In [ ]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind
```

In [ ]:

```python
# Load transaction data
transactionData = pd.read_excel('QVI_transaction_data.xlsx')
transactionData.head()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

In [ ]:

```python
# Load customer data
customerData = pd.read_csv('QVI_purchase_behaviour.csv')
customerData.head()
```

Out[ ]:

	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

Explore Transaction data

In [ ]:

```python
# Examine structure and size of the data
transactionData
```

Out[ ]:

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

264836 rows × 8 columns

In [ ]:

```python
# Describe the data
transactionData.describe()
```

Out[ ]:

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

In [ ]:

```python
# Check for missing values
transactionData.isnull().sum()
```

Out[ ]:

```
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 [ ]:

```python
# Convert date column to date format
transactionData['DATE'] = pd.to_datetime(transactionData['DATE'], unit='D', origin='1899-12-30')
transactionData.head()
```

Out[ ]:

	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 [ ]:

```
transactionData['PROD_NAME']
```

Out[ ]:

```
0           Natural Chip        Compny SeaSalt175g
1                         CCs Nacho Cheese    175g
2           Smiths Crinkle Cut  Chips Chicken 170g
3           Smiths Chip Thinly  S/Cream&Onion 175g
4         Kettle Tortilla ChpsHny&Jlpno Chili 150g
                            ...
264831     Kettle Sweet Chilli And Sour Cream 175g
264832               Tostitos Splash Of  Lime 175g
264833                    Doritos Mexicana    170g
264834     Doritos Corn Chip Mexican Jalapeno 150g
264835               Tostitos Splash Of  Lime 175g
Name: PROD_NAME, Length: 264836, dtype: object
```

In [ ]:

```
# Split the values in the 'PROD_NAME' column by space and create a new DataFrame.
productWords = transactionData['PROD_NAME'].str.split(' ').explode().reset_index(drop=True).to_frame()

# Rename the resulting column to 'words'.
productWords = productWords.rename(columns={'PROD_NAME':'words'})
productWords
```

```
Out[ ]:            words
               0  Natural
               1     Chip
               2
               3
               4
             ...      ...
         1863917   Splash
         1863918       Of
         1863919
         1863920     Lime
         1863921     175g

        1863922 rows × 1 columns
```

```
In [ ]: # Remove all special characters and digits
        mask = productWords['words'].str.contains(r'^[a-zA-Z\s]+$', case=False, na=False)
        productWords = productWords[mask]
        productWords
```

Out[ ]:

```
            words
      0   Natural
      1      Chip
      9   Compny
     11       CCs
     12     Nacho
    ...       ...
1863914  Jalapeno
1863916  Tostitos
1863917    Splash
1863918        Of
1863920      Lime

1008776 rows × 1 columns
```

In [ ]:

```python
# Count unique words and sort in order of frequency used
word_counts = productWords['words'].value_counts().reset_index()
word_counts.columns = ['words', 'frequency']
word_counts = word_counts.sort_values(by='frequency', ascending=False)
```

In [ ]:

```python
# Display popular words
word_counts
```

Out[ ]:

	words	frequency
0	Chips	49770
1	Kettle	41288
2	Smiths	28860
3	Salt	27976
4	Cheese	27890
...	...	...
163	Whlegrn	1432
164	Pc	1431
165	NCC	1419
166	Garden	1419
167	Fries	1418

168 rows × 2 columns

In [ ]:

```
# Remove the rows in which product names contain 'salsa' by returning an inverted bool
transactionData = transactionData[~transactionData['PROD_NAME'].str.contains('salsa', case=False)]
# Display sum of true bools to ensure that it is zero, indicating success
transactionData['PROD_NAME'].str.contains('salsa', case=False).sum()
```

Out[ ]:

```
0
```

In [ ]:

```
# Statistical summary of dataframe
transactionData.describe()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
count	246742	246742.000000	2.467420e+05	2.467420e+05	246742.000000	246742.000000	246742.000000
mean	2018-12-30 01:19:01.211467520	135.051098	1.355310e+05	1.351311e+05	56.351789	1.908062	7.321322
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.700000
25%	2018-09-30 00:00:00	70.000000	7.001500e+04	6.756925e+04	26.000000	2.000000	5.800000
50%	2018-12-30 00:00:00	130.000000	1.303670e+05	1.351830e+05	53.000000	2.000000	7.400000
75%	2019-03-31 00:00:00	203.000000	2.030840e+05	2.026538e+05	87.000000	2.000000	8.800000
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.000000	200.000000	650.000000
std	NaN	76.787096	8.071528e+04	7.814772e+04	33.695428	0.659831	3.077828

In [ ]:

```
# Assess outliers in product quantity
transactionData[transactionData['PROD_QTY'] == 200]
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

In [ ]:

```
# Examine customers other purchases
transactionData[transactionData['LYLTY_CARD_NBR'] == 226000]
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

In [ ]:

```
# As customer is not a normal customer we shall filter them out
transactionData = transactionData[transactionData['LYLTY_CARD_NBR'] != 226000]
```

```
# Confirm that they have been filtered out
transactionData
```

Out[ ]:

	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
...	...	...	...	...	...	...	...	...
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2	10.8
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	1	4.4
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	2	8.8
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2	7.8
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	2	8.8

246740 rows × 8 columns

In [ ]:

```
# Rexamine statistical date now without the outlier
transactionData.describe()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
count	246740	246740.000000	2.467400e+05	2.467400e+05	246740.000000	246740.000000	246740.000000
mean	2018-12-30 01:18:58.448569344	135.050361	1.355303e+05	1.351304e+05	56.352213	1.906456	7.316113
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.700000
25%	2018-09-30 00:00:00	70.000000	7.001500e+04	6.756875e+04	26.000000	2.000000	5.800000
50%	2018-12-30 00:00:00	130.000000	1.303670e+05	1.351815e+05	53.000000	2.000000	7.400000
75%	2019-03-31 00:00:00	203.000000	2.030832e+05	2.026522e+05	87.000000	2.000000	8.800000
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.000000	5.000000	29.500000
std	NaN	76.786971	8.071520e+04	7.814760e+04	33.695235	0.342499	2.474897

In [ ]:

```
# Count numbers of rows actually containing dates
transactionsPerDay = transactionData.groupby(transactionData['DATE']).size().reset_index(name='N')
transactionsPerDay
```

Out[ ]:

	DATE	N
0	2018-07-01	663
1	2018-07-02	650
2	2018-07-03	674
3	2018-07-04	669
4	2018-07-05	660
...	...	...
359	2019-06-26	657
360	2019-06-27	669
361	2019-06-28	673
362	2019-06-29	703
363	2019-06-30	704

364 rows × 2 columns

In [ ]:

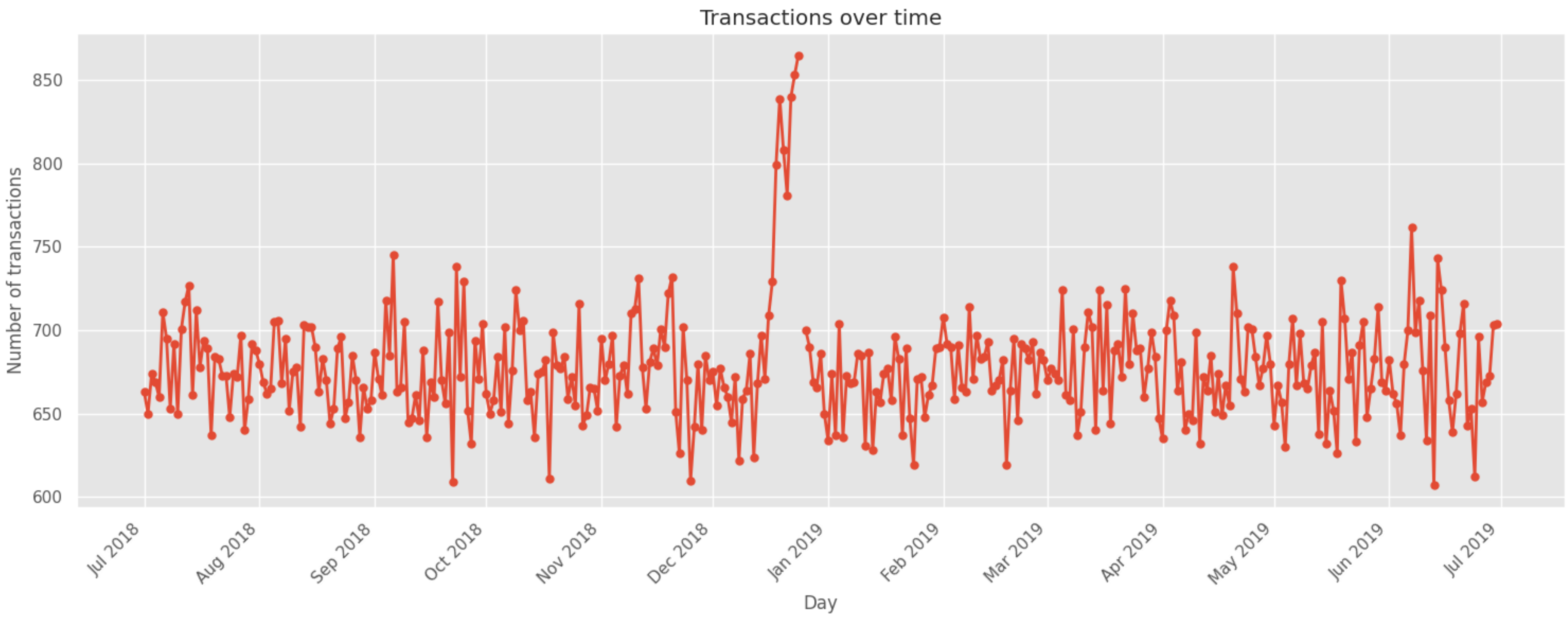
```
#Create a column of dates from 1 July 2018 to 30 June 2019 and join to data
date_range = pd.date_range(start='2018-07-01', end='2019-06-30')
date_df = pd.DataFrame({'DATE': date_range})
transactionsPerDay = date_df.merge(transactionsPerDay, on='DATE', how='left')
transactionsPerDay.head()
```

Out[ ]:

	DATE	N
0	2018-07-01	663.0
1	2018-07-02	650.0
2	2018-07-03	674.0
3	2018-07-04	669.0
4	2018-07-05	660.0

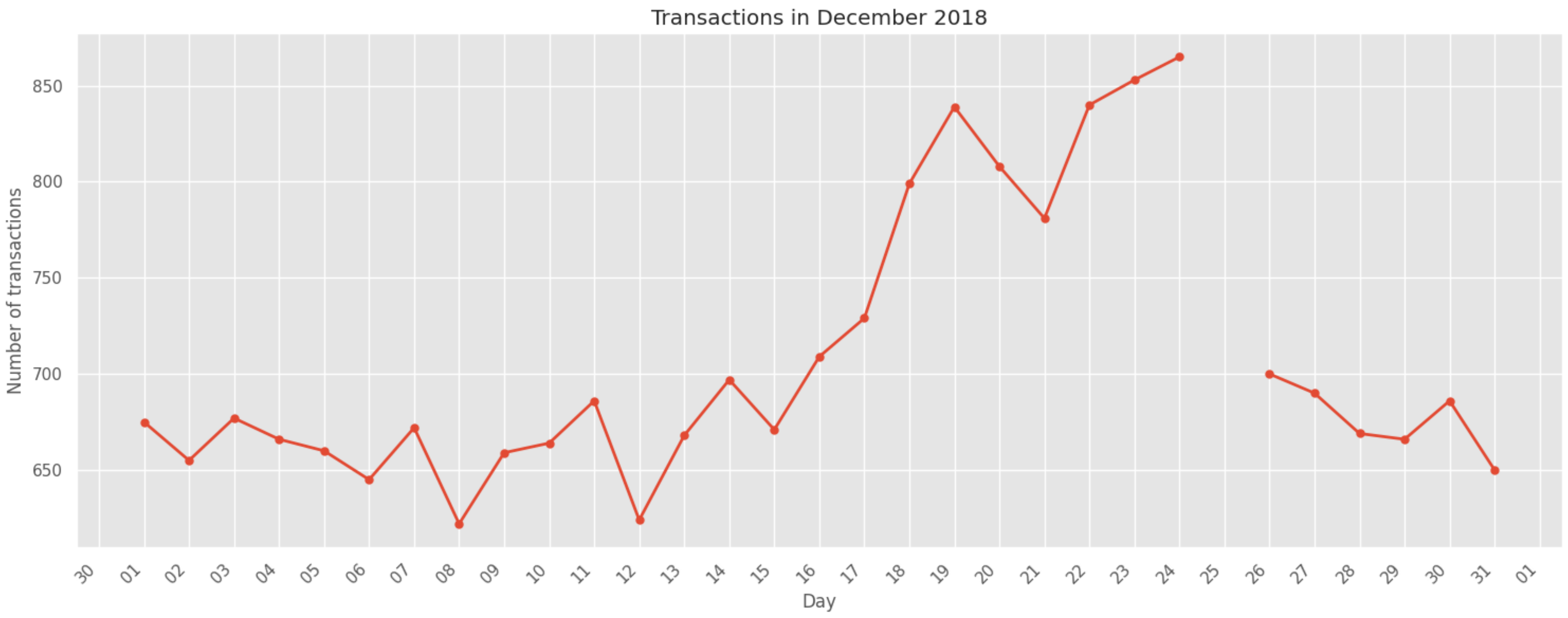
In [ ]:

```python
# Plot graph to show where the missing date is

# Plot transactions over time
plt.figure(figsize=(15, 6))  # Set the figure size
plt.plot(transactionsPerDay['DATE'], transactionsPerDay['N'], '-o', linewidth=2, markersize=5)

# Set axis labels and title
plt.xlabel("Day")
plt.ylabel("Number of transactions")
plt.title("Transactions over time")

# Format the x-axis for monthly breaks
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%b %Y'))
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.MonthLocator(interval=1))

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Display the plot
plt.tight_layout()
plt.show()
```



In [ ]:

```
# We can see that the missing value is in December 2018, so let us filter only for the december month
december_data = transactionsPerDay[(transactionsPerDay['DATE'] >= '2018-12-01') & (transactionsPerDay['DATE'] <= '2
# Plot transactions for December 2018
plt.figure(figsize=(15, 6))
plt.plot(december_data['DATE'], december_data['N'], '-o', linewidth=2, markersize=5)
plt.xlabel("Day")
plt.ylabel("Number of transactions")
plt.title("Transactions in December 2018")

# Format the x-axis for daily breaks
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%d'))
plt.gca().xaxis.set_major_locator(plt.matplotlib.dates.DayLocator(interval=1))

plt.xticks(rotation=45, ha='right')

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



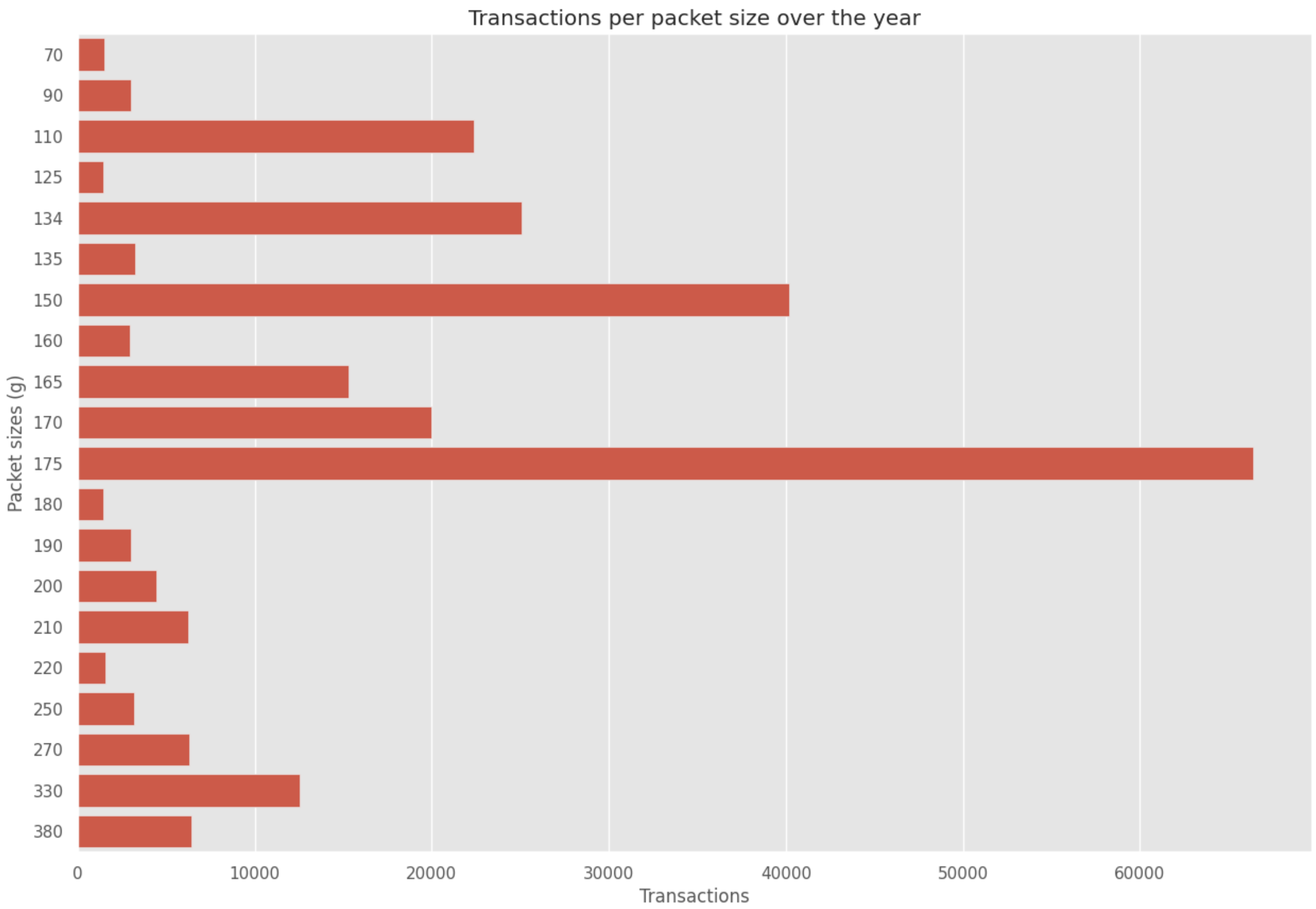
```
In [ ]: # Create column of the packet size for each transaction
        transactionData['PACK_SIZE'] = transactionData['PROD_NAME'].apply(lambda x : int(''.join(filter(str.isdigit, x))))
        transactionData.describe()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES	PACK_
count	246740	246740.000000	2.467400e+05	2.467400e+05	246740.000000	246740.000000	246740.000000	246740.00
mean	2018-12-30 01:18:58.448569344	135.050361	1.355303e+05	1.351304e+05	56.352213	1.906456	7.316113	175.58
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.700000	70.00
25%	2018-09-30 00:00:00	70.000000	7.001500e+04	6.756875e+04	26.000000	2.000000	5.800000	150.00
50%	2018-12-30 00:00:00	130.000000	1.303670e+05	1.351815e+05	53.000000	2.000000	7.400000	170.00
75%	2019-03-31 00:00:00	203.000000	2.030832e+05	2.026522e+05	87.000000	2.000000	8.800000	175.00
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.000000	5.000000	29.500000	380.00
std	NaN	76.786971	8.071520e+04	7.814760e+04	33.695235	0.342499	2.474897	59.43

In [ ]:

```python
# Plot bargraph based on the transactions by packet size
plt.figure(figsize=(15,10))
sns.countplot(transactionData, y='PACK_SIZE')
plt.ylabel('Packet sizes (g)')
plt.xlabel('Transactions')
plt.title('Transactions per packet size over the year')
plt.show()
```



```
In [ ]: # Create a brand name column in transactionData
        transactionData['BRAND'] = transactionData['PROD_NAME'].str.split(' ').str[0]
```

```
transactionData
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	Natural
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	175	CCs
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	170	Smiths
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0	175	Smiths
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8	150	Kettle
...	...	...	...	...	...	...	...	...	...	...
264831	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2	10.8	175	Kettle
264832	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	1	4.4	175	Tostitos
264833	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	2	8.8	170	Doritos
264834	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	2	7.8	150	Doritos
264835	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	2	8.8	175	Tostitos

246740 rows × 10 columns

In [ ]:

```
# Examine brand names for cleaning
transactionData['BRAND'].value_counts(sort=False, ascending=True)
```

Out[ ]:

```
BRAND
Natural        6050
CCs            4551
Smiths        27390
Kettle        41288
Grain          6272
Doritos       22041
Twisties       9454
WW            10320
Thins         14075
Burger         1564
NCC            1419
Cheezels       4603
Infzns         3144
Red            4427
Pringles      25102
Dorito         3183
Infuzions     11057
Smith          2963
GrnWves        1468
Tyrrells       6442
Cobs           9693
French         1418
RRD           11894
Tostitos       9471
Cheetos        2927
Woolworths     1516
Snbts          1576
Sunbites       1432
Name: count, dtype: int64
```

In [ ]:

```
# Replace duplicate brands that have been named differently
transactionData['BRAND'].replace('Dorito', 'Doritos', inplace=True)
transactionData['BRAND'].replace('Red', 'RRD', inplace=True)
transactionData['BRAND'].replace('Infzns', 'Infuzions' ,inplace=True)
```

```python
transactionData['BRAND'].replace('Snbts', 'Sunbites' ,inplace=True)
transactionData['BRAND'].replace('WW', 'Woolworths' ,inplace=True)
transactionData['BRAND'].replace('Natural', 'NCC' ,inplace=True)
transactionData['BRAND'].replace('GrnWves', 'GrainWaves' ,inplace=True)
transactionData['BRAND'].replace('Grain', 'GrainWaves' ,inplace=True)
```

In [ ]:

```python
# Rexamine brand list after cleaning
transactionData['BRAND'].value_counts(sort=False, ascending=True)
```

Out[ ]:

```
BRAND
NCC             7469
CCs             4551
Smiths         27390
Kettle         41288
GrainWaves      7740
Doritos        25224
Twisties        9454
Woolworths     11836
Thins          14075
Burger          1564
Cheezels        4603
Infuzions      14201
RRD            16321
Pringles       25102
Smith           2963
Tyrrells        6442
Cobs            9693
French          1418
Tostitos        9471
Cheetos         2927
Sunbites        3008
Name: count, dtype: int64
```

Explore Customer Dataset

In [ ]:

```python
# Examine structure and size of data
customerData
```

Out[ ]:

	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

In [ ]:

```
# Search for null values
customerData.isnull().sum()
```

Out[ ]:

```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [ ]:

```
# Examine dataframe info
customerData.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 [ ]:

```python
# Merge the dataframes into 1
data = transactionData.merge(customerData, on="LYLTY_CARD_NBR")
data
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NCC
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	175	CCs
2	2018-11-10	1	1307	346	96	WW Original Stacked Chips 160g	2	3.8	160	Woolworths
3	2019-03-09	1	1307	347	54	CCs Original 175g	1	2.1	175	CCs
4	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	170	Smiths
...	...	...	...	...	...	...	...	...	...	...
246735	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	2	10.8	175	Kettle
246736	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	1	4.4	175	Tostitos
246737	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	2	8.8	170	Doritos
246738	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno	2	7.8	150	Doritos

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND
						150g				
246739	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	2	8.8	175	Tostitos

246740 rows × 12 columns

In [ ]:

```
# Check for missing values
data.isnull().sum()
```

Out[ ]:

```
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PACK_SIZE           0
BRAND               0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [ ]:

```
data.to_csv('full_transaction_data.csv')
```

Data Analysis

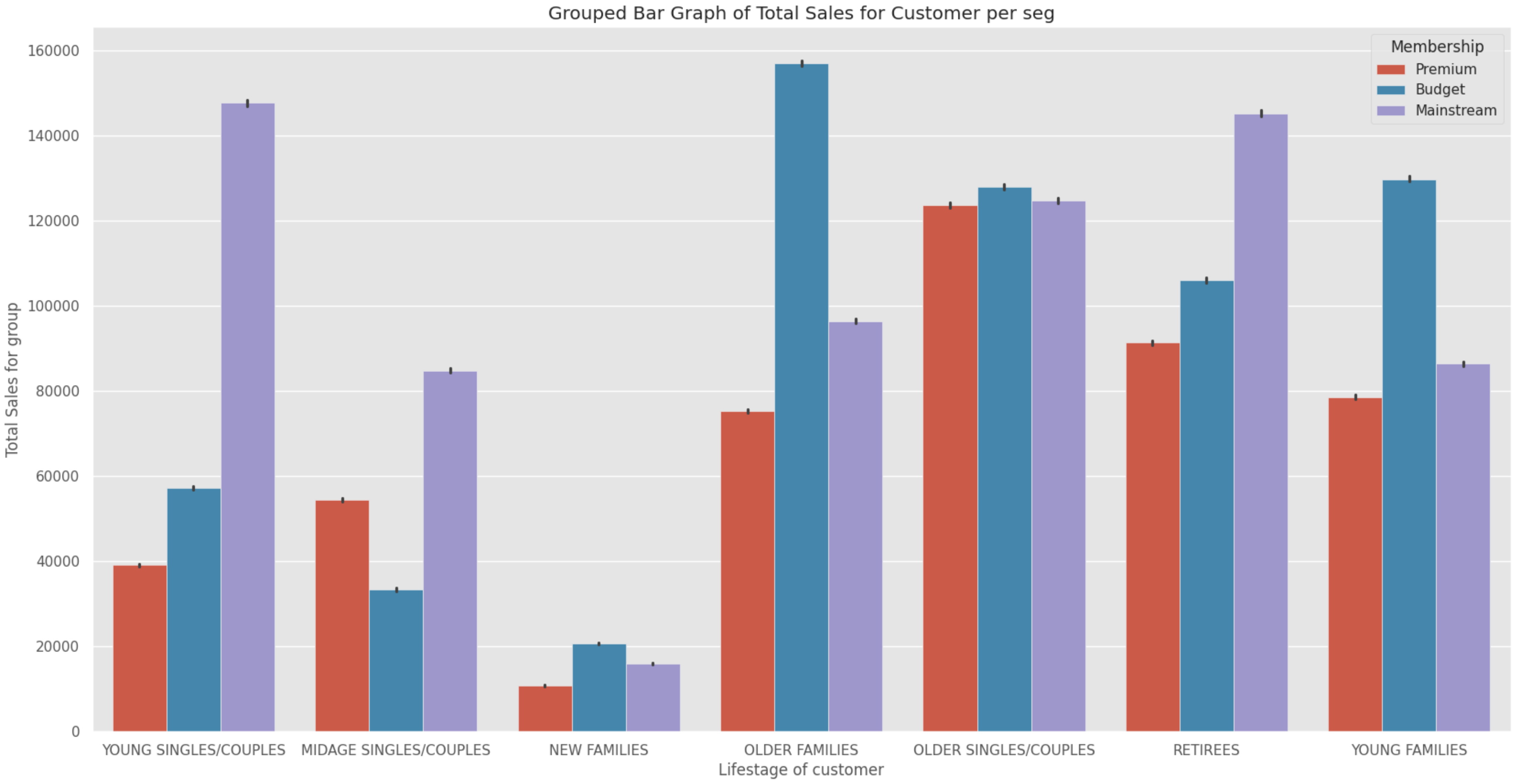
- Who spends the most on chips (total sales), describing customers by lifestage and how premium is their general purchasing behaviour?

In [ ]:

```
# Create the bar graph
plt.figure(figsize=(20,10))
```

```
sns.barplot(data, x ="LIFESTAGE", y="TOT_SALES", hue="PREMIUM_CUSTOMER", estimator=sum)

# Customize the graph
plt.xlabel('Lifestage of customer')
plt.ylabel('Total Sales for group')
plt.title('Grouped Bar Graph of Total Sales for Customer per seg')
plt.legend(title='Membership')
```

```
Out[ ]:  <matplotlib.legend.Legend at 0x7fea45f59d90>
```



We can see that the top 3 main contributors to our sales are:

- Budget Older Families
- Mainstream Young Singles/Couples
- Mainstream Retirees

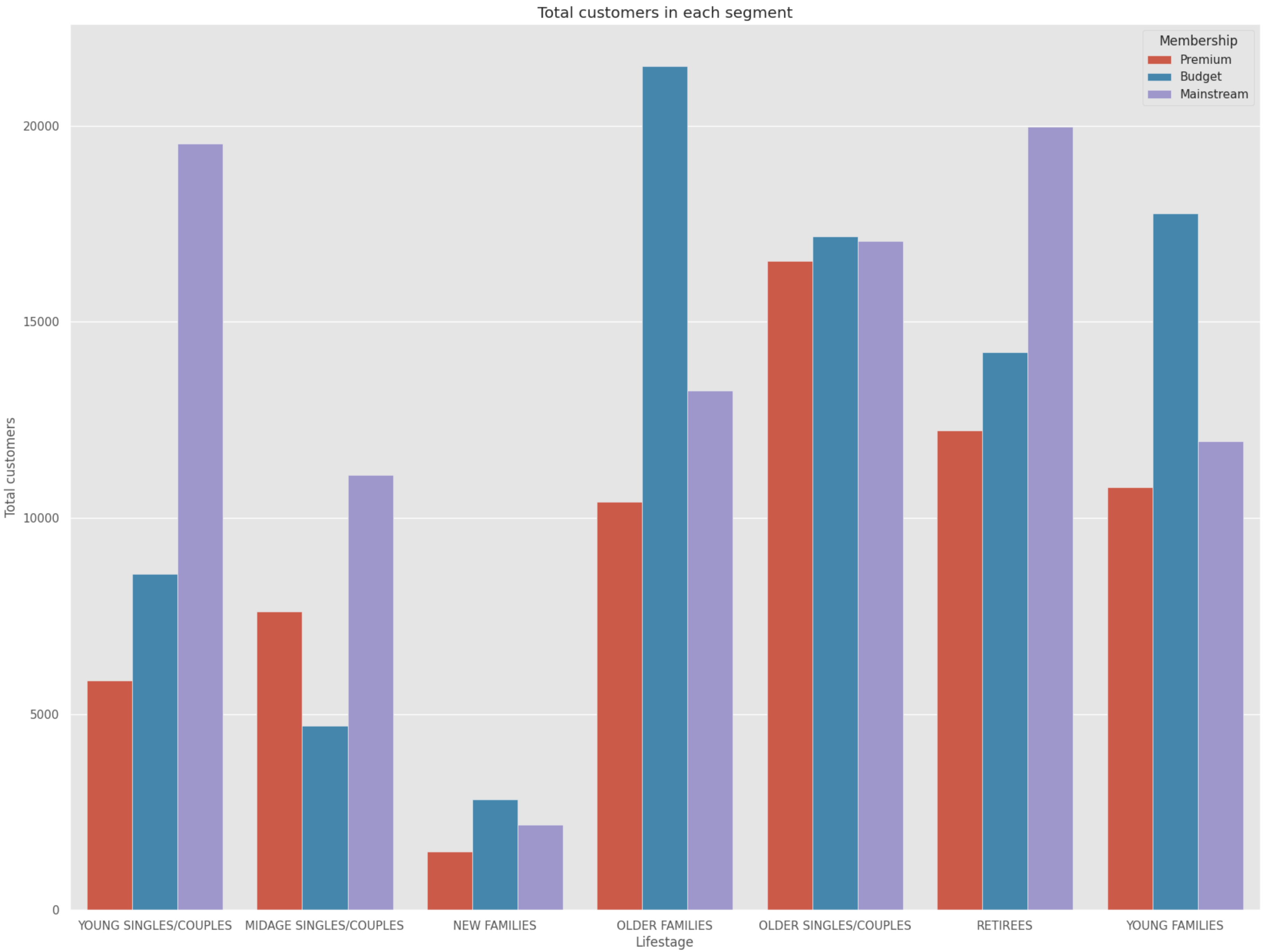
- How many customers are in each segment?

In [ ]:

```
# Create histogram
plt.figure(figsize=(20,15))
sns.countplot(data, x='LIFESTAGE', hue='PREMIUM_CUSTOMER')

# Customize graph
plt.xlabel('Lifestage')
plt.ylabel('Total customers')
plt.title('Total customers in each segment')
plt.legend(title='Membership')
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7fea45ff0350>
```


Total customers in each segment
Membership
Premium
Budget
Mainstream
Total customers
0
5000
10000
15000
20000
YOUNG SINGLES/COUPLES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
OLDER FAMILIES
OLDER SINGLES/COUPLES
RETIREES
YOUNG FAMILIES
Lifestage

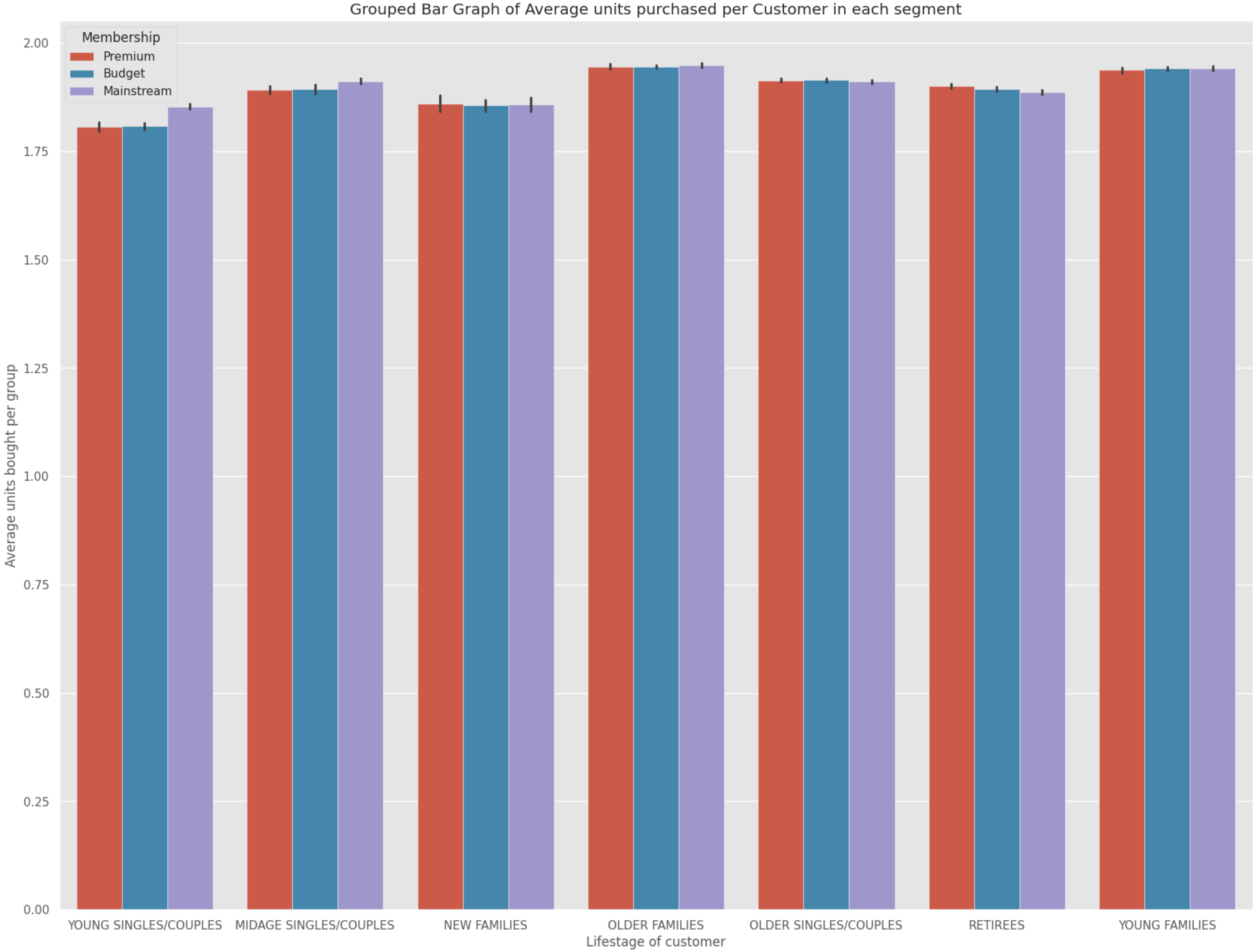
- How many chips are bought per customer by segment?

In [ ]:

```
plt.figure(figsize=(20,15))
sns.barplot(data, x ="LIFESTAGE", y="PROD_QTY", hue="PREMIUM_CUSTOMER", estimator='mean')

# Customize the graph
plt.xlabel('Lifestage of customer')
plt.ylabel('Average units bought per group')
plt.title('Grouped Bar Graph of Average units purchased per Customer in each segment')
plt.legend(title='Membership')
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7fea460b3e90>
```


Grouped Bar Graph of Average units purchased per Customer in each segment
Membership
Premium
Budget
Mainstream
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Average units bought per group
YOUNG SINGLES/COUPLES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
OLDER FAMILIES
OLDER SINGLES/COUPLES
RETIREES
YOUNG FAMILIES
Lifestage of customer

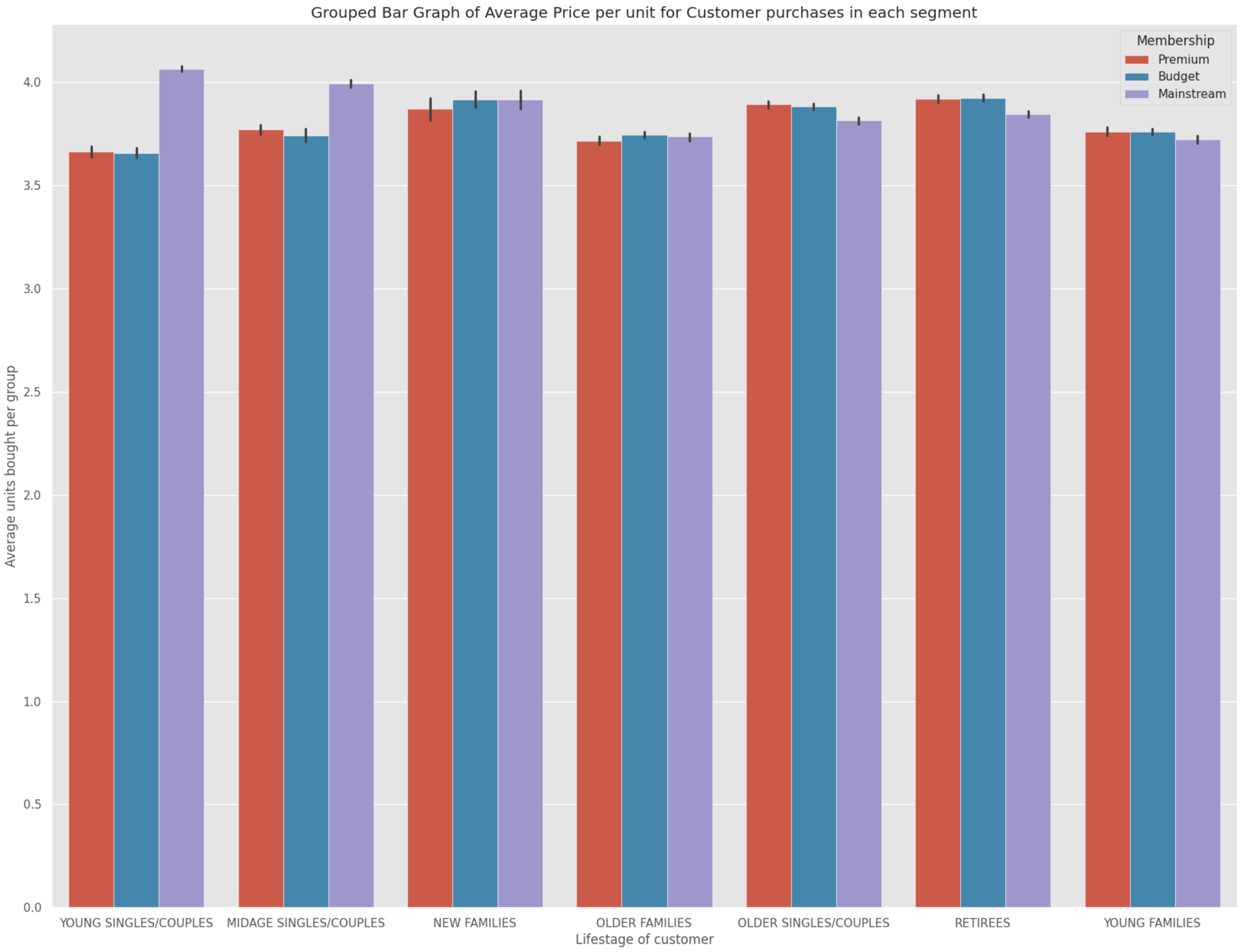
- What's the average chip price by customer segment?

In [ ]:

```
# Create unit price column
data['UNIT_PRICE'] = data['TOT_SALES']/data['PROD_QTY']
# Create graph
plt.figure(figsize=(20,15))
sns.barplot(data, x ="LIFESTAGE", y="UNIT_PRICE", hue="PREMIUM_CUSTOMER", estimator='mean')

# Customize the graph
plt.xlabel('Lifestage of customer')
plt.ylabel('Average units bought per group')
plt.title('Grouped Bar Graph of Average Price per unit for Customer purchases in each segment')
plt.legend(title='Membership')
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7fea46037150>
```


Grouped Bar Graph of Average Price per unit for Customer purchases in each segment
Membership
🟥 Premium
🟦 Budget
🟪 Mainstream
Average units bought per group
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
YOUNG SINGLES/COUPLES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
OLDER FAMILIES
OLDER SINGLES/COUPLES
RETIREES
YOUNG FAMILIES
Lifestage of customer

Let's now perform an independent t-test on the average price per unit between mainstream vs premium and budget customers, to determine is there there a statistical difference between them

In [ ]:
```
# Define sample groups
group1 = data[data['PREMIUM_CUSTOMER']== 'Mainstream'][(data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') | (data['LIFE
group2 = data[(data['PREMIUM_CUSTOMER']== 'Premium') | (data['PREMIUM_CUSTOMER']== 'Budget')][(data['LIFESTAGE'] ==
```

```
/tmp/ipykernel_6021/350695917.py:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  group1 = data[data['PREMIUM_CUSTOMER']== 'Mainstream'][(data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') | (data['LIF
ESTAGE'] == 'MIDAGE SINGLES/COUPLES')]
/tmp/ipykernel_6021/350695917.py:3: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  group2 = data[(data['PREMIUM_CUSTOMER']== 'Premium') | (data['PREMIUM_CUSTOMER']== 'Budget')][(data['LIFESTAGE'] =
= 'YOUNG SINGLES/COUPLES') | (data['LIFESTAGE'] == 'MIDAGE SINGLES/COUPLES')]
```

In [ ]:
```
# Perform t-test
stat, pval = ttest_ind(group1['UNIT_PRICE'], group2['UNIT_PRICE'])
print(f'The p-value is {pval}')
```

```
The p-value is 2.235645611540966e-309
```

As the p-value is less than 0.05, we can reject the null hypothesis and conclude that the unit price for mainstream, young and mid-age singles and couples are significantly higher than that of budget or premium, young and midage singles and couples.

Deep dive into specific customer segments

With the initial defining of metrics for the client now completed, we can perform a more indepth analysis on the most prominent groups. This will help us to better understand the customers who contribute the most to sales and how to retain them

First we shall assess whether there are any preferred brands for the mainstream, young singles/couples segments

In [ ]:
```
data['BRAND'][(data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (data['PREMIUM_CUSTOMER'] == 'Mainstream')].value_co
```

Out[ ]:

```
BRAND
Kettle        3844
Doritos       2379
Pringles      2315
Smiths        1790
Infuzions     1250
Thins         1166
Twisties       900
Tostitos       890
RRD            875
Cobs           864
GrainWaves     646
Tyrrells       619
Woolworths     479
NCC            394
Cheezels       346
CCs            222
Cheetos        166
Smith          131
Sunbites       128
French          78
Burger          62
Name: count, dtype: int64
```

We can see at a preliminary stage the ranking of brands based on total purchase transactions. The top 3 in this view are Kettle, Doritos and Pringles.

However for a deeper analysis we will also have to determine the total units purchased for each brand:

In [ ]:

```
brand_purchasing = data[(data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (data['PREMIUM_CUSTOMER'] == 'Mainstream')
brand_purchasing.groupby('BRAND')['PROD_QTY'].sum().sort_values(ascending=False)
```

Out[ ]:

```
BRAND
Kettle        7172
Doritos       4447
Pringles      4326
Smiths        3252
Infuzions     2343
Thins         2187
Twisties      1673
Tostitos      1645
Cobs          1617
RRD           1587
GrainWaves    1185
Tyrrells      1143
Woolworths     873
NCC            710
Cheezels       651
CCs            405
Cheetos        291
Smith          239
Sunbites       230
French         143
Burger         106
Name: PROD_QTY, dtype: int64
```

Here we can see that the previous top 3 brands still remain when looking at both the number of transactions and the total units purchased

Next let us find out if the target segment tends to buy larger pack sizes

In [ ]:

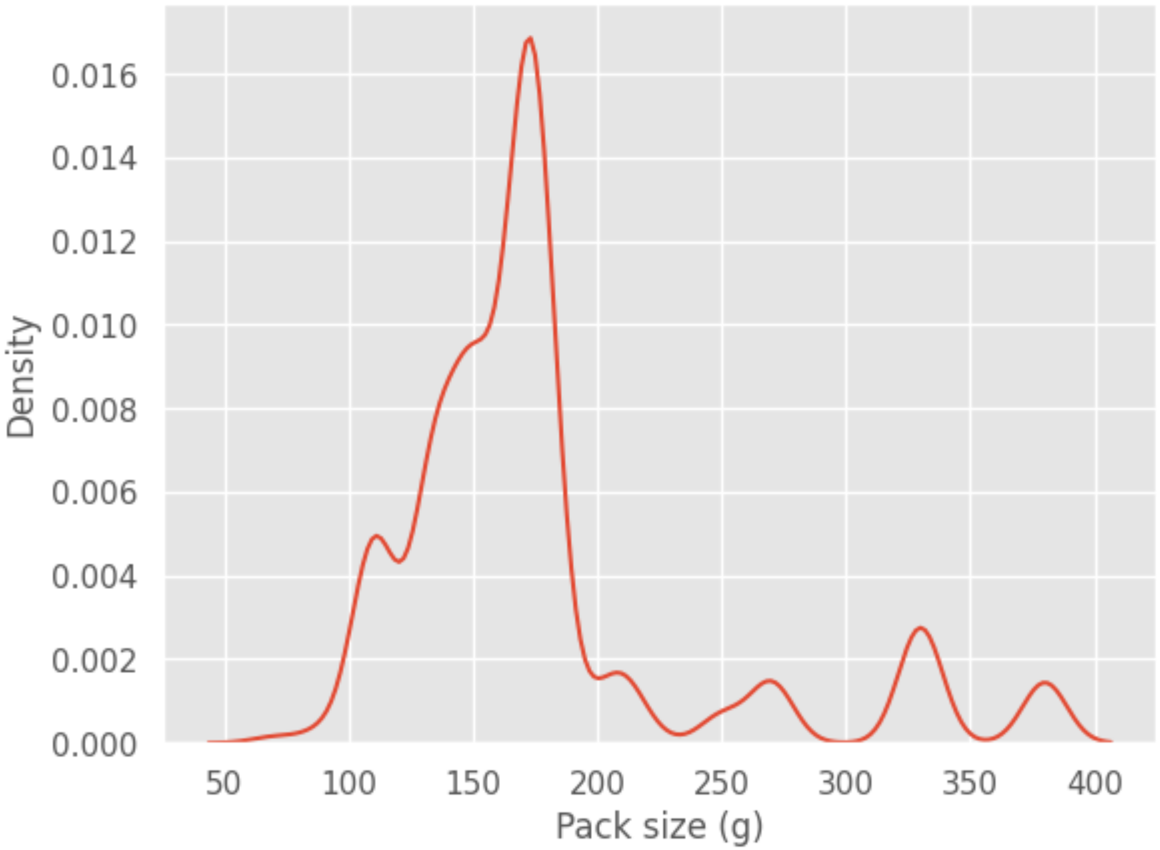
```
data['PACK_SIZE'][(data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (data['PREMIUM_CUSTOMER'] == 'Mainstream')].valu
```

```
Out[ ]: PACK_SIZE
        175    4997
        150    3080
        134    2315
        110    2051
        170    1575
        330    1195
        165    1102
        380     626
        270     620
        210     576
        135     290
        250     280
        200     179
        190     148
        90      128
        160     128
        180      70
        70       63
        220      62
        125      59
        Name: count, dtype: int64
```

We can get a general sense from the table that the most commonly purchased sizes are 175g, 150g and 134g. However a better display of this would be with a graph

```
In [ ]: pack_size = data[(data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (data['PREMIUM_CUSTOMER'] == 'Mainstream')]
        plt.figure()
        sns.kdeplot(pack_size, x='PACK_SIZE')
        plt.xlabel('Pack size (g)')
```

```
Out[ ]: Text(0.5, 0, 'Pack size (g)')
```



This graph shows us that most transactions are occuring below the halfway mark in the size of packs the client sells. Therefore we can determine that the customers prefer smaller pack sizes