting trends and/or insights found. These will all form part of our report to Julia.

Deep dive into customer segments – define recommendation from the insights, determine which segments we should be targeting, if packet sizes are relative and form an overall conclusion based on the analysis.

Dataset

`transaction` dataframe is imported from `QVI_transaction_data.xlsx` : a year's worth of product transcations
feature:

- DATE: the days after the base date (1899-12-30)
- STORE_NBR: the store ID number
- LYLTY_CARD_NBR: loyalty card id number
- TXN_ID: transaction ID
- PROD_NBR: Product ID Number
- PROD_NAME: product name
- PROD_QTY: product quantity sold
- TOT_SALES: total sales

`customer` dataframe is imported from `QVI_purchase_behavious.csv` : customer segmentation
feature:

- LYLTY_CARD_NBR: loyalty card id number
- LIFESTAGE: customer attribute that identifies whether a customer has a family or not and what point in life they are at: RETIREES, OLDER SINGLES/COUPLES, YOUNG SINGLES/COUPLES, OLDER FAMILIES, YOUNG FAMILIES, MIDAGE SINGLES/COUPLES, NEW FAMILIES.
- PREMIUM_CUSTOMER: Customer segmentation used to differentiate shoppers by the price point of products they buy and the types of products they buy. It is used too identify whether customers may spend more for quality or brand or wehter they will purchase the cheapest options. Budget, Mainstream, Premium

In [143]:
```
# import transaction data
transaction= pd.read_excel('data/QVI_transaction_data.xlsx')
```

In [144]:
```
# data summary
print('df shape:', transaction.shape)
print('missing values:\n', transaction.isnull().sum())
print('description:\n',transaction.describe())
transaction.head()
```

```
df shape: (264836, 8)
missing values:
 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
description:
                 DATE      STORE_NBR  LYLTY_CARD_NBR        TXN_ID  \
count  264836.000000  264836.00000    2.648360e+05  2.648360e+05
mean    43464.036260     135.08011    1.355495e+05  1.351583e+05
std       105.389282      76.78418    8.057998e+04  7.813303e+04
min     43282.000000       1.00000    1.000000e+03  1.000000e+00
25%     43373.000000      70.00000    7.002100e+04  6.760150e+04
50%     43464.000000     130.00000    1.303575e+05  1.351375e+05
75%     43555.000000     203.00000    2.030942e+05  2.027012e+05
max     43646.000000     272.00000    2.373711e+06  2.415841e+06

            PROD_NBR       PROD_QTY      TOT_SALES
count  264836.000000  264836.000000  264836.000000
mean       56.583157       1.907309       7.304200
std        32.826638       0.643654       3.083226
min         1.000000       1.000000       1.500000
25%        28.000000       2.000000       5.400000
50%        56.000000       2.000000       7.400000
75%        85.000000       2.000000       9.200000
max       114.000000     200.000000     650.000000
```

Out[144]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT
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 [145]:
```
# import purchase behaviour customer data
customer = pd.read_csv('data/QVI_purchase_behaviour.csv')
```

In [146]:
```
# data summary
print(customer.LIFESTAGE.value_counts())
print(customer.PREMIUM_CUSTOMER.value_counts())
print(customer.isnull().sum())
customer.head()
```

```
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
Mainstream    29245
Budget        24470
Premium       18922
Name: PREMIUM_CUSTOMER, dtype: int64
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

Out[146]:

	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

Data Wrangling

- DATE

> - Converting the days from base date to date

- PROD_NBR, PROD_NAME

> - There are total 114 products sold. Some of them are salsa dipping sause. In order to find them out, the product table is extracted from the transaction table, containing 114 products (PROD_NBR, PROD_NAME). After investigating 114 products, there are 7 salsa products. Since we are inly interesting at the chips products, the records in transaction table only containing 7 salsa prodocts will be removed from the transaction table.
>   The 7 salsa products are (PROD_NBR, PROD_NAME):
>   35, Woolworths Mild Salsa 300g
>   76, Woolworths Medium Salsa 300g 57, Old El Paso Salsa Dip Tomato Mild 300g
>   59, Old El Paso Salsa Dip Tomato Med 300g
>   65, Old El Paso Salsa Dip Chnky Tom Ht300g
>   41, Doritos Salsa Mild 300g
>   101, Doritos Salsa Medium 300g
> - Extracting the brand name (BRAND) from PROD_NAME
> - Extracting the pack net weight (NET_WT) from PROD_NAME

- Outliers

> By the boxplots of PROD_QTY and TOT_SALES, there are clearly outliers on both features and look at the trasaction table. I saw the the two records where 200 packets of Doritos chips are bought in one transaction made by same customer. It looks like this customer has only had the two transactions over the year and is not an ordinary retail customer. The customer are premium customer from elder family. The customer might be buying chips for commericial purposes instead. I remove this loyalty card number in the customer table and the transaction made by this loyalty card number in the transaction table.

- Time Series

> These is one day missing was 2018-12-25 since all store close on Christmas day.
> Let's look at the number of transaction lines over time to see if there are any obvious data issues such as missing data. We see the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day.

- NET_WT:

> The largeest size is 380g and the smallest size is 70g. The distribution of pack size seems sensible

- BRAND:

> After cleaning the brand name, the barplot of brand seems reasonable. 41288 transactions contains Kettle brand chips, making Kettle the most popular chip brand.

- UNIT PRICE:

> Dividing TOT_SALES by PROD_QTY to get price by unit sold

In [147]:
```
##### Correct Date Data Type
# date datatype
transaction['DATE'] = pd.to_datetime(transaction['DATE'], unit = 'D', or
igin = pd.Timestamp('1899-12-30'))
```

In [148]:
```
##### Make product table and Exclude the salsa product
product = transaction[['PROD_NBR', 'PROD_NAME']].drop_duplicates()

# exclude the salsa product
mask = product['PROD_NBR'].isin([35, 76, 57, 59, 65, 41, 101])
product = product[~mask]
```

In [149]:
```
# extract net weight
product['NET_WT'] = product.PROD_NAME.str.extract('(\d+)').astype(int)
product['PROD_NAME'] = product.PROD_NAME.str.replace('(\d+)[gG]', '')
```

In [150]:
```
# correct erroneous brand name
product.loc[7, 'PROD_NAME'] = 'Sunbites Grain Waves Sweet Chilli'
product.loc[9, 'PROD_NAME'] = 'Sunbites Grain Waves Sour Cream&Chives'
product.loc[16, 'PROD_NAME'] = 'Smiths Burger Rings'
product.loc[56, 'PROD_NAME'] = 'Sunbites Grain Waves Plus Btroot & Chill
i Jam'

incorrect_brandname = np.array(['NCC', 'Natural Chip        Compny', 'Na
tural Chip Co', 'Natural ChipCo',
                       'Smith ', 'Infzns', 'Red Rock Deli', 'Snbts', 'Do
rito ', 'WW', 'French Fries', 'GrnWves'])
a = np.repeat('Natural_Chip_Company', 4)
correct_brandname = np.append(a, ['Smiths ', 'Infuzions', 'RRD', 'Sunbit
es', 'Doritos ', 'Woolworths', 'French_Fries', 'Grain Waves'])
brand_name_corrected = dict(zip(incorrect_brandname, correct_brandname))
# extract product name
product['PROD_NAME'] = product[['PROD_NAME']].replace(brand_name_correct
ed, regex=True)
product['BRAND'] = product['PROD_NAME'].str.split(' ', 1, expand = True)
[0]
```

In [151]:
```
# construct the product table
product = product[['PROD_NBR', 'NET_WT', 'BRAND']]
product.head()
```

Out[151]:

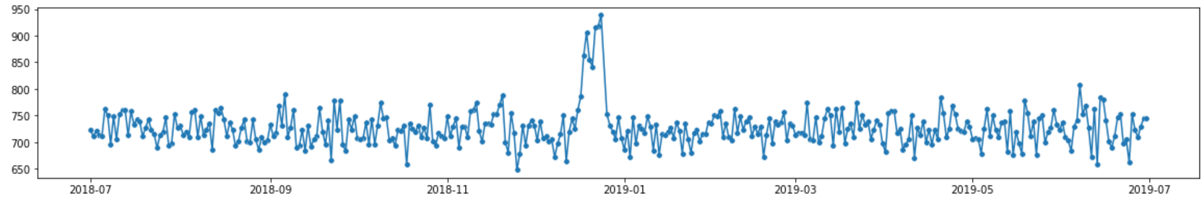
	PROD_NBR	NET_WT	BRAND
0	5	175	Natural_Chip_Company
1	66	175	CCs
2	61	170	Smiths
3	69	175	Smiths
4	108	150	Kettle

In [152]:
```
# join the product table back into the transaction table
# transaction.merge(product, how = 'inner', on = 'PROD_NBR')
```

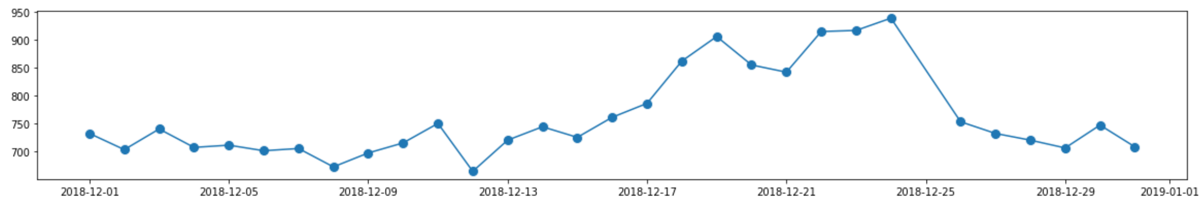
In [153]:
```
##### Missing one date
a = set(transaction.DATE.to_list())
b = set(pd.date_range(start=transaction.DATE.min(), end=transaction.DATE
.max(), freq='D').to_list())
print('one missiing day is ', b-a)
```

```
one missiing day is  {Timestamp('2018-12-25 00:00:00', freq='D')}
```

In [154]:
```
plt.figure(figsize = (20, 3))
sns.lineplot(x = transaction.DATE.value_counts().index ,
             y = transaction.DATE.value_counts().values);
sns.scatterplot(x = transaction.DATE.value_counts().index ,
             y = transaction.DATE.value_counts().values,
                s = 30);
```

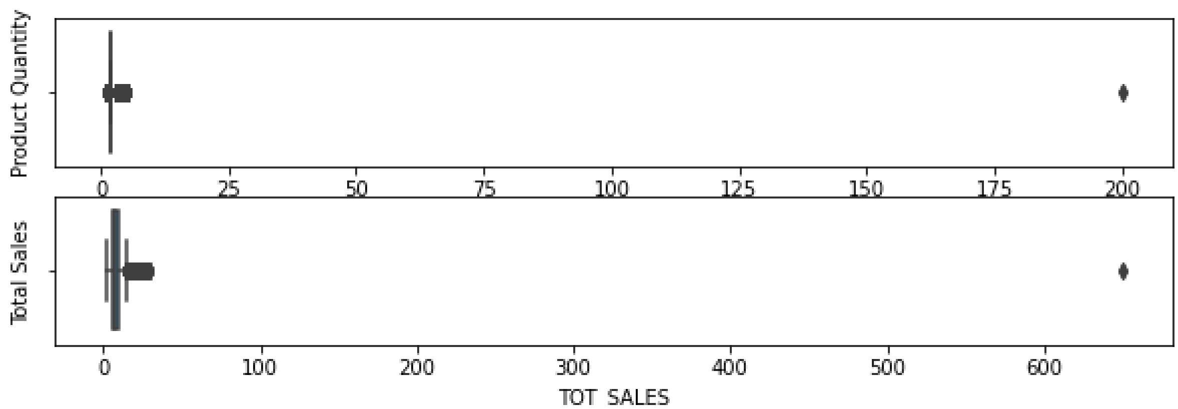


In [155]:
```
dec_tran = transaction[transaction.DATE.dt.month == 12]
plt.figure(figsize = (20, 3))
sns.lineplot(x = dec_tran.DATE.value_counts().index ,
             y = dec_tran.DATE.value_counts().values);
sns.scatterplot(x = dec_tran.DATE.value_counts().index ,
             y = dec_tran.DATE.value_counts().values,
                s = 100);
```



In [156]:
```
##### exclude outliers

plt.figure(figsize = (10, 3))

plt.subplot(2,1,1)
sns.boxplot(x = 'PROD_QTY', data = transaction);
plt.ylabel('Product Quantity')

plt.subplot(2,1,2)
sns.boxplot(x = 'TOT_SALES', data = transaction);
plt.ylabel('Total Sales')
#ax2 = sns.boxplot(x = 'TOT_SALES', data = transaction);
```

Out[156]:
```
Text(0, 0.5, 'Total Sales')
```



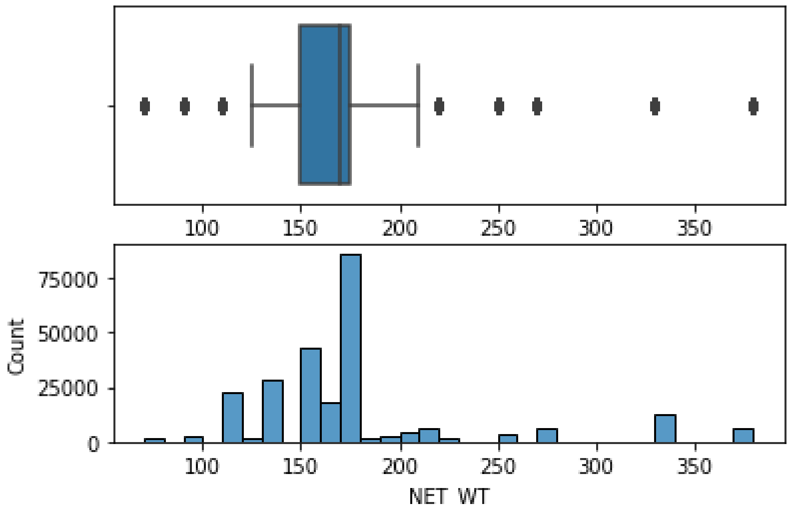
In [157]:
```
# drop LYLTY_CARD_NBR 22600 from both
transaction = transaction[transaction.LYLTY_CARD_NBR != 226000]
customer = customer[customer.LYLTY_CARD_NBR != 226000]
```

In [158]:
```
# remove multiple spaces in Prod_name
transaction['PROD_NAME'] = transaction.PROD_NAME.replace('\s+', ' ', reg
ex=True)
```

In [159]:
```
# merging product table to
transaction =  transaction.merge(product, how = 'inner', on = 'PROD_NBR'
)
```

In [160]:
```
# check pack size
print('Smallest Pack Size:', transaction.NET_WT.min(), 'Largest Pack Siz
e:', transaction.NET_WT.max())
plt.subplot(2,1,1)
sns.boxplot(x = transaction['NET_WT']);
plt.subplot(2,1,2)
sns.histplot(data = transaction, x = 'NET_WT', binwidth = 10);
```

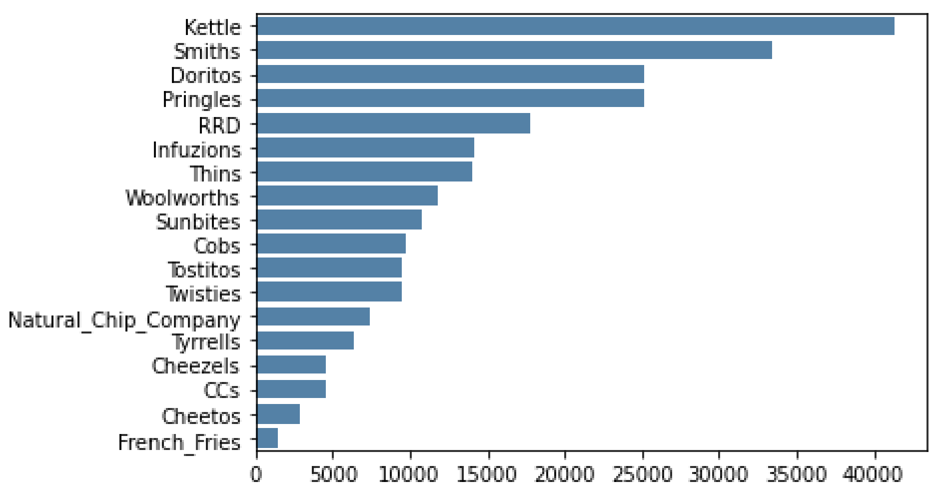
```
Smallest Pack Size: 70 Largest Pack Size: 380
```



In [161]:
```
# check brand
print(transaction.BRAND.value_counts())
sns.barplot(x = transaction.BRAND.value_counts().values, y = transaction
.BRAND.value_counts().index, color = 'steelblue')
```

```
Kettle                 41288
Smiths                 33387
Doritos                25224
Pringles               25102
RRD                    17779
Infuzions              14201
Thins                  14075
Woolworths             11836
Sunbites               10748
Cobs                    9693
Tostitos                9471
Twisties                9454
Natural_Chip_Company    7469
Tyrrells                6442
Cheezels                4603
CCs                     4551
Cheetos                 2927
French_Fries            1418
Name: BRAND, dtype: int64
```

Out[161]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb9641fad90>



In [162]:
```
# merge transaction and customer
transaction_customer = transaction.merge(customer, on='LYLTY_CARD_NBR',
how = 'left')
```

In [163]:
```
# feature engineering: unit_price, combine two cusomter segment: lifestage + premium customer
transaction_customer['UNIT_PRICE'] = transaction_customer['TOT_SALES'].astype('float')/transaction_customer['PROD_QTY'].astype('float')
transaction_customer['LIFESTAGE_PREMIUM'] = transaction_customer['LIFESTAGE']+ '-'+ transaction_customer['PREMIUM_CUSTOMER']


transaction_customer.head()
```

Out[163]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	
1	2018-12-05	5	5050	4667	5	Natural Chip Compny SeaSalt175g	2	
2	2018-08-04	16	16364	14497	5	Natural Chip Compny SeaSalt175g	1	
3	2018-07-18	35	35359	31902	5	Natural Chip Compny SeaSalt175g	1	
4	2019-05-06	39	39167	35645	5	Natural Chip Compny SeaSalt175g	2	

In [164]:
```
# save the dataset
# transaction_customer.to_csv('transaction_customer.csv', index = False, header = True)
```

Data Analysis and Customer Segmentation

Now that data is ready for analysis, we can define some metrics of interset to the client:

- Who spends the most on chips (total sales), describing customer by lifestage and how premium their general pruchasing 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

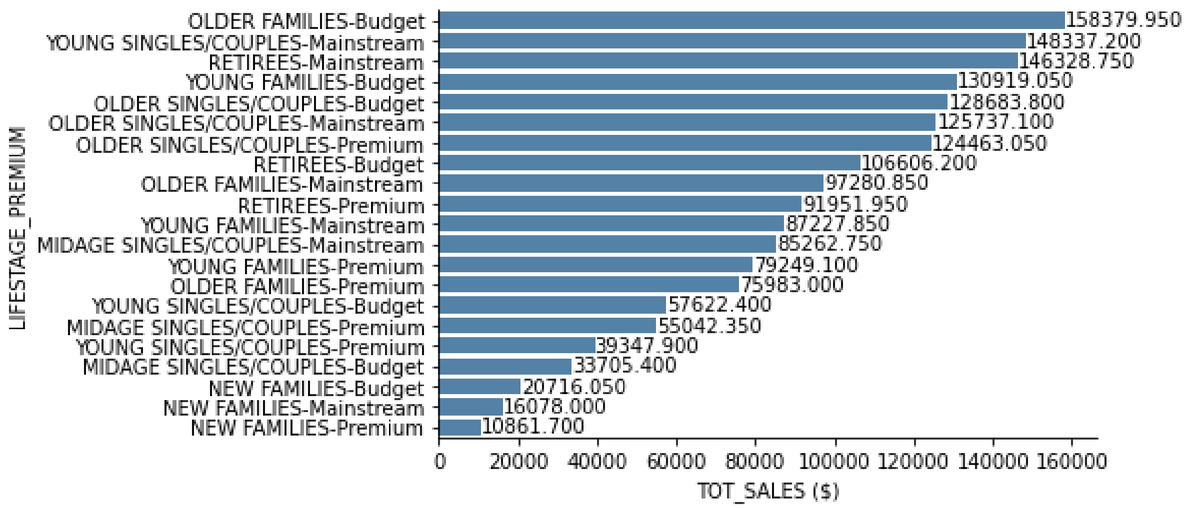
In [165]:
```
def groupby_df(group_feature, target_feature, agg_method, df = transacti
on_customer):
    '''
    df: transaction_customer (default),
    group_feature: df grouped by the group_feature,
    target_feature: target feature
    '''

    temp = df\
    .groupby(group_feature)[[target_feature]]\
    .agg(agg_method)\
    .sort_values(target_feature, ascending = False)\
    .reset_index()

    temp['LIFESTAGE_PREMIUM'] = temp['LIFESTAGE']+'-'+temp['PREMIUM_CUST
OMER']

    return temp



def groupby_df_self_agg(group_feature, target_feature, agg_method, df =
transaction_customer):
    '''
    df: transaction_customer (default),
    group_feature: df grouped by the group_feature,
    target_feature: target feature
    '''

    temp = df\
    .groupby(group_feature)[[target_feature]]\
    .agg(agg_method)\
    .reset_index()

    temp['sales_per_capita'] = temp[target_feature][agg_method[0]]/temp[
target_feature][agg_method[1]]

    temp.columns = ['_'.join(col).rstrip('_') for col in temp.columns.va
lues]

    temp['LIFESTAGE_PREMIUM'] = temp['LIFESTAGE']+'-'+temp['PREMIUM_CUST
OMER']

    temp = temp.sort_values('sales_per_capita', ascending = False).reset
_index()

    return temp



def plot_barplot(df, feature_target):
    sns.barplot(y = 'LIFESTAGE_PREMIUM', x=feature_target, data = df, co
lor = 'steelblue');
    for i in range(len(df)):
        count = df[feature_target][i]
        pct_string = '{0:.3f}'.format(count)
```

```
        plt.text(count + 0.1, i, pct_string, va='center')
    sns.despine();
    plt.xlabel('{} ($)'.format(feature_target))
    plt.show()
```

- Who spends the most on chips (total sales), describing customer by lifestage and how premium their general pruchasing behaviour is

Top total sales are coming from Order Family - Budget, Young Singles/Couples - Mainstream, Retirees - Mainstream

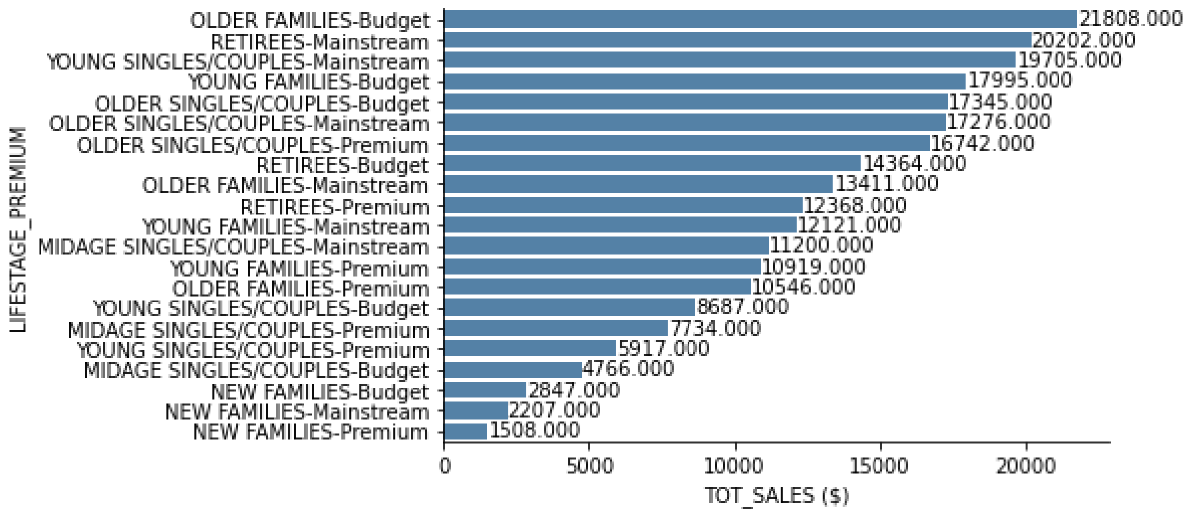
In [166]:
```
plot_barplot(groupby_df(['LIFESTAGE', 'PREMIUM_CUSTOMER'], 'TOT_SALES', 'sum'), 'TOT_SALES')
```



- How many customers are in each segment

Let's see if the higher sales are due to there being more customer who buy chips. Calculating that the number customers by lifestage and premium status.
There are more Retirees - Mainstream and Young Singles/Couples - Mainstream who buy chips. This contributes to there being more sales to these customer segments but this is not a major driver for the Olderr Families - Budget.

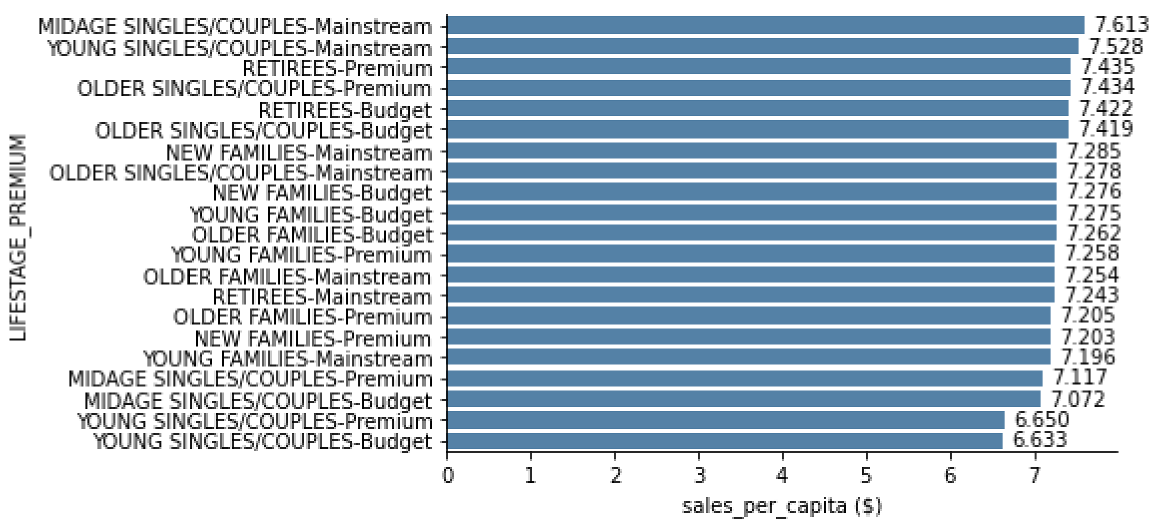
In [167]:
```
plot_barplot(groupby_df(['LIFESTAGE', 'PREMIUM_CUSTOMER'], 'TOT_SALES',
'count'), 'TOT_SALES')
```



- How the sales per capita are in each segment

We divide the total sales by the number of customer in each segment nd we got sales per capita in each segment. We look at the sales per capita in each segments. The midage Singles/Couples - Mainstream and Young Singles/Couples - Mainstream have highest customer value regarding sales per capita.

In [168]:
```
plot_barplot(groupby_df_self_agg(['LIFESTAGE', 'PREMIUM_CUSTOMER'], 'TOT
_SALES', ['sum', 'count']), 'sales_per_capita')
```

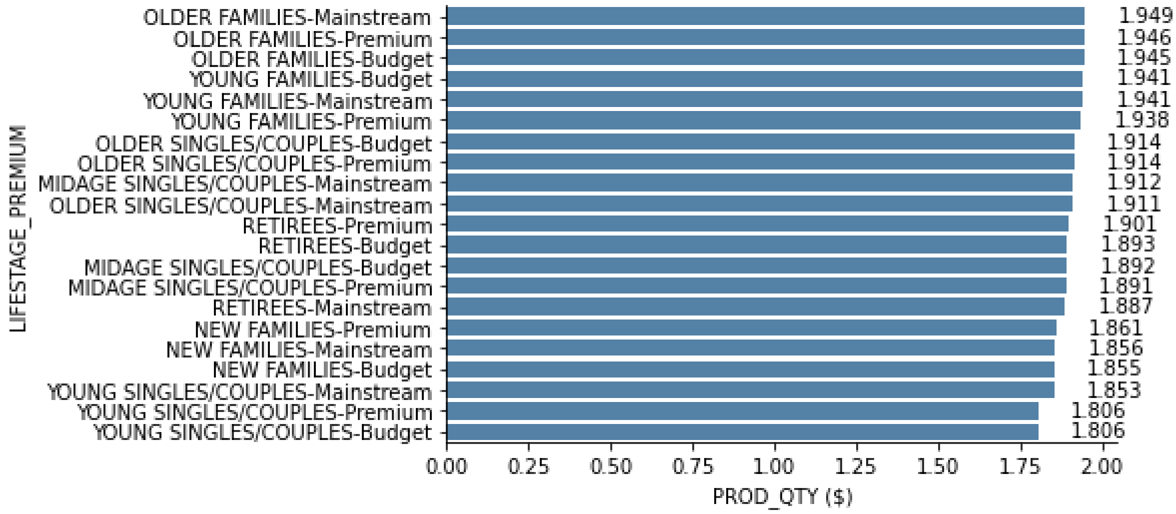


- How many chips are bought per customer by segment

Higher sales may also be driven by more units of chips being bought per customer. So we calculate the average number of units per customer by lifestage and premium status.

Older families and young families in general buy more per customer

In [169]:
```
plot_barplot(groupby_df(['LIFESTAGE', 'PREMIUM_CUSTOMER'], 'PROD_QTY', 'mean'), 'PROD_QTY')
```

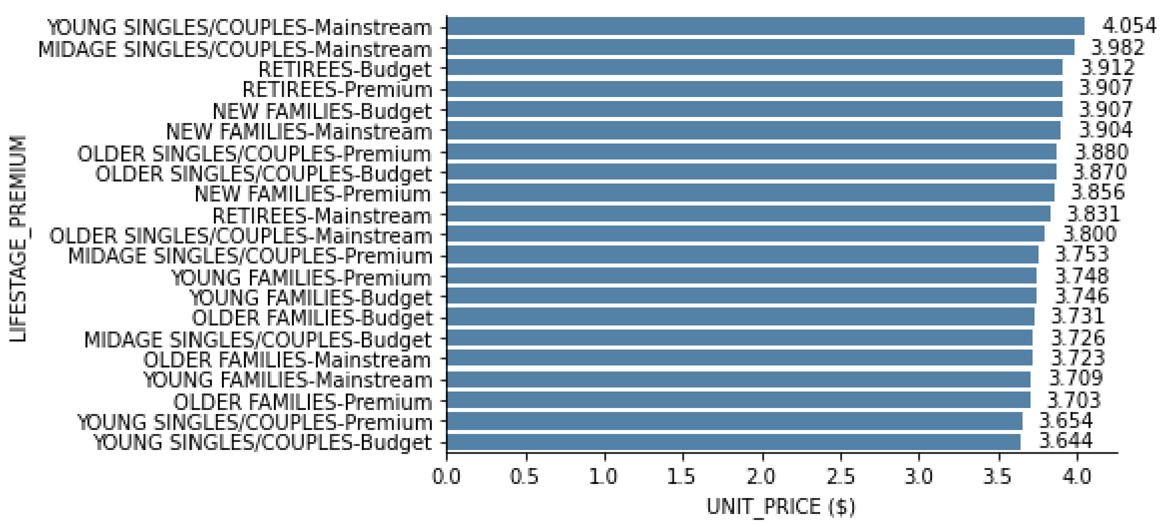


- What's the average chip price by customer segment

We also investigate the averrage price per unit chips bought for each customer segment as this is also a driver of total sales. We calculate average price per unit by lifestage and premium status.

Mainstream Midage and Young Singles/Couples are more willing to pay more per packet of chips compared to their budget and premium counterrparts. This may be due to premium shoppers being more likely to buy healthy snacks and when they buy chips, this is mainly for entertainment purposes rather than their own consuption. This is also supported by there mainstream counterparts.

In [170]:
```
plot_barplot(groupby_df(['LIFESTAGE', 'PREMIUM_CUSTOMER'], 'UNIT_PRICE', 'mean'), 'UNIT_PRICE')
```



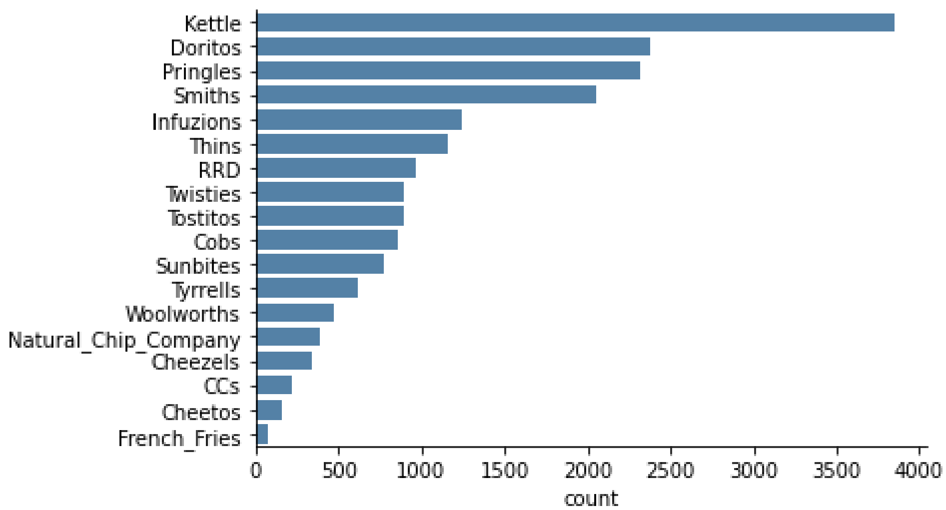
Target Customer Segment - Mainstream Young Singles/Couples

Above investigating metrics by customer segment, Our Target Customer Segment is Mainstream Young Singles/Couples.

Popular Brands for Targar Segment

The most popular brand for target segment is Kettle.

In [171]:
```
sns.barplot(x = target_segment.BRAND.value_counts().values, y = target_s
egment.BRAND.value_counts().index, color = 'steelblue');
sns.despine();
plt.xlabel('count');
```



Hypothesis Test for Average Unit Price

As the difference in average price per unit is not large, we can check if this difference is statistically different.

μ : average price per unit
group_A : mainstream midage and young singles/couples
group_B : premium and budget midage and young singles/couples

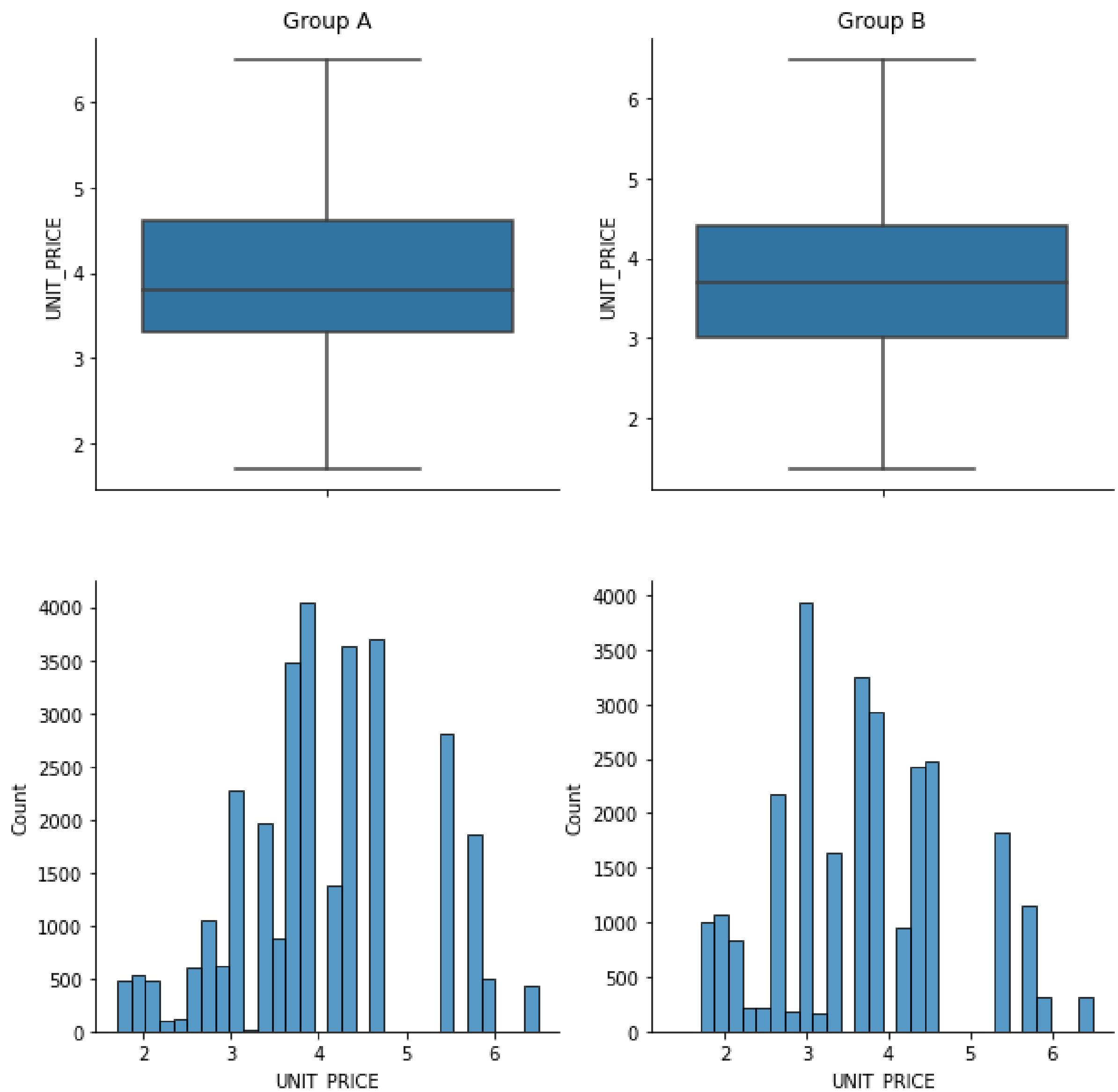
$$H_0 : \mu_{\text{group\_A}} \leq \mu_{\text{group\_B}}$$
$$H_a : \mu_{\text{group\_A}} > \mu_{\text{group\_B}}$$

We conduct t test comparing two average and telling us if they are different from each other. The t test also tells you how significant the differents are.

The t-test results in a p-value of 0.0, we could reject the hypothesis null $H_0 : \mu_{\text{group\_A}} \leq \mu_{\text{group\_B}}$. The unit price for mainstream, young and mid-age singles and coules are significant higher than that of budget or premium, young and midage singkes and couples

In [172]:
```
group_A = transaction_customer[(transaction_customer.PREMIUM_CUSTOMER == 'Mainstream') & (transaction_customer.LIFESTAGE.isin(['YOUNG SINGLES/COUPLES', 'MIDAGE SINGLES/COUPLES']))][['UNIT_PRICE']]
group_B = transaction_customer[(transaction_customer.PREMIUM_CUSTOMER.isin(['Budget', 'Premium'])) & transaction_customer.LIFESTAGE.isin(['YOUNG SINGLES/COUPLES', 'MIDAGE SINGLES/COUPLES'])][['UNIT_PRICE']]
```

In [173]:
```
# boxplot and histplot
plt.subplots(2,2, sharey=True, figsize = (10, 10));
plt.subplot(2,2,1);
sns.boxplot(y = group_A['UNIT_PRICE']);
plt.title('Group A')
plt.subplot(2,2,2);
sns.boxplot(y = group_B['UNIT_PRICE']);
plt.title('Group B')

plt.subplot(2,2,3);
sns.histplot(x = group_A['UNIT_PRICE'], bins = 30);

plt.subplot(2,2,4);
sns.histplot(x = group_B['UNIT_PRICE'], bins = 30);
sns.despine()
```



In [174]:
```
# calculate the Standard Deviation
# set alpha
alpha = 0.05
# calculate the variance to get standard deviation
# For unbiased max likelyhood estimate we have to divide the var by N-1
# and therefore the parameter ddof = 1
var_a = group_A['UNIT_PRICE'].var(ddof = 1)
var_b = group_B['UNIT_PRICE'].var(ddof = 1)
# calculate the means
m_a = group_A['UNIT_PRICE'].mean()
m_b = group_B['UNIT_PRICE'].mean()
# calculate the sample size
n_a = len(group_A)
n_b = len(group_B)
# t-statistic
t = (m_a - m_b)/np.sqrt(var_a/n_a + var_b/n_b)
# compare with critical values
# degree of freedom
df = n_a + n_b - 2
# p_value after comparison with the t
p = 1 - stats.t.cdf(t, df = df)

print('p-value: ', p/2)
print('Is p-value smaller than alpha of 0.05?', p/2 < alpha)
```

```
p-value:  0.0
Is p-value smaller than alpha of 0.05? True
```

Sample t-Test for Pack Size (unequal sample size and unequal variances)

We are interesting to look at if our target segment tends to buy larger packs of chips. The averge of pack size for our target segment (mainstream - young singles/couples) is 175g, and the average of pack size for other segment is 178g. We construct a t-test to test if the target segment is buying the larger pack than other segment.

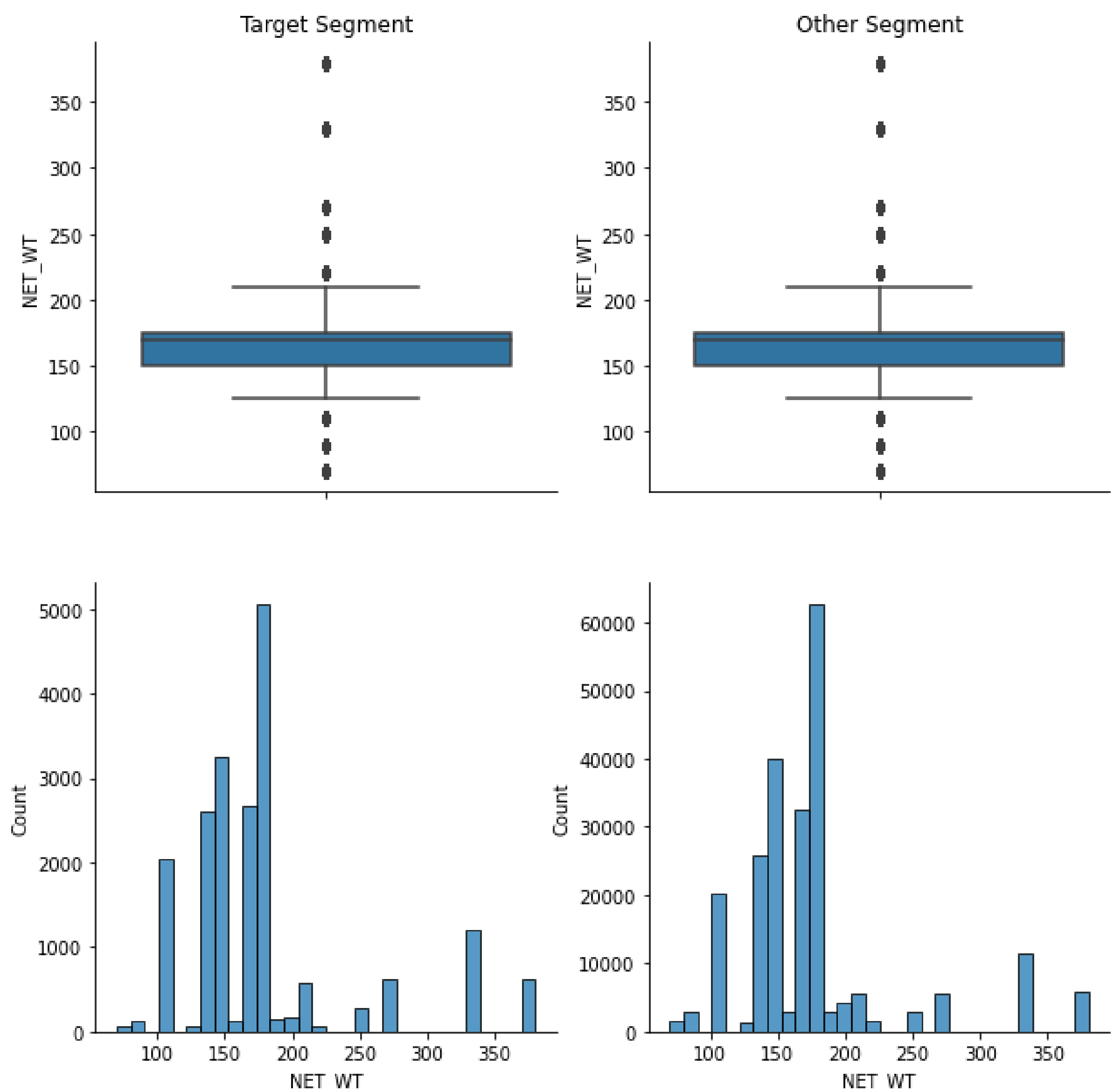
μ : average pack size
group_A : mainstream young singles/couples
group_B : other segment

$$H_0 : \mu_{\text{group\_A}} \leq \mu_{\text{group\_B}}$$
$$H_a : \mu_{\text{group\_A}} > \mu_{\text{group\_B}}$$

The t-test results in a p-value of 1.6×10^{-11}, which less then 0,0. Therefore we could reject the hypothesis null. It is significant that target segment is buying larger pack size than other segment.

In [175]:
```
mask =(transaction_customer.LIFESTAGE == 'YOUNG SINGLES/COUPLES') & (tra
nsaction_customer.PREMIUM_CUSTOMER == 'Mainstream')
target_segment = transaction_customer[mask]
other_segment = transaction_customer[~mask]
```

In [176]:
```
# boxplot and histplot
plt.subplots(2,2, sharey=True, figsize = (10, 10));
plt.subplot(2,2,1);
sns.boxplot(y = target_segment['NET_WT']);
plt.title('Target Segment')
plt.subplot(2,2,2);
sns.boxplot(y = other_segment['NET_WT']);
plt.title('Other Segment')

plt.subplot(2,2,3);
sns.histplot(x = target_segment['NET_WT'], bins = 30);

plt.subplot(2,2,4);
sns.histplot(x = other_segment['NET_WT'], bins = 30);
sns.despine()
```



In [177]:
```
# calculate the Standard Deviation
# set alpha
alpha = 0.05
# calculate the variance to get standard deviation
# For unbiased max likelyhood estimate we have to divide the var by N-1
# and therefore the parameter ddof = 1
var_a = target_segment['NET_WT'].var(ddof=1)
var_b = other_segment['NET_WT'].var(ddof = 1)
# calculate the means
m_a = target_segment['NET_WT'].mean()
m_b = other_segment['NET_WT'].mean()
# calculate the sample size
n_a = len(target_segment)
n_b = len(other_segment)
# t-statistic
t = (m_a - m_b)/np.sqrt(var_a/n_a + var_b/n_b)
# compare with critical values
# degree of freedom
df = n_a + n_b - 2
# p_value after comparison with the t
p = 1 - stats.t.cdf(t, df = df)

print('p-value: ', p/2)
print('Is p-value smaller than alpha of 0.05?', p/2 < alpha)
```

```
p-value:  1.5992929203179074e-11
Is p-value smaller than alpha of 0.05? True
```

Insights

- Top 3 total sales contributed by
  - Older Families - Budget ($\$158,380$)
  - Young Singles/Couples - Mainstream ($\$148,337$)
  - Retirees - Mainstream ($\$146,328$)
- Top 3 sales per capita contributed by
  - Midage Singles/Couples - Mainstream ($\$7.613$)
  - Young Singles/Couples - Mainstream ($\$7.528$)
  - Retirees - Premium ($\$7.435$)
- Top 3 number of pack bought by
  - Older Families - Mainstream 1.949 pack
  - Older Families - Premium 1.946 pack
  - Older Families - Budget 1.945 pack
- Top 3 unit price bought by
  - Young Singles/Couples - Mainstream ($\$4.054$)
  - Midage Singles/Couples - Mainstream ($\$3.982$)
  - Retirees - Budget ($\$3.912$)

Since Young Single/Couples are the willing to pay more unit price aroung 4 dollars. We look at the popluar brands amoung this customer segment.

- Top 3 popular brands among the segment
  - Kettle
  - Doritos
  - Pringles

Compared two segments, The unit price for mainstream, young and mid-age singles and coules are significant higher than that of budget or premium, young and midage singkes and couples

Compared two segments, It is significant that young singles/couples - mainstream is buying larger pack size than other segment.