QVI customer analysis

We have been provided by two datasets by Quantium to analysis chip purchasing behaviour and categorize customers into segments according to behaviour.

In [1]:

```
#getting the dataset
import pandas as pd
qvipb = pd.read_csv("C:\\Users\\sujoydutta\\Downloads\\QVI_purchase_behaviour.csv")
qvitd = pd.read_excel("C:\\Users\\sujoydutta\\Downloads\\QVI_transaction_data.xlsx")
```

In [2]:

```
#examining first dataset
qvipb.head()
```

Out[2]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream

In [3]:

```
#examining second dataset
qvitd.head()
```

Out[3]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

In [4]:

```
# Merging the DataFrames on the common column 'LYLTY_CARD_NBR'
data = pd.merge(qvipb, qvitd, on='LYLTY_CARD_NBR')

data.head()
```

Out[4]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_N
0	1000	YOUNG SINGLES/COUPLES	Premium	2018-10-17	1	1	5	Natural Con SeaSalt
1	1002	YOUNG SINGLES/COUPLES	Mainstream	2018-09-16	1	2	58	Red Rock Chikn&G Aioli

2	1003	YOUNG FAMILIES	Budget	2019-03-07	1	3	52	Grain W Cream&Cl 2
3	1003	YOUNG FAMILIES	Budget	2019-03-08	1	4	106	Na ChipCo H Chckn
4	1004	OLDER SINGLES/COUPLES	Mainstream	2018-11-02	1	5	96	WW Ori Stacked C

In [5]:

```python
#examining the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 10 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   LYLTY_CARD_NBR    264836 non-null  int64         
 1   LIFESTAGE         264836 non-null  object        
 2   PREMIUM_CUSTOMER  264836 non-null  object        
 3   DATE              264836 non-null  datetime64[ns]
 4   STORE_NBR         264836 non-null  int64         
 5   TXN_ID            264836 non-null  int64         
 6   PROD_NBR          264836 non-null  int64         
 7   PROD_NAME         264836 non-null  object        
 8   PROD_QTY          264836 non-null  int64         
 9   TOT_SALES         264836 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
memory usage: 20.2+ MB
```

In [6]:

```python
import regex as re
# Function to extract the pack size
def extract_pack_size(prod_name):
    match = re.search(r'(\d+)g', prod_name, re.IGNORECASE)
    return match.group(1) if match else None
```

In [7]:

```python
# Function to extract the brand name
def extract_brand_name(prod_name):
    words = prod_name.split()
    return ' '.join(words[:1])
```

In [8]:

```python
# Function to remove brand name and pack size from product name
def clean_prod_name(prod_name):
    brand_name = extract_brand_name(prod_name)
    pack_size = extract_pack_size(prod_name)
    if pack_size:
        pack_size += 'g'
    cleaned_name = prod_name.replace(brand_name, '').replace(pack_size, '').strip()

    cleaned_name = re.sub(' +', ' ', cleaned_name)
    return cleaned_name
```

In [9]:

```python
# Applying the functions to create new columns
data['PACK_SIZE'] = data['PROD_NAME'].apply(extract_pack_size)
data['BRAND_NAME'] = data['PROD_NAME'].apply(extract_brand_name)
data['PRODUCT_NAME'] = data['PROD_NAME'].apply(clean_prod_name)
```

In [10]:

```python
#examining the new dataset
data.head()
```

Out[10]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_N
0	1000	YOUNG SINGLES/COUPLES	Premium	2018-10-17	1	1	5	Natural Con SeaSalt
1	1002	YOUNG SINGLES/COUPLES	Mainstream	2018-09-16	1	2	58	Red Rock Chikn&C Aioli
2	1003	YOUNG FAMILIES	Budget	2019-03-07	1	3	52	Grain W Cream&Cl 2
3	1003	YOUNG FAMILIES	Budget	2019-03-08	1	4	106	Na ChipCo H Chckn
4	1004	OLDER SINGLES/COUPLES	Mainstream	2018-11-02	1	5	96	WW Ori Stacked C

In [11]:

```
#removing useless column
data=data.drop(['PROD_NAME'],axis=1)
data.head()
```

Out[11]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_QT
0	1000	YOUNG SINGLES/COUPLES	Premium	2018-10-17	1	1	5	
1	1002	YOUNG SINGLES/COUPLES	Mainstream	2018-09-16	1	2	58	
2	1003	YOUNG FAMILIES	Budget	2019-03-07	1	3	52	
3	1003	YOUNG FAMILIES	Budget	2019-03-08	1	4	106	
4	1004	OLDER SINGLES/COUPLES	Mainstream	2018-11-02	1	5	96	

In [12]:

```
# Convert DATE to datetime
data['DATE'] = pd.to_datetime(data['DATE'])
```

In [13]:

```
# Converting PACK_SIZE,PROD_QTY and TOT_SALES to numeric types
data['PROD_QTY'] = data['PROD_QTY'].astype(int)
data['TOT_SALES'] = data['TOT_SALES'].astype(float)
data['PACK_SIZE'] = data['PACK_SIZE'].astype(int)
```

In [14]:

```
# Ensure the remaining columns are of type object
object_columns = ['LYLTY_CARD_NBR', 'LIFESTAGE', 'PREMIUM_CUSTOMER', 'STORE_NBR', 'TXN_I
data[object_columns] = data[object_columns].astype(object)
```

In [15]:

```
# Display the DataFrame
print(data.dtypes)
```

```
LYLTY_CARD_NBR              object
LIFESTAGE                   object
PREMIUM_CUSTOMER            object
```

```
DATE                    datetime64[ns]
STORE_NBR                       object
TXN_ID                          object
PROD_NBR                        object
PROD_QTY                         int32
TOT_SALES                      float64
PACK_SIZE                        int32
BRAND_NAME                      object
PRODUCT_NAME                    object
dtype: object
```

In [16]:

```
# Function to replace outliers with the median
def replace_outliers_with_median(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    median = series.median()
    return series.apply(lambda x: median if x < lower_bound or x > upper_bound else x)
```

In [17]:

```
# Applying the function to the specified columns

data['TOT_SALES'] = replace_outliers_with_median(data['TOT_SALES'])
data['PACK_SIZE'] = replace_outliers_with_median(data['PACK_SIZE'])
```

In [18]:

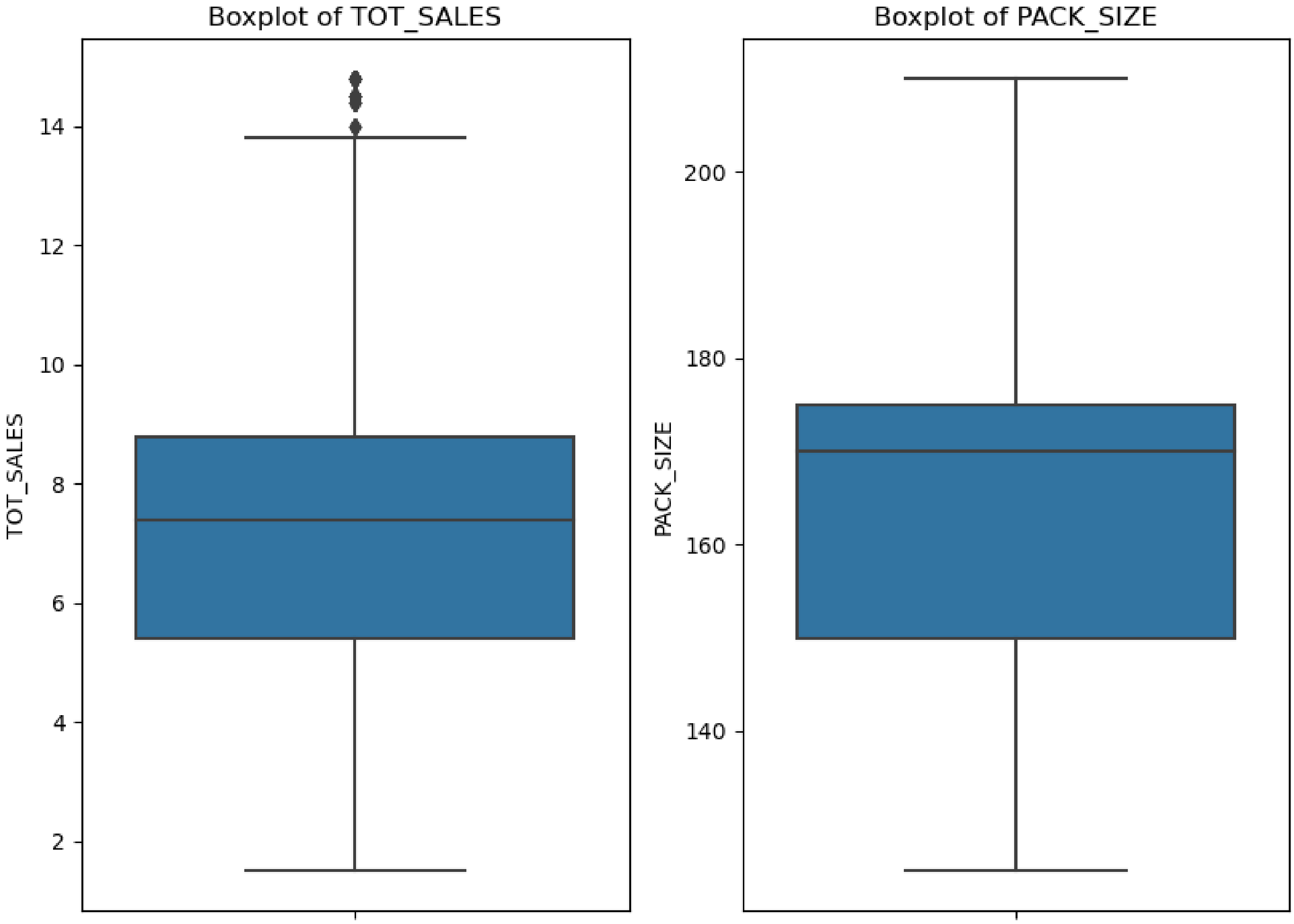
```
import seaborn as sns
import matplotlib.pyplot as plt

# Generate boxplot
plt.figure(figsize=(12, 6))


# Boxplot for TOT_SALES
plt.subplot(1, 3, 2)
sns.boxplot(y=data['TOT_SALES'])
plt.title('Boxplot of TOT_SALES')

# Boxplot for PACK_SIZE
plt.subplot(1, 3, 3)
sns.boxplot(y=data['PACK_SIZE'])
plt.title('Boxplot of PACK_SIZE')

plt.tight_layout()
plt.show()
```



In [19]:

```
#getting summary stats
data.describe()
```

Out[19]:

	DATE	PROD_QTY	TOT_SALES	PACK_SIZE
count	264836	264836.000000	264836.000000	264836.000000
mean	2018-12-30 00:52:12.879215360	1.907309	7.272554	165.241198
min	2018-07-01 00:00:00	1.000000	1.500000	125.000000
25%	2018-09-30 00:00:00	2.000000	5.400000	150.000000
50%	2018-12-30 00:00:00	2.000000	7.400000	170.000000
75%	2019-03-31 00:00:00	2.000000	8.800000	175.000000
max	2019-06-30 00:00:00	200.000000	14.800000	210.000000
std	NaN	0.643654	2.453754	16.078671

In [21]:

```
# Calculating the sum of PROD_QTY by DATE
sum_prod_qty_by_date = data.groupby('DATE')['PROD_QTY'].sum().reset_index()
sum_prod_qty_by_date
```

Out[21]:

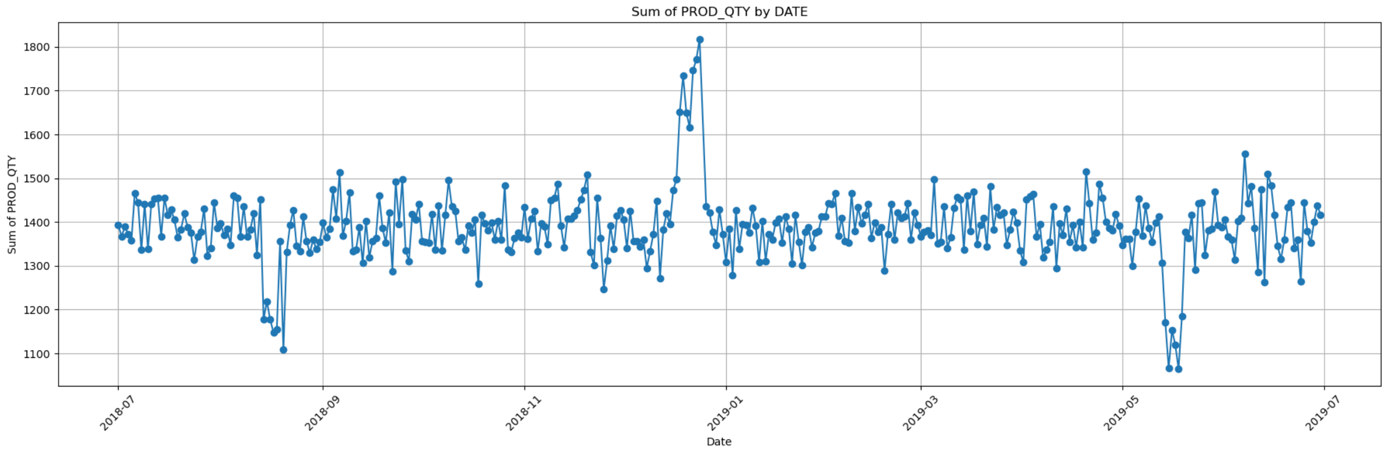
	DATE	PROD_QTY
0	2018-07-01	1394
1	2018-07-02	1367
2	2018-07-03	1389
3	2018-07-04	1373

4	2018-07-05	1358
...	...	...
359	2019-06-26	1380
360	2019-06-27	1352
361	2019-06-28	1400
362	2019-06-29	1438
363	2019-06-30	1416

364 rows × 2 columns

In [23]:

```
# Plotting the amount of products sold by date
plt.figure(figsize=(18, 6))
plt.plot(sum_prod_qty_by_date['DATE'], sum_prod_qty_by_date['PROD_QTY'], marker='o', lin
plt.title('Products sold by date')
plt.xlabel('Date')
plt.ylabel('Total Products sold on that date')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [26]:

```
#creating unit price for better analysis
data['UNITPRICE']=data['TOT_SALES']/data['PROD_QTY']
data.head()
```

Out[26]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_QT
0	1000	YOUNG SINGLES/COUPLES	Premium	2018-10-17	1	1	5	
1	1002	YOUNG SINGLES/COUPLES	Mainstream	2018-09-16	1	2	58	
2	1003	YOUNG FAMILIES	Budget	2019-03-07	1	3	52	
3	1003	YOUNG FAMILIES	Budget	2019-03-08	1	4	106	
4	1004	OLDER SINGLES/COUPLES	Mainstream	2018-11-02	1	5	96	

In [27]:

```
import scipy.stats as stats
```

```
# Convert categorical column to category type
data['LIFESTAGE'] = data['LIFESTAGE'].astype('category')
data['PREMIUM_CUSTOMER'] = data['PREMIUM_CUSTOMER'].astype('category')
```

In [29]:

```
# Prepare data for plotting
df_melted = data.melt(id_vars='LIFESTAGE', value_vars=['PROD_QTY', 'PACK_SIZE', 'TOT_SAL
                    var_name='Metric', value_name='Value')
```

In [28]:

```
# Hypothesis test to see if customer of different life stage consume the same amount or
anova_prod_qty = stats.f_oneway(*(data[data['LIFESTAGE'] == group]['PROD_QTY'] for group
print('ANOVA for LIFESTAGE and PROD_QTY:', anova_prod_qty)
```

```
ANOVA for LIFESTAGE and PROD_QTY: F_onewayResult(statistic=159.33675367200007, pvalue=6.
809757428214285e-203)
```

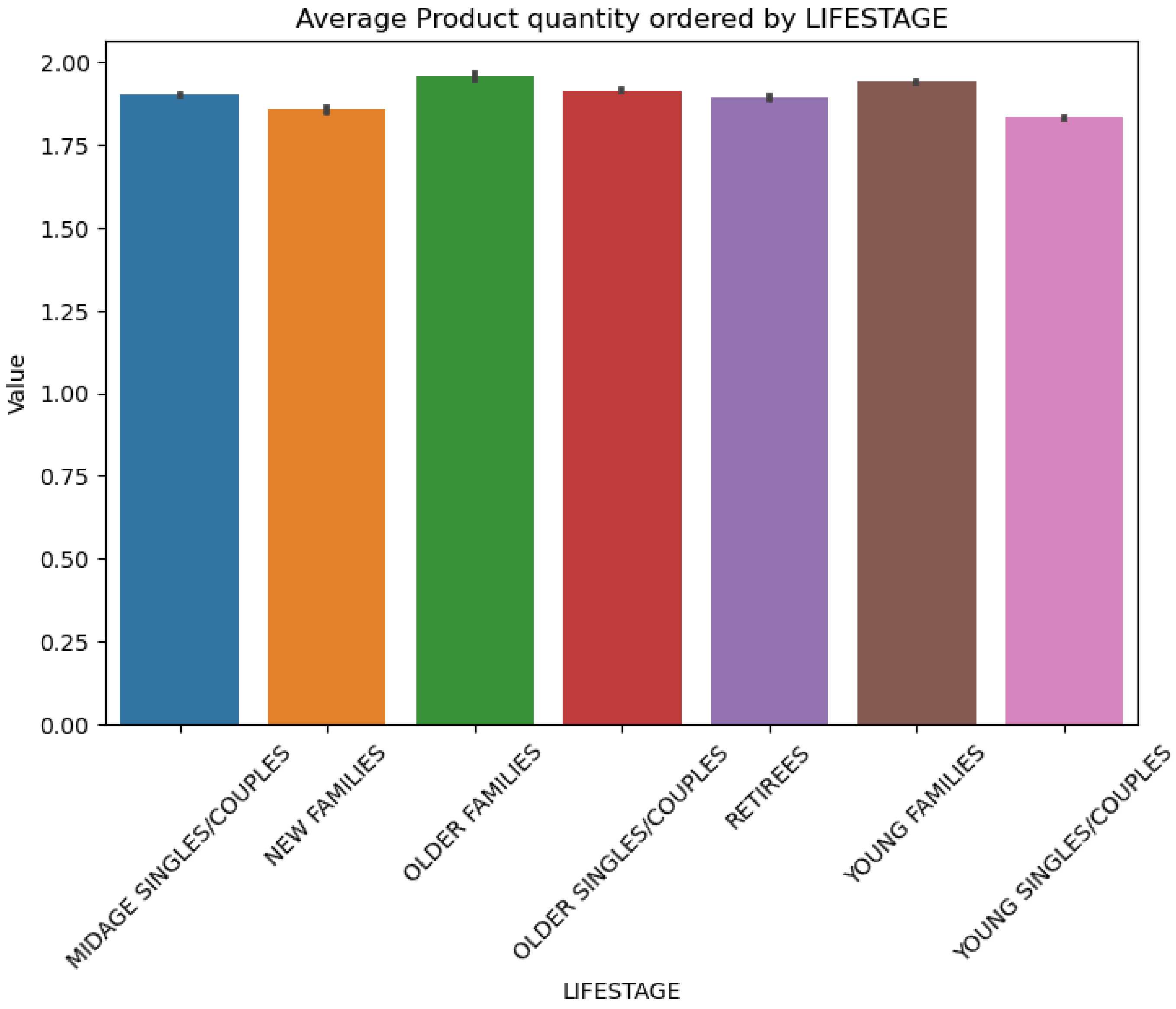
In [31]:

```
# Bar plot for PROD_QTY consumed by lifestage
plt.figure(figsize=(18, 12))

plt.subplot(2, 2, 1)
sns.barplot(x='LIFESTAGE', y='Value', data=df_melted[df_melted['Metric'] == 'PROD_QTY'])
plt.title('Average Product quantity ordered by LIFESTAGE')
plt.xticks(rotation=45)
```

Out[31]:

```
(array([0, 1, 2, 3, 4, 5, 6]),
 [Text(0, 0, 'MIDAGE SINGLES/COUPLES'),
  Text(1, 0, 'NEW FAMILIES'),
  Text(2, 0, 'OLDER FAMILIES'),
  Text(3, 0, 'OLDER SINGLES/COUPLES'),
  Text(4, 0, 'RETIREES'),
  Text(5, 0, 'YOUNG FAMILIES'),
  Text(6, 0, 'YOUNG SINGLES/COUPLES')])
```



In [33]:

```
# Hypothesis test to see if customer of different life stage like different sizes or not
anova_packsize = stats.f_oneway(*(data[data['LIFESTAGE'] == group]['PACK_SIZE'] for grou
print('ANOVA for LIFESTAGE and PACK_SIZE:', anova_packsize)
```

```
ANOVA for LIFESTAGE and PACK_SIZE: F_onewayResult(statistic=7.779375892097349, pvalue=2.
1747352758636294e-08)
```

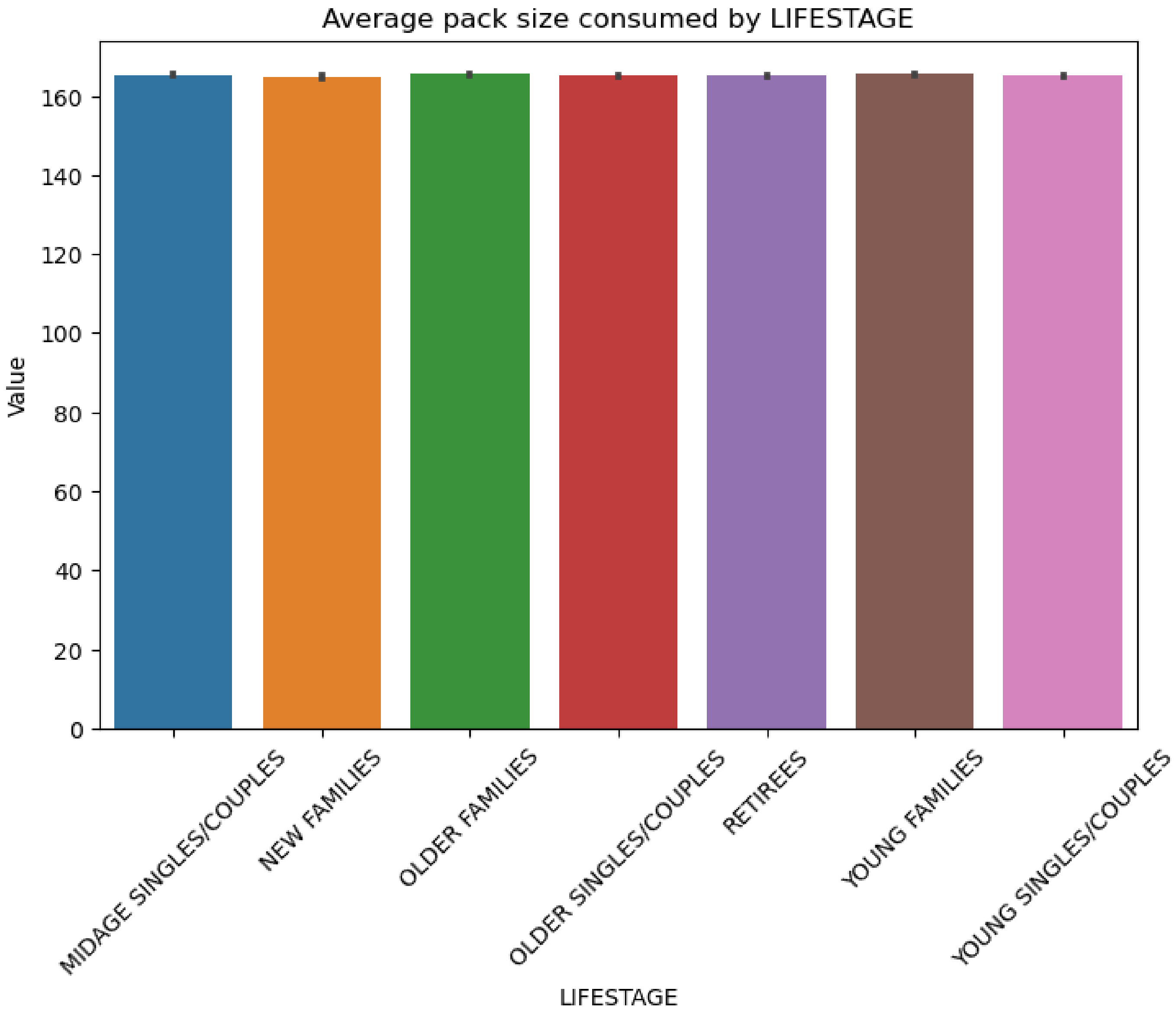
In [35]:

```
# Bar plot for PACK_SIZE consumed by lifestage
plt.figure(figsize=(18, 12))

plt.subplot(2, 2, 1)
sns.barplot(x='LIFESTAGE', y='Value', data=df_melted[df_melted['Metric'] == 'PACK_SIZE']
plt.title('Average pack size consumed by LIFESTAGE')
plt.xticks(rotation=45)
```

Out[35]:

```
(array([0, 1, 2, 3, 4, 5, 6]),
 [Text(0, 0, 'MIDAGE SINGLES/COUPLES'),
  Text(1, 0, 'NEW FAMILIES'),
  Text(2, 0, 'OLDER FAMILIES'),
  Text(3, 0, 'OLDER SINGLES/COUPLES'),
  Text(4, 0, 'RETIREES'),
  Text(5, 0, 'YOUNG FAMILIES'),
  Text(6, 0, 'YOUNG SINGLES/COUPLES')])
```



In [42]:

```
# Prepare data for plotting
df_melted = data.melt(id_vars='PREMIUM_CUSTOMER', value_vars=['PROD_QTY', 'PACK_SIZE', '
                    var_name='Metric', value_name='Value')
```

In [38]:

```
# Hypothesis test to see if customer of different life stage contribute same revenue or
anova_revenue = stats.f_oneway(*(data[data['LIFESTAGE'] == group]['TOT_SALES'] for group
print('ANOVA for LIFESTAGE and TOT_SALES:', anova_revenue)
```

```
ANOVA for LIFESTAGE and TOT_SALES: F_onewayResult(statistic=43.576812371131425, pvalue=
1.5488907741525745e-53)
```

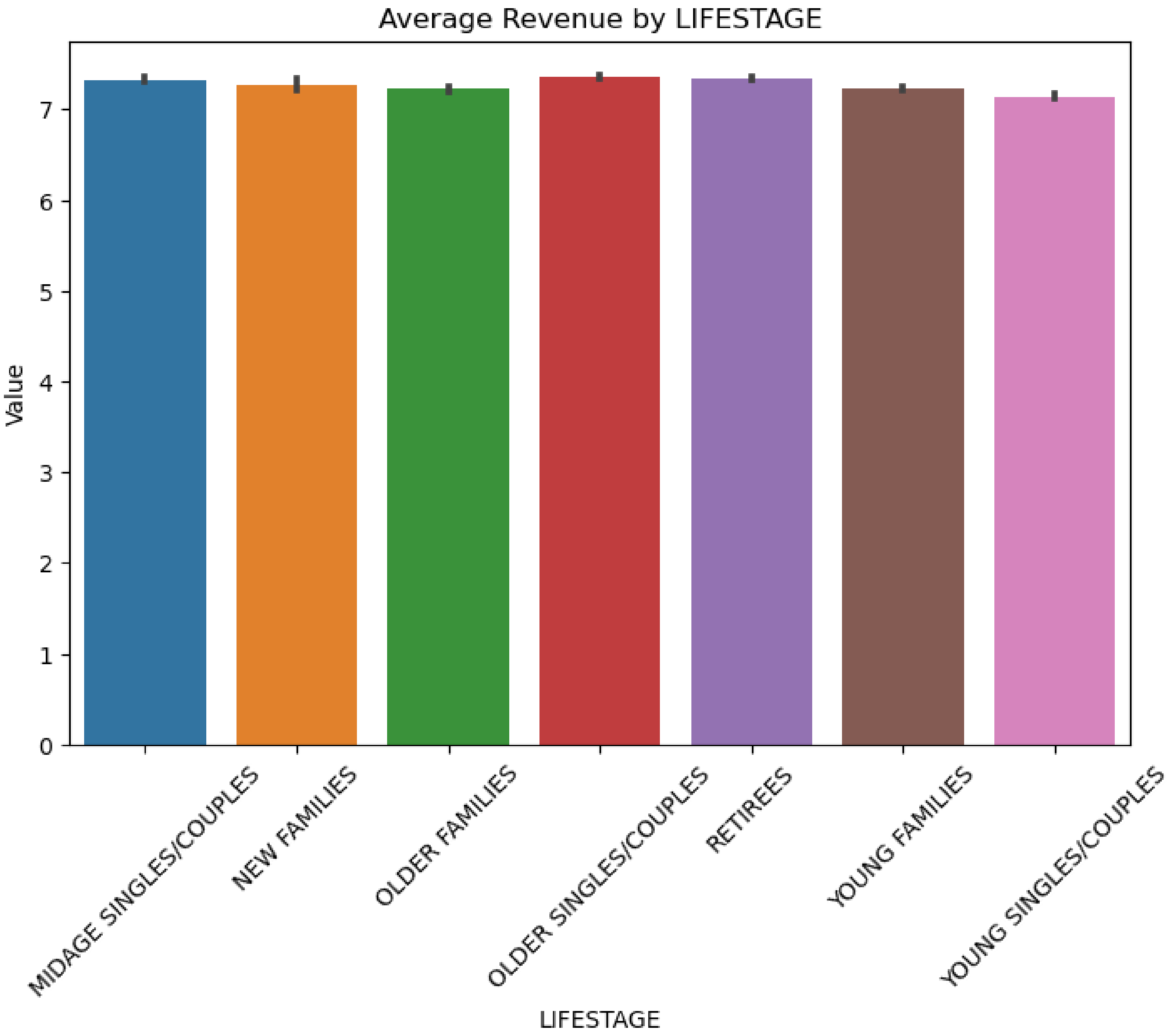
In [39]:

```
# Bar plot for average TOT_SALES by lifestage
plt.figure(figsize=(18, 12))

plt.subplot(2, 2, 1)
sns.barplot(x='LIFESTAGE', y='Value', data=df_melted[df_melted['Metric'] == 'TOT_SALES']
plt.title('Average Revenue by LIFESTAGE')
plt.xticks(rotation=45)
```

Out[39]:

```
(array([0, 1, 2, 3, 4, 5, 6]),
 [Text(0, 0, 'MIDAGE SINGLES/COUPLES'),
  Text(1, 0, 'NEW FAMILIES'),
  Text(2, 0, 'OLDER FAMILIES'),
  Text(3, 0, 'OLDER SINGLES/COUPLES'),
  Text(4, 0, 'RETIREES'),
  Text(5, 0, 'YOUNG FAMILIES'),
  Text(6, 0, 'YOUNG SINGLES/COUPLES')])
```



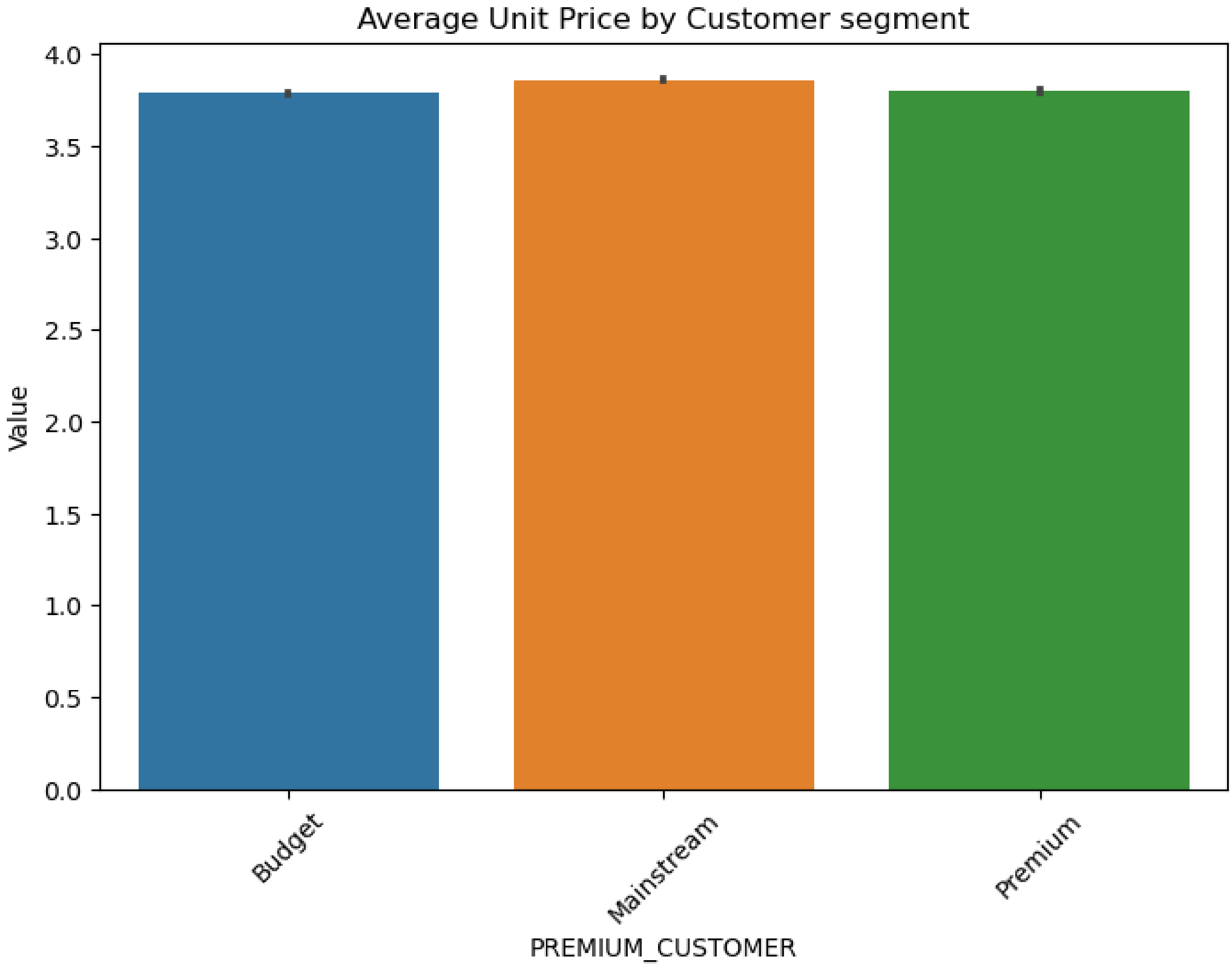
```
In [40]: # Hypothesis test to see if customer of different segment are price sensitive or not
         anova_unit_qty = stats.f_oneway(*(data[data['PREMIUM_CUSTOMER'] == group]['UNITPRICE'] f
         print('ANOVA for PREMIUM_CUSTOMER and UNITPRICE:', anova_unit_qty)

         ANOVA for PREMIUM_CUSTOMER and UNITPRICE: F_onewayResult(statistic=124.79028745132659, p
         value=6.757529672254181e-55)
```

```
In [44]: # Bar plot for average unit price by Customer segment
         plt.figure(figsize=(18, 12))

         plt.subplot(2, 2, 1)
         sns.barplot(x='PREMIUM_CUSTOMER', y='Value', data=df_melted[df_melted['Metric'] == 'UNIT
         plt.title('Average Unit Price by Customer segment')
         plt.xticks(rotation=45)

Out[44]: (array([0, 1, 2]),
          [Text(0, 0, 'Budget'), Text(1, 0, 'Mainstream'), Text(2, 0, 'Premium')])
```



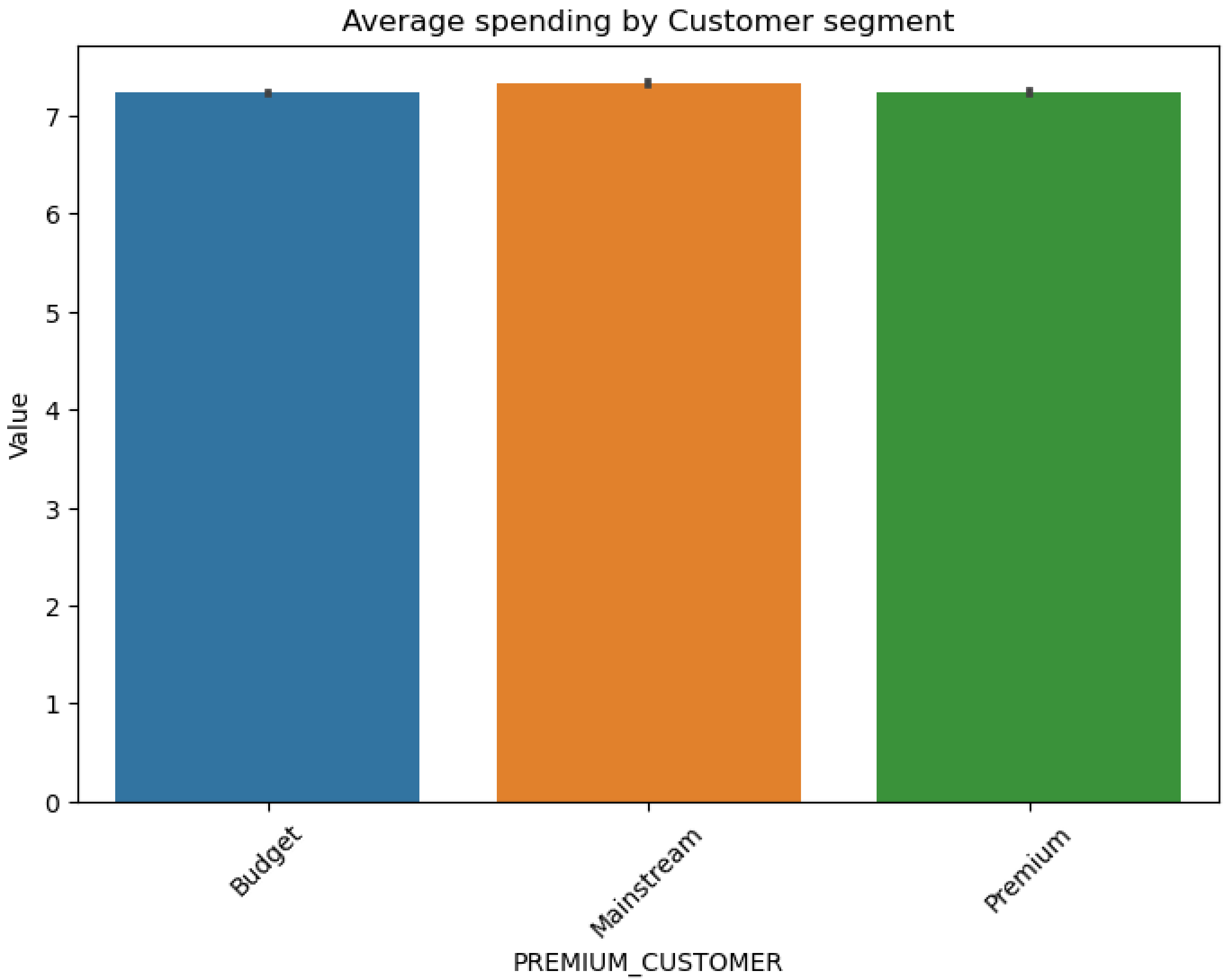
```
In [45]: # Hypothesis test to see if customer of different segment are spending the same or not
         anova_total_qty = stats.f_oneway(*(data[data['PREMIUM_CUSTOMER'] == group]['TOT_SALES']
         print('ANOVA for PREMIUM_CUSTOMER and UNITPRICE:', anova_total_qty)
```

```
ANOVA for PREMIUM_CUSTOMER and UNITPRICE: F_onewayResult(statistic=54.00624113387171, pv
alue=3.549516375774072e-24)
```

```
In [46]: # Bar plot for average spending by Customer segment
         plt.figure(figsize=(18, 12))

         plt.subplot(2, 2, 1)
         sns.barplot(x='PREMIUM_CUSTOMER', y='Value', data=df_melted[df_melted['Metric'] == 'TOT_
         plt.title('Average spending by Customer segment')
         plt.xticks(rotation=45)
```

```
Out[46]: (array([0, 1, 2]),
          [Text(0, 0, 'Budget'), Text(1, 0, 'Mainstream'), Text(2, 0, 'Premium')])
```



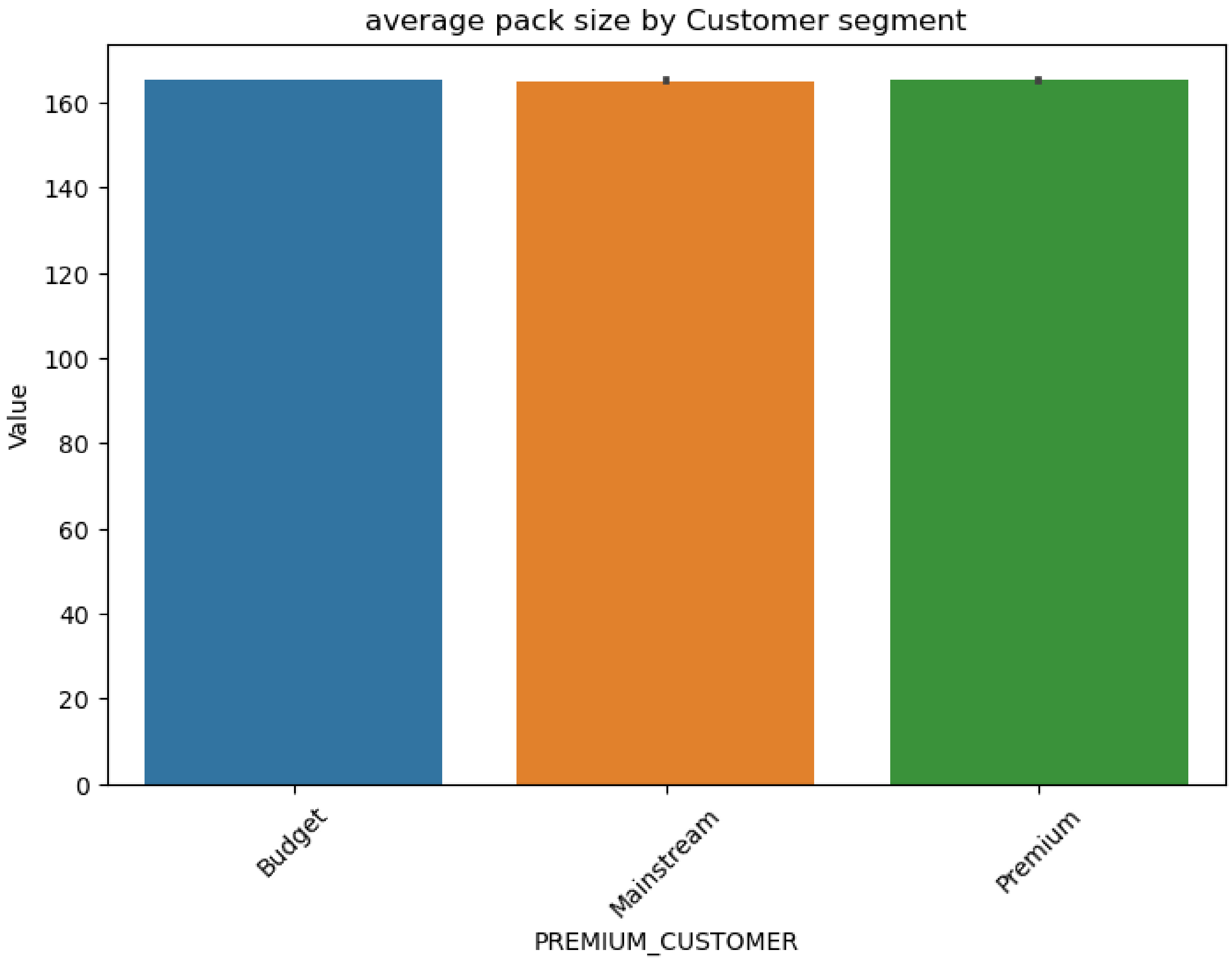
In [47]:
```
# Hypothesis test to see if customer of different segment like the same pack size or not
anova_psize = stats.f_oneway(*(data[data['PREMIUM_CUSTOMER'] == group]['PACK_SIZE'] for
print('ANOVA for PREMIUM_CUSTOMER and PACK_SIZE:', anova_psize)
```

```
ANOVA for PREMIUM_CUSTOMER and PACK_SIZE: F_onewayResult(statistic=3.365267450855786, pv
alue=0.03455425129373402)
```

In [49]:
```
# Bar plot for average pack size by Customer segment
plt.figure(figsize=(18, 12))

plt.subplot(2, 2, 1)
sns.barplot(x='PREMIUM_CUSTOMER', y='Value', data=df_melted[df_melted['Metric'] == 'PACK
plt.title('average pack size by Customer segment')
plt.xticks(rotation=45)
```

Out[49]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'Budget'), Text(1, 0, 'Mainstream'), Text(2, 0, 'Premium')])
```



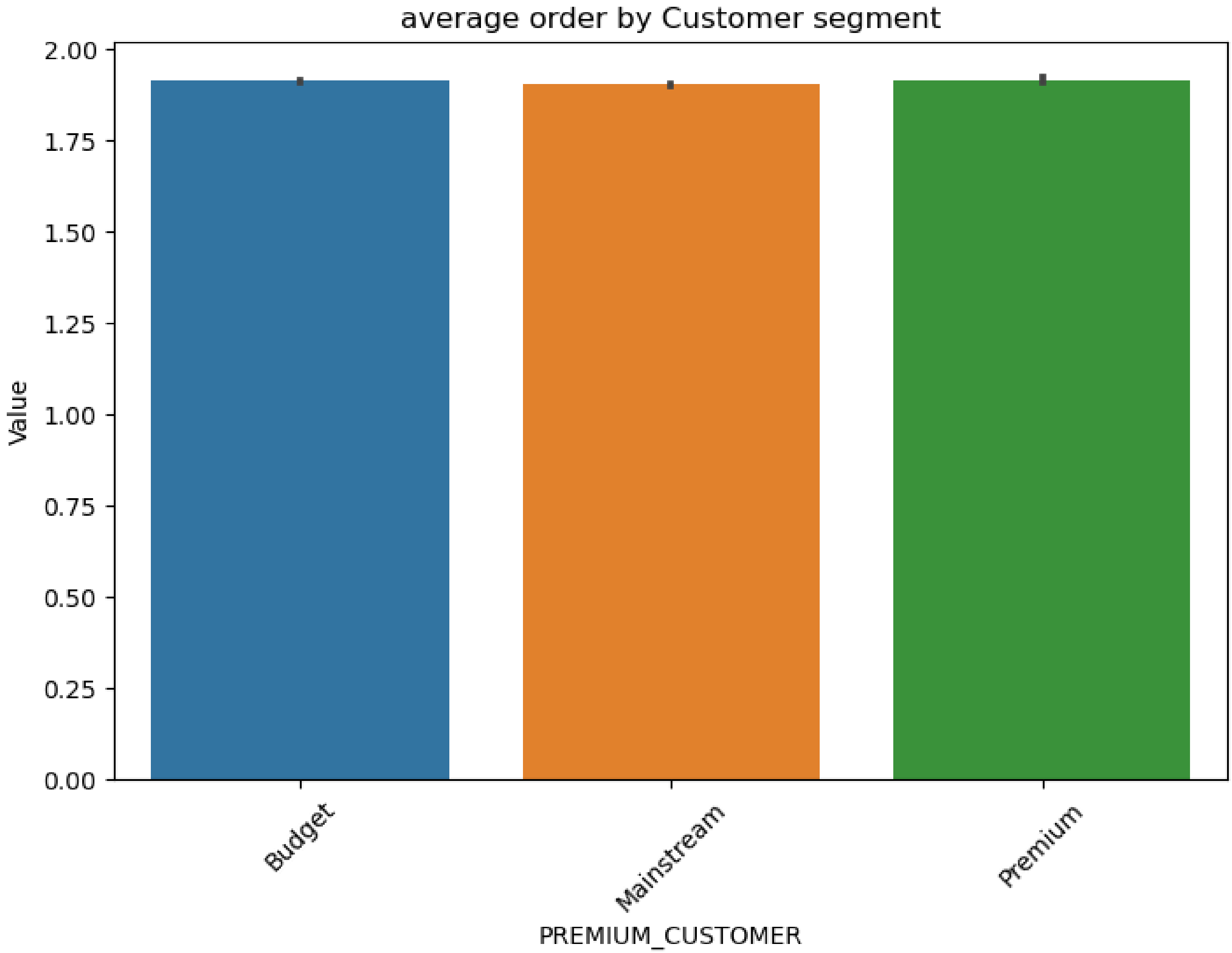
```
In [50]: # Hypothesis test to see if customer of different segment order the same number or not
         anova_orderqty = stats.f_oneway(*(data[data['PREMIUM_CUSTOMER'] == group]['PROD_QTY'] fo
         print('ANOVA for PREMIUM_CUSTOMER and PROD_QTY:', anova_orderqty)
```

```
ANOVA for PREMIUM_CUSTOMER and PROD_QTY: F_onewayResult(statistic=6.28719751819346, pval
ue=0.0018602427975793138)
```

```
In [51]: # Bar plot for average order by Customer segment
         plt.figure(figsize=(18, 12))

         plt.subplot(2, 2, 1)
         sns.barplot(x='PREMIUM_CUSTOMER', y='Value', data=df_melted[df_melted['Metric'] == 'PROD
         plt.title('average order by Customer segment')
         plt.xticks(rotation=45)
```

```
Out[51]: (array([0, 1, 2]),
          [Text(0, 0, 'Budget'), Text(1, 0, 'Mainstream'), Text(2, 0, 'Premium')])
```


average order by Customer segment
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Value
Budget
Mainstream
Premium
PREMIUM_CUSTOMER

```
In [ ]:
```