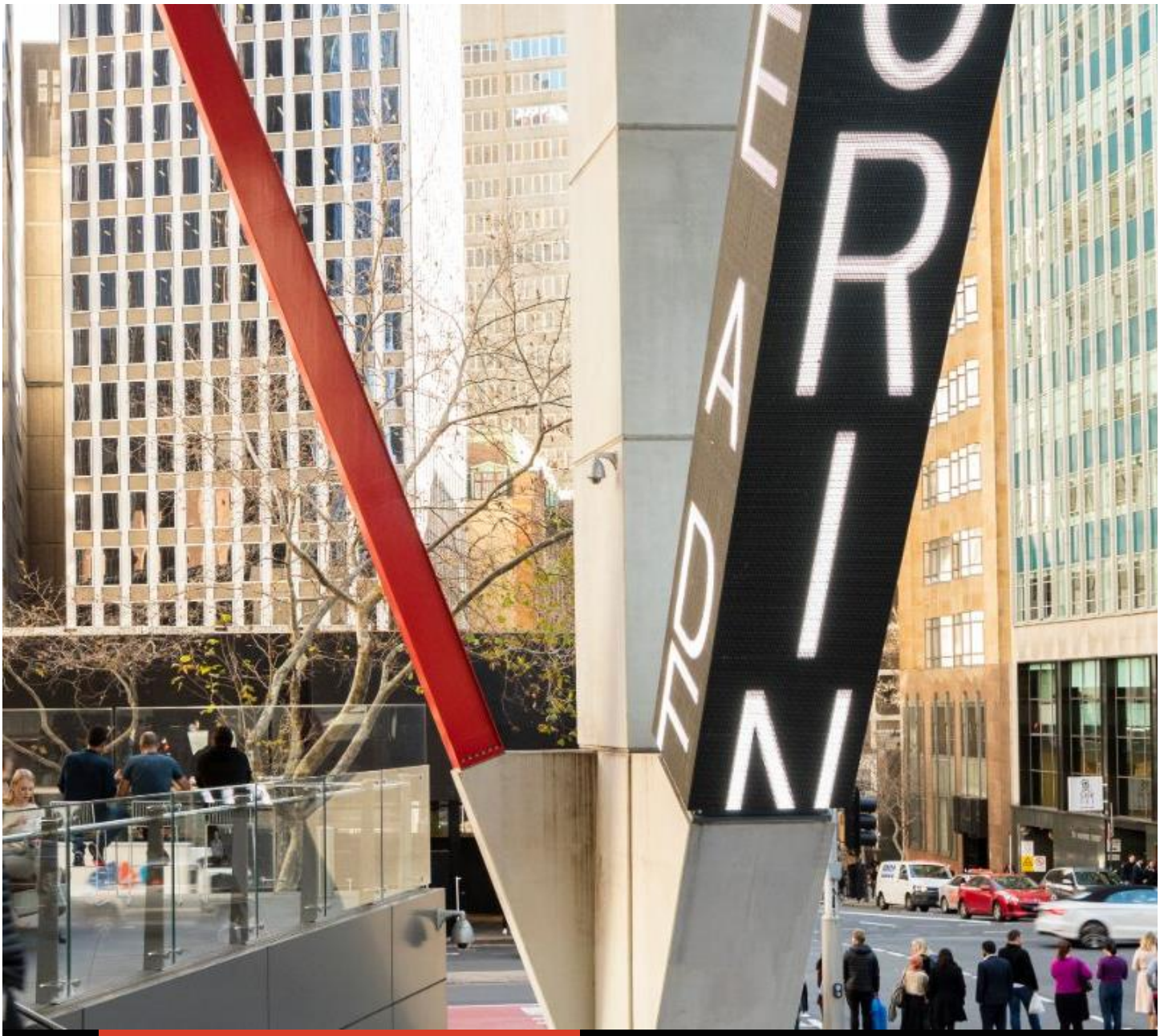


QUANTIUM VIRTUAL INTERNSHIP

TASK 1

JOEL DOMINGO Joeldomingo117@gmail.com

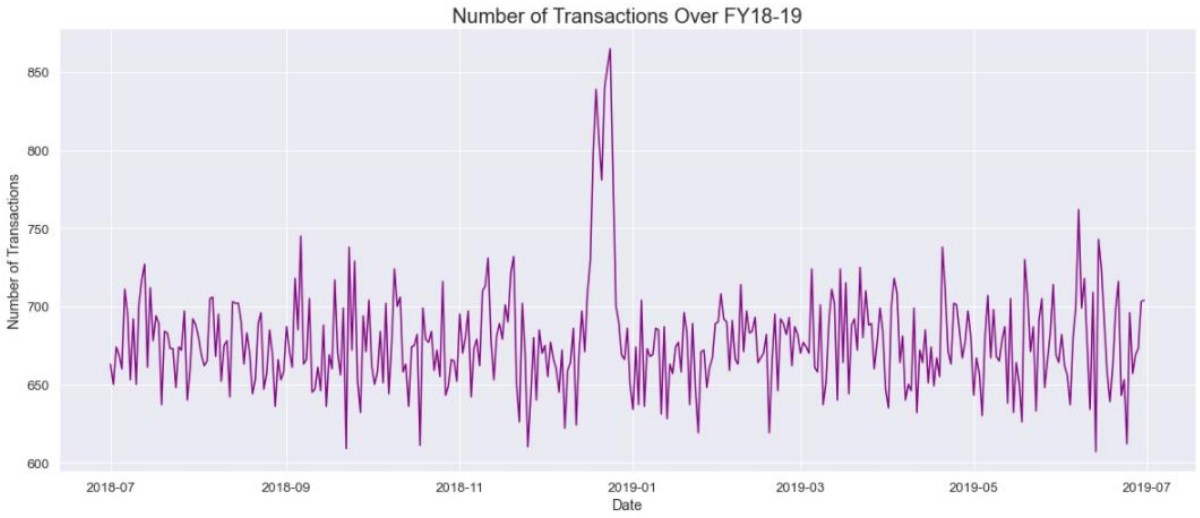
SUMMARY OF CODE / INSIGHTS

FOR RAW CODE: PROCEED TO **PAGE 20**

Number of Transactions Over FY18-19

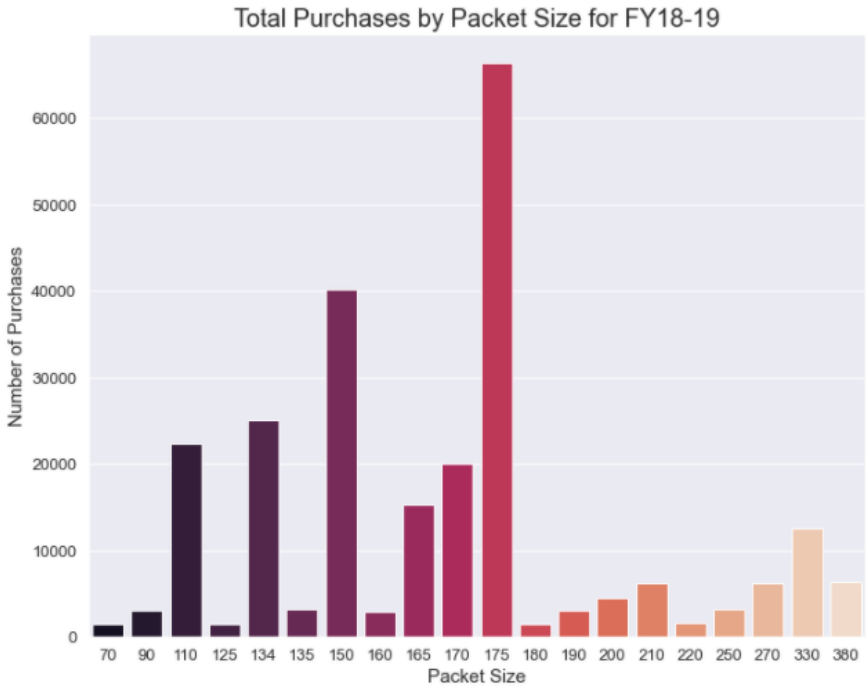
We see a large increase in sales building up to Christmas, and a sudden drop off afterwards back to the regular amount. The data does not include the drop to 0 sales on Christmas day due to closure on the 25th of December.

The data shows no significant outliers besides December sales, which is already explained by the lead up to the Christmas period.



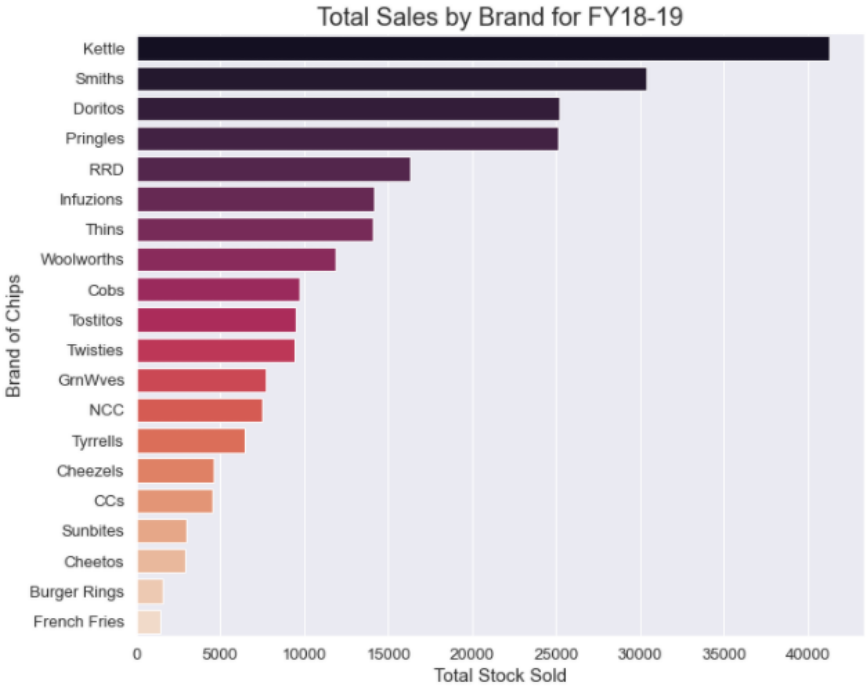
Packet Size Analysis

The smallest packet is 70g, whilst the upper range goes up to 380g packets. The most common packet purchased is 175g. We see a common pattern with the distribution, with 110g, 134g, 150g and 175g topping the purchases. This may be an indication that consumers see these sizes as being the best value for the price. In the figure below, we see that the upper ranges of the packet sizes are not as popular as those stated above, in fact they are significantly lower in popularity.



Brand Analysis

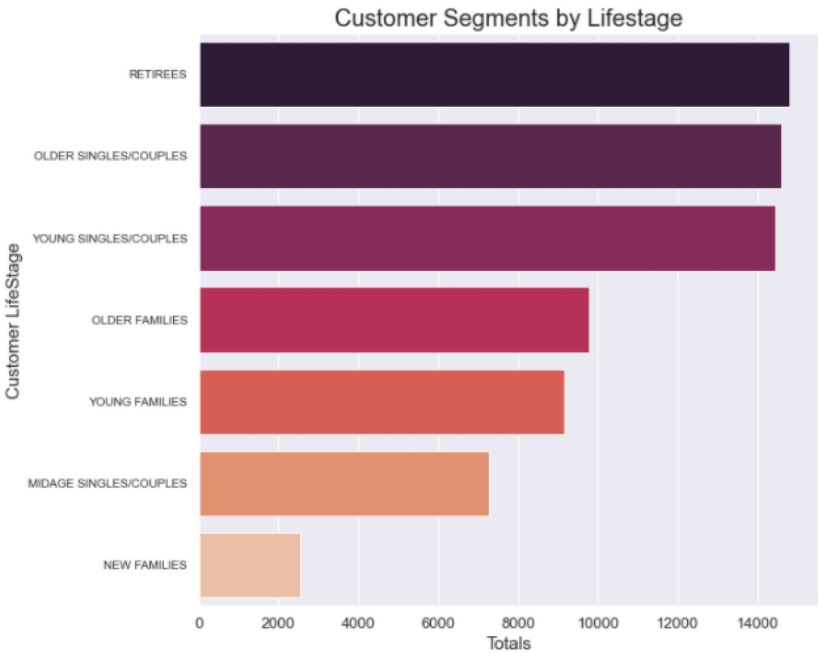
Kettle is the most popular brand of chips, followed by Smiths. Doritos and Pringles are relatively equal in terms of popularity, while Burger Rings and French Fries are the least popular brand of snacks.

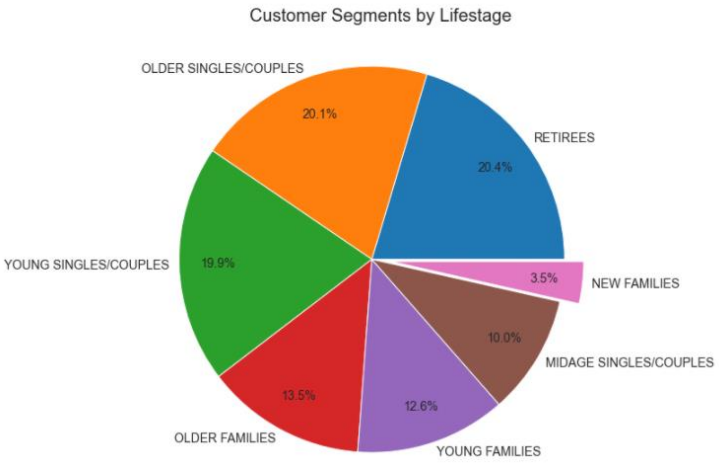


	Stock Sold
Kettle	41288
Smiths	30353
Doritos	25224
Pringles	25102
RRD	16321
Infuzions	14201
Thins	14075
Woolworths	11836
Cobs	9693
Tostitos	9471
Twisties	9454
GrnWves	7740
NCC	7469
Tyrrells	6442
Cheezels	4603
CCs	4551
Sunbites	3008
Cheetos	2927
Burger Rings	1564
French Fries	1418

Customer Segments - Life stage

The business categorizes customer life stage segments into 7 classes. Most customers are either retirees or older/younger singles/couples, being 20.4%, 20.1% and 19.9% respectively, making up 60.4% of the total customer population.

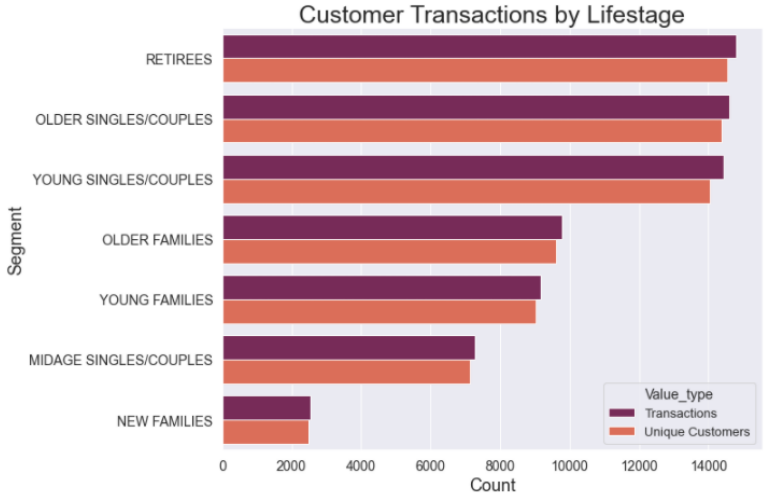


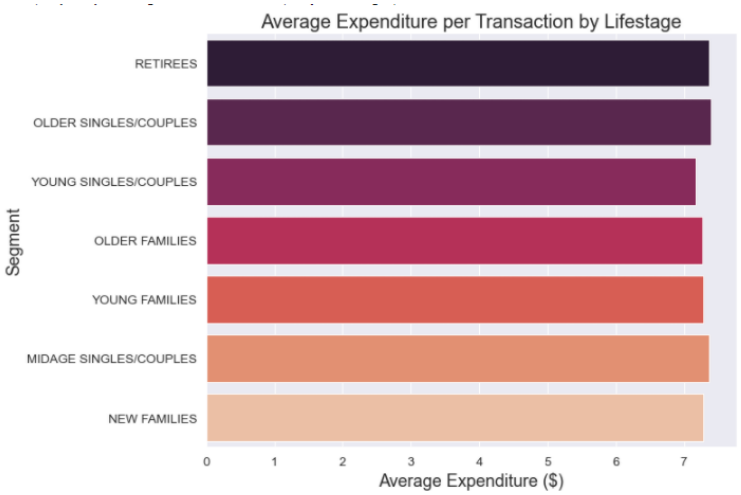


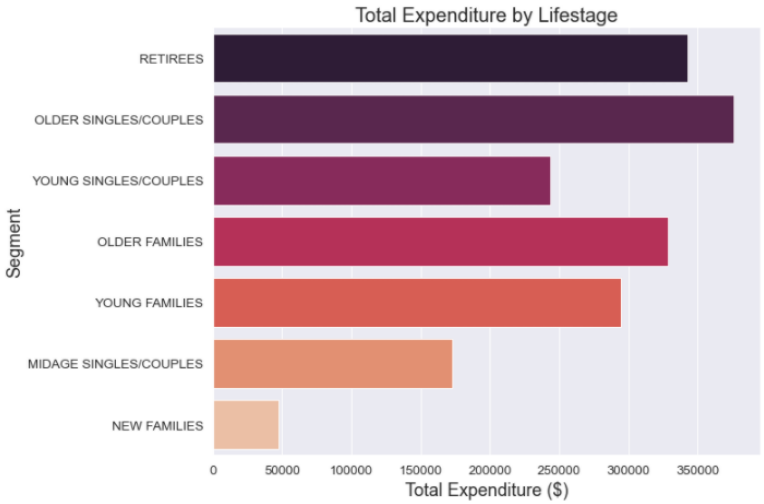
Customer Segments - Life stage (Purchasing Behavior)

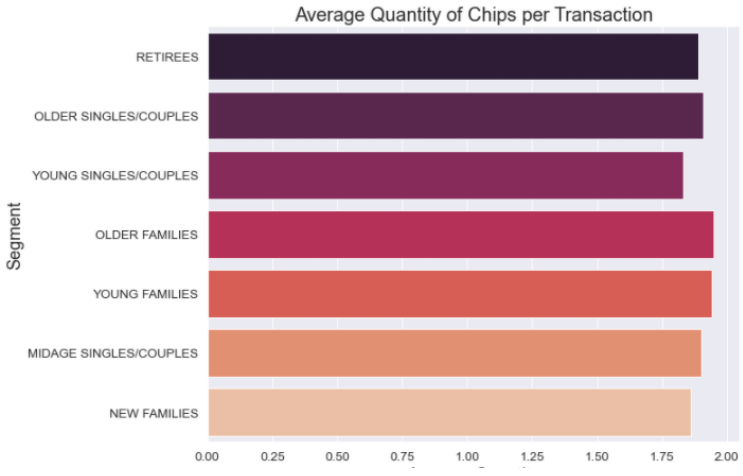
The number of transactions by each lifestage closely follows the population distribution of that segment. In terms of expenditure, all segments spend a very similar amount (+$7), with only slight deviation of each other. Older singles/couples spend the most money on snacks, followed by Retirees and Older Families. Regarding quantities per transaction, all segments average between 1.75 to 2 chip packets per transaction. Older families have bought the most, with new families the least. For all subsets, the average price of each unit sold for all subsets are all between $3.7 and $3.9.

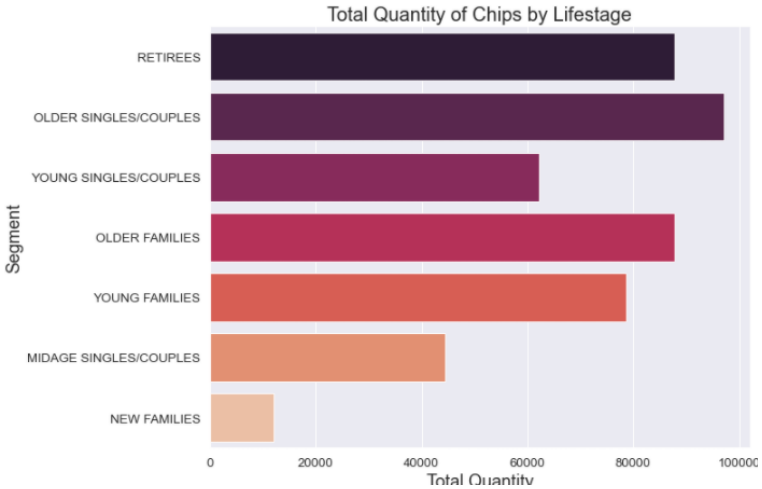
SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY	TOT_QTY	AVG_CHIP_PRICE
RETIREES	14805	14555	7.37	342381.90	1.89	87875	3.89
OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91	97184	3.86
YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83	62298	3.89
OLDER FAMILIES	9780	9630	7.27	328519.90	1.95	87896	3.74
YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94	78577	3.75
MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90	44496	3.87
NEW FAMILIES	2549	2492	7.29	47347.95	1.86	12070	3.91

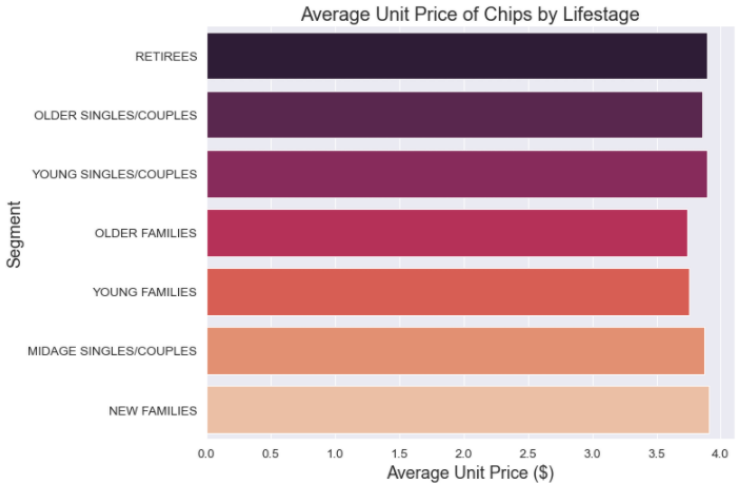




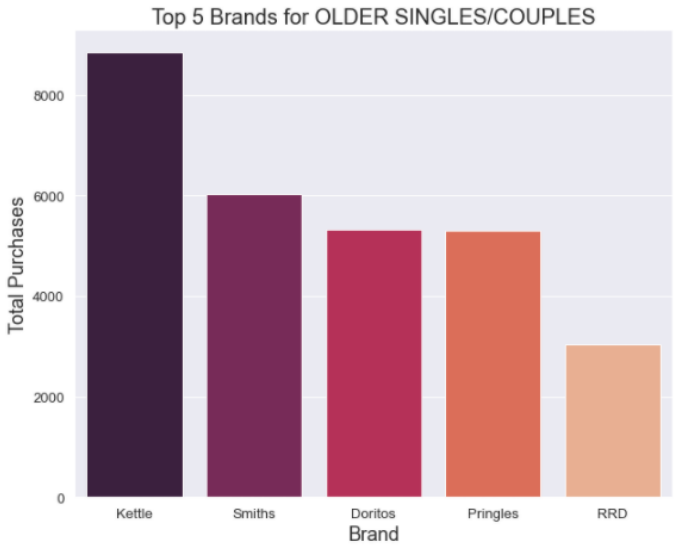




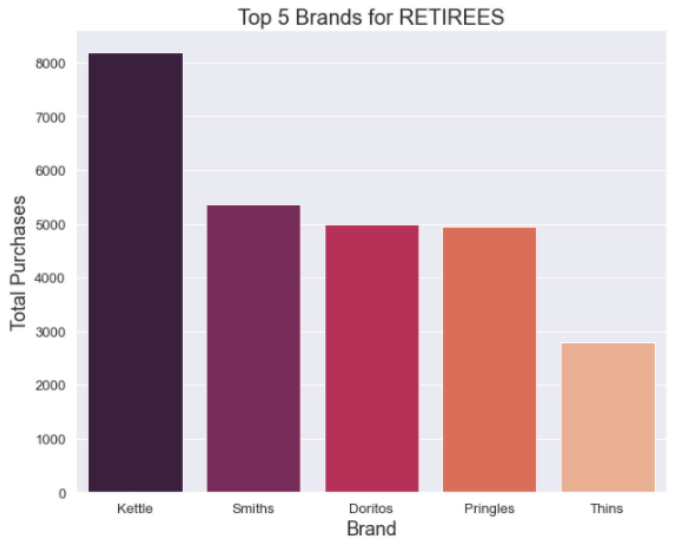




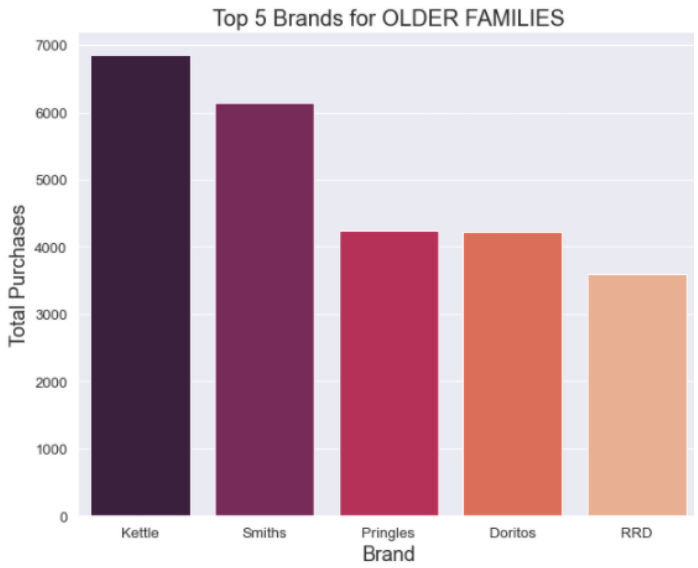
Customer Segments - Life stage (Favorite Brands)
For all lifestage customer segments, the top 5 brands appear to be the same. The top to brands per segment are Kettle and Smiths, followed by either Doritos or Pringles.



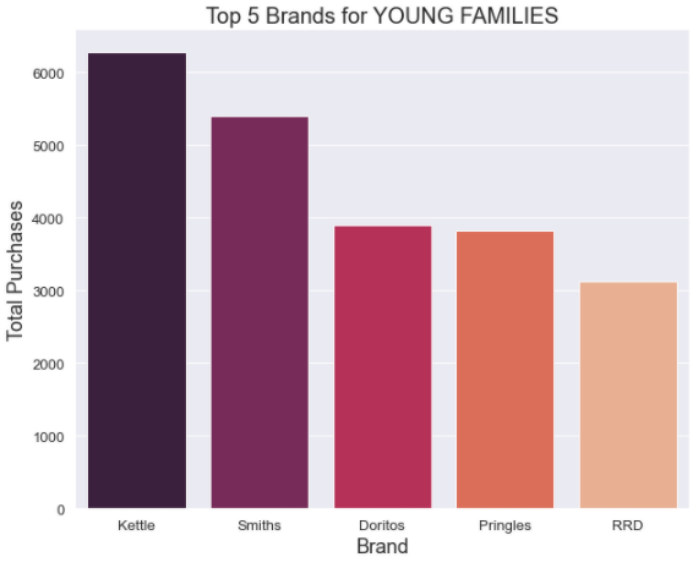
OLDER SINGLES/COUPLES	
Kettle	8847
Smiths	6031
Doritos	5326
Pringles	5307
RRD	3042
Thins	2969
Infuzions	2962
Woolworths	2316
Tostitos	2039
Cobs	2036
Twisties	1949
GrnWves	1603
NCC	1466
Tyrrells	1340
Cheezels	966
CCs	851
Sunbites	584
Cheetos	580
Burger Rings	292
French Fries	286



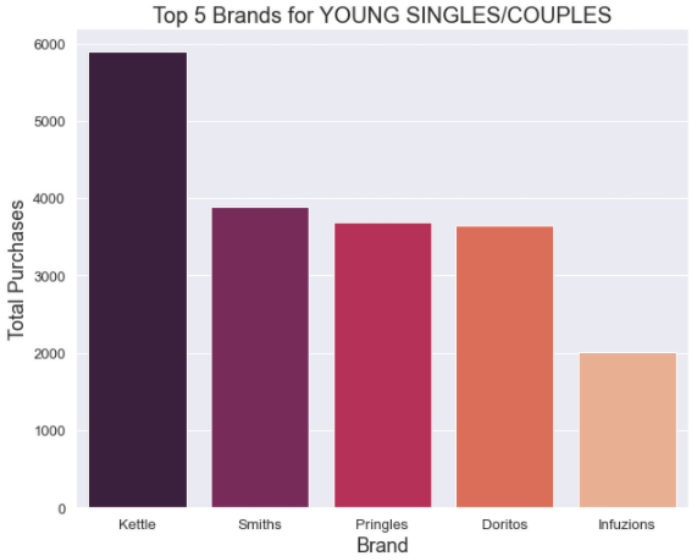
RETIREES	
Kettle	8194
Smiths	5374
Doritos	4987
Pringles	4951
Thins	2792
Infuzions	2719
RRD	2716
Woolworths	1945
Twisties	1890
Cobs	1884
Tostitos	1850
GrnWves	1466
NCC	1286
Tyrrells	1259
Cheezels	867
CCs	741
Sunbites	535
Cheetos	491
Burger Rings	256
French Fries	228



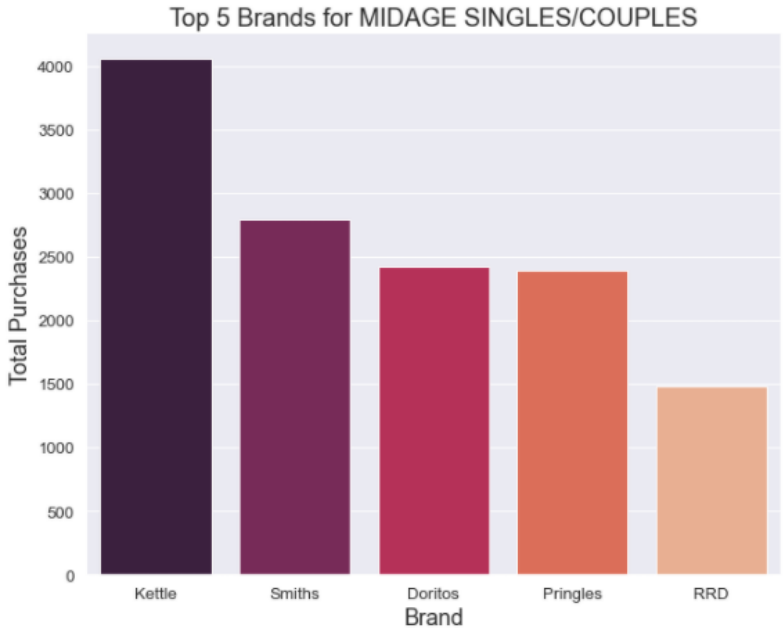
OLDER FAMILIES	
Kettle	6851
Smiths	6138
Pringles	4244
Doritos	4218
RRD	3593
Woolworths	2609
Infuzions	2496
Thins	2475
Twisties	1644
Cobs	1624
NCC	1571
Tostitos	1546
GrnWves	1429
Tyrrells	1093
CCs	941
Cheezels	813
Sunbites	622
Cheetos	615
Burger Rings	353
French Fries	283



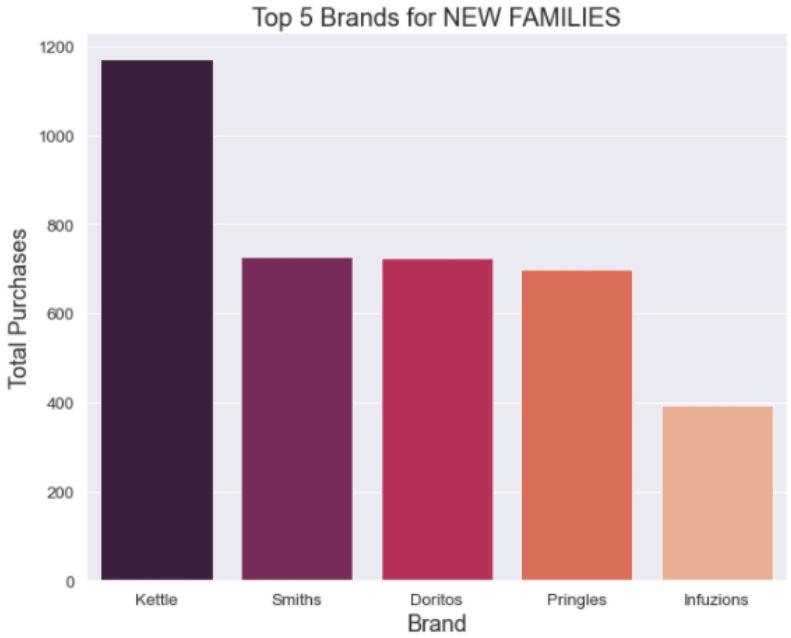
YOUNG FAMILIES	
Kettle	6277
Smiths	5399
Doritos	3894
Pringles	3829
RRD	3129
Infuzions	2215
Woolworths	2211
Thins	2186
Cobs	1504
Tostitos	1467
Twisties	1412
NCC	1367
GrnWves	1221
Tyrrells	997
CCs	898
Cheezels	771
Sunbites	596
Cheetos	550
Burger Rings	293
French Fries	278



YOUNG SINGLES/COUPLES	
Kettle	5893
Smiths	3893
Pringles	3684
Doritos	3650
Infuzions	2013
RRD	2008
Thins	1959
Woolworths	1447
Cobs	1396
Twisties	1395
Tostitos	1368
GrnWves	1076
Tyrrells	955
NCC	927
Cheezels	613
CCs	594
Cheetos	364
Sunbites	361
French Fries	194
Burger Rings	178



MIDAGE SINGLES/COUPLES	
Kettle	4055
Smiths	2790
Doritos	2423
Pringles	2389
RRD	1478
Infuzions	1403
Thins	1316
Woolworths	1032
Cobs	961
Twisties	931
Tostitos	924
GrnWves	732
NCC	694
Tyrrells	611
Cheezels	443
CCs	433
Cheetos	265
Sunbites	249
Burger Rings	152
French Fries	117



NEW FAMILIES	
Kettle	1171
Smiths	727
Doritos	726
Pringles	698
Infuzions	393
Thins	378
RRD	355
Cobs	288
Tostitos	277
Woolworths	276
Twisties	233
GrnWves	213
Tyrrells	187
NCC	158
Cheezels	129
CCs	93
Cheetos	62
Sunbites	61
Burger Rings	40
French Fries	32

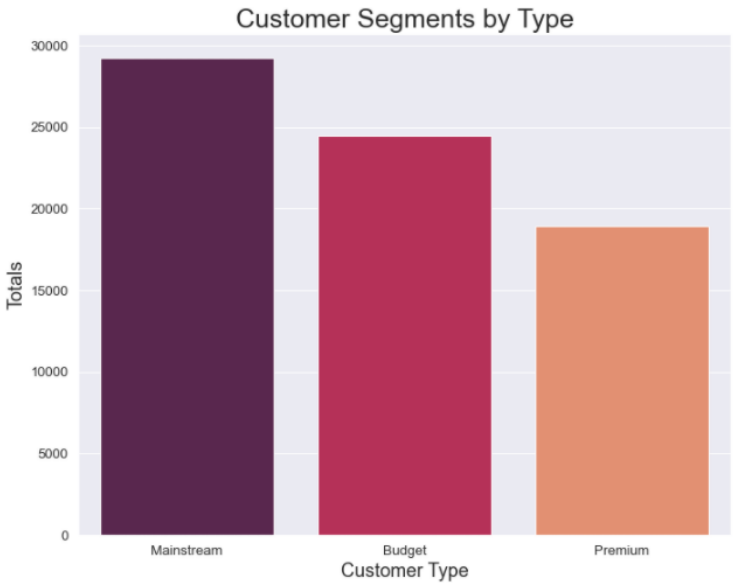
Customer Segments – Premium or Type

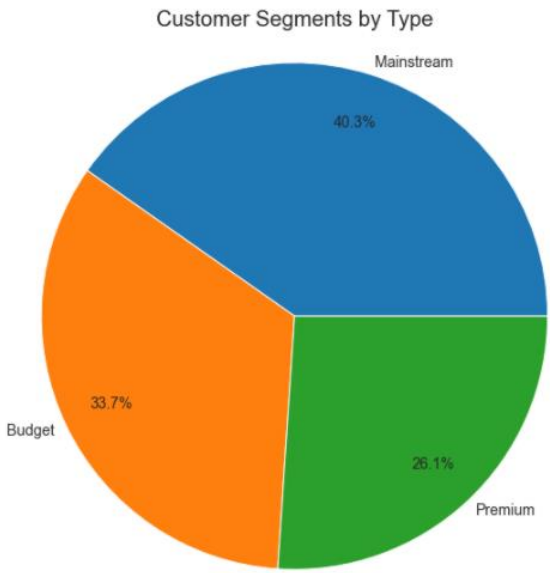
There are 3 Customer Premium Types:

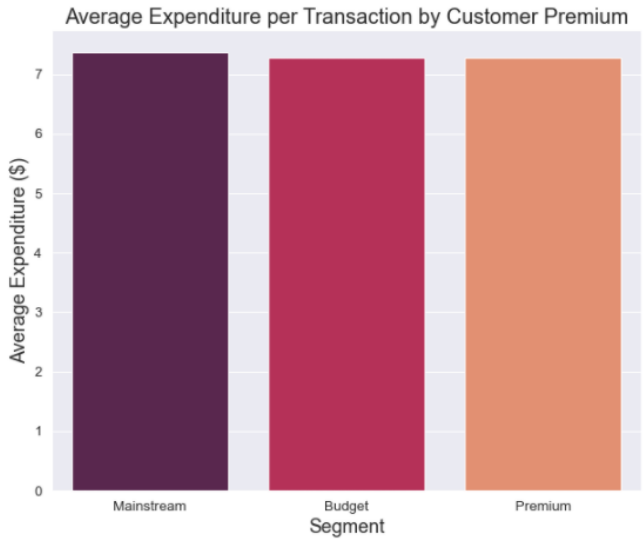
- Mainstream
- Budget
- Premium

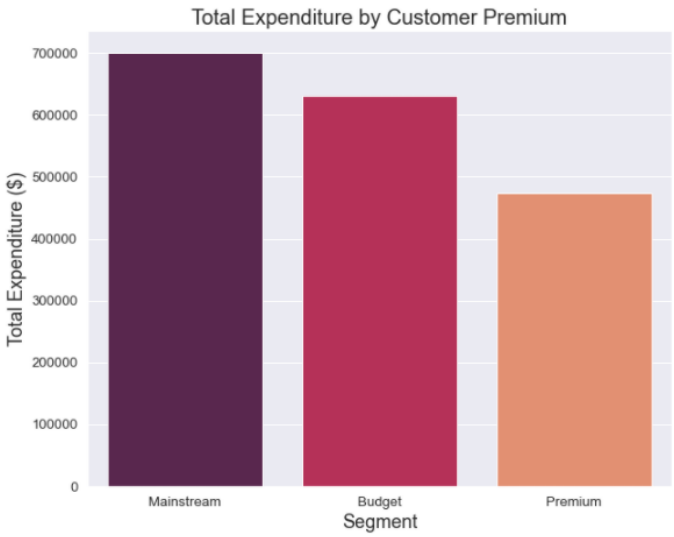
Mainstream makes up majority of customers.

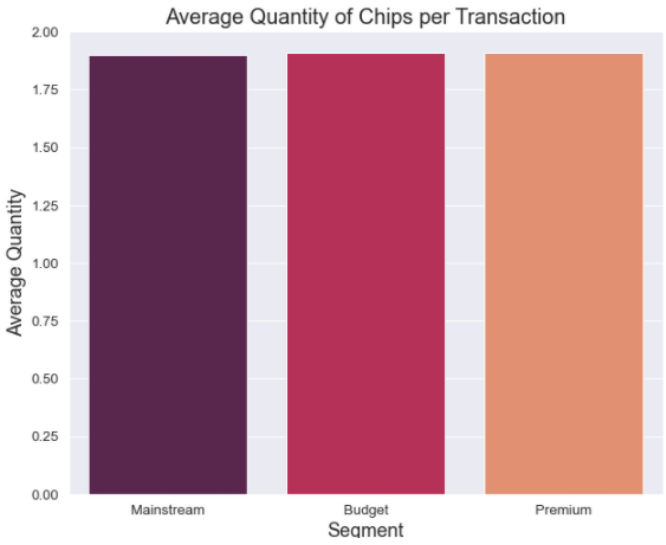
Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	AVG_CHIP_PRICE
Mainstream	29245	28734	7.37	700859.70	1.90	180779	3.87
Budget	24470	24006	7.28	631402.65	1.91	165772	3.80
Premium	18922	18547	7.28	472905.45	1.91	123845	3.81

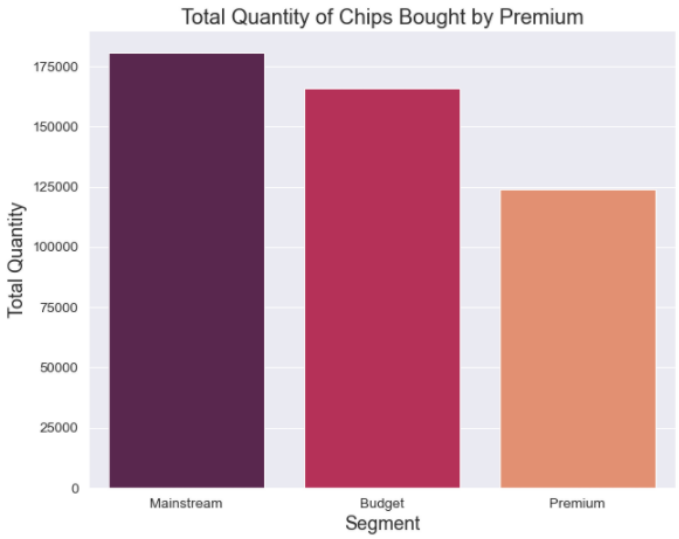


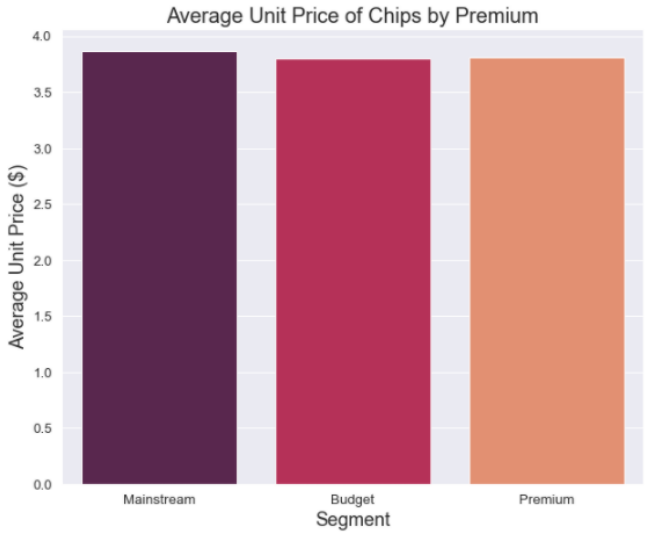


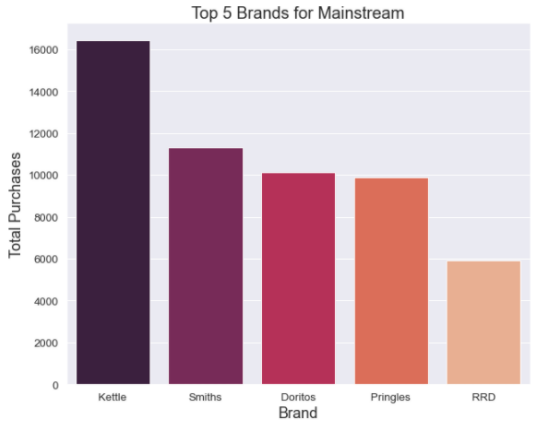




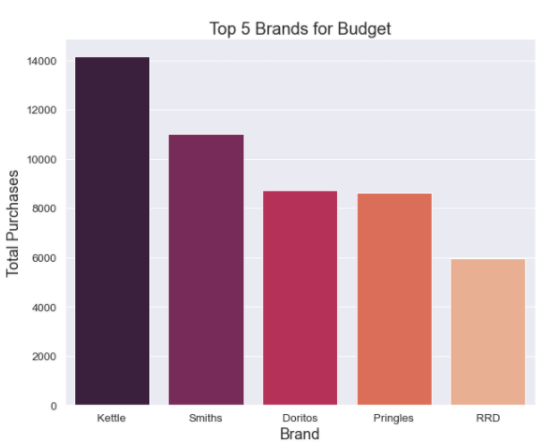




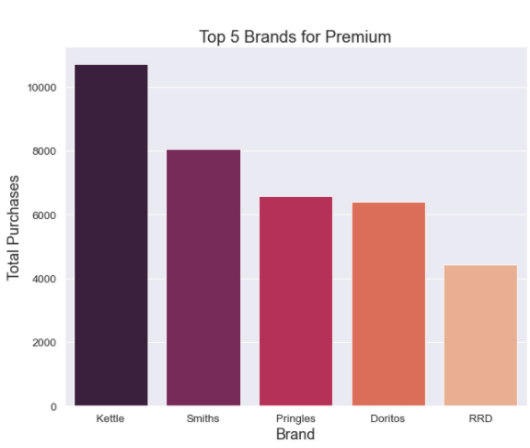




Mainstream	
Kettle	16423
Smiths	11300
Doritos	10114
Pringles	9903
RRD	5924
Infuzions	5550
Thins	5436
Woolworths	4130
Cobs	3889
Twisties	3785
Tostitos	3737
GrnWves	3037
NCC	2657
Tyrrells	2583
Cheezels	1735
CCs	1631
Cheetos	1111
Sunbites	1042
Burger Rings	548
French Fries	507



Budget	
Kettle	14154
Smiths	11008
Doritos	8718
Pringles	8620
RRD	5970
Thins	4931
Infuzions	4922
Woolworths	4444
Cobs	3274
Tostitos	3236
Twisties	3229
NCC	2785
GrnWves	2656
Tyrrells	2195
CCs	1679
Cheezels	1625
Sunbites	1146
Cheetos	1051
Burger Rings	579
French Fries	539



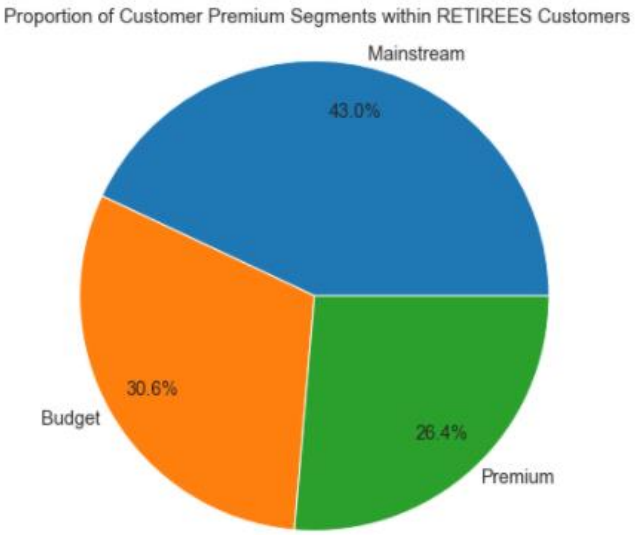
Premium	
Kettle	10711
Smiths	8044
Pringles	6579
Doritos	6392
RRD	4427
Infuzions	3729
Thins	3708
Woolworths	3262
Cobs	2530
Tostitos	2498
Twisties	2440
GrnWves	2047
NCC	2027
Tyrrells	1664
Cheezels	1242
CCs	1241
Sunbites	820
Cheetos	765
Burger Rings	437
French Fries	372

OBSERVATIONS (CUSTMOMER PREMIUM TOP BRANDS)

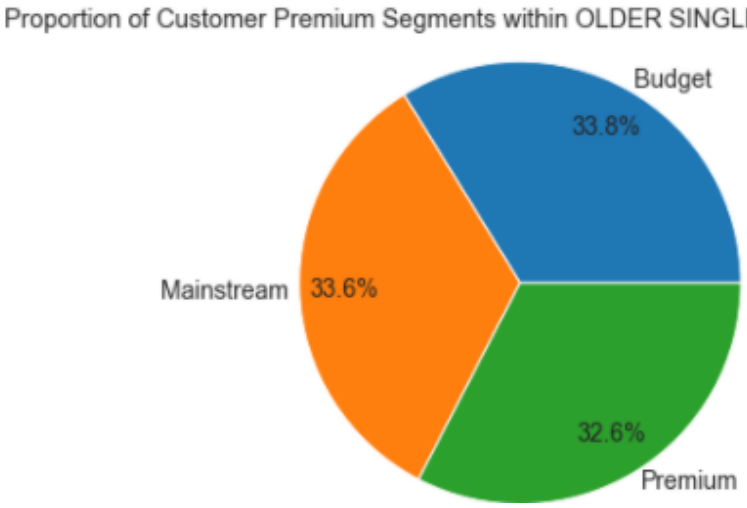
For all customer premium segments, the top 5 brands are all homogeneous, including order. The only observable difference is slight variation between ranks 3 & 4 (Doritos and pringles), and 5 & 6 (RRD and Infuzions).

Customer Segments – Proportion of Lifestage Segments within Premium Segments

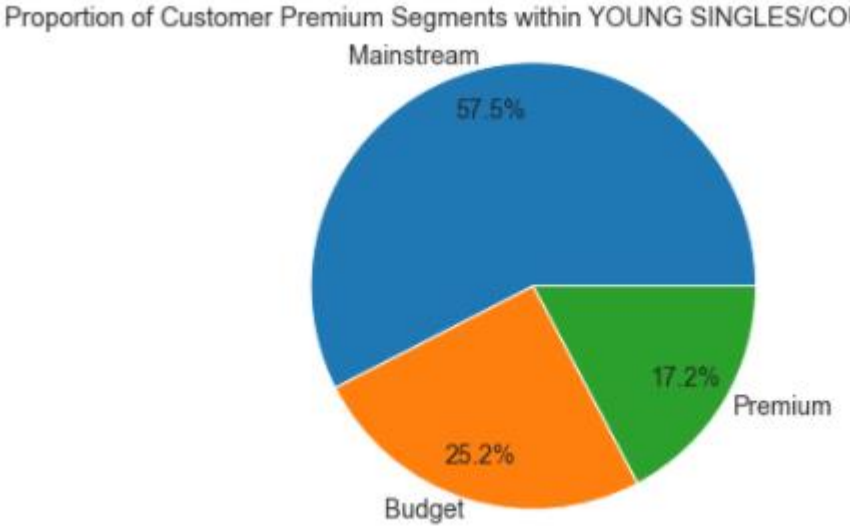
The following charts illustrate the proportion of lifestage segments make up the customer premium segments, and visa versa.



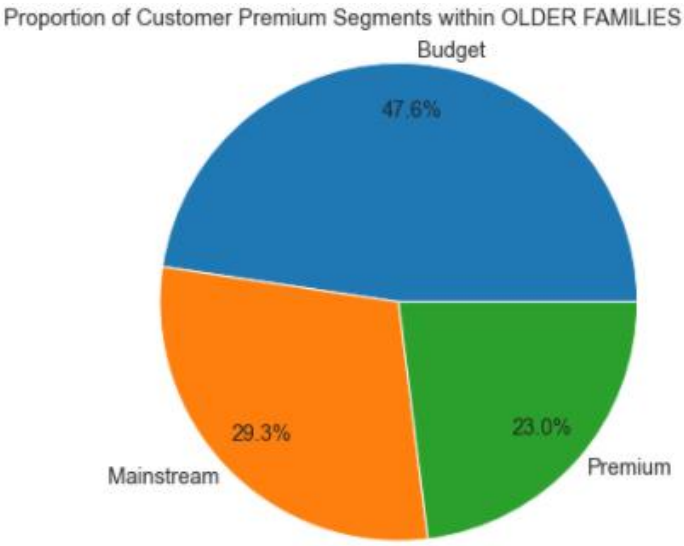
RETIREES	Count
Mainstream	19970
Budget	14225
Premium	12236



OLDER SINGLES/COUPLES	Count
Budget	17172
Mainstream	17060
Premium	16560

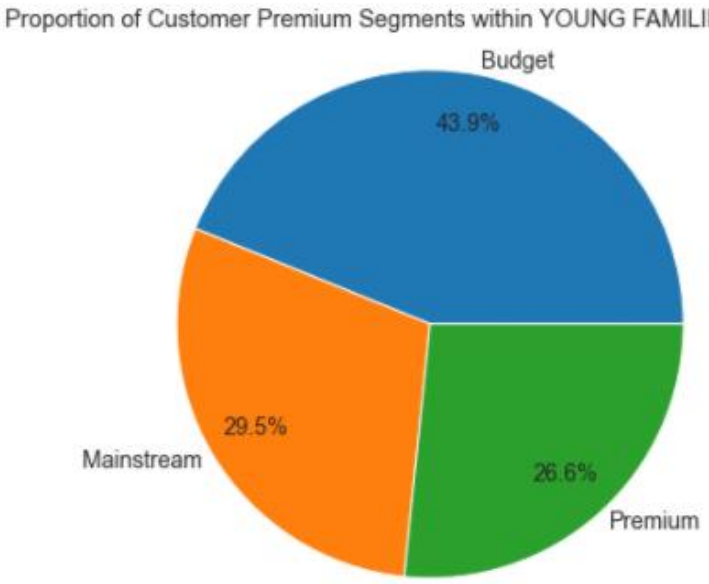


YOUNG SINGLES/COUPLES	Count
Mainstream	19544
Budget	8572
Premium	5852



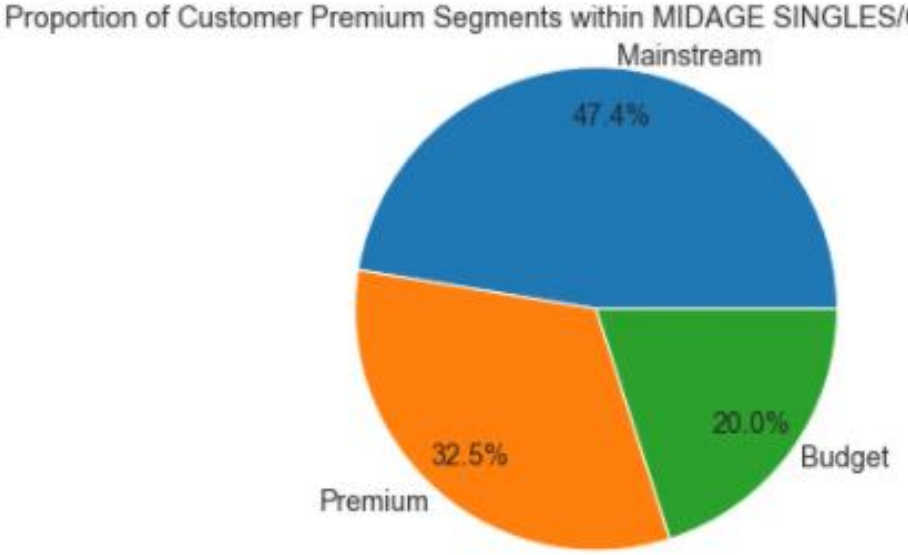
OLDER FAMILIES

	Count
Budget	21514
Mainstream	13241
Premium	10403



YOUNG FAMILIES

	Count
Budget	17763
Mainstream	11947
Premium	10784



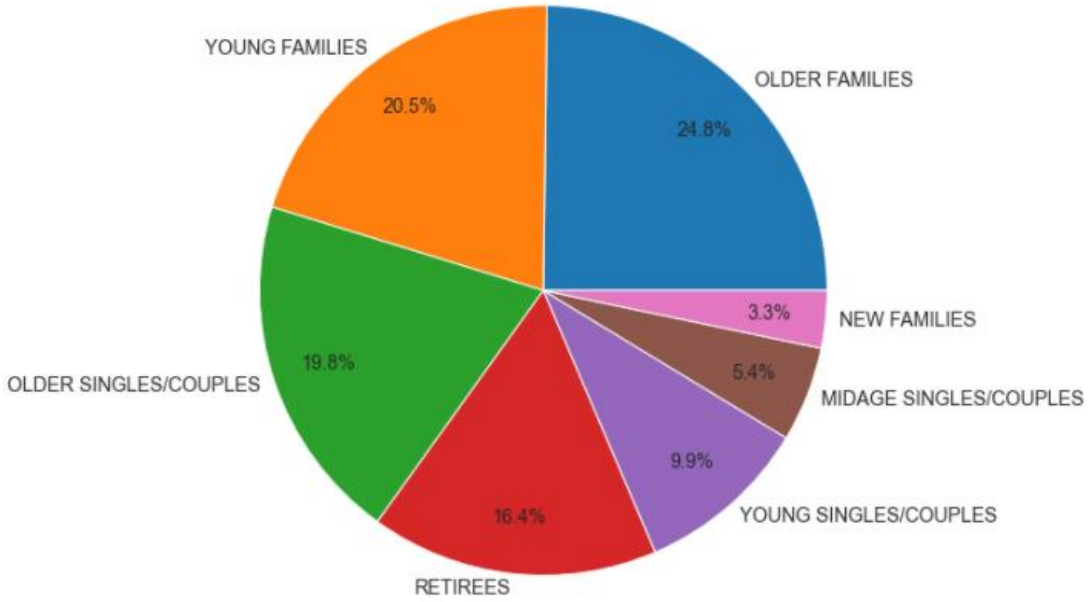
MIDAGE SINGLES/COUPLES

	Count
Mainstream	11095
Premium	7612
Budget	4691

Budget

	Count
OLDER FAMILIES	21514
YOUNG FAMILIES	17763
OLDER SINGLES/COUPLES	17172
RETIREES	14225
YOUNG SINGLES/COUPLES	8572
MIDAGE SINGLES/COUPLES	4691
NEW FAMILIES	2824

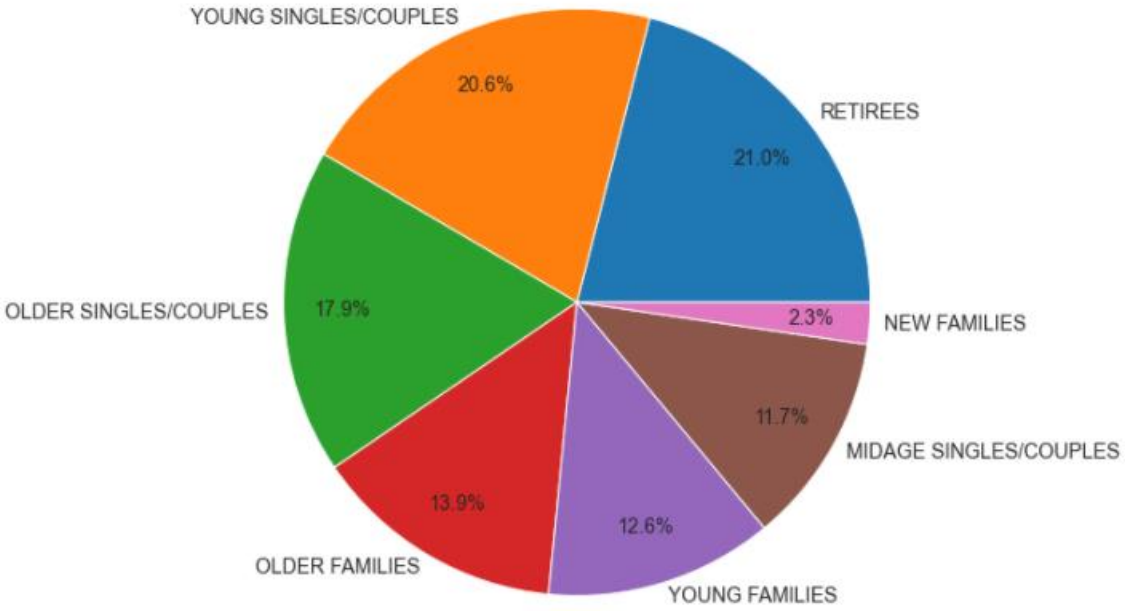
Proportion of Customer Lifestages within Budget Customers



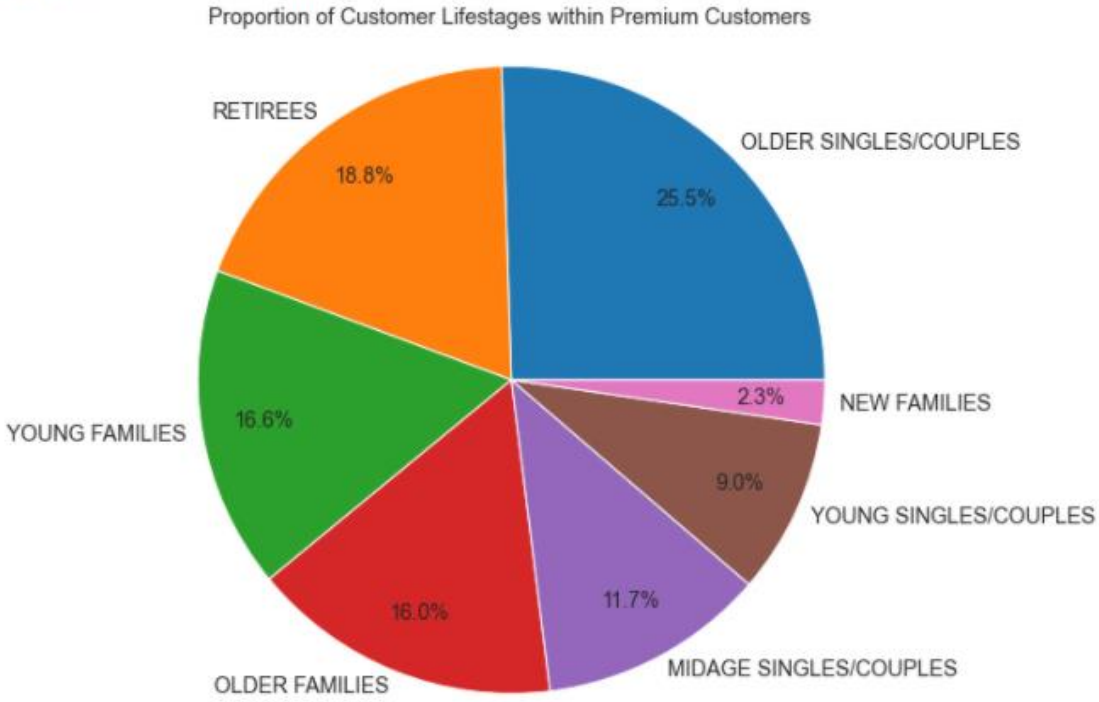
Mainstream

	Count
RETIREES	19970
YOUNG SINGLES/COUPLES	19544
OLDER SINGLES/COUPLES	17060
OLDER FAMILIES	13241
YOUNG FAMILIES	11947
MIDAGE SINGLES/COUPLES	11095
NEW FAMILIES	2185

Proportion of Customer Lifestages within Mainstream Customers



Premium	Count
OLDER SINGLES/COUPLES	16560
RETIREES	12236
YOUNG FAMILIES	10784
OLDER FAMILIES	10403
MIDAGE SINGLES/COUPLES	7612
YOUNG SINGLES/COUPLES	5852
NEW FAMILIES	1488

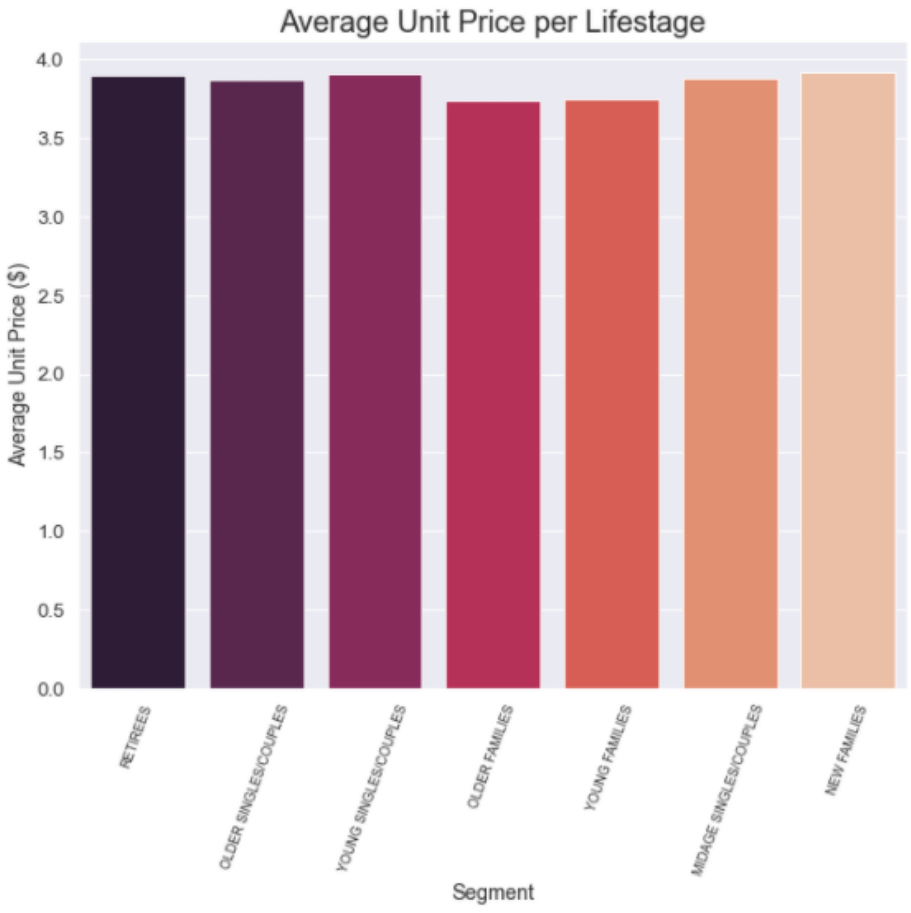


The We can further dive into their purchasing behavior by looking at their average unit price per transaction.

Customer Segments – Average Unit Price Analysis Per Lifestage

Looking at the data, we see that younger singles/couples and new families are more willing to spend on more premium ranges of snacks.

SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY	TOT_QTY	AVG_CHIP_PRICE	PER_UNIT_PRICE_TOTAL
RETIREES	14805	14555	7.37	342381.90	1.89	87875	3.89	3.896238
OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91	97184	3.86	3.869093
YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83	62298	3.89	3.912684
OLDER FAMILIES	9780	9630	7.27	328519.90	1.95	87896	3.74	3.737598
YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94	78577	3.75	3.749544
MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90	44496	3.87	3.877288
NEW FAMILIES	2549	2492	7.29	47347.95	1.86	12070	3.91	3.922780



Customer Segments – Average Unit Price Analysis Per Premium

Looking at the data, we see that Mainstream customers are more willing to spend on more premium ranges of snacks.

Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	AVG_CHIP_PRICE	PER_UNIT_PRICE_TOTAL
Mainstream	29245	28734	7.37	700859.70	1.90	180779	3.87	3.876887
Budget	24470	24006	7.28	631402.65	1.91	165772	3.80	3.808862
Premium	18922	18547	7.28	472905.45	1.91	123845	3.81	3.818527

Total Observations: Average Unit Price

Based on the data shown, young singles/couples who are registered as the mainstream premiums are willing to pay more per unit compared to other segments. Reasons for this could lie with a more health-oriented purchasing behavior. This is further backed up by there being fewer purchases by premium middle-aged and young singles/couples' customers, compared to mainstream segments of the same kind.

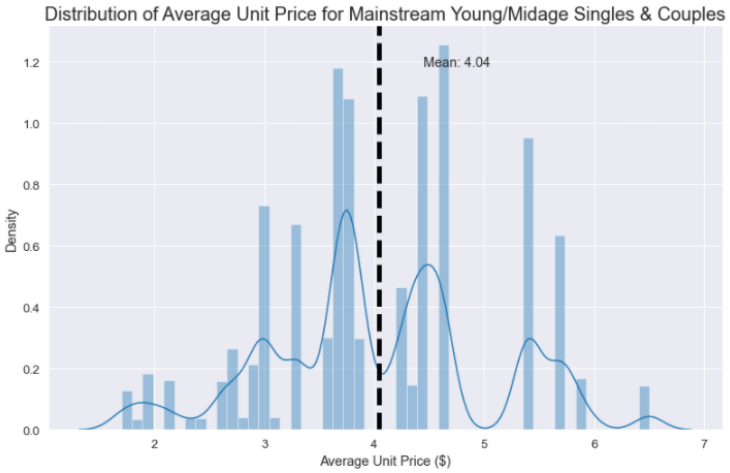
We must check if this difference in unit price is statistically significant for these segments. We can do this by performing a t-test of significance.

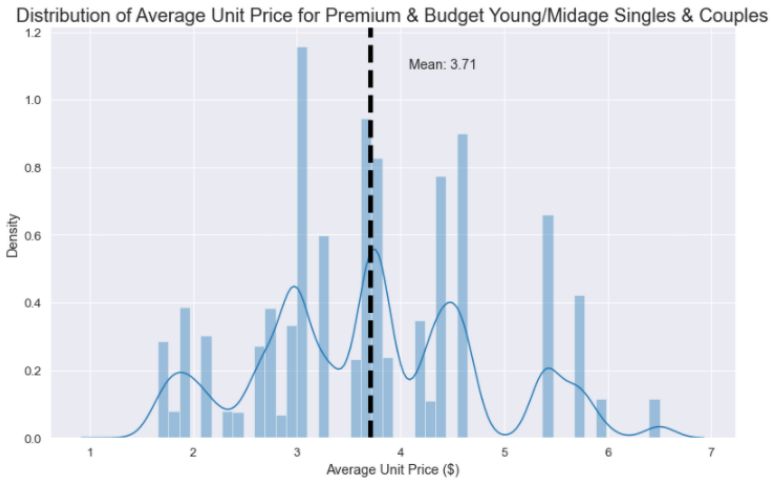
T-Test: Average Unit Price

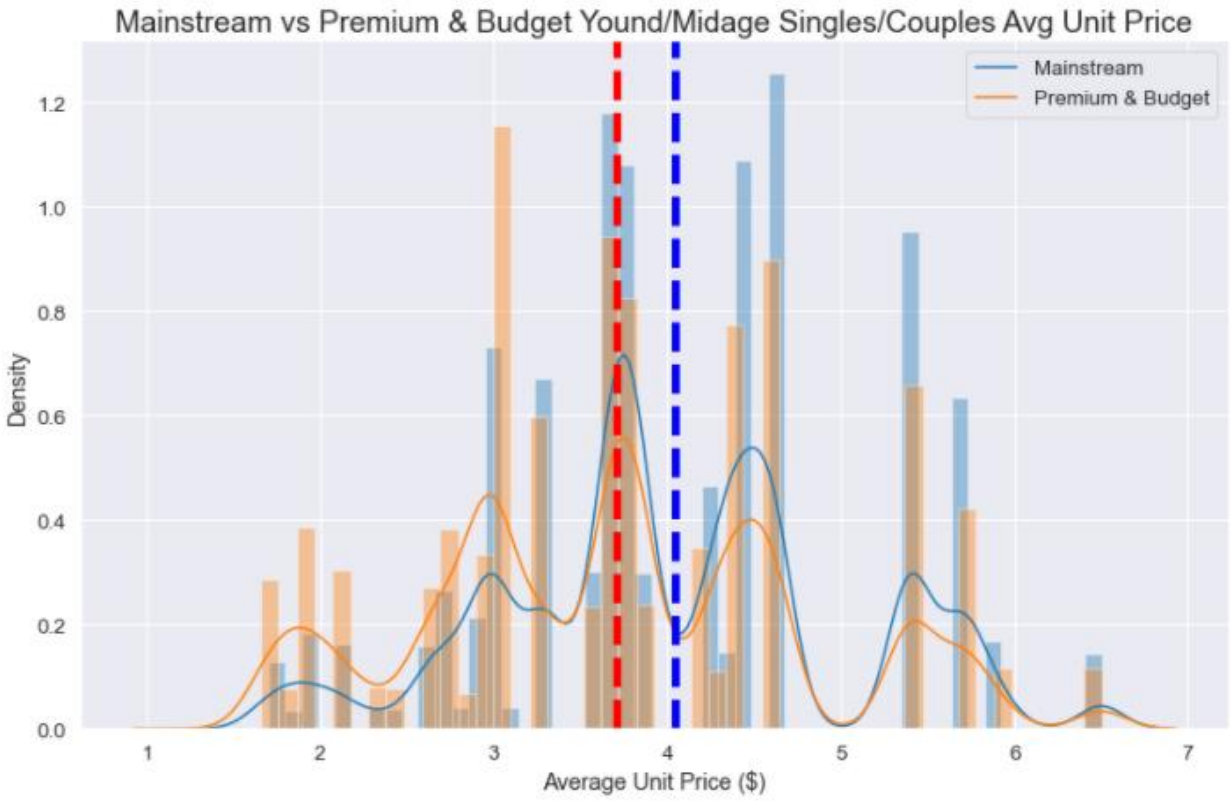
The following distributions show the distribution for Mainstream young and middle-aged singles/couples compared to other premiums of the same lifestage. When performing the t test, we get a p-value of **0.015** (with our limit being 0.05). This suggests that the unit price for mainstream, young and middle-aged singles and couples are significantly higher than that of budget or premium segments of the same lifestage.

T-Test Results:

Statistics	2.445
p-value	0.015
Result	Different distributions (reject h0)







Further Insights: Total Expenditure

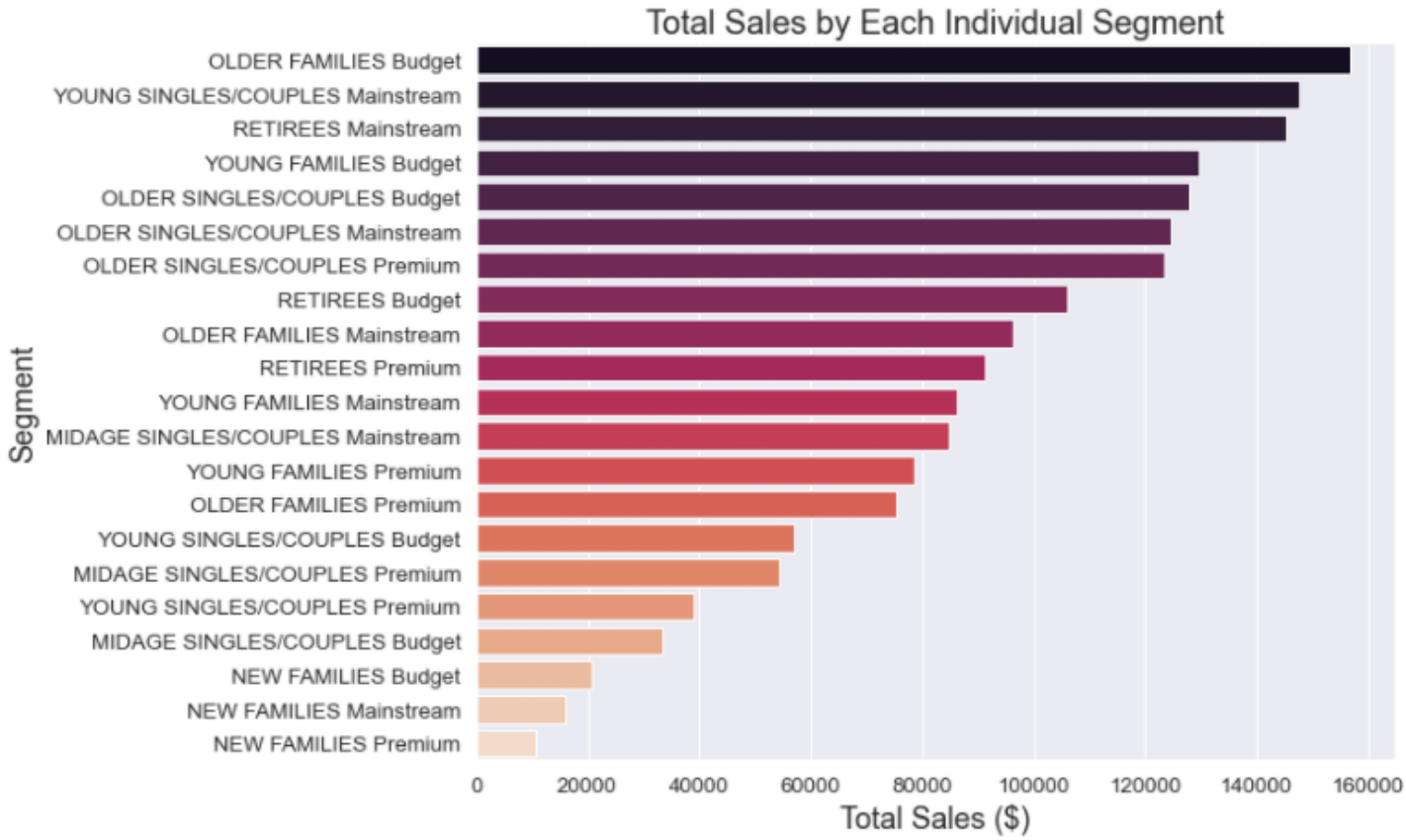
When looking at the total expenditure by each individual segment, the highest paying segments include:

- Older Families with Budget Premiums
- Young singles/couples with Mainstream Premiums
- Retirees with Mainstream Premiums

Recommendations:

- Target segments which provide the most sales.
- Cater to these segments will further increase sales. We can do this by looking at their favorite brands of snacks.

Segment	Total_sales
OLDER FAMILIES Budget	156863.8
YOUNG SINGLES/COUPLES Mainstream	147582.2
RETIREES Mainstream	145169.0
YOUNG FAMILIES Budget	129718.0
OLDER SINGLES/COUPLES Budget	127833.6
OLDER SINGLES/COUPLES Mainstream	124642.8
OLDER SINGLES/COUPLES Premium	123537.6
RETIREES Budget	105916.3
OLDER FAMILIES Mainstream	96413.6
RETIREES Premium	91296.6
YOUNG FAMILIES Mainstream	86338.2
MIDAGE SINGLES/COUPLES Mainstream	84734.2
YOUNG FAMILIES Premium	78571.7
OLDER FAMILIES Premium	75242.6
YOUNG SINGLES/COUPLES Budget	57117.9
MIDAGE SINGLES/COUPLES Premium	54443.8
YOUNG SINGLES/COUPLES Premium	39052.3
MIDAGE SINGLES/COUPLES Budget	33345.7
NEW FAMILIES Budget	20607.4
NEW FAMILIES Mainstream	15979.7
NEW FAMILIES Premium	10760.8

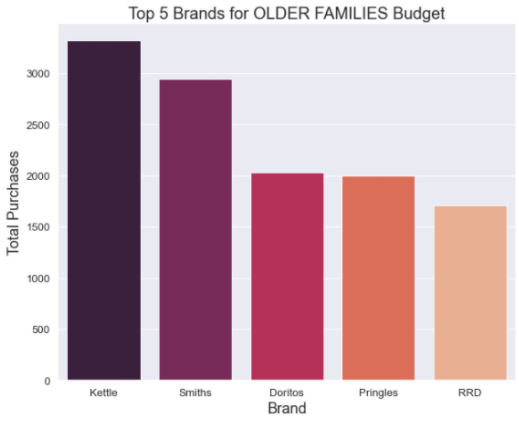


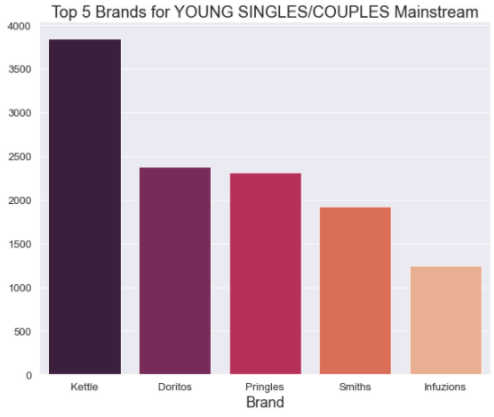
Further Insights: Top Brands by Top Spending Segments

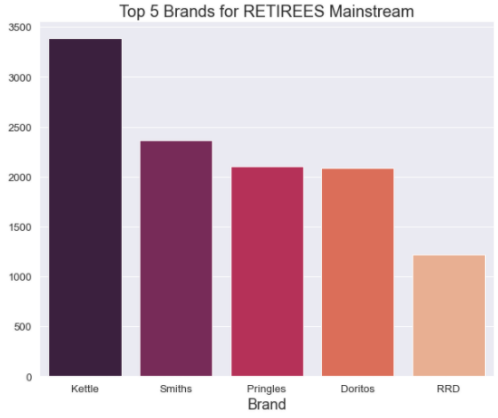
Based on the expenditure analysis, the top 3 segments have the following favorite brands:

	Older Families | Budget	Young Singles/Couples | Mainstream	Retirees | Mainstream
1.	Kettle	Kettle	Kettle
2.	Smiths	Smiths	Smiths
3.	Doritos	Doritos	Pringles
4.	Pringles	Pringles	Doritos
5.	Red Rock Deli	Red Rock Deli	Red Rock Deli

Recommendations: Run promotions on these brands to attract higher sales quantities.








```
----------------------------------------
OLDER FAMILIES Budget Top Brands Ranked
----------------------------------------
Kettle          3320
Smiths          2948
Doritos         2032
Pringles        1996
RRD             1708
Woolworths      1213
Infuzions       1185
Thins           1171
Twisties         810
Cobs             760
NCC              741
Tostitos         705
GrnWves          671
Tyrrells         489
CCs              451
Cheezels         427
Sunbites         305
Cheetos          281
Burger Rings     159
French Fries     142
```

```
----------------------------------------
YOUNG SINGLES/COUPLES Mainstream Top Brands Ranked
----------------------------------------
Kettle          3844
Doritos         2379
Pringles        2315
Smiths          1921
Infuzions       1250
Thins           1166
Twisties         900
Tostitos         890
RRD              875
Cobs             864
GrnWves          646
Tyrrells         619
Woolworths       479
NCC              394
Cheezels         346
CCs              222
Cheetos          166
Sunbites         128
French Fries      78
Burger Rings      62
```

```
----------------------------------------
RETIREES Mainstream Top Brands Ranked
----------------------------------------
Kettle          3386
Smiths          2367
Pringles        2103
Doritos         2089
RRD             1223
Thins           1199
Infuzions       1182
Woolworths       902
Twisties         802
Cobs             776
Tostitos         739
GrnWves          667
NCC              587
Tyrrells         514
Cheezels         382
CCs              355
Sunbites         243
Cheetos          236
Burger Rings     122
French Fries      96
```

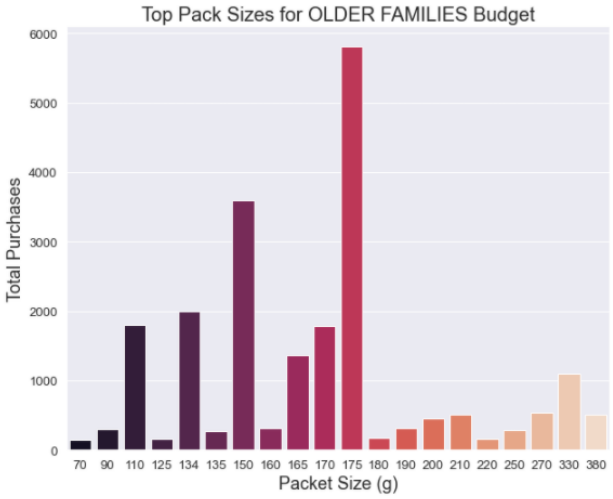
Further Insights: Top Unit Sizes per Top Spending Segments

Across the top spending segments, we want to look at the most popular pack sizes. This will give an idea on how to handle stock management.

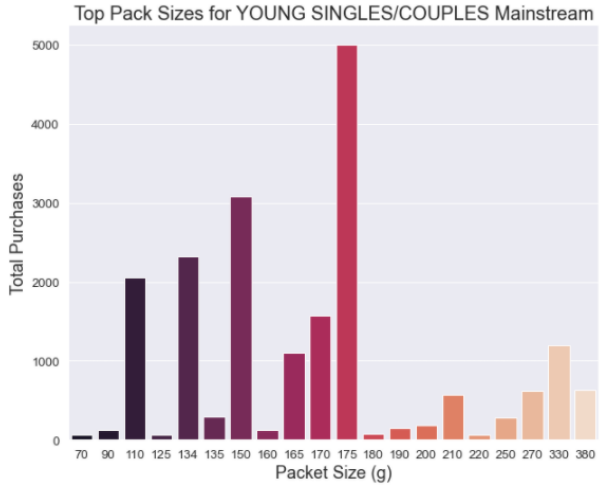
Across the top 3 segments, the most popular pack sizes are homogeneous. The sizes are:

	Older Families | Budget	Young Singles/Couples | Mainstream	Retirees | Mainstream
1.	175g	175g	175g
2.	150g	150g	150g
3.	134g	134g	134g

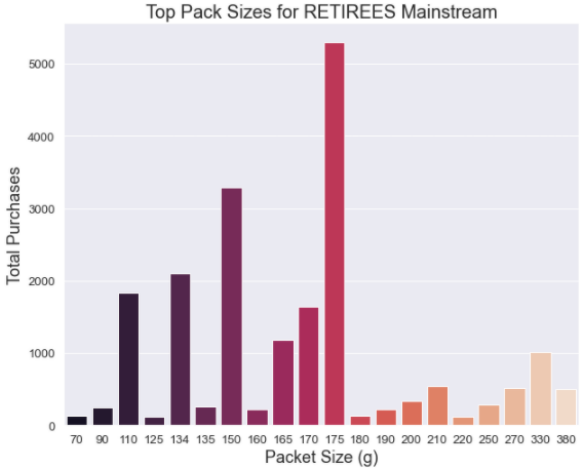
Recommendations: Prioritize these sizes when replenishing stock and negotiating with suppliers.




```
----------------------------------------
OLDER FAMILIES Budget Favorite Chip Sizes
----------------------------------------
175    5808
150    3588
134    1996
110    1803
170    1786
165    1358
330    1092
270     532
380     510
210     505
200     448
190     312
160     306
90      305
250     278
135     268
180     166
220     159
125     152
70      142
```




```
----------------------------------------
YOUNG SINGLES/COUPLES Mainstream Favorite Chip Sizes
----------------------------------------
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
```




```
----------------------------------------
RETIREES Mainstream Favorite Chip Sizes
----------------------------------------
175    5295
150    3290
134    2103
110    1829
170    1636
165    1182
330    1010
210     540
270     514
380     497
200     342
250     288
135     263
90      243
160     221
190     218
70      129
180     127
220     122
125     121
```

PROJECT CODE (JUPYTER NOTEBOOK)

Language Used: Python & SQL

In [1]:

```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
from scipy.ndimage.filters import gaussian_filter1d
import pandasql as ps
import warnings
from pylab import rcParams
from scipy.stats import f_oneway
from scipy.stats import ttest_ind
```

In [2]:

```
# Read File
file = r'C:\Users\Joel\Dropbox\Vitrual Internships\Quantium\Task 1\Data\QVI_transaction_data.csv'
data = pd.read_csv(file, index_col = None)
data.head()
```

Out[2]:

	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

Data Cleaning and Feature Engineering

In [3]:

```
# Extract packet size from product name
data['PACKET_SIZE'] = data.PROD_NAME.str.extract('(\d+)')
data['PACKET_SIZE'] = data['PACKET_SIZE'].astype(str).astype(int)

# Extract brand name from product name
data['BRAND_NAME'] = data['PROD_NAME'].str.split().str.get(0)

# Extract real date from excel form date
data['real_date'] = pd.TimedeltaIndex(data['DATE'], unit = 'd') + dt.datetime(1899,12,30)

# Only contain records which are chips (not salsa)
data = data[data['PROD_NAME'].str.contains('Salsa') == False]
data
```

Out[3]:

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

246742 rows × 11 columns

Check for Nulls

In [4]:

```
# Check nulls
data.isnull().sum()
```

Out[4]:

```
DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
PACKET_SIZE       0
BRAND_NAME        0
real_date         0
dtype: int64
```

In [5]:

```
# Create clean df

df = data[['real_date', 'STORE_NBR','LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NAME', 'BRAND_NAM
E', 'PACKET_SIZE', 'PROD_QTY', 'TOT_SALES']]
df.sort_values(by = ['real_date'], inplace = True, ignore_index = True)
df
```

```
<ipython-input-5-5a071f63a699>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.sort_values(by = ['real_date'], inplace = True, ignore_index = True)
```

Out[5]:

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAME
0	2018-07-01	9	9341	8808	Smiths Thinly Cut Roast Chicken 175g	Smiths
1	2018-07-01	86	86016	84237	Red Rock Deli Sp Salt & Truffle 150G	Red
2	2018-07-01	129	129046	132474	Smith Crinkle Cut Mac N Cheese 150g	Smith
3	2018-07-01	58	58072	53145	Pringles Sthrn FriedChicken 134g	Pringles
4	2018-07-01	97	97164	97311	WW Crinkle Cut Chicken 175g	WW
...	...	...	...	...	...	...
246737	2019-06-30	91	91076	89519	Thins Chips Seasonedchicken 175g	Thins
246738	2019-06-30	84	84116	83704	Doritos Corn Chips Nacho Cheese 170g	Doritos
246739	2019-06-30	24	24115	20917	Smiths Crinkle Cut Chips Chs&Onion170g	Smiths
246740	2019-06-30	199	199117	198068	Doritos Corn Chips Nacho Cheese 170g	Doritos
246741	2019-06-30	220	220032	219497	Dorito Corn Chp Supreme 380g	Dorito

246742 rows × 9 columns

In [6]:

```
df['BRAND_NAME'].value_counts()
```

Out[6]:

```
Kettle        41288
Smiths        27390
Pringles      25102
Doritos       22041
Thins         14075
RRD           11894
Infuzions     11057
WW            10320
Cobs           9693
Tostitos       9471
Twisties       9454
Tyrrells       6442
Grain          6272
Natural        6050
Cheezels       4603
CCs            4551
Red            4427
Dorito         3185
Infzns         3144
Smith          2963
Cheetos        2927
Snbts          1576
Burger         1564
Woolworths     1516
GrnWves        1468
Sunbites       1432
NCC            1419
French         1418
Name: BRAND_NAME, dtype: int64
```

In [7]:

```
# Check Unique Brand Instances
df.groupby('BRAND_NAME', group_keys=False).apply(lambda df: df.sample(1))
```

Out[7]:

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAM
232474	2019-06-09	156	156061	157030	Burger Rings 220g	Burge
973	2018-07-02	266	266009	263846	CCs Nacho Cheese 175g	CC
192716	2019-04-12	93	93174	91781	Cheetos Puffs 165g	Cheeto
137752	2019-01-20	172	172088	173268	Cheezels Cheese Box 125g	Cheezel
232618	2019-06-10	95	95230	95149	Cobs Popd Sea Salt Chips 110g	Cob
103556	2018-12-01	271	271179	269276	Dorito Corn Chp Supreme 380g	Dorit
181584	2019-03-26	236	236133	239362	Doritos Corn Chip Southern Chicken 150g	Dorito
186117	2019-04-02	106	106063	107121	French Fries Potato Chips 175g	Frenc
73405	2018-10-17	79	79246	77806	Grain Waves Sweet Chilli 210g	Grai
121072	2018-12-26	184	184197	187532	GrnWves Plus Btroot & Chilli Jam 180g	GrnWve
71640	2018-10-15	23	23040	18984	Infuzions SourCream&Herbs Veg Strws 110g	Infuzion
148618	2019-02-06	146	146382	145730	Infzns Crn Crnchers Tangy Gcamole 110g	Infzn
64908	2018-10-05	180	180191	182230	Kettle Honey Soy Chicken 175g	Kettl
137217	2019-01-20	67	67247	65294	NCC Sour Cream & Garden Chives 175g	NC(
42899	2018-09-02	10	10238	10430	Natural Chip Compny SeaSalt175g	Natura
126577	2019-01-04	201	201233	200946	Pringles Sweet&Spcy BBQ 134g	Pringle
195872	2019-04-16	67	67104	64428	RRD Pc Sea Salt 165g	RRI
73987	2018-10-18	227	227165	229260	Red Rock Deli Chikn&Garlic Aioli 150g	Re
11663	2018-07-18	271	271029	268378	Smith Crinkle Cut Mac N Cheese 150g	Smit
167858	2019-03-06	160	160050	160464	Smiths Crinkle Cut French OnionDip 150g	Smith

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAM
134016	2019-01-15	16	16029	14178	Snbts Whlgrn Crisps Cheddr&Mstrd 90g	Snbt
180048	2019-03-24	116	116176	120215	Sunbites Whlegrn Crisps Frch/Onin 90g	Sunbite
133883	2019-01-15	103	103167	103260	Thins Chips Seasonedchicken 175g	Thin
78568	2018-10-25	88	88181	87126	Tostitos Smoked Chipotle 175g	Tostito
236505	2019-06-15	153	153290	153115	Twisties Cheese 270g	Twistie
230032	2019-06-06	65	65348	63107	Tyrrells Crisps Lightly Salted 165g	Tyrrell
223803	2019-05-28	78	78000	75471	WW Original Stacked Chips 160g	WV
126306	2019-01-03	180	180070	181432	Woolworths Cheese Rings 190g	Woolworth

In [8]:

```
# Combine Duplicate Brands
df['BRAND_NAME'].replace({'Burger':'Burger Rings',
                          'Dorito':'Doritos',
                          'French':'French Fries',
                          'Grain': 'GrnWves',
                          'Infzns':'Infuzions',
                          'Natural':'NCC',
                          'Old':'Old El Paso',
                          'Red':'RRD',
                          'Smith':'Smiths',
                          'Snbts':'Sunbites',
                          'WW':'Woolworths'}, inplace = True)
```

```
C:\Users\Joel\anaconda3\lib\site-packages\pandas\core\generic.py:6746: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._update_inplace(new_data)
```

In [9]:

```
df['BRAND_NAME'].value_counts()
```

Out[9]:

```
Kettle          41288
Smiths          30353
Doritos         25226
Pringles        25102
RRD             16321
Infuzions       14201
Thins           14075
Woolworths      11836
Cobs             9693
Tostitos         9471
Twisties         9454
GrnWves          7740
NCC              7469
Tyrrells         6442
Cheezels         4603
CCs              4551
Sunbites         3008
Cheetos          2927
Burger Rings     1564
French Fries     1418
Name: BRAND_NAME, dtype: int64
```

Check for Outliers

In [10]:

```
# Check for outliers in transactions
print(df['PROD_QTY'].describe())
```

```
count    246742.000000
mean          1.908062
std           0.659831
min           1.000000
25%           2.000000
50%           2.000000
75%           2.000000
max         200.000000
Name: PROD_QTY, dtype: float64
```

- Seems like an outlier of 200 purchases, may be a bulk order for business

In [11]:

```
# Check the features for records over 6
df[df['PROD_QTY'] > 6]
```

Out[11]:

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAME	P
33534	2018-08-19	226	226000	226201	Dorito Corn Chp Supreme 380g	Doritos	
218684	2019-05-20	226	226000	226210	Dorito Corn Chp Supreme 380g	Doritos	

In [12]:

```
# Appears to be the same customer, did that customer have any other transactions? if no
t then drop those outliers
df[df['LYLTY_CARD_NBR'] == 226000]
```

Out[12]:

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAME	P
33534	2018-08-19	226	226000	226201	Dorito Corn Chp Supreme 380g	Doritos	
218684	2019-05-20	226	226000	226210	Dorito Corn Chp Supreme 380g	Doritos	

In [13]:

```
# Drop those rows
df.drop(df[df.LYLTY_CARD_NBR == 226000].index, inplace = True)
```

```
C:\Users\Joel\anaconda3\lib\site-packages\pandas\core\frame.py:3990: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

In [14]:

```
# Check dropped rows
df[df['LYLTY_CARD_NBR'] == 226000]
```

Out[14]:

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAME	PACKET

Check for complete dates

In [15]:

```
print(df['real_date'].value_counts())
```

```
2018-12-24    865
2018-12-23    853
2018-12-22    840
2018-12-19    839
2018-12-20    808
             ...
2019-06-24    612
2018-10-18    611
2018-11-25    610
2018-09-22    609
2019-06-13    607
Name: real_date, Length: 364, dtype: int64
```

- It appears we are missing one day of data.

In [16]:

```
# Find date which is missing for FY 2018-2019
pd.date_range(start = '2018-07-01', end = '2019-06-30').difference(df['real_date'])
```

Out[16]:

```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

- Date missing is christmas day, which is a public holiday. We can assume this particular business did not operate on that day.

Describe Data and Check Correlation

In [17]:

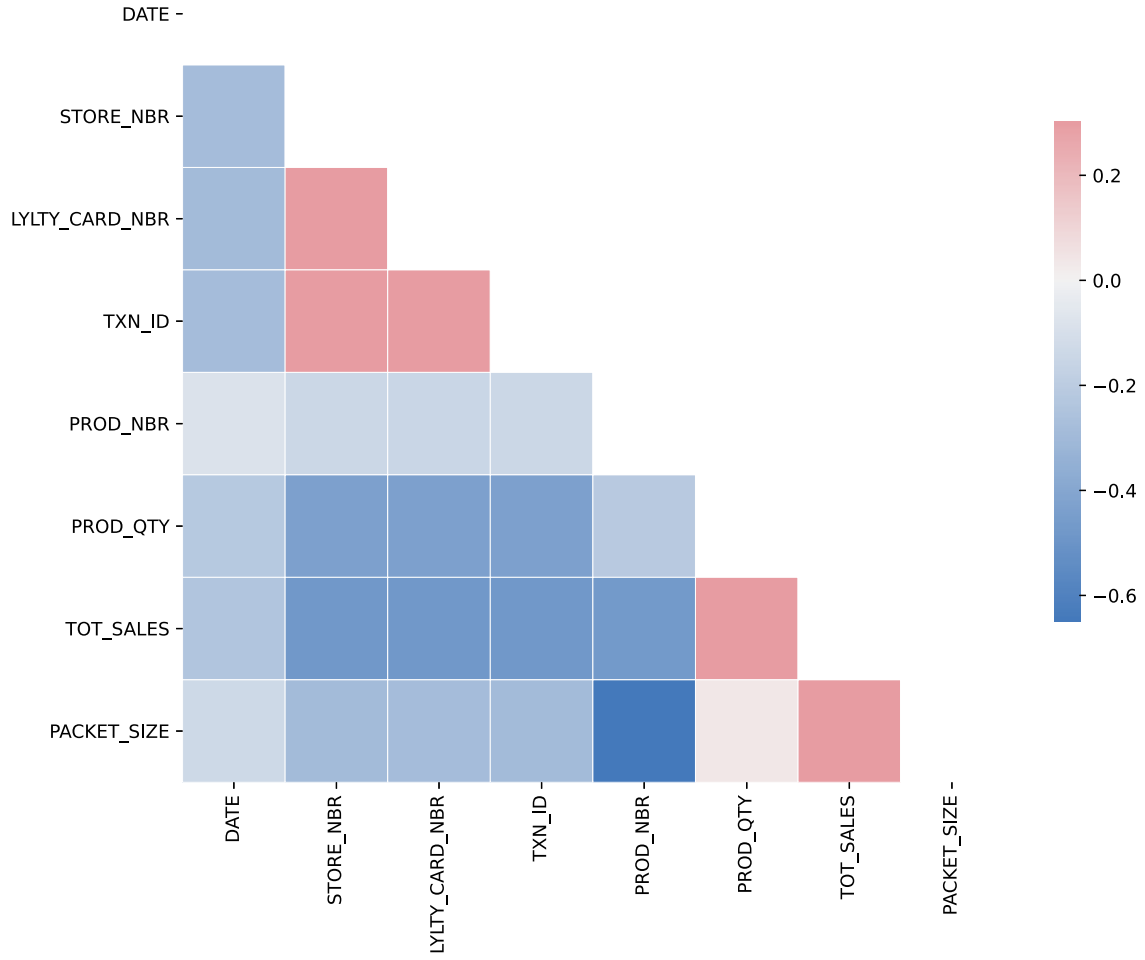
```
df.describe()
```

Out[17]:

	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PACKET_SIZE	PROD_QTY	TO
count	246740.000000	2.467400e+05	2.467400e+05	246740.000000	246740.000000	2467
mean	135.050361	1.355303e+05	1.351304e+05	175.583521	1.906456	
std	76.786971	8.071520e+04	7.814760e+04	59.432118	0.342499	
min	1.000000	1.000000e+03	1.000000e+00	70.000000	1.000000	
25%	70.000000	7.001500e+04	6.756875e+04	150.000000	2.000000	
50%	130.000000	1.303670e+05	1.351815e+05	170.000000	2.000000	
75%	203.000000	2.030832e+05	2.026522e+05	175.000000	2.000000	
max	272.000000	2.373711e+06	2.415841e+06	380.000000	5.000000	2

In [18]:

```
# Correlation
data_corr = data.corr()

corr = data_corr.corr()

mask = np.triu(np.ones_like(corr, dtype=np.bool))

f, ax = plt.subplots(figsize=(10, 10))

cmap = sns.diverging_palette(250, 10, as_cmap=True)

sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})

plt.title('Correlation Heatmap', fontsize = 20)

plt.show()
```



Transactions Through the Year

In [19]:

```
# Create Dataframe for transaction counts
transactions = pd.DataFrame.from_dict(dict(df['real_date'].value_counts()), orient = 'i
ndex', columns = ['transactions'])
transactions.reset_index(inplace = True)
transactions.rename(columns = {'index': 'date'}, inplace = True)
transactions.sort_values(by = 'date', inplace = True)
transactions
```

Out[19]:

	date	transactions
252	2018-07-01	663
302	2018-07-02	650
177	2018-07-03	674
214	2018-07-04	669
266	2018-07-05	660
...	...	...
280	2019-06-26	657
211	2019-06-27	669
185	2019-06-28	673
56	2019-06-29	703
54	2019-06-30	704

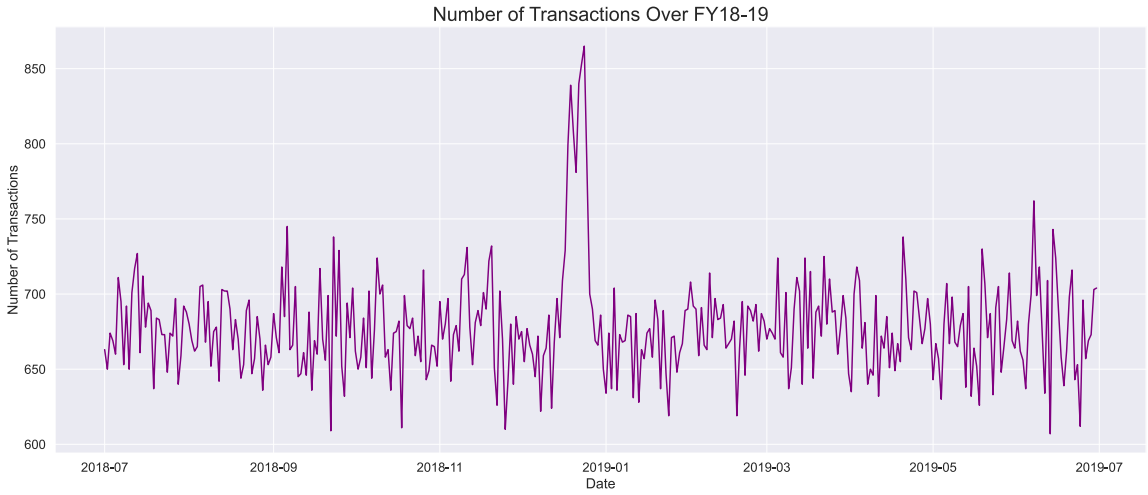
364 rows × 2 columns

In [20]:

```
plt.figure(figsize = (20,8))
sns.set_style('darkgrid')
sns.set_context('notebook', font_scale = 1.2)
sns.lineplot(data = transactions, x = 'date', y = 'transactions', palette = 'rocket', c
olor = 'purple')
plt.xlabel('Date')
plt.ylabel('Number of Transactions')
plt.title('Number of Transactions Over FY18-19', fontsize = 20)
```

Out[20]:

```
Text(0.5, 1.0, 'Number of Transactions Over FY18-19')
```



OBSERVATIONS:

- We see a large increase in sales building up to christmas, and a sudden drop off afterwards back to the regular amount. The data does not inculde the drop to 0 sales on christmas day due to closure on the 25th.
- Data shows no significant outliers besides december sales, which is already explained by lead up to christmas period.

Pack Size Analysis

In [21]:

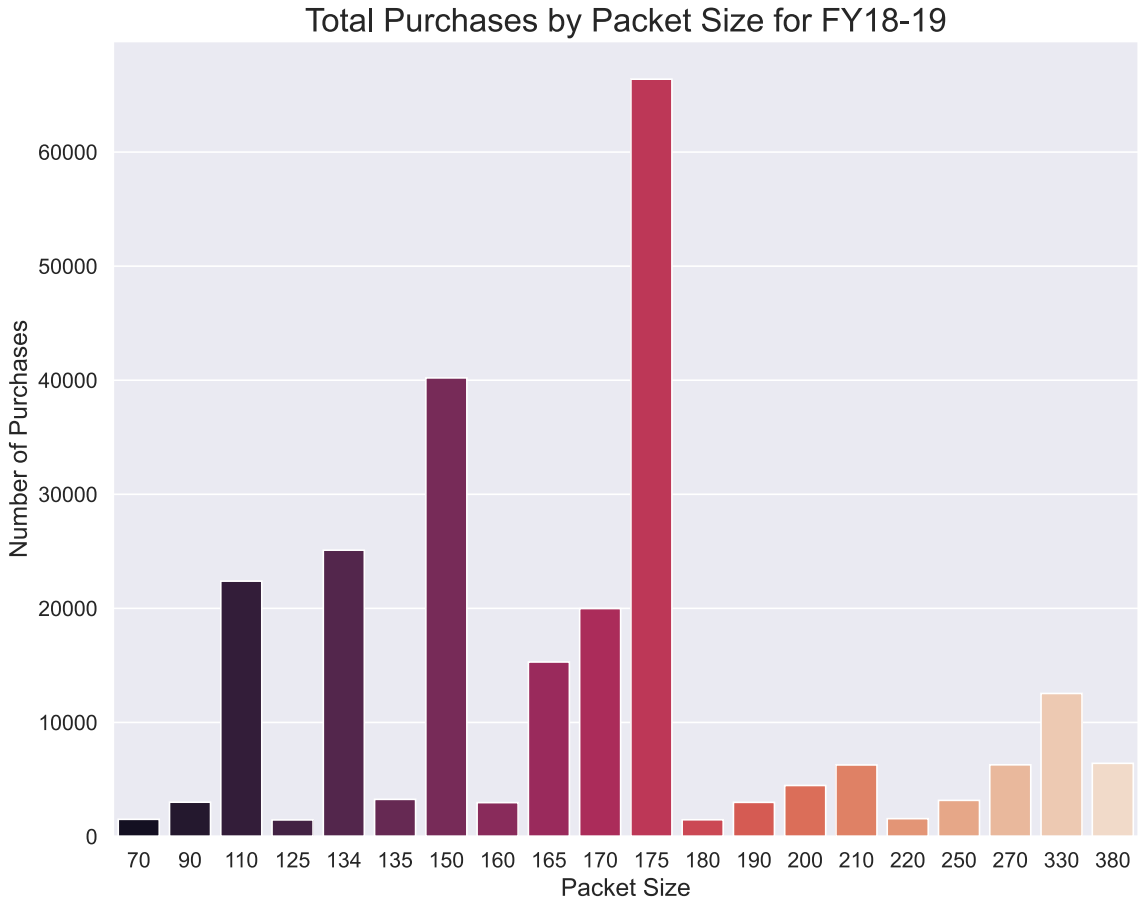
```
# List all types of packet sizes
pack_sizes = pd.DataFrame.from_dict(dict(df['PACKET_SIZE'].value_counts()), orient = 'i
ndex', columns = ['counts'])
pack_sizes.reset_index(inplace = True)
pack_sizes.rename(columns = {'index':'size (g)'}, inplace = True)
pack_sizes.sort_values(by = 'size (g)', inplace = True)
pack_sizes
```

Out[21]:

	size (g)	counts
17	70	1507
13	90	3008
3	110	22387
19	125	1454
2	134	25102
11	135	3257
1	150	40203
15	160	2970
5	165	15297
4	170	19983
0	175	66390
18	180	1468
14	190	2995
10	200	4473
9	210	6272
16	220	1564
12	250	3169
8	270	6285
6	330	12540
7	380	6416

In [22]:

```
plt.figure(figsize = (10,8))
ax = sns.set_style('darkgrid')
ax = sns.barplot(x = pack_sizes['size (g)'], y = pack_sizes['counts'], palette = 'rocke
t')
ax.set_xlabel('Packet Size', fontsize = 15)
ax.set_ylabel('Number of Purchases', fontsize = 15)
ax.axes.set_title('Total Purchases by Packet Size for FY18-19', fontsize = 20)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
plt.tight_layout()
```



OBSERVATIONS:

- The smallest packet is 70g, whilst the upper range goes up to 380g pakcets.
- The most common packet is 175g.
- In the histogram, we see that the upper ranges of packet sizes are not as popular.
- We see a common pattern with distribution, with 110g, 134g, 150g, and 175g topping the purchases. This may be indication that consumers see these sizes being the best value for money.

Brand Analysis

In [23]:

```
df['BRAND_NAME'].describe()
```

Out[23]:

```
count     246740
unique        20
top       Kettle
freq       41288
Name: BRAND_NAME, dtype: object
```

In [24]:

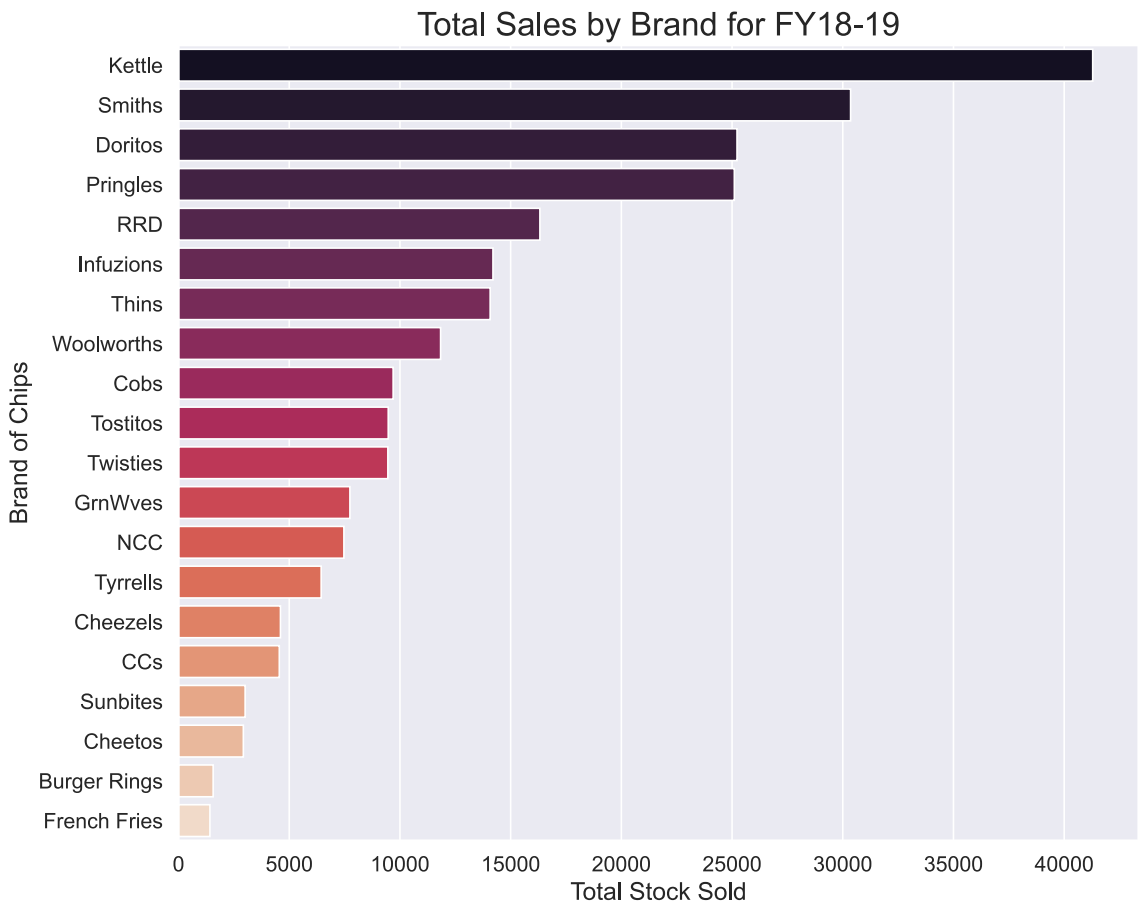
```
brand_name_dist = dict(df['BRAND_NAME'].value_counts())
brands = pd.DataFrame.from_dict(data = brand_name_dist,columns = ['Stock Sold'], orient
= 'index')
brands
```

Out[24]:

	Stock Sold
Kettle	41288
Smiths	30353
Doritos	25224
Pringles	25102
RRD	16321
Infuzions	14201
Thins	14075
Woolworths	11836
Cobs	9693
Tostitos	9471
Twisties	9454
GrnWves	7740
NCC	7469
Tyrrells	6442
Cheezels	4603
CCs	4551
Sunbites	3008
Cheetos	2927
Burger Rings	1564
French Fries	1418

In [25]:

```
plt.figure(figsize = (10,8))
ax = sns.set_style('darkgrid')
ax = sns.barplot(x = brands['Stock Sold'], y = brands.index, palette = 'rocket')
ax.set_xlabel('Total Stock Sold', fontsize = 15)
ax.set_ylabel('Brand of Chips', fontsize = 15)
ax.axes.set_title('Total Sales by Brand for FY18-19', fontsize = 20)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
plt.tight_layout()
```



PRELIMINARY OBSERVATONS:

- Kettle is the most popular brand ofchips
- Followed by Smiths
- Dirotis and Pringles are relatively equal in terms of popularity
- Burger Rings and French Fries are the least popular

Customer Data

In [26]:

```
# import data
file = r'C:\Users\Joel\Dropbox\Vitrual Internships\Quantium\Task 1\Data\QVI_purchase_behaviour.csv'
data2 = pd.read_csv(file, index_col = None)
data2
```

Out[26]:

	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 [27]:

```
# Check for nulls
data2.isnull().sum()
```

Out[27]:

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

In [28]:

```
# Check value counts for duplicate customers
data2['LYLTY_CARD_NBR'].value_counts()
```

Out[28]:

```
2047      1
197109    1
121326    1
119279    1
74225     1
         ..
251088    1
77007     1
81101     1
79052     1
131072    1
Name: LYLTY_CARD_NBR, Length: 72637, dtype: int64
```

- Length matches length of data frame, no duplicate customer id's.

Customer Lifestage Segments

In [29]:

```
data2['LIFESTAGE'].describe()
```

Out[29]:

```
count        72637
unique           7
top       RETIREES
freq         14805
Name: LIFESTAGE, dtype: object
```

In [30]:

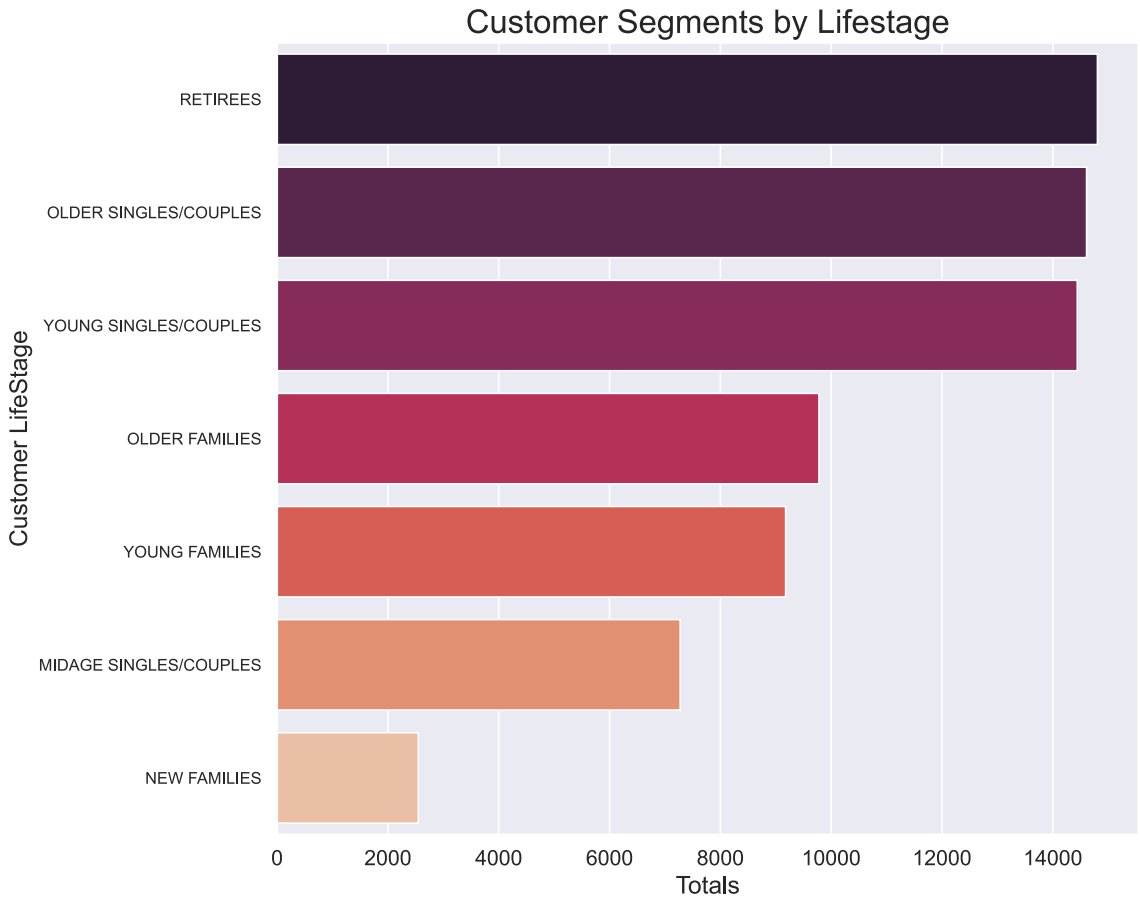
```
data2['LIFESTAGE'].value_counts()
```

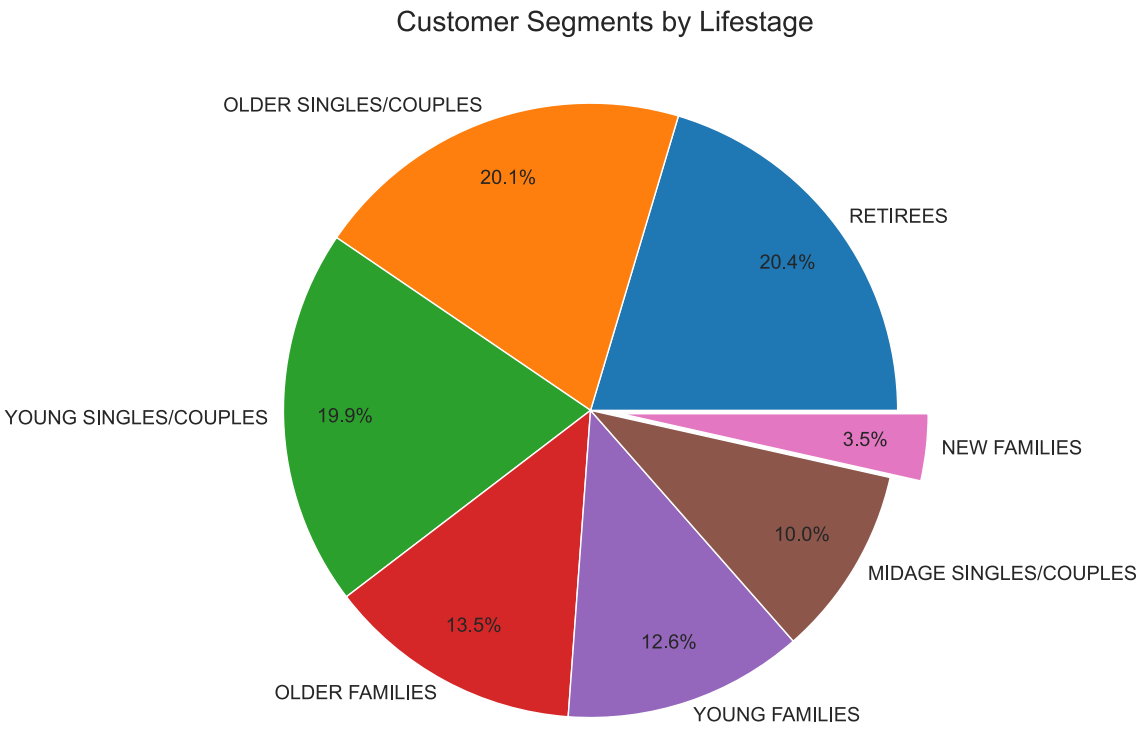
Out[30]:

```
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
```

In [31]:

```
# Make DataFrame
lifestage = pd.DataFrame.from_dict(dict(data2['LIFESTAGE'].value_counts()), orient = 'i
ndex', columns = ['Counts'])

# Show Distribution
plt.figure(figsize = (10,8))
ax = sns.set_style('darkgrid')
ax = sns.barplot(y = lifestage.index, x = lifestage['Counts'], palette = 'rocket')
ax.set_ylabel('Customer LifeStage', fontsize = 15)
ax.set_xlabel('Totals', fontsize = 15)
ax.axes.set_title('Customer Segments by Lifestage', fontsize = 20)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 10)
plt.tight_layout()

# Show Proportions
# Pie chart
pie, ax = plt.subplots(figsize = (10, 8))
labels = lifestage.index
plt.pie(x = lifestage['Counts'], autopct="%.1f%%", explode=(0, 0, 0, 0, 0, 0, .05), lab
els=labels, pctdistance=0.8, labeldistance = 1.05, textprops = {'fontsize': 14}, radius
= 0.5)
plt.title("Customer Segments by Lifestage", fontsize=20)
ax.axis('square')
plt.tight_layout()
```


Customer Segments by Lifestage
RETIREES
OLDER SINGLES/COUPLES
YOUNG SINGLES/COUPLES
OLDER FAMILIES
YOUNG FAMILIES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
Customer LifeStage
0
2000
4000
6000
8000
10000
12000
14000
Totals



- Business categorizes this segment into 7 classes.
- Majority of customers are either retirees or older/younger singles/couples, being 20.4%, 20.1% and 19.9% respectively, making up 60.4% of the population.

Customer Type Segments

In [32]:

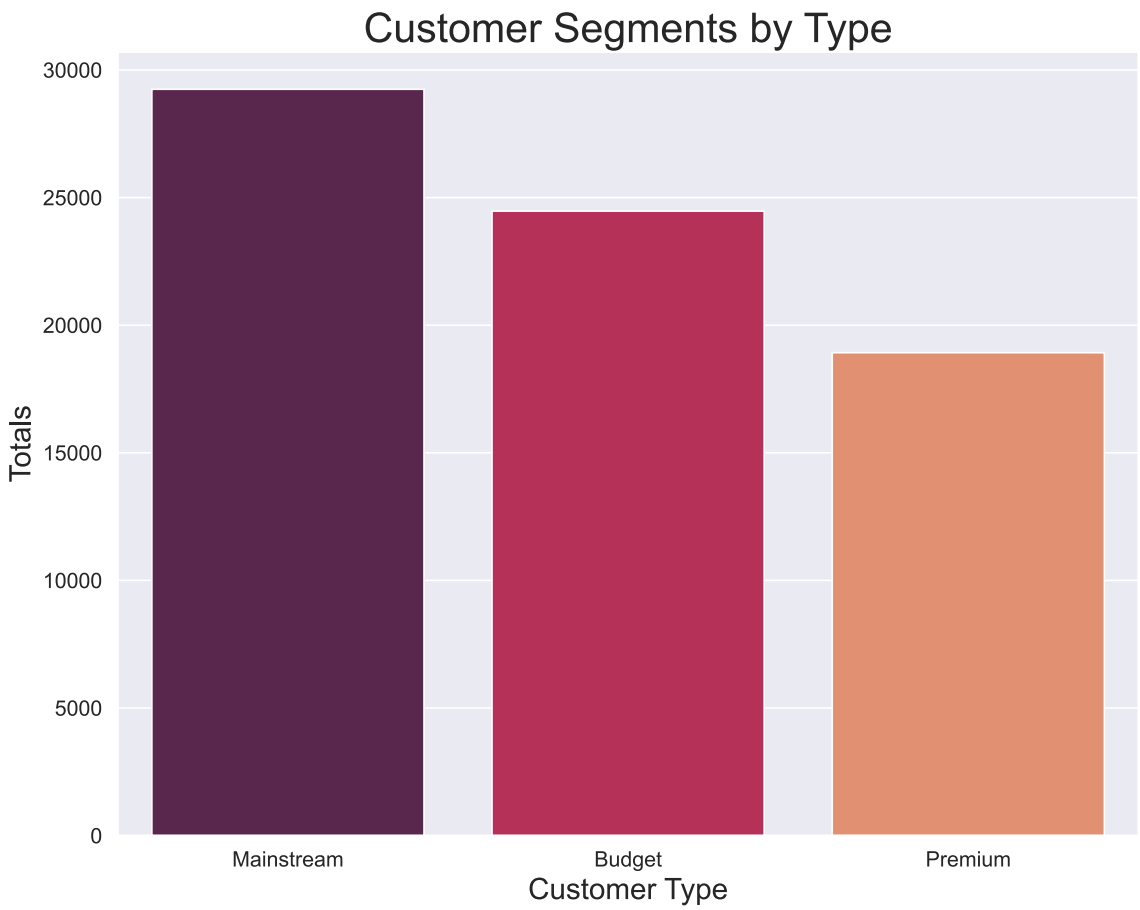
```
data2['PREMIUM_CUSTOMER'].value_counts()
```

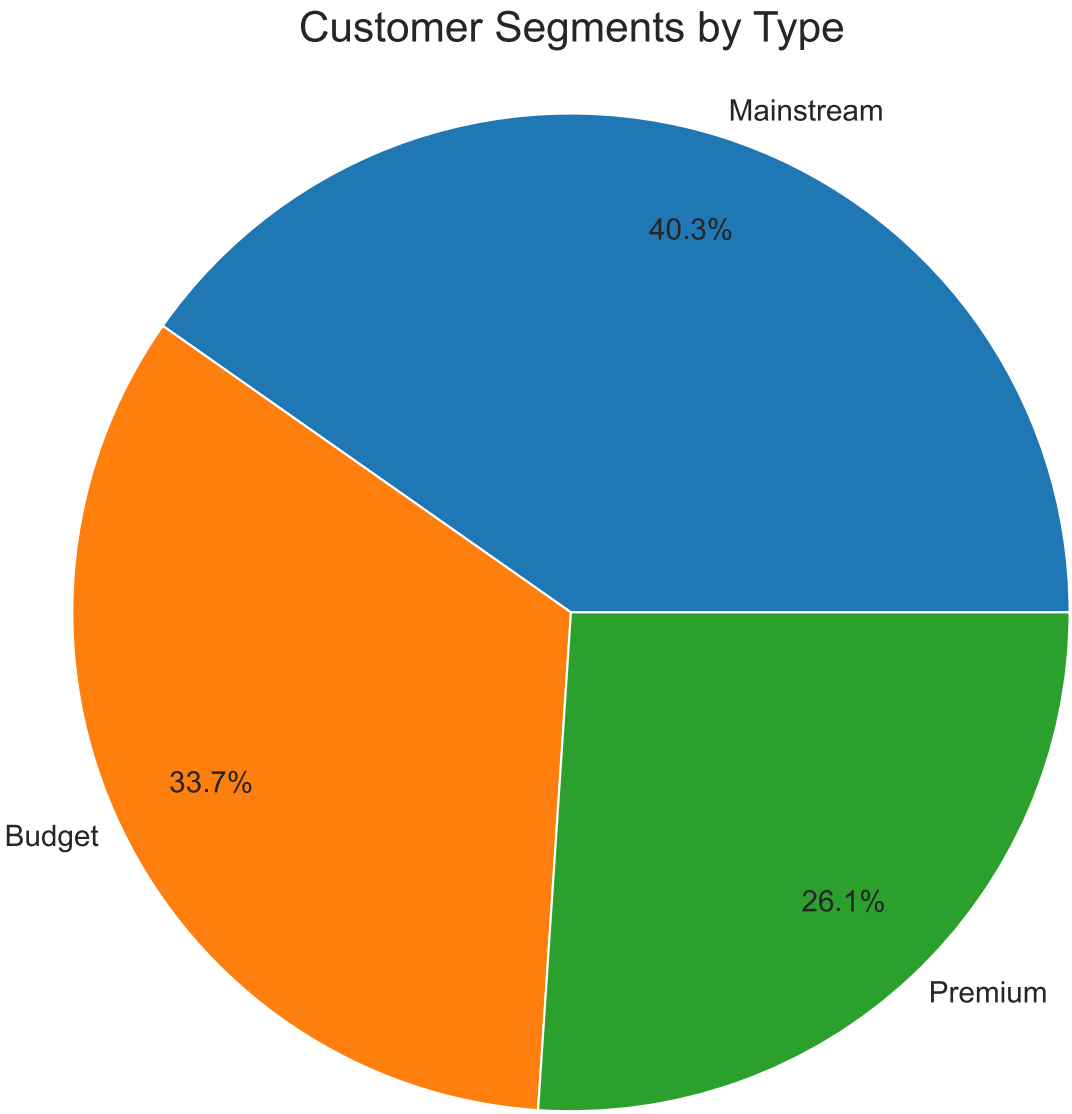
Out[32]:

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

In [33]:

```
# Make DataFrame
cstype = pd.DataFrame.from_dict(dict(data2['PREMIUM_CUSTOMER'].value_counts()), orient
= 'index', columns = ['Counts'])

# Show Distribution
plt.figure(figsize = (10,8))
ax = sns.set_style('darkgrid')
ax = sns.barplot(x = cstype.index, y = cstype['Counts'], palette = 'rocket')
ax.set_xlabel('Customer Type', fontsize = 18)
ax.set_ylabel('Totals', fontsize = 18)
ax.axes.set_title('Customer Segments by Type', fontsize = 25)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
plt.tight_layout()

# Show proportionate distribution (Pie Chart)
pie, ax = plt.subplots(figsize = (10, 8))
labels = cstype.index
plt.pie(x = cstype['Counts'], autopct="%.1f%%", labels=labels, pctdistance=0.8, labeldi
stance = 1.05, textprops = {'fontsize': 14}, radius = 0.5)
plt.title("Customer Segments by Type", fontsize=20)
ax.axis('square')
plt.tight_layout()
```


Customer Segments by Type
30000
25000
20000
15000
10000
5000
0
Totals
Mainstream
Budget
Premium
Customer Type



Merging the 2 Datasets

In [34]:

```
df
```

Out[34]:

	real_date	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NAME	BRAND_NAME
0	2018-07-01	9	9341	8808	Smiths Thinly Cut Roast Chicken 175g	Smiths
1	2018-07-01	86	86016	84237	Red Rock Deli Sp Salt & Truffle 150G	RRD
2	2018-07-01	129	129046	132474	Smith Crinkle Cut Mac N Cheese 150g	Smiths
3	2018-07-01	58	58072	53145	Pringles Sthrn FriedChicken 134g	Pringles
4	2018-07-01	97	97164	97311	WW Crinkle Cut Chicken 175g	Woolworths
...	...	...	...	...	...	...
246737	2019-06-30	91	91076	89519	Thins Chips Seasonedchicken 175g	Thins
246738	2019-06-30	84	84116	83704	Doritos Corn Chips Nacho Cheese 170g	Doritos
246739	2019-06-30	24	24115	20917	Smiths Crinkle Cut Chips Chs&Onion170g	Smiths
246740	2019-06-30	199	199117	198068	Doritos Corn Chips Nacho Cheese 170g	Doritos
246741	2019-06-30	220	220032	219497	Dorito Corn Chp Supreme 380g	Doritos

246740 rows × 9 columns

In [35]:

```
data2
```

Out[35]:

	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 [36]:

```
# LEFT JOIN using SQL
query = """
        SELECT df.real_date,
                df.STORE_NBR,
                df.TXN_ID,
                df.PROD_NAME,
                df.BRAND_NAME,
                df.PACKET_SIZE,
                df.PROD_QTY,
                df.TOT_SALES,
                df.LYLTY_CARD_NBR,
                data2.LIFESTAGE,
                data2.PREMIUM_CUSTOMER
        FROM df
        LEFT JOIN data2
        ON df.LYLTY_CARD_NBR = data2.LYLTY_CARD_NBR
        """
data = ps.sqldf(query, locals())

# Convert datetime to just date
data['real_date'] = pd.to_datetime(df['real_date']).dt.date

# Display merged dataframe
data
```

Out[36]:

	real_date	STORE_NBR	TXN_ID	PROD_NAME	BRAND_NAME	PACKET_SIZE	PR
0	2018-07-01	9	8808	Smiths Thinly Cut Roast Chicken 175g	Smiths	175	
1	2018-07-01	86	84237	Red Rock Deli Sp Salt & Truffle 150G	RRD	150	
2	2018-07-01	129	132474	Smith Crinkle Cut Mac N Cheese 150g	Smiths	150	
3	2018-07-01	58	53145	Pringles Sthrn FriedChicken 134g	Pringles	134	
4	2018-07-01	97	97311	WW Crinkle Cut Chicken 175g	Woolworths	175	
...	...	...	...	...	...	...	
246735	2019-06-30	91	89519	Thins Chips Seasonedchicken 175g	Thins	175	
246736	2019-06-30	84	83704	Doritos Corn Chips Nacho Cheese 170g	Doritos	170	
246737	2019-06-30	24	20917	Smiths Crinkle Cut Chips Chs&Onion170g	Smiths	170	
246738	2019-06-30	199	198068	Doritos Corn Chips Nacho Cheese 170g	Doritos	170	
246739	2019-06-30	220	219497	Dorito Corn Chp Supreme 380g	Doritos	380	

246740 rows × 11 columns

Check merged dataframe for nulls

In [37]:

```
data.isnull().sum()
```

Out[37]:

```
real_date           2
STORE_NBR           0
TXN_ID              0
PROD_NAME           0
BRAND_NAME          0
PACKET_SIZE         0
PROD_QTY            0
TOT_SALES           0
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [38]:

```
# Find which records have null date
data[data['real_date'].isnull()]
```

Out[38]:

	real_date	STORE_NBR	TXN_ID	PROD_NAME	BRAND_NAME	PACKET_SIZE	PROD
33534	NaN	89	88591	Cheezels Cheese Box 125g	Cheezels	125	
218684	NaN	250	251884	Smiths Crinkle Original 330g	Smiths	330	

In [39]:

```
# drop those 2 rows
data.drop(index = [33534, 218684], inplace = True)
```

In [40]:

```
# Check for nulls
data.isnull().sum()
```

Out[40]:

```
real_date           0
STORE_NBR           0
TXN_ID              0
PROD_NAME           0
BRAND_NAME          0
PACKET_SIZE         0
PROD_QTY            0
TOT_SALES           0
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [41]:

```
# Export Clean data to CSV
data.to_csv('QVI_clean_data.csv')
```

Check Active Customers

In [42]:

```
data.LYLTY_CARD_NBR.value_counts()
```

Out[42]:

```
230078    17
162039    17
113080    16
23192     16
105026    16
          ..
66236      1
195267     1
187079     1
148180     1
146351     1
Name: LYLTY_CARD_NBR, Length: 71287, dtype: int64
```

Total unique customers registered:

- 72,637

Total active customers in current FY:

- 71,287

Total inactive customers: 72,637 - 71,287 = 1,350

- 1.9% of registered customers are inactive, meaning they haven't made any purchases in the past year.

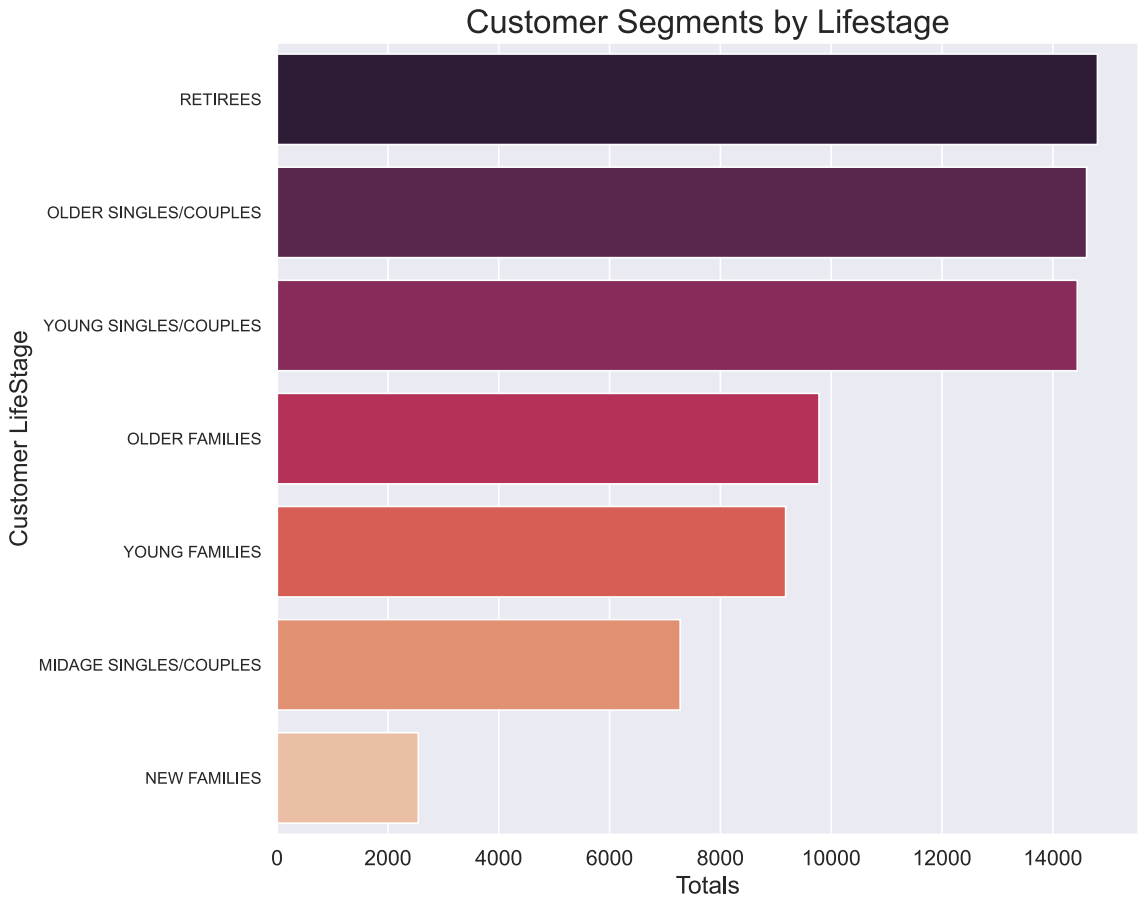
Customer Segmentation Analysis

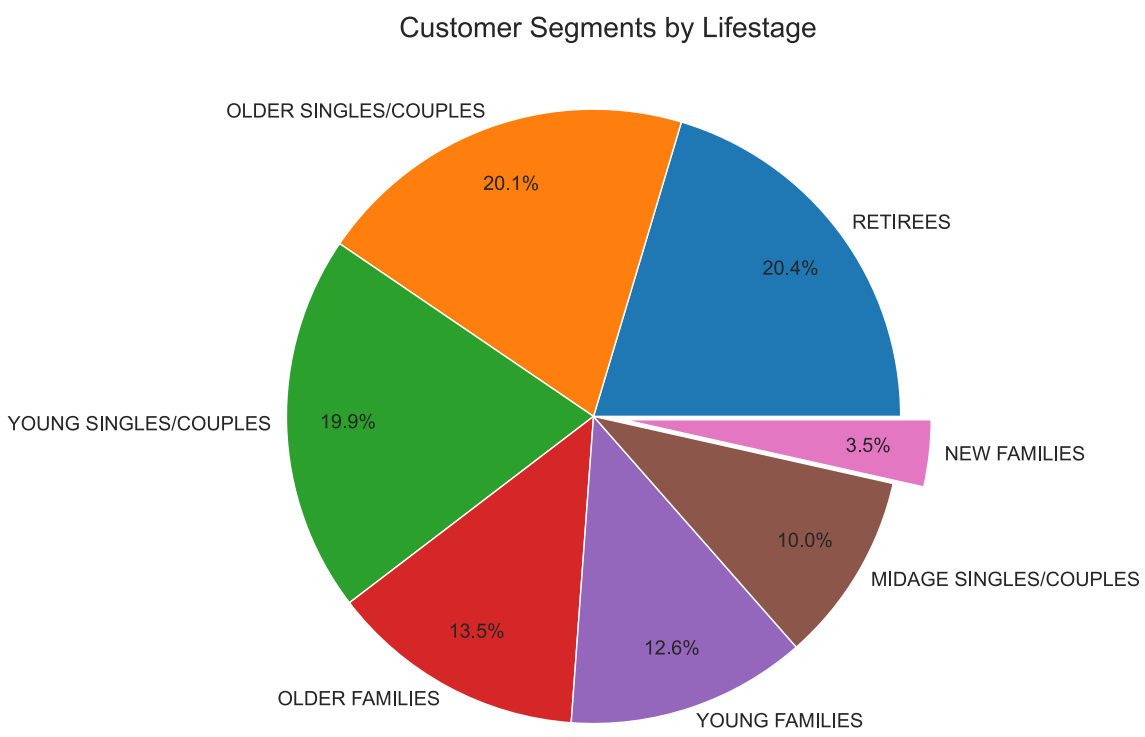
- Some prior code may repeat. This is the formal analysis section of the code.

CUSTOMER LIFE STAGE ANALYSIS

In [43]:

```
# Make a DataFrame
lifestage = pd.DataFrame.from_dict(dict(data2['LIFESTAGE'].value_counts()), orient = 'i
ndex', columns = ['Counts'])

# Show Distribution
plt.figure(figsize = (10,8))
ax = sns.set_style('darkgrid')
ax = sns.barplot(y = lifestage.index, x = lifestage['Counts'], palette = 'rocket')
ax.set_ylabel('Customer LifeStage', fontsize = 15)
ax.set_xlabel('Totals', fontsize = 15)
ax.axes.set_title('Customer Segments by Lifestage', fontsize = 20)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 10)
plt.tight_layout()

# Show Proportions
# Pie chart
pie, ax = plt.subplots(figsize = (10, 8))
labels = lifestage.index
plt.pie(x = lifestage['Counts'], autopct="%.1f%%", explode=(0, 0, 0, 0, 0, 0, .05), lab
els=labels, pctdistance=0.8, labeldistance = 1.05, textprops = {'fontsize': 14}, radius
= 0.5)
plt.title("Customer Segments by Lifestage", fontsize=20)
ax.axis('square')
plt.tight_layout()
```


Customer Segments by Lifestage
RETIREES
OLDER SINGLES/COUPLES
YOUNG SINGLES/COUPLES
OLDER FAMILIES
YOUNG FAMILIES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
Customer LifeStage
0
2000
4000
6000
8000
10000
12000
14000
Totals



In [44]:

```
# Check Unique customers, grouping by lifestage
query = """
        SELECT LIFESTAGE, COUNT(DISTINCT(LYLTY_CARD_NBR)) AS 'UNIQUE_COUNTS'
        FROM data
        GROUP BY LIFESTAGE
        """
unique_lifestage_count = ps.sqldf(query, locals())
unique_lifestage_count
```

Out[44]:

	LIFESTAGE	UNIQUE_COUNTS
0	MIDAGE SINGLES/COUPLES	7141
1	NEW FAMILIES	2492
2	OLDER FAMILIES	9630
3	OLDER SINGLES/COUPLES	14389
4	RETIREES	14555
5	YOUNG FAMILIES	9036
6	YOUNG SINGLES/COUPLES	14044

In [45]:

```
lifestage
```

Out[45]:

	Counts
RETIREES	14805
OLDER SINGLES/COUPLES	14609
YOUNG SINGLES/COUPLES	14441
OLDER FAMILIES	9780
YOUNG FAMILIES	9178
MIDAGE SINGLES/COUPLES	7275
NEW FAMILIES	2549

OBSERVATIONS

- Business categorizes this segment into 7 classes.
- Majority of customers are either retirees or older/younger singles/couples, being 20.4%, 20.1% and 19.9% respectively, making up 60.4% of the population.

In [46]:

```
# Add average chip price per transaction
data['AVG_CHIP_PRICE'] = data['TOT_SALES']/data['PROD_QTY']
data
```

Out[46]:

	real_date	STORE_NBR	TXN_ID	PROD_NAME	BRAND_NAME	PACKET_SIZE	PR
0	2018-07-01	9	8808	Smiths Thinly Cut Roast Chicken 175g	Smiths	175	
1	2018-07-01	86	84237	Red Rock Deli Sp Salt & Truffle 150G	RRD	150	
2	2018-07-01	129	132474	Smith Crinkle Cut Mac N Cheese 150g	Smiths	150	
3	2018-07-01	58	53145	Pringles Sthrn FriedChicken 134g	Pringles	134	
4	2018-07-01	97	97311	WW Crinkle Cut Chicken 175g	Woolworths	175	
...	...	...	...	...	...	...	
246735	2019-06-30	91	89519	Thins Chips Seasonedchicken 175g	Thins	175	
246736	2019-06-30	84	83704	Doritos Corn Chips Nacho Cheese 170g	Doritos	170	
246737	2019-06-30	24	20917	Smiths Crinkle Cut Chips Chs&Onion170g	Smiths	170	
246738	2019-06-30	199	198068	Doritos Corn Chips Nacho Cheese 170g	Doritos	170	
246739	2019-06-30	220	219497	Dorito Corn Chp Supreme 380g	Doritos	380	

246738 rows × 12 columns

In [47]:

```
# make list of unique customers grouped by lifestage
unique_customers = [14555,14389,14044,9630,9036,7141,2492]
lifestage_analysis = lifestage
lifestage_analysis['UNIQUE_CUST'] = unique_customers
lifestage_analysis
```

Out[47]:

	Counts	UNIQUE_CUST
RETIREES	14805	14555
OLDER SINGLES/COUPLES	14609	14389
YOUNG SINGLES/COUPLES	14441	14044
OLDER FAMILIES	9780	9630
YOUNG FAMILIES	9178	9036
MIDAGE SINGLES/COUPLES	7275	7141
NEW FAMILIES	2549	2492

In [48]:

```
# initialize lists
lifestage_sales_sum = []
lifestage_sales_avg = []
lifestage_avg_qty = []
lifestage_tot_qty = []
lifestage_avg_unit_price = []

# Get average spend and total sum of sales by lifestage
# Check average quantity and total quantity of chips bought per lifestage segment
for i in lifestage.index:
    subset = data[data['LIFESTAGE'] == i]
    lifestage_sales_sum.append(round(np.sum(subset['TOT_SALES']), 2))
    lifestage_sales_avg.append(round(np.mean(subset['TOT_SALES']), 2))
    lifestage_avg_qty.append(round(np.mean(subset['PROD_QTY']), 2))
    lifestage_tot_qty.append(np.sum(subset['PROD_QTY']))
    lifestage_avg_unit_price.append(round(np.mean(subset['AVG_CHIP_PRICE']), 2))



lifestage_analysis['AVG_SPEND'] = lifestage_sales_avg
lifestage_analysis['TOTAL_SPEND'] = lifestage_sales_sum
lifestage_analysis['AVG_QTY'] = lifestage_avg_qty
lifestage_analysis['TOT_QTY'] = lifestage_tot_qty
lifestage_analysis['AVG_CHIP_PRICE'] = lifestage_avg_unit_price
lifestage.reset_index(inplace=True)
lifestage_analysis.rename(columns = {'Counts':'TXN_COUNTS', 'index':'SEGMENT'}, inplace
= True)
lifestage_analysis
```

Out[48]:

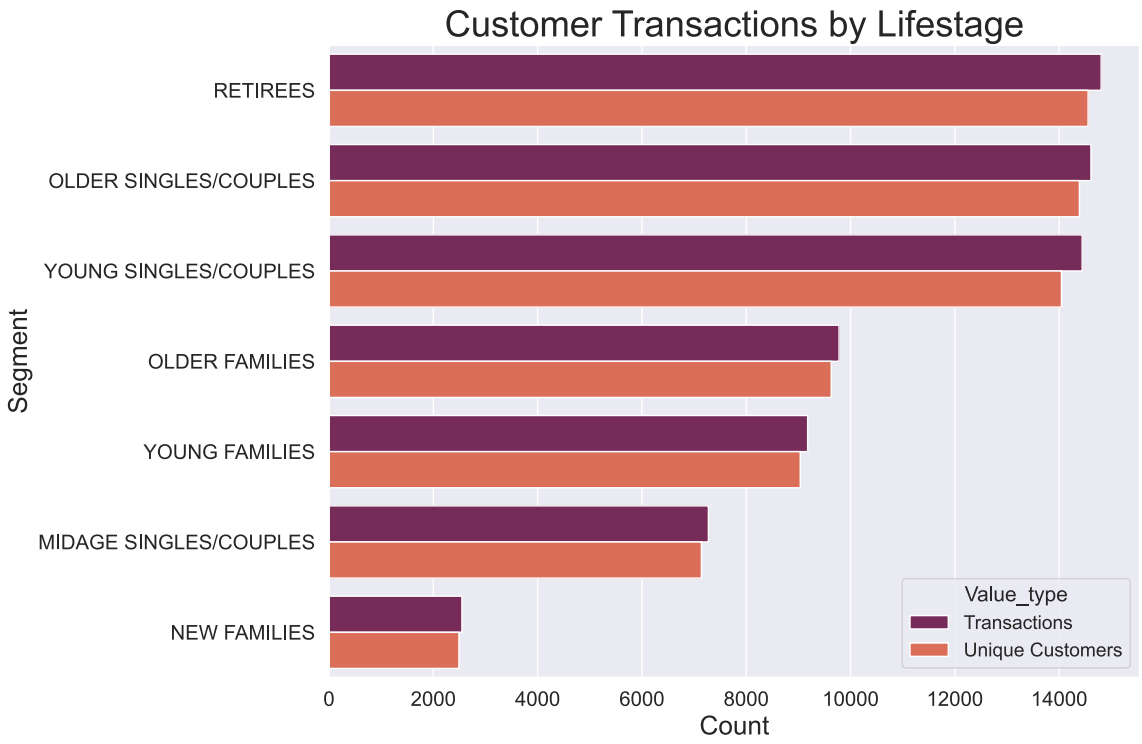
	SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY
0	RETIREES	14805	14555	7.37	342381.90	1.89
1	OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91
2	YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83
3	OLDER FAMILIES	9780	9630	7.27	328519.90	1.95
4	YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94
5	MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90
6	NEW FAMILIES	2549	2492	7.29	47347.95	1.86

In [49]:

```
# Transaction analysis Dataframe
segments = []
for i in lifestage_analysis.SEGMENT:
    segments.append(i)

for i in lifestage_analysis.SEGMENT:
    segments.append(i)

counts = []
for i in lifestage_analysis['TXN_COUNTS']:
    counts.append(i)
for i in lifestage_analysis['UNIQUE_CUST']:
    counts.append(i)

types = ['Transactions','Transactions','Transactions','Transactions','Transactions','Tr
ansactions','Transactions',
        'Unique Customers', 'Unique Customers','Unique Customers','Unique Customers','U
nique Customers','Unique Customers','Unique Customers']

segments = pd.DataFrame(segments, columns = ['SEGMENT'])
segments['COUNTS'] = counts
segments['Value_type'] = types

# Make Visualization
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'SEGMENT', x = 'COUNTS', data = segments, hue = 'Value_type', pale
tte= 'rocket')
ax.set_ylabel('Segment', fontsize = 18)
ax.set_xlabel('Count', fontsize = 18)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.tick_params(labelsize = 14)
plt.title('Customer Transactions by Lifestage', fontsize = 25)
plt.tight_layout
```

Out[49]:

```
<function matplotlib.pyplot.tight_layout(pad=1.08, h_pad=None, w_pad=None,
rect=None)>
```



OBSERVATIONS:

- The number of transactions by each segment closely follows the population distribution of each segment.

In [50]:

```
lifestage_analysis
```

Out[50]:

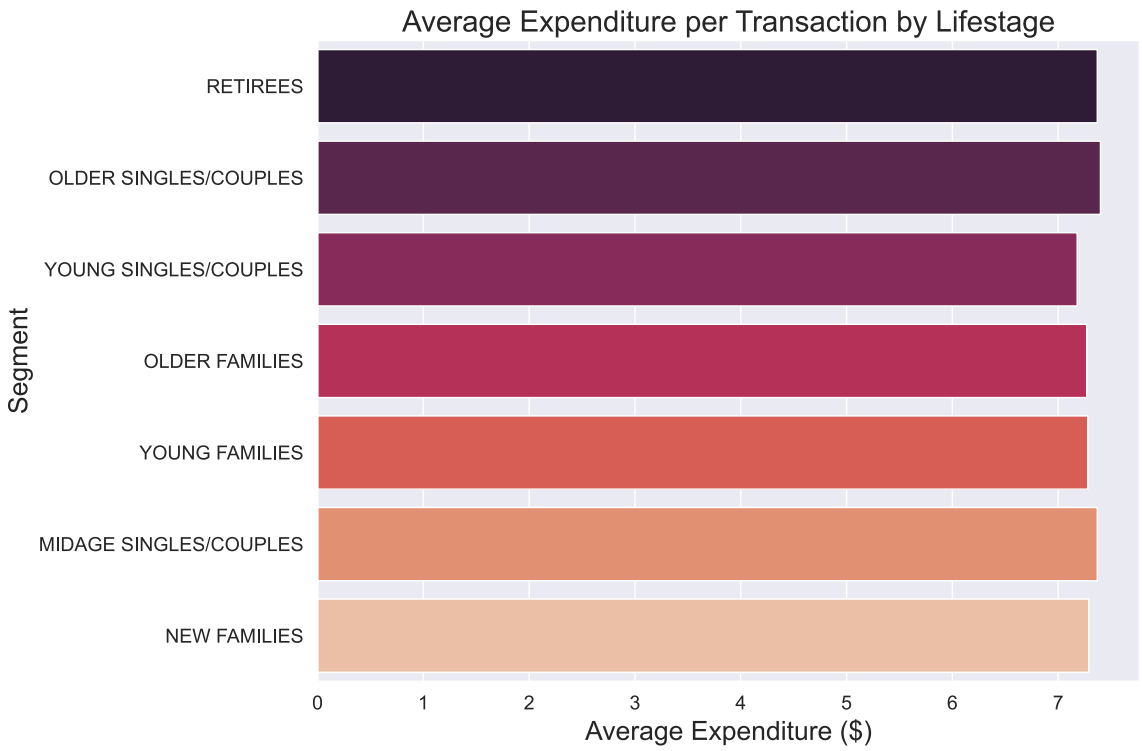
	SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY
0	RETIREES	14805	14555	7.37	342381.90	1.89
1	OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91
2	YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83
3	OLDER FAMILIES	9780	9630	7.27	328519.90	1.95
4	YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94
5	MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90
6	NEW FAMILIES	2549	2492	7.29	47347.95	1.86

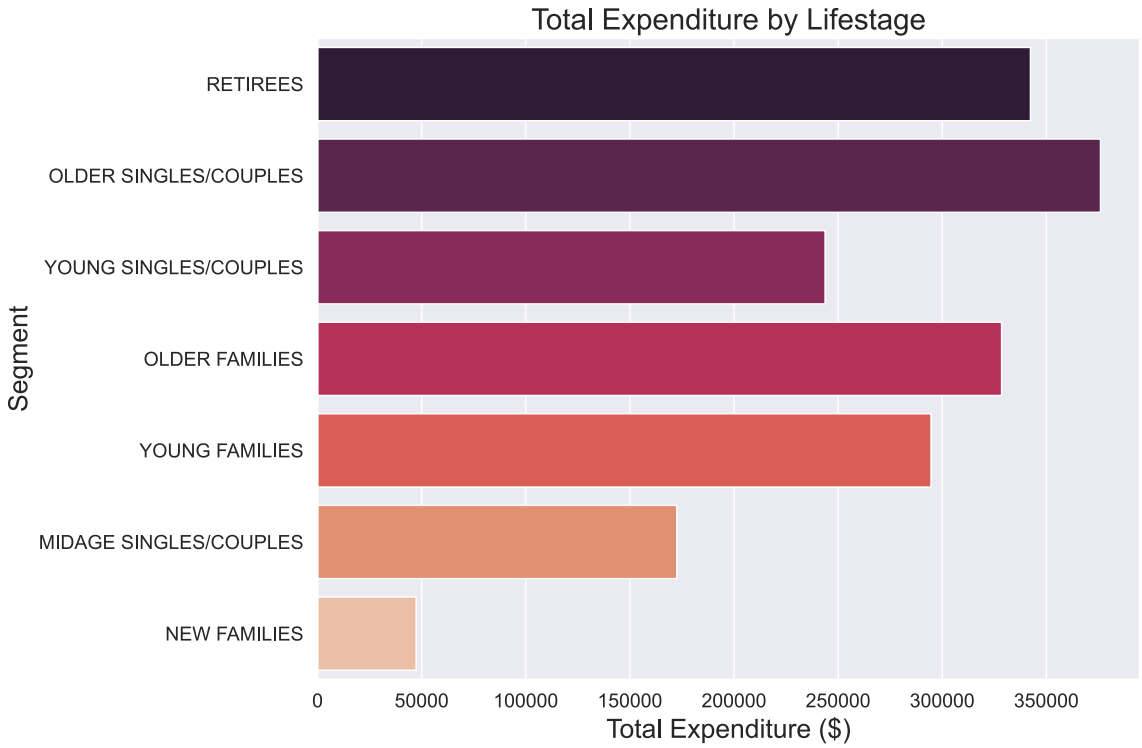
In [51]:

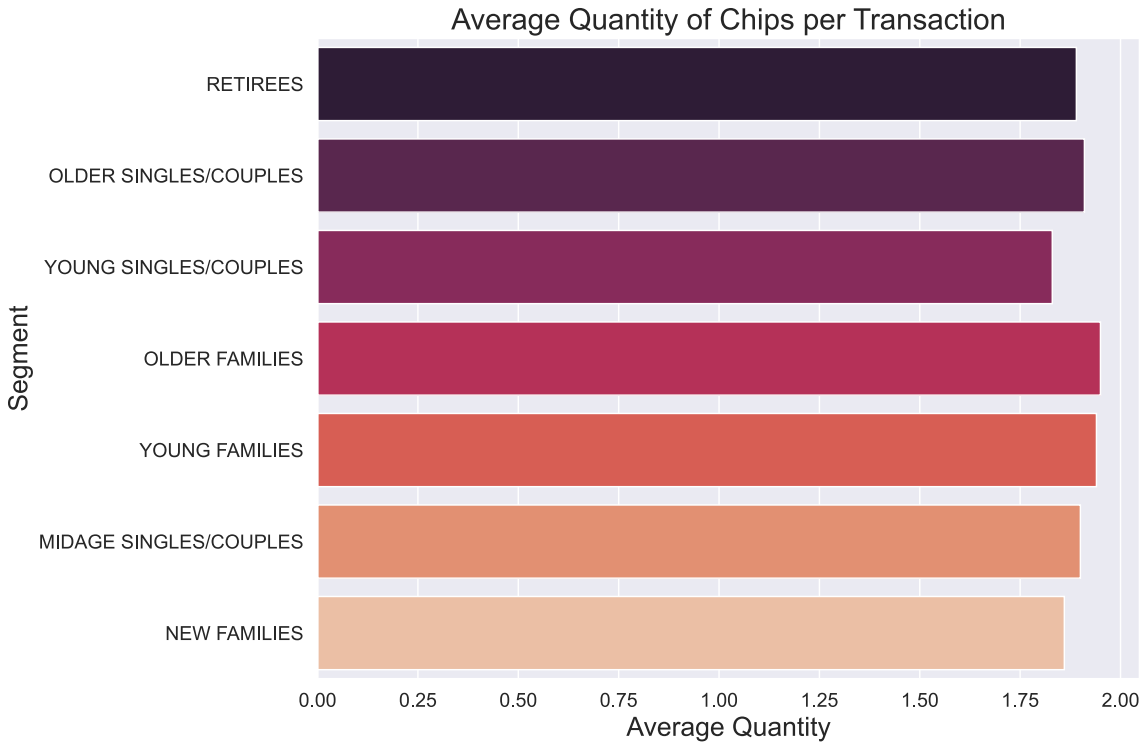
```
# QTY and Expenditure visualizations
# Average Expenditure
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'SEGMENT', x = 'AVG_SPEND', data = lifestage_analysis, palette =
'rocket')
ax.set_xlabel('Average Expenditure ($)', fontsize = 18)
ax.set_ylabel('Segment', fontsize = 18)
plt.title('Average Expenditure per Transaction by Lifestage', fontsize = 20)

# Total Expenditure
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'SEGMENT', x = 'TOTAL_SPEND', data = lifestage_analysis, palette =
'rocket')
ax.set_xlabel('Total Expenditure ($)', fontsize = 18)
ax.set_ylabel('Segment', fontsize = 18)
plt.title('Total Expenditure by Lifestage', fontsize = 20)

# Average QTY per transaction
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'SEGMENT', x = 'AVG_QTY', data = lifestage_analysis, palette = 'ro
cket')
ax.set_xlabel('Average Quantity', fontsize = 18)
ax.set_ylabel('Segment', fontsize = 18)
plt.title('Average Quantity of Chips per Transaction', fontsize = 20)

# Total QTY
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'SEGMENT', x = 'TOT_QTY', data = lifestage_analysis, palette = 'ro
cket')
ax.set_xlabel('Total Quantity', fontsize = 18)
ax.set_ylabel('Segment', fontsize = 18)
plt.title('Total Quantity of Chips by Lifestage', fontsize = 20)

# Avg Chip/unit price
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'SEGMENT', x = 'AVG_CHIP_PRICE', data = lifestage_analysis, palett
e = 'rocket')
ax.set_xlabel('Average Unit Price ($)', fontsize = 18)
ax.set_ylabel('Segment', fontsize = 18)
plt.title('Average Unit Price of Chips by Lifestage', fontsize = 20)
```

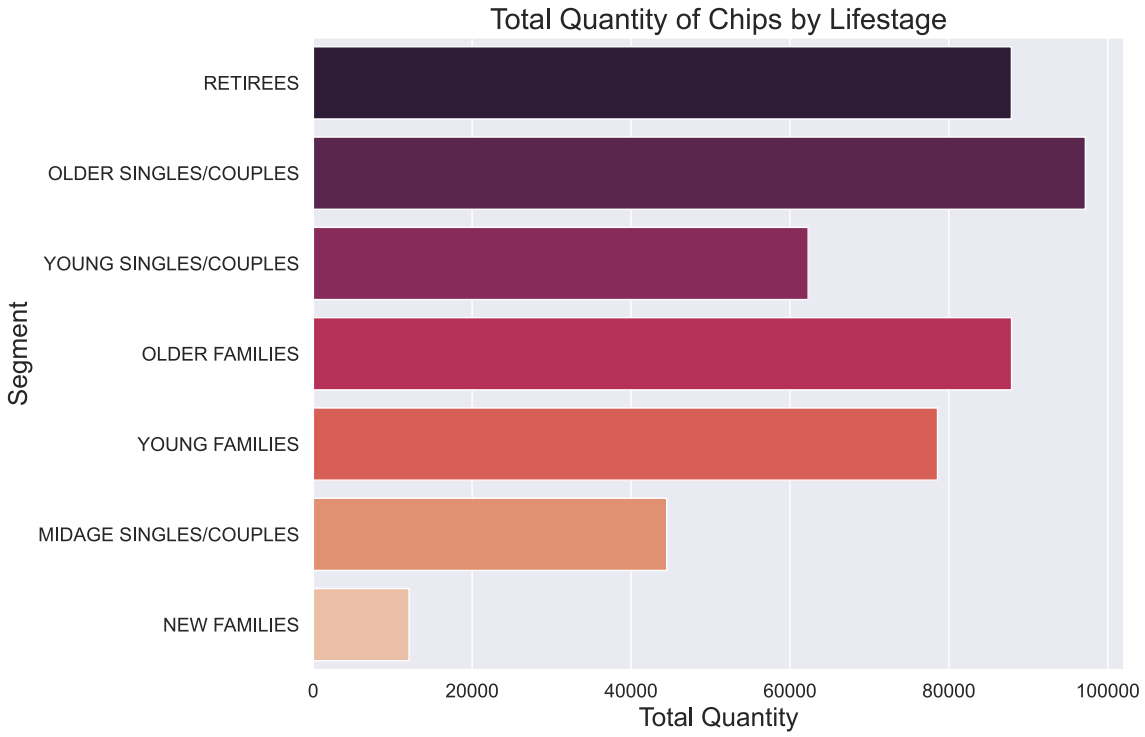
Out[51]:

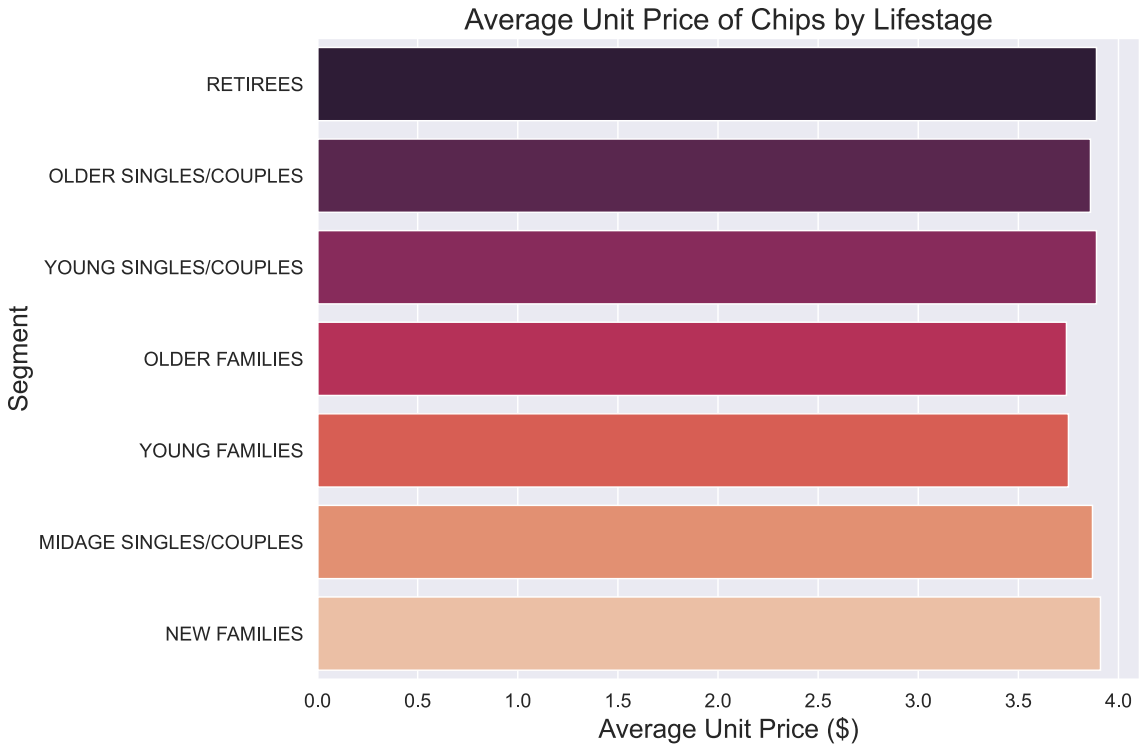
```
Text(0.5, 1.0, 'Average Unit Price of Chips by Lifestage')
```




Total Expenditure by Lifestage
RETIREES
OLDER SINGLES/COUPLES
YOUNG SINGLES/COUPLES
OLDER FAMILIES
YOUNG FAMILIES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
Segment
0
50000
100000
150000
200000
250000
300000
350000
Total Expenditure ($)


Average Quantity of Chips per Transaction
RETIREES
OLDER SINGLES/COUPLES
YOUNG SINGLES/COUPLES
OLDER FAMILIES
YOUNG FAMILIES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
Segment
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Average Quantity


Total Quantity of Chips by Lifestage
RETIREES
OLDER SINGLES/COUPLES
YOUNG SINGLES/COUPLES
OLDER FAMILIES
YOUNG FAMILIES
MIDAGE SINGLES/COUPLES
NEW FAMILIES
Segment
0
20000
40000
60000
80000
100000
Total Quantity



OBSERVATIONS:

EXPENDITURE

- All segments spend a very similar amount (+$7), with only slight deviation of 20c of each other.
- Older Singles/Couples spend the most on chips, followed by Retirees and Older Families.

QUANTITY

- All segments average between 1.75 to 2 chip packets per transaction.
- Older families have bought the most, with New Families the least.

AVERAGE UNIT PRICE

- The average price of each unit of chip sold for all subsets are all between $3.7-3.9$

In [52]:

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

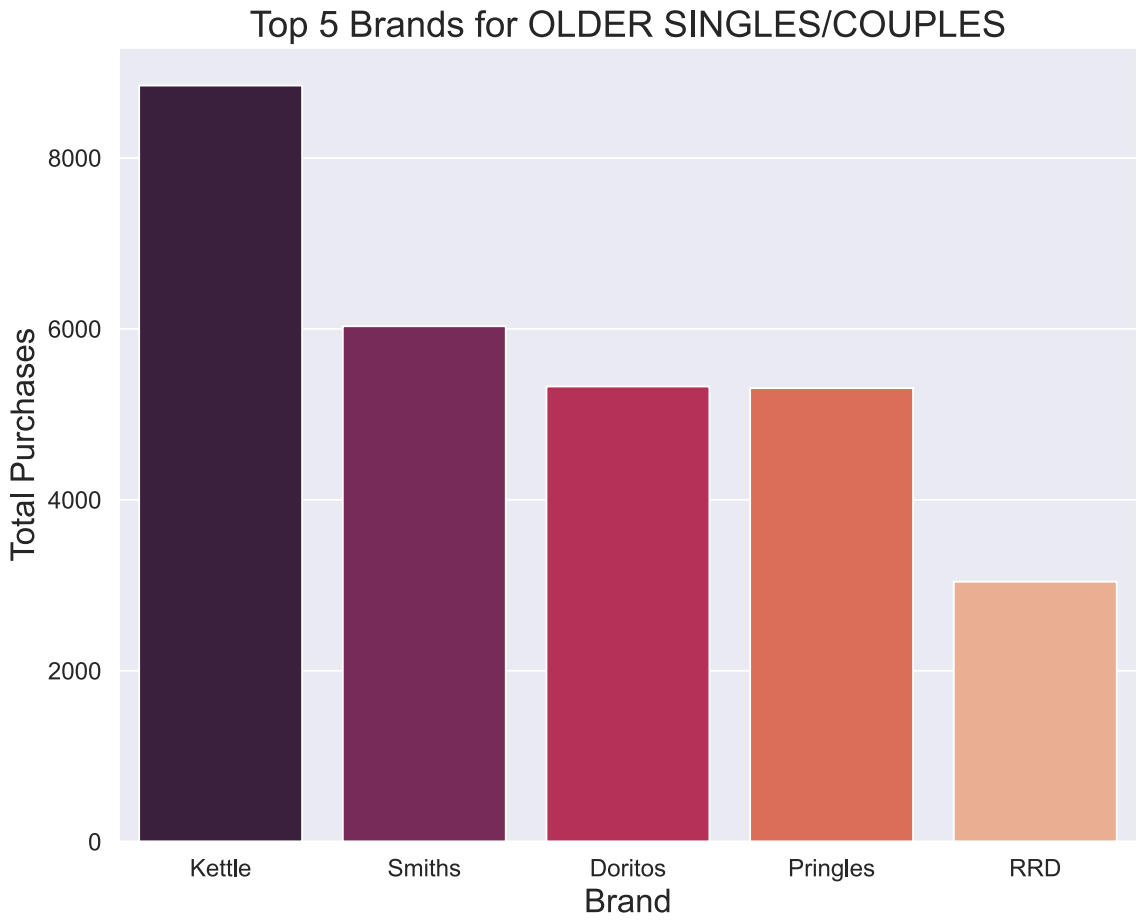
Out[52]:

```
OLDER SINGLES/COUPLES     50792
RETIREES                  46431
OLDER FAMILIES            45158
YOUNG FAMILIES            40494
YOUNG SINGLES/COUPLES     33968
MIDAGE SINGLES/COUPLES    23398
NEW FAMILIES               6497
Name: LIFESTAGE, dtype: int64
```

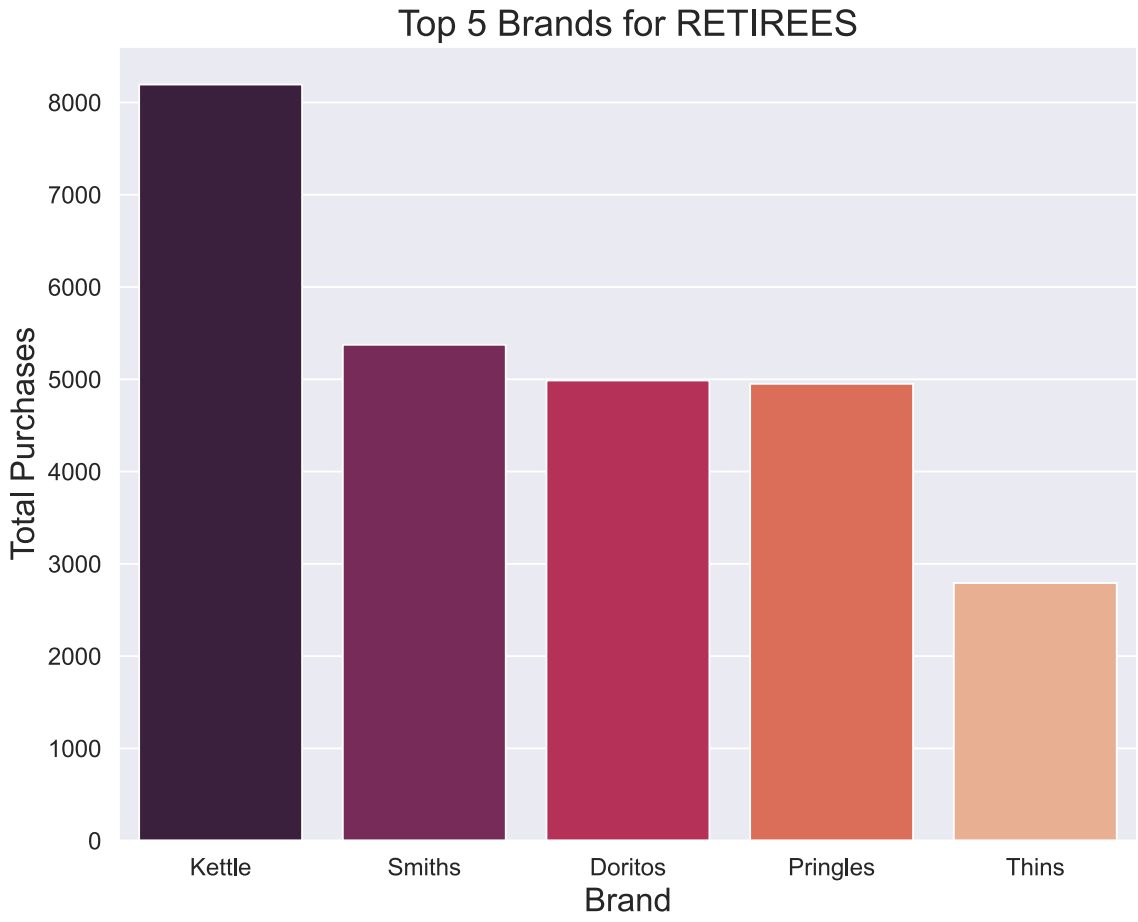
In [53]:

```
# Check favorite brands of customers by lifestage

lifestages = []
for i in dict(data['LIFESTAGE'].value_counts()):
    lifestages.append(i)

for subset in lifestages:
    dataframe = data[data['LIFESTAGE'] == subset]
    print('-------------------------------------------------------------------')
    print(subset)
    print(dataframe['BRAND_NAME'].value_counts())
    viz = pd.DataFrame.from_dict(dict(dataframe['BRAND_NAME'].value_counts()), orient =
'index', columns = ['count'])
    #Show visualization of top 5 brands
    values = viz['count'][:5]
    labels = viz.index[:5]
    plt.figure(figsize = (10,8))
    ax = sns.barplot(x = labels, y = values, palette = 'rocket')
    ax.set_xlabel('Brand', fontsize = 18)
    ax.set_ylabel('Total Purchases', fontsize = 18)
    plt.title(f'Top 5 Brands for {subset}', fontsize = 20)
    plt.show()
```

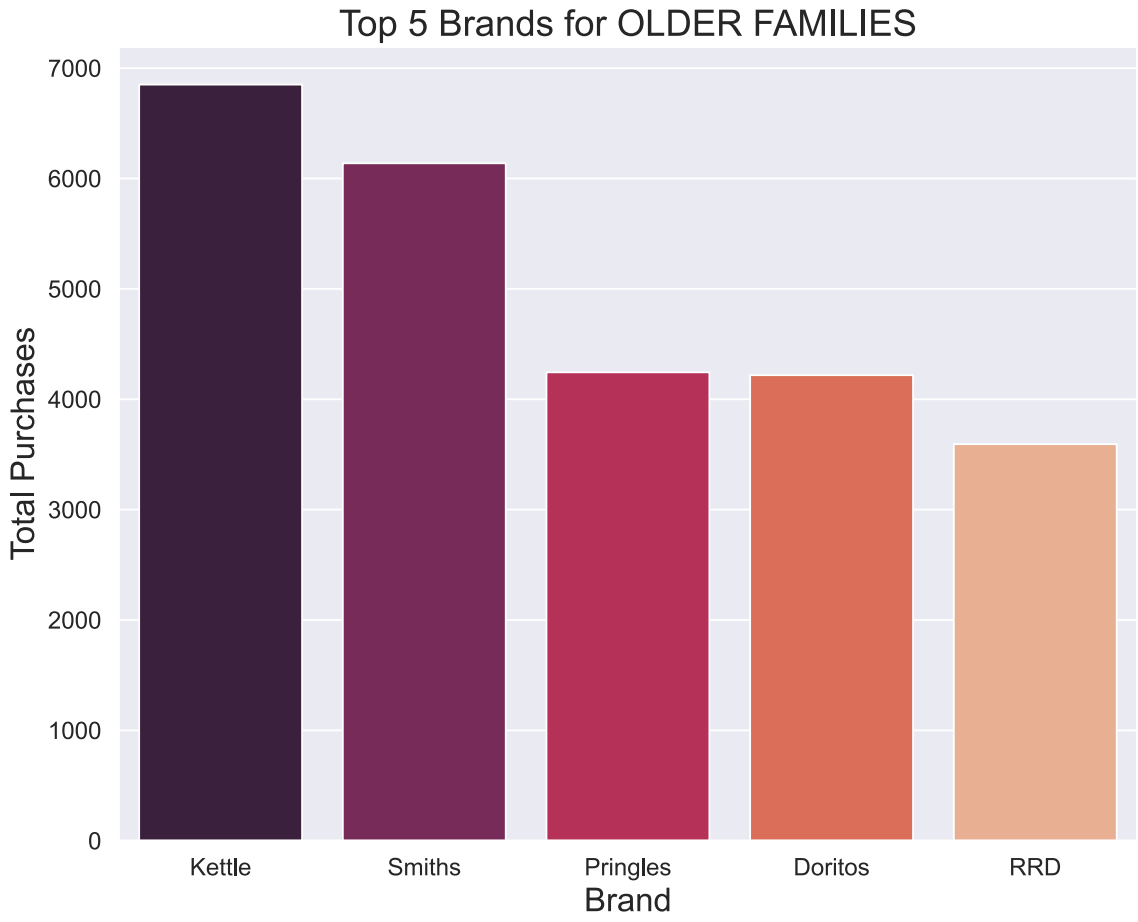
```
------------------------------------------------------------------
OLDER SINGLES/COUPLES
Kettle          8847
Smiths          6031
Doritos         5326
Pringles        5307
RRD             3042
Thins           2969
Infuzions       2962
Woolworths      2316
Tostitos        2039
Cobs            2036
Twisties        1949
GrnWves         1603
NCC             1466
Tyrrells        1340
Cheezels         966
CCs              851
Sunbites         584
Cheetos          580
Burger Rings     292
French Fries     286
Name: BRAND_NAME, dtype: int64
```



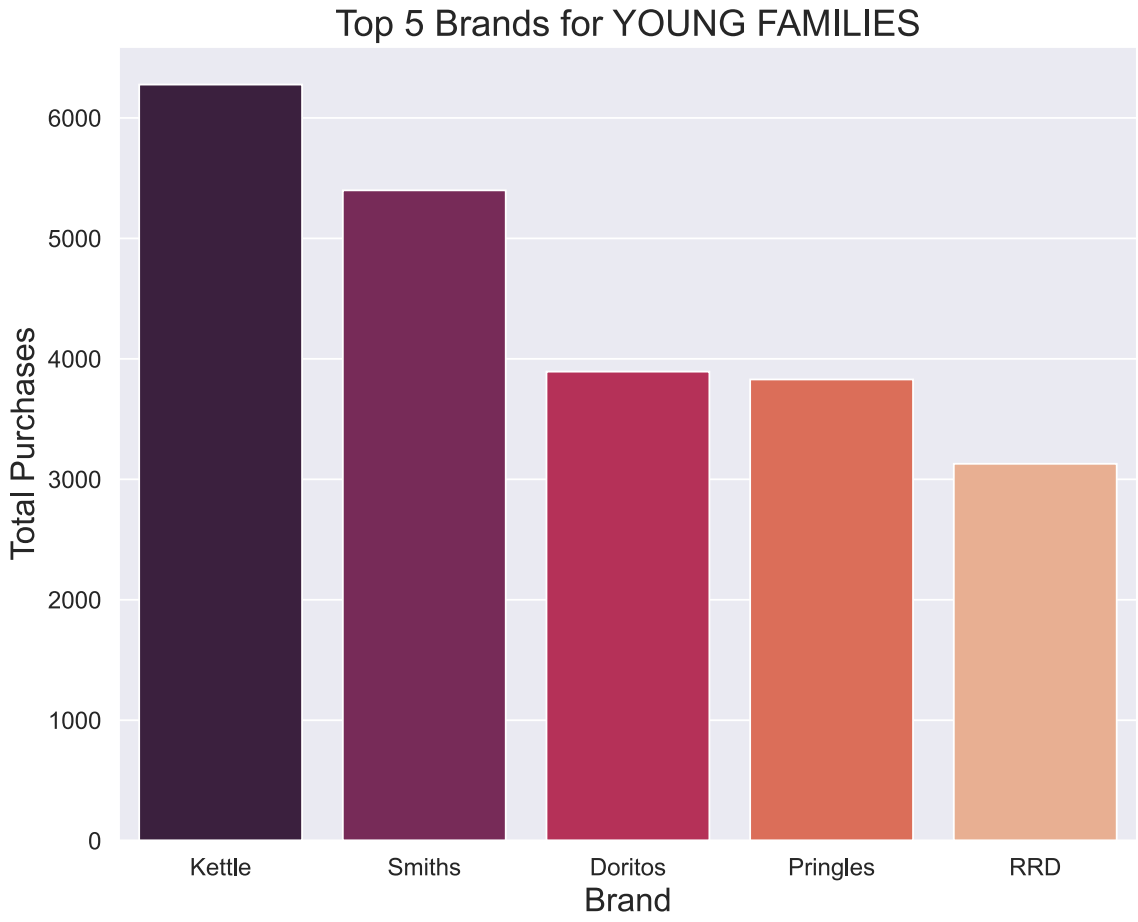
```
------------------------------------------------------------------------
RETIREES
Kettle          8194
Smiths          5374
Doritos         4987
Pringles        4951
Thins           2792
Infuzions       2719
RRD             2716
Woolworths      1945
Twisties        1890
Cobs            1884
Tostitos        1850
GrnWves         1466
NCC             1286
Tyrrells        1259
Cheezels         867
CCs              741
Sunbites         535
Cheetos          491
Burger Rings     256
French Fries     228
Name: BRAND_NAME, dtype: int64
```



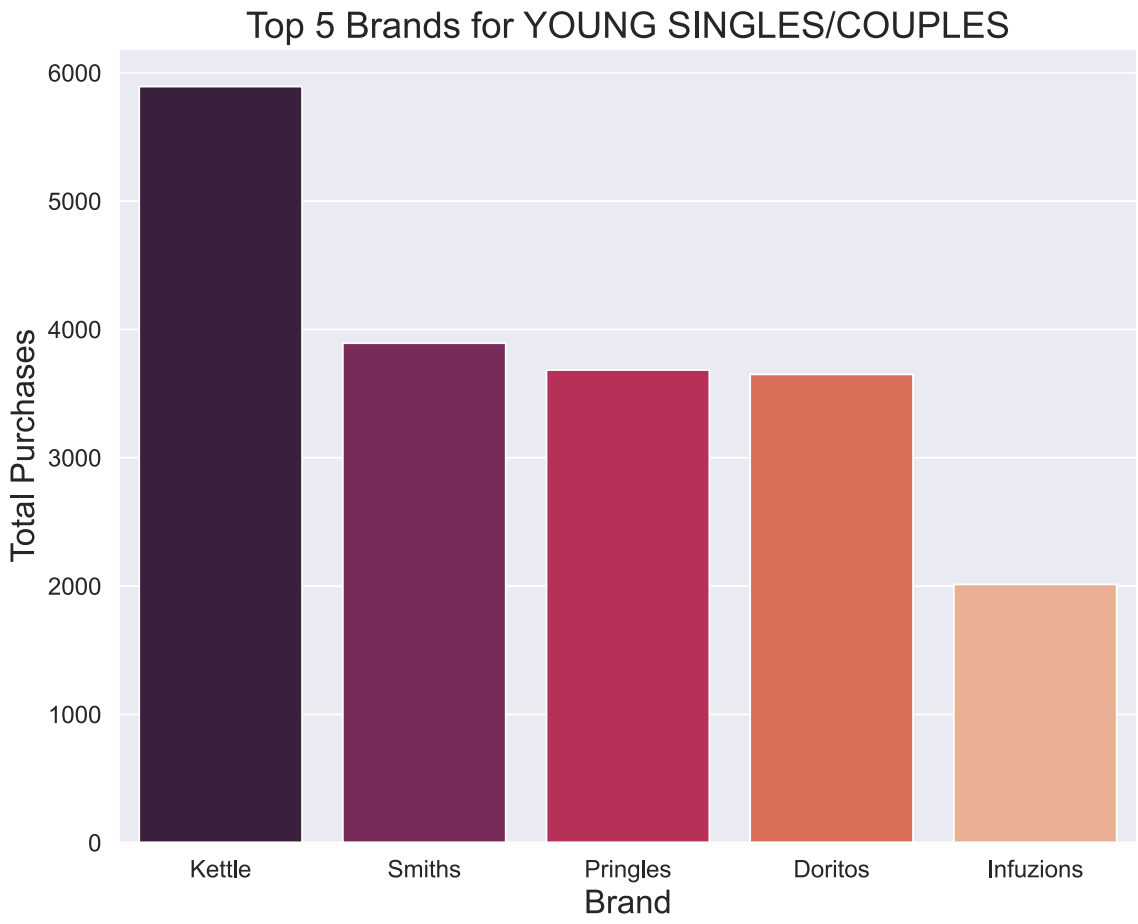
```
------------------------------------------------------------------------
OLDER FAMILIES
Kettle          6851
Smiths          6138
Pringles        4244
Doritos         4218
RRD             3593
Woolworths      2609
Infuzions       2496
Thins           2475
Twisties        1644
Cobs            1624
NCC             1571
Tostitos        1546
GrnWves         1429
Tyrrells        1093
CCs              941
Cheezels         813
Sunbites         622
Cheetos          615
Burger Rings     353
French Fries     283
Name: BRAND_NAME, dtype: int64
```



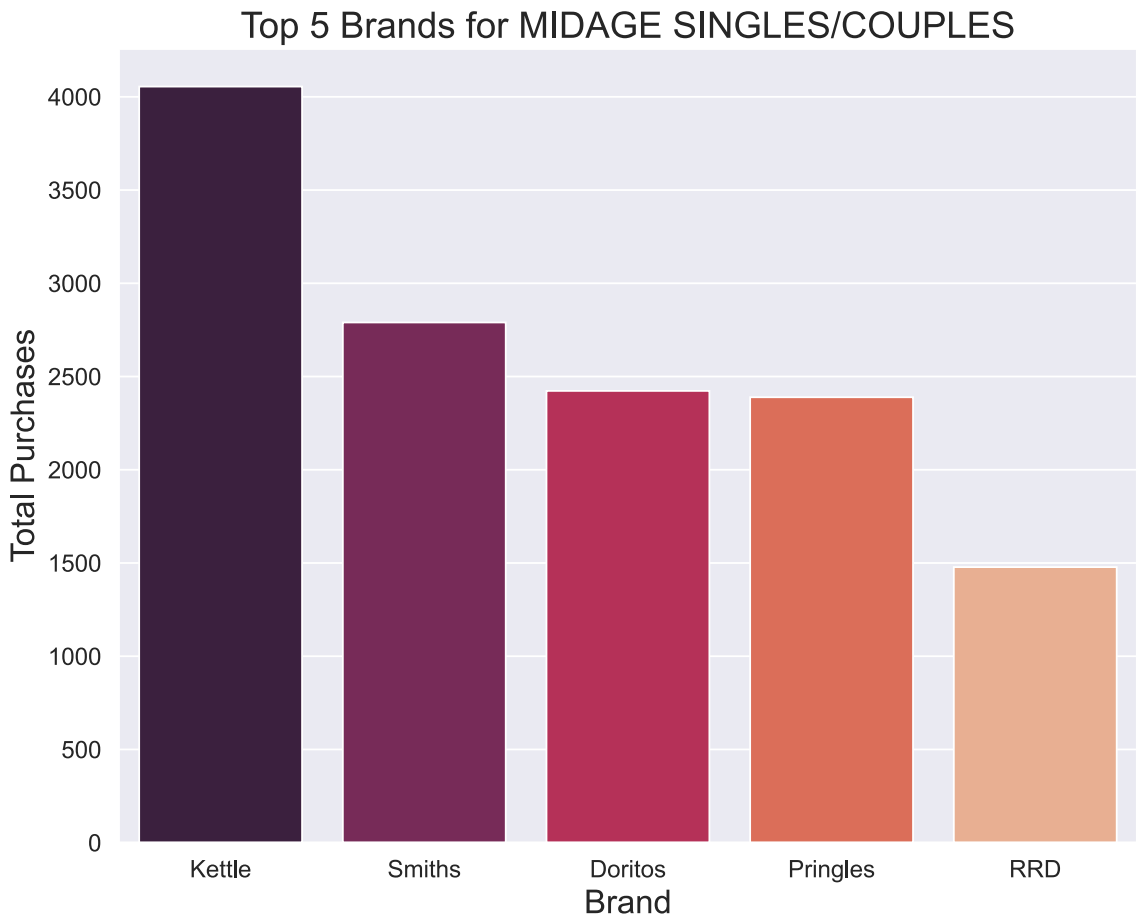
```
------------------------------------------------------------------------
YOUNG FAMILIES
Kettle          6277
Smiths          5399
Doritos         3894
Pringles        3829
RRD             3129
Infuzions       2215
Woolworths      2211
Thins           2186
Cobs            1504
Tostitos        1467
Twisties        1412
NCC             1367
GrnWves         1221
Tyrrells         997
CCs              898
Cheezels         771
Sunbites         596
Cheetos          550
Burger Rings     293
French Fries     278
Name: BRAND_NAME, dtype: int64
```



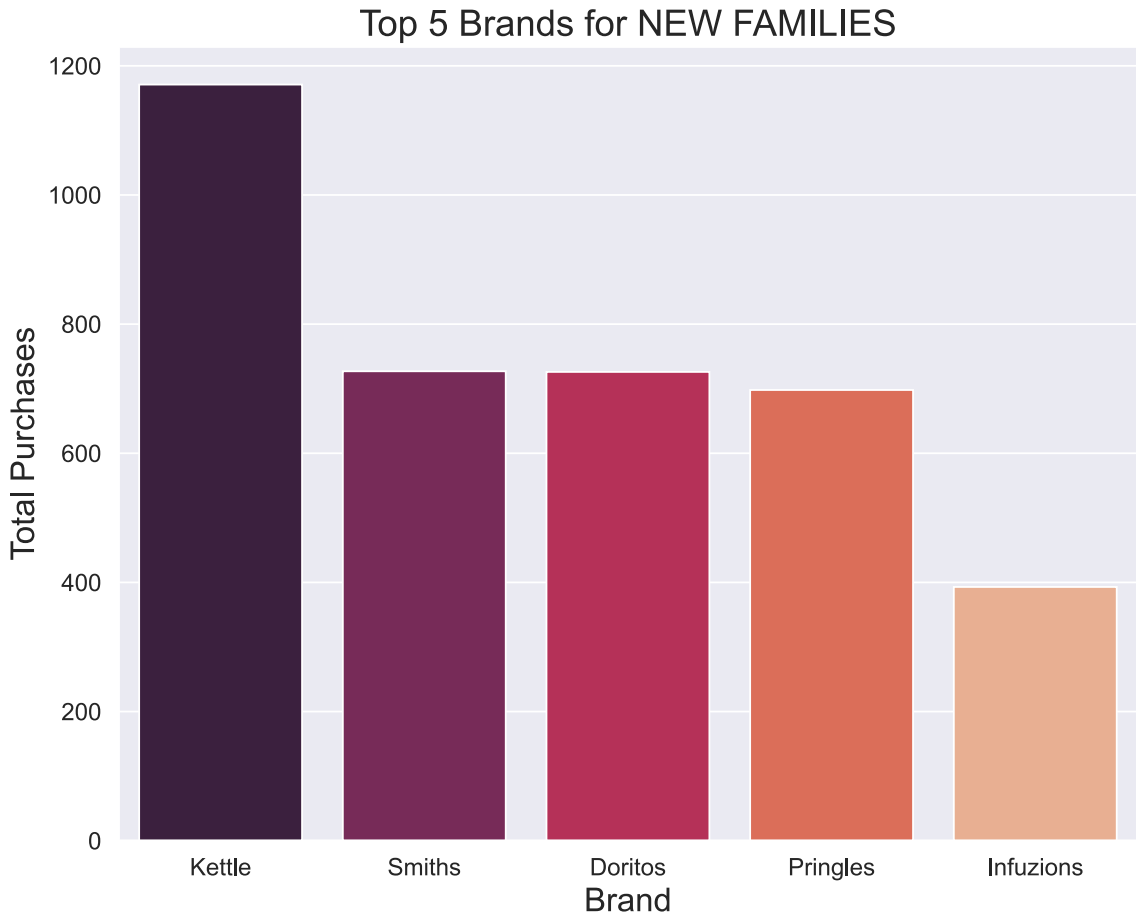
```
----------------------------------------------------------------------
YOUNG SINGLES/COUPLES
Kettle          5893
Smiths          3893
Pringles        3684
Doritos         3650
Infuzions       2013
RRD             2008
Thins           1959
Woolworths      1447
Cobs            1396
Twisties        1395
Tostitos        1368
GrnWves         1076
Tyrrells         955
NCC              927
Cheezels         613
CCs              594
Cheetos          364
Sunbites         361
French Fries     194
Burger Rings     178
Name: BRAND_NAME, dtype: int64
```



```
----------------------------------------------------------------------
MIDAGE SINGLES/COUPLES
Kettle          4055
Smiths          2790
Doritos         2423
Pringles        2389
RRD             1478
Infuzions       1403
Thins           1316
Woolworths      1032
Cobs             961
Twisties         931
Tostitos         924
GrnWves          732
NCC              694
Tyrrells         611
Cheezels         443
CCs              433
Cheetos          265
Sunbites         249
Burger Rings     152
French Fries     117
Name: BRAND_NAME, dtype: int64
```



```
------------------------------------------------------------------------
NEW FAMILIES
Kettle          1171
Smiths           727
Doritos          726
Pringles         698
Infuzions        393
Thins            378
RRD              355
Cobs             288
Tostitos         277
Woolworths       276
Twisties         233
GrnWves          213
Tyrrells         187
NCC              158
Cheezels         129
CCs               93
Cheetos           62
Sunbites          61
Burger Rings      40
French Fries      32
Name: BRAND_NAME, dtype: int64
```



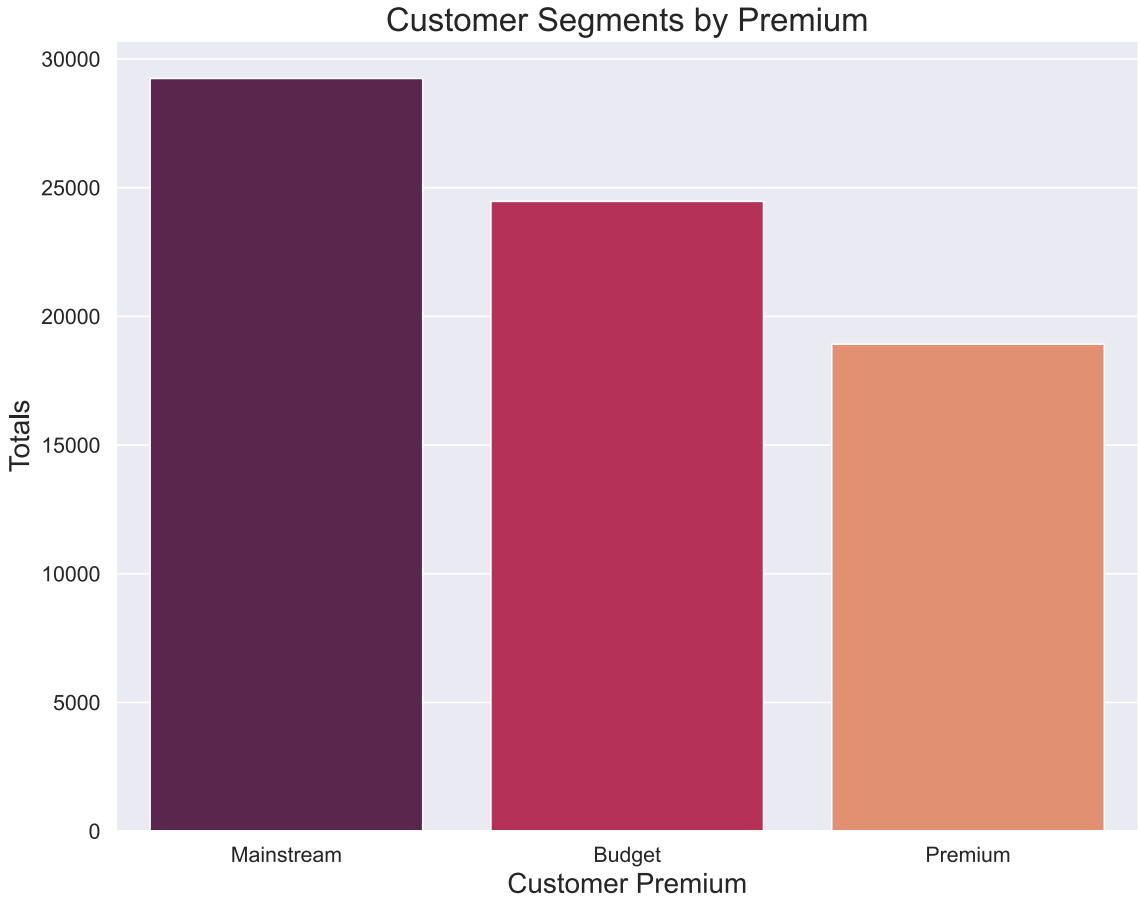
OBSERVATIONS:

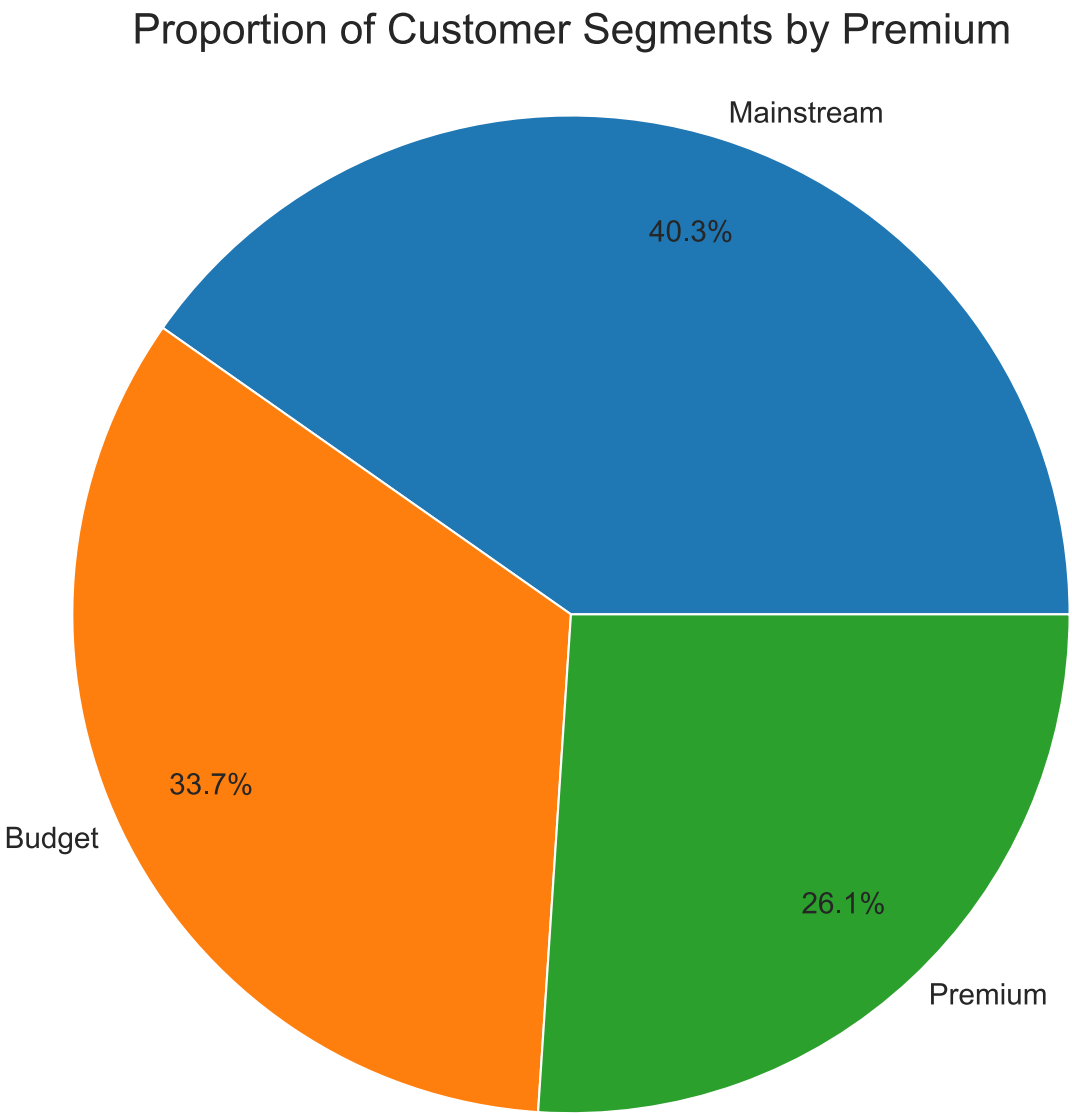
- For all lifestage customer segments, the top 5 brands appear to be the same.
- The top 2 brands per segment are Kettle and Smiths, followed by either Doritos or Pringles.

CUSTOMER PREMIUM ANALYSIS

In [54]:

```
# Make a DataFrame
premium = pd.DataFrame.from_dict(dict(data2['PREMIUM_CUSTOMER'].value_counts()), orient
= 'index', columns = ['Counts'])
premium
# Show Distribution
plt.figure(figsize = (10,8))
ax = sns.set_style('darkgrid')
ax = sns.barplot(x = premium.index, y = premium['Counts'], palette = 'rocket')
ax.set_xlabel('Customer Premium', fontsize = 17)
ax.set_ylabel('Totals', fontsize = 17)
ax.axes.set_title('Customer Segments by Premium', fontsize = 20)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
plt.tight_layout()

# Show Proportions
# Pie chart
pie, ax = plt.subplots(figsize = (10, 8))
labels = premium.index
plt.pie(x = premium['Counts'], autopct="%.1f%%", labels=labels, pctdistance=0.8, labeld
istance = 1.05, textprops = {'fontsize': 14}, radius = 0.5)
plt.title("Proportion of Customer Segments by Premium", fontsize=20)
ax.axis('square')
plt.tight_layout()
```


Customer Segments by Premium
30000
25000
20000
15000
10000
5000
0
Totals
Mainstream
Budget
Premium
Customer Premium


Proportion of Customer Segments by Premium
Mainstream
40.3%
33.7%
Budget
26.1%
Premium

OBSERVATIONS:

- There are 3 segments for which describe a customer's premium: Budget, Mainstream and Premium.
- Majority of customer transactions are part of the Mainstream segment.

In [55]:

```
# Check unique customers, grouping by premium

query = """
        SELECT PREMIUM_CUSTOMER, COUNT(DISTINCT(LYLTY_CARD_NBR)) as 'UNIQUE_COUNTS'
        FROM data
        GROUP BY PREMIUM_CUSTOMER
        """
unique_premium_count = ps.sqldf(query,locals())
unique_premium_count
```

Out[55]:

	PREMIUM_CUSTOMER	UNIQUE_COUNTS
0	Budget	24006
1	Mainstream	28734
2	Premium	18547

In [56]:

```
# Merge Columns
uc = [28734,24006,18547]
premium['unique_cust'] = uc
premium.rename(columns = {'Counts':'Transactions'}, inplace = True)
premium
```

Out[56]:

	Transactions	unique_cust
Mainstream	29245	28734
Budget	24470	24006
Premium	18922	18547

In [57]:

```
# Perform sales summary for premiums

premium_sales_sum = []
premium_sales_avg = []
premium_avg_qty = []
premium_tot_qty = []
premium_avg_unit_price = []

# Get average spend and total sum
# Check average qty and total quantity of chips
# Check average unit price per segment

for i in premium.index:
    subset = data[data['PREMIUM_CUSTOMER'] == i]
    premium_sales_sum.append(round(np.sum(subset['TOT_SALES']), 2))
    premium_sales_avg.append(round(np.mean(subset['TOT_SALES']), 2))
    premium_avg_qty.append(round(np.mean(subset['PROD_QTY']), 2))
    premium_tot_qty.append(round(np.sum(subset['PROD_QTY'])))
    premium_avg_unit_price.append(round(np.mean(subset['AVG_CHIP_PRICE']), 2))

# Make columns in premium df
premium['AVG_SPEND'] = premium_sales_avg
premium['TOT_SPEND'] = premium_sales_sum
premium['AVG_QTY'] = premium_avg_qty
premium['TOT_QTY'] = premium_tot_qty
premium['AVG_CHIP_PRICE'] = premium_avg_unit_price
premium.reset_index(inplace=True)
premium.rename(columns = {'index':'Segment'}, inplace = True)
premium
```

Out[57]:

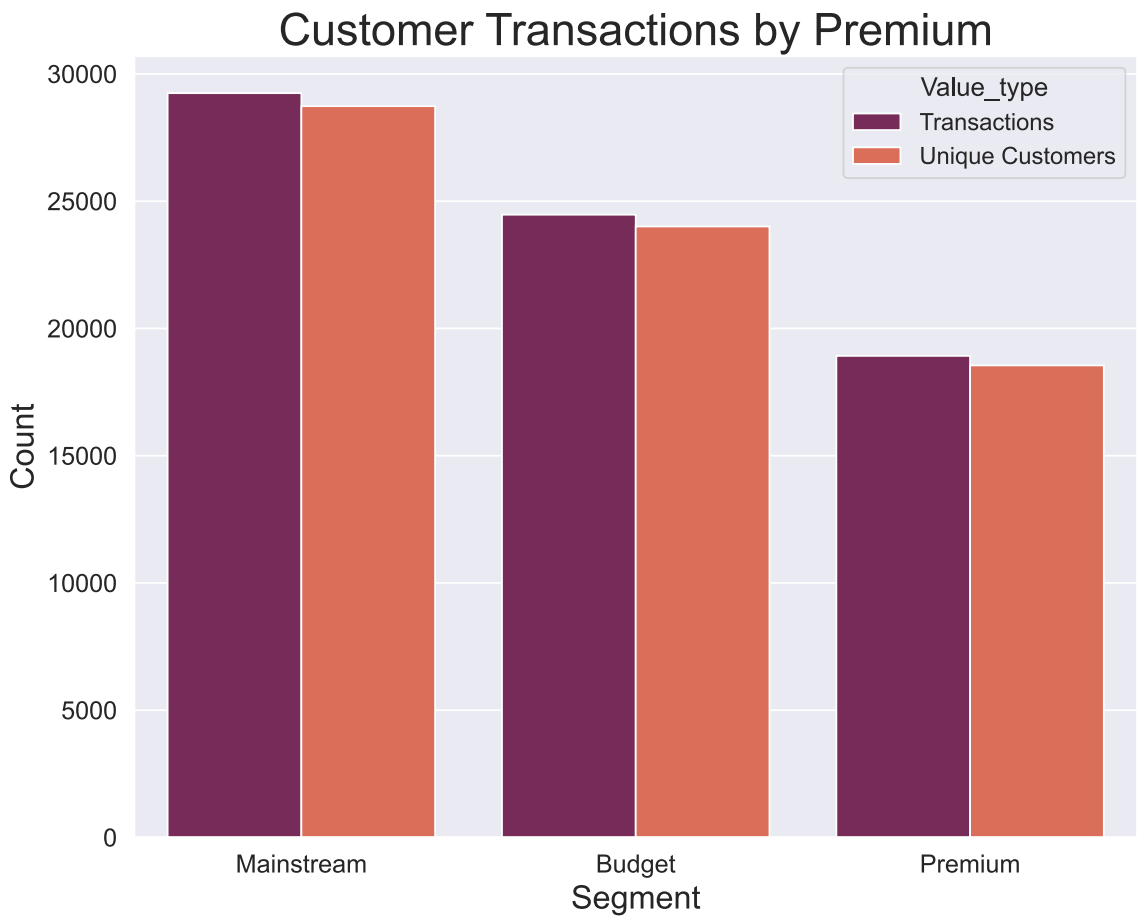
	Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	A
0	Mainstream	29245	28734	7.37	700859.70	1.90	180779	
1	Budget	24470	24006	7.28	631402.65	1.91	165772	
2	Premium	18922	18547	7.28	472905.45	1.91	123845	

In [58]:

```
# Transaction analysis dataframe

segments = []
for i in premium.Segment:
    segments.append(i)
for i in premium.Segment:
    segments.append(i)

counts = []
for i in premium.Transactions:
    counts.append(i)
for i in premium.unique_cust:
    counts.append(i)

types = ['Transactions', 'Transactions', 'Transactions',
        'Unique Customers', 'Unique Customers', 'Unique Customers']

segments = pd.DataFrame(segments, columns = ['Segment'])
segments['Counts'] = counts
segments['Value_type'] = types

# Make viz
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'Counts', data = segments, hue = 'Value_type', pale
tte= 'rocket')
ax.set_xlabel('Segment', fontsize = 18)
ax.set_ylabel('Count', fontsize = 18)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.tick_params(labelsize = 14)
plt.title('Customer Transactions by Premium', fontsize = 25)
plt.tight_layout
```

Out[58]:

```
<function matplotlib.pyplot.tight_layout(pad=1.08, h_pad=None, w_pad=None,
rect=None)>
```



OBSERVATIONS:

- The number of transactions by each segment closely follows the populatio distribution of each segment.

In [59]:

```
premium
```

Out[59]:

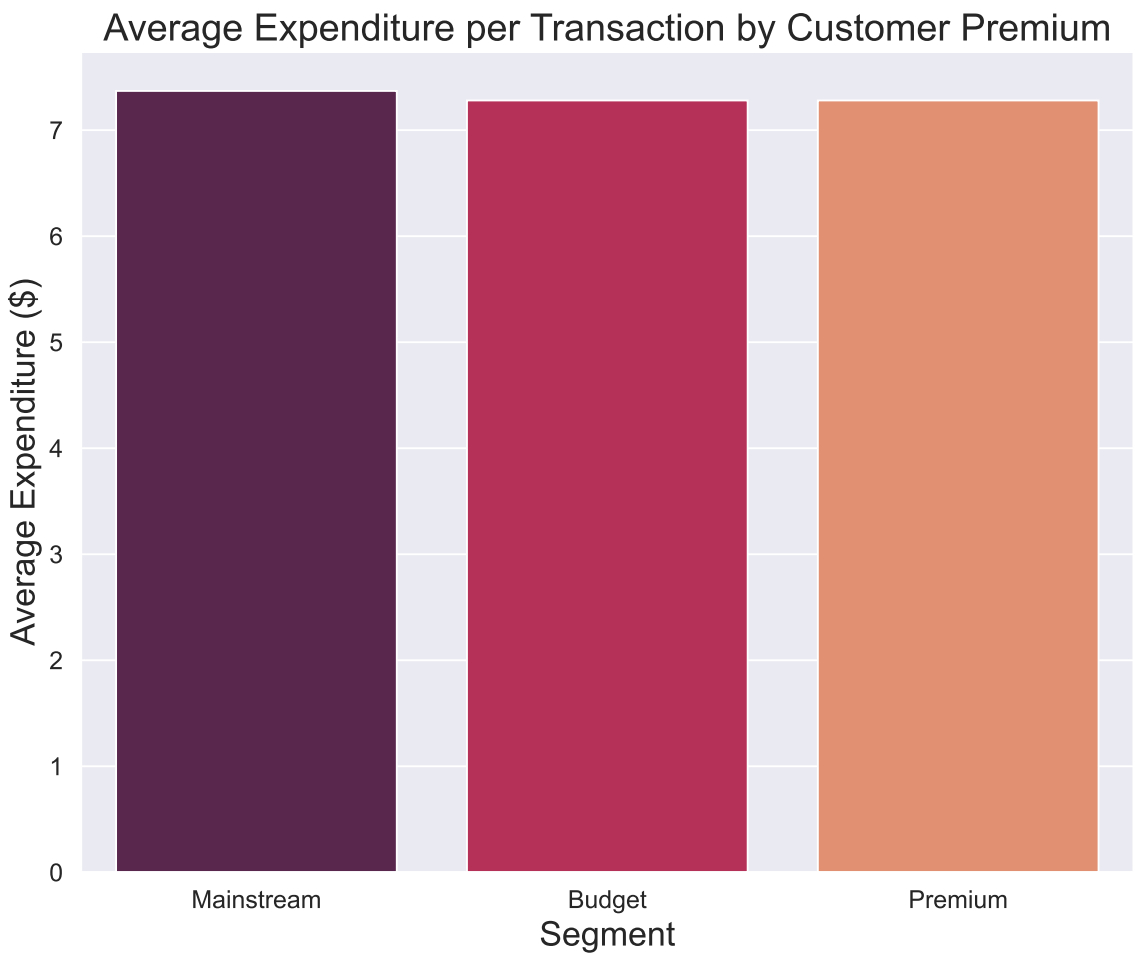
	Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	A
0	Mainstream	29245	28734	7.37	700859.70	1.90	180779	
1	Budget	24470	24006	7.28	631402.65	1.91	165772	
2	Premium	18922	18547	7.28	472905.45	1.91	123845	

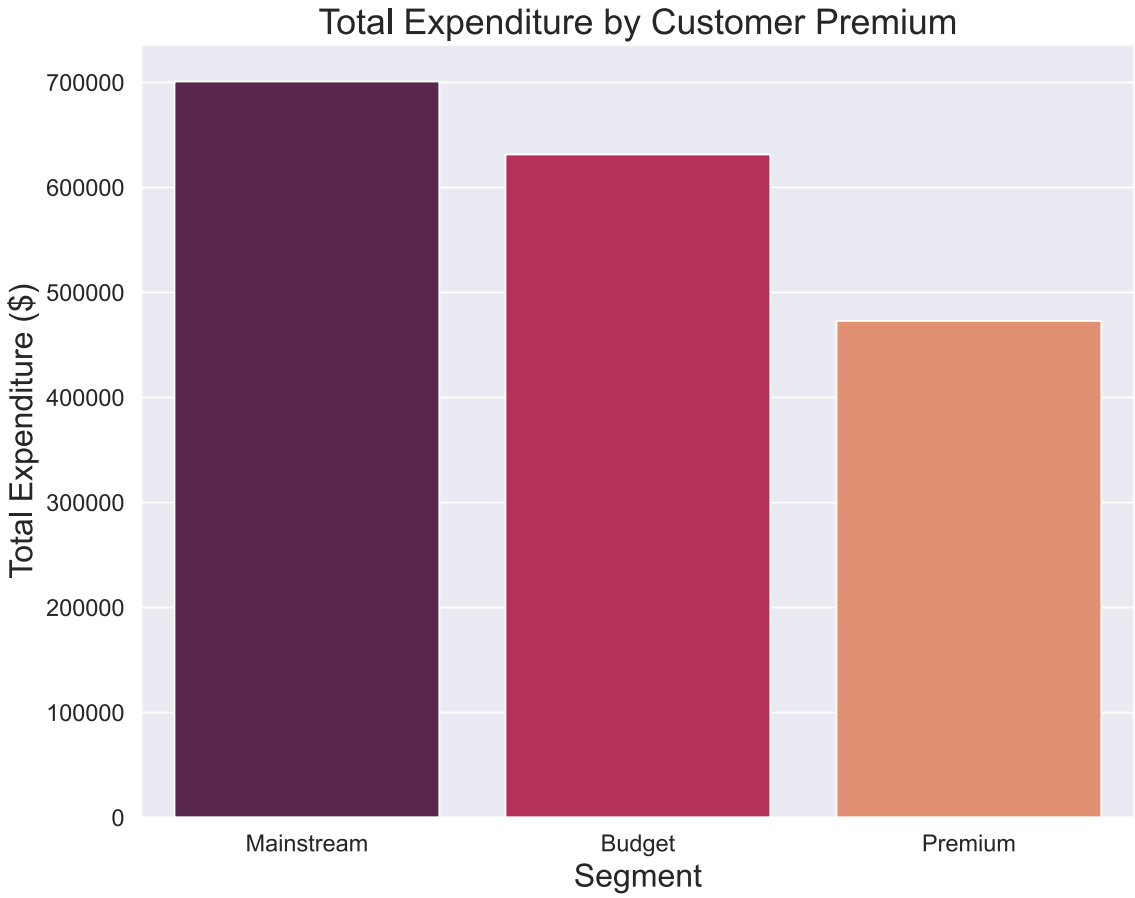
In [60]:

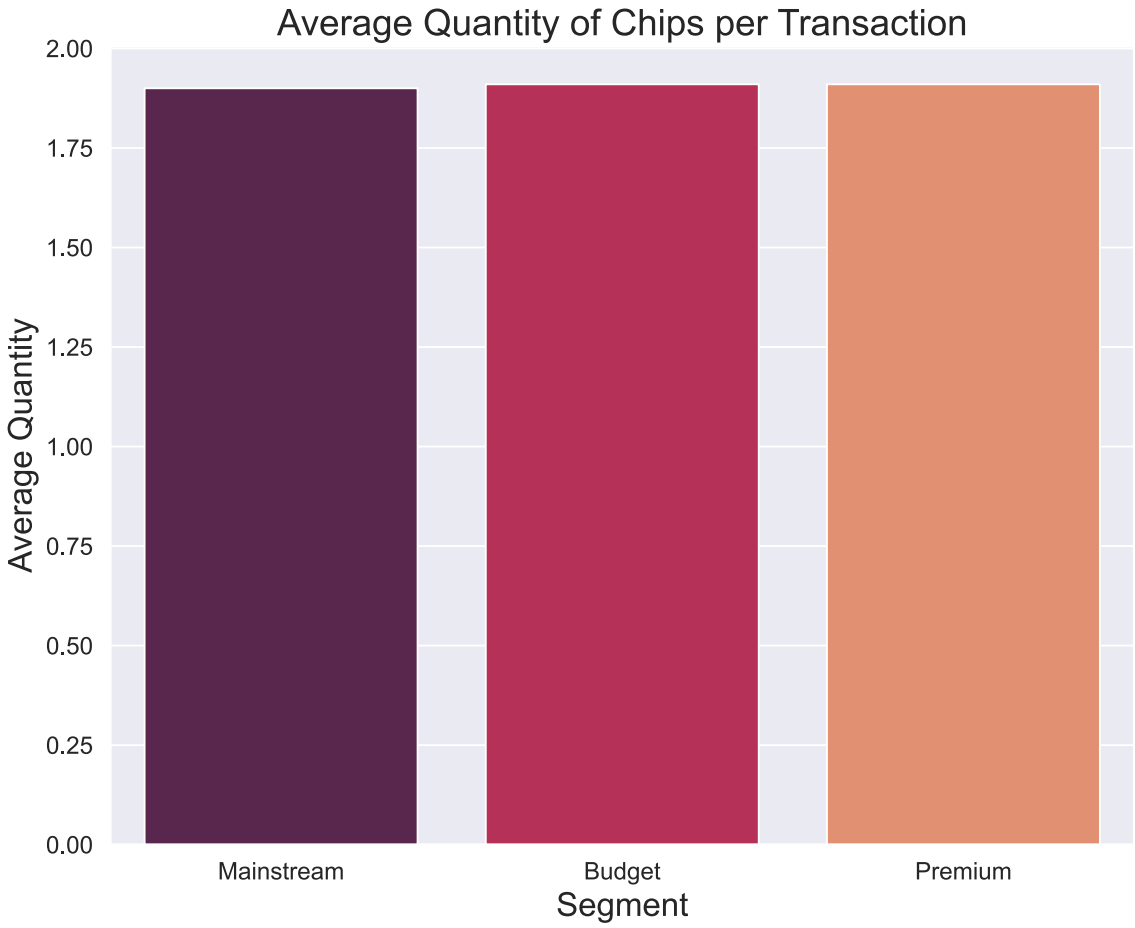
```
# QTY and Expenditure visualizations
# Average Expenditure
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'AVG_SPEND', data = premium, palette = 'rocket')
ax.set_ylabel('Average Expenditure ($)', fontsize = 18)
ax.set_xlabel('Segment', fontsize = 18)
plt.title('Average Expenditure per Transaction by Customer Premium', fontsize = 20)

# Total Expenditure
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'TOT_SPEND', data = premium, palette = 'rocket')
ax.set_ylabel('Total Expenditure ($)', fontsize = 18)
ax.set_xlabel('Segment', fontsize = 18)
plt.title('Total Expenditure by Customer Premium', fontsize = 20)

# Average QTY per transaction
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'AVG_QTY', data = premium, palette = 'rocket')
ax.set_ylabel('Average Quantity', fontsize = 18)
ax.set_xlabel('Segment', fontsize = 18)
plt.title('Average Quantity of Chips per Transaction', fontsize = 20)

# Total QTY
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'TOT_QTY', data = premium, palette = 'rocket')
ax.set_ylabel('Total Quantity', fontsize = 18)
ax.set_xlabel('Segment', fontsize = 18)
plt.title('Total Quantity of Chips Bought by Premium', fontsize = 20)

# Avg Chip/unit price
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'AVG_CHIP_PRICE', data = premium, palette = 'rocke
t')
ax.set_ylabel('Average Unit Price ($)', fontsize = 18)
ax.set_xlabel('Segment', fontsize = 18)
plt.title('Average Unit Price of Chips by Premium', fontsize = 20)
```

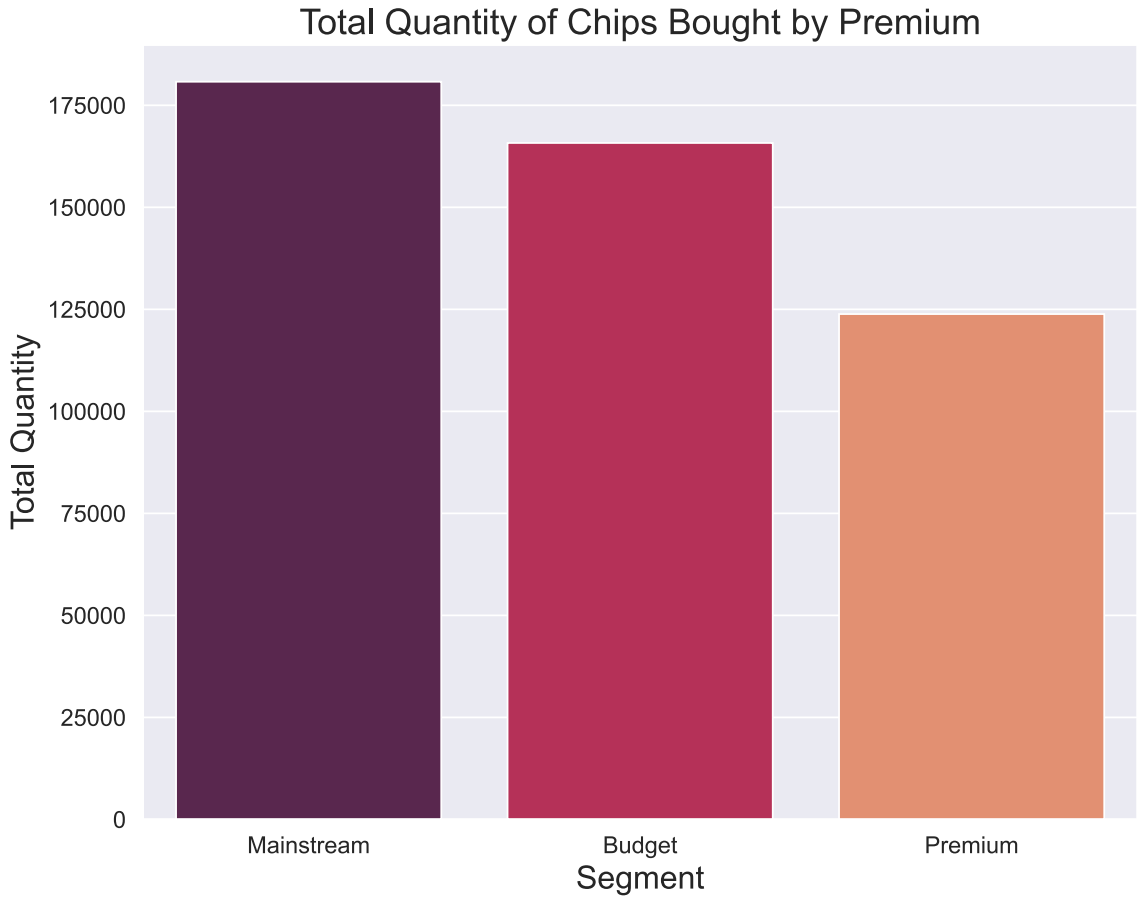
Out[60]:

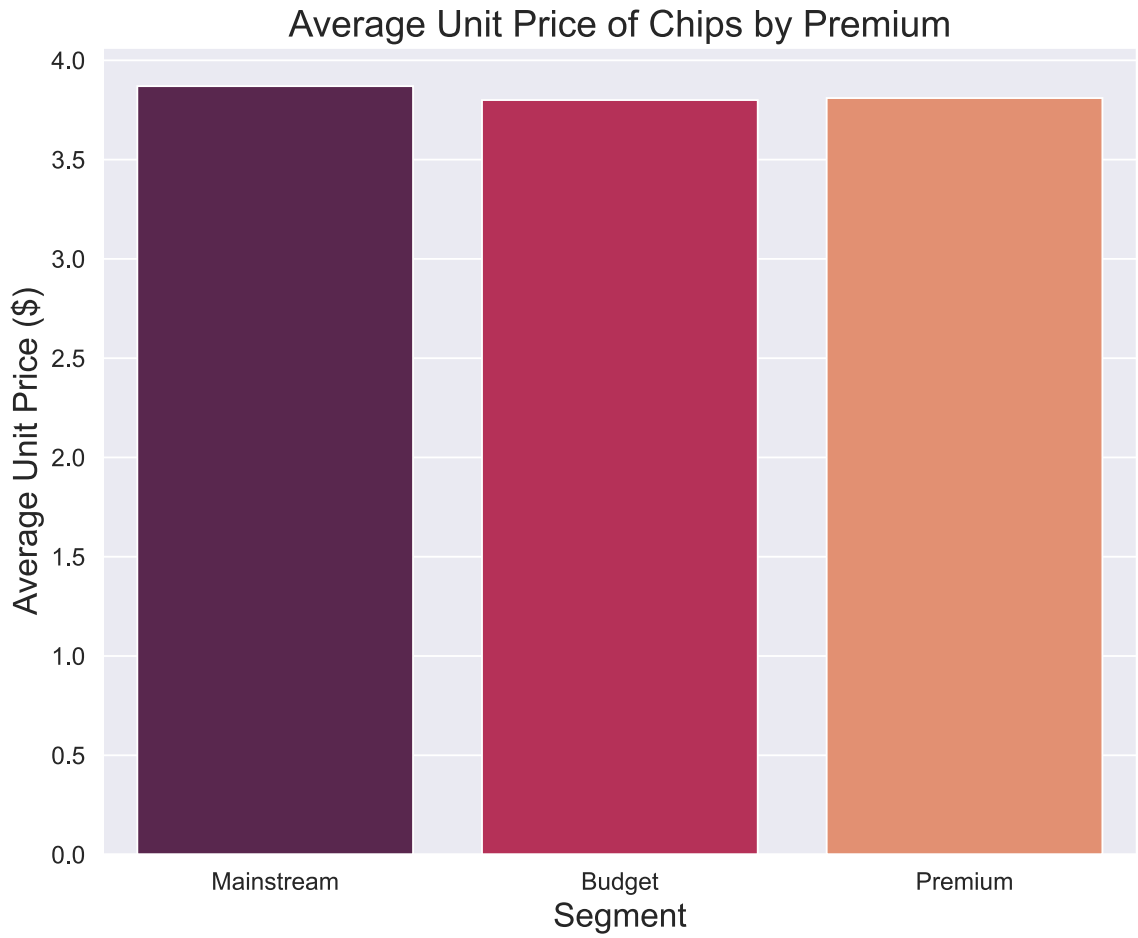
```
Text(0.5, 1.0, 'Average Unit Price of Chips by Premium')
```




Total Expenditure by Customer Premium
700000
600000
500000
400000
300000
200000
100000
0
Total Expenditure ($)
Mainstream
Budget
Premium
Segment


Average Quantity of Chips per Transaction
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Average Quantity
Mainstream
Budget
Premium
Segment

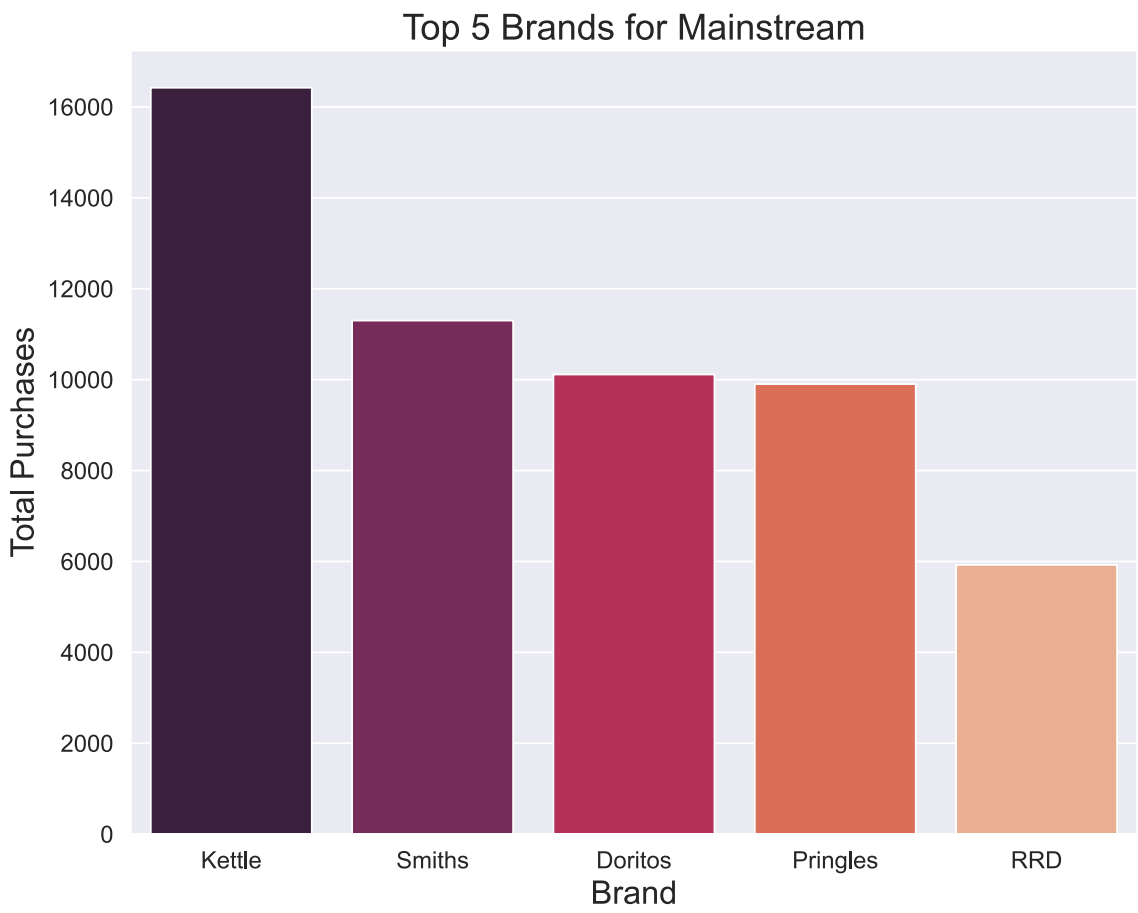

Total Quantity of Chips Bought by Premium
Total Quantity
175000
150000
125000
100000
75000
50000
25000
0
Mainstream
Budget
Premium
Segment


Average Unit Price of Chips by Premium
Average Unit Price ($)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Mainstream
Budget
Premium
Segment

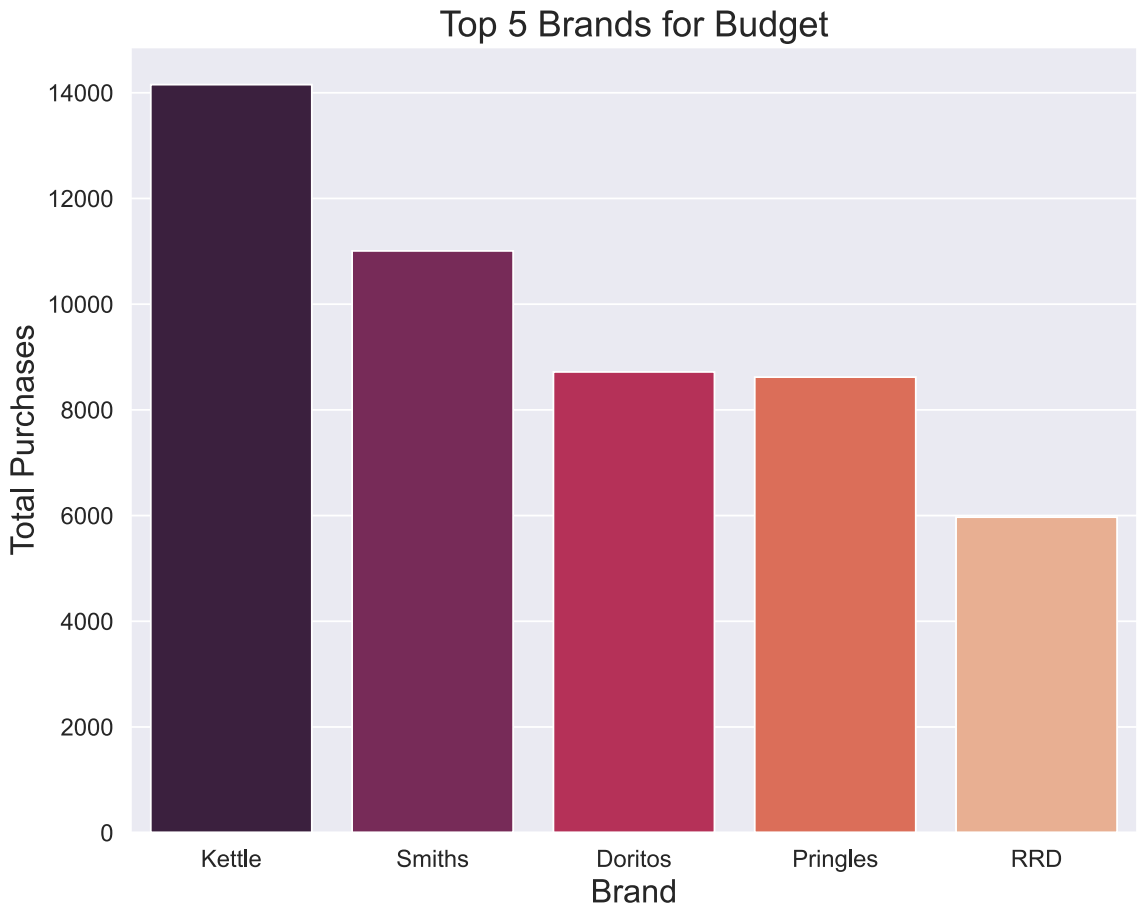
In [61]:

```
# Check favorite brands of customers by their premium

premiums = []
for i in dict(data['PREMIUM_CUSTOMER'].value_counts()):
    premiums.append(i)

for subset in premiums:
    dataframe = data[data['PREMIUM_CUSTOMER'] == subset]
    print('----------------------------------------------------------------------')
    print(subset)
    print(dataframe['BRAND_NAME'].value_counts())
    viz = pd.DataFrame.from_dict(dict(dataframe['BRAND_NAME'].value_counts()), orient =
'index', columns = ['count'])
    #Show visualization of top 5 brands
    values = viz['count'][:5]
    labels = viz.index[:5]
    plt.figure(figsize = (10,8))
    ax = sns.barplot(x = labels, y = values, palette = 'rocket')
    ax.set_xlabel('Brand', fontsize = 18)
    ax.set_ylabel('Total Purchases', fontsize = 18)
    plt.title(f'Top 5 Brands for {subset}', fontsize = 20)
    plt.show()
```

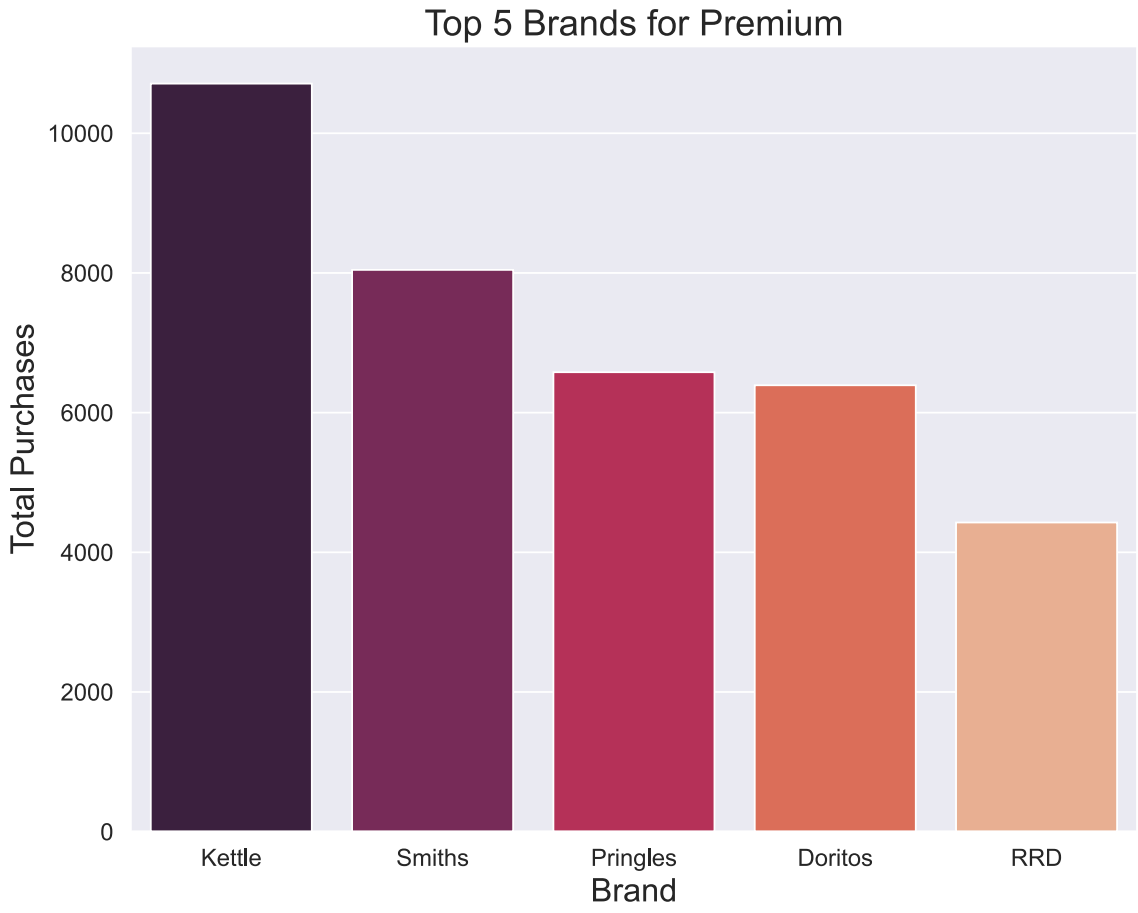
```
-----------------------------------------------------------------------
Mainstream
Kettle          16423
Smiths          11300
Doritos         10114
Pringles         9903
RRD              5924
Infuzions        5550
Thins            5436
Woolworths       4130
Cobs             3889
Twisties         3785
Tostitos         3737
GrnWves          3037
NCC              2657
Tyrrells         2583
Cheezels         1735
CCs              1631
Cheetos          1111
Sunbites         1042
Burger Rings      548
French Fries      507
Name: BRAND_NAME, dtype: int64
```


Top 5 Brands for Mainstream
Total Purchases
16000
14000
12000
10000
8000
6000
4000
2000
0
Kettle
Smiths
Doritos
Pringles
RRD
Brand

```
----------------------------------------------------------------------
Budget
Kettle          14154
Smiths          11008
Doritos          8718
Pringles         8620
RRD              5970
Thins            4931
Infuzions        4922
Woolworths       4444
Cobs             3274
Tostitos         3236
Twisties         3229
NCC              2785
GrnWves          2656
Tyrrells         2195
CCs              1679
Cheezels         1625
Sunbites         1146
Cheetos          1051
Burger Rings      579
French Fries      539
Name: BRAND_NAME, dtype: int64
```



```
------------------------------------------------------------------------
Premium
Kettle          10711
Smiths           8044
Pringles         6579
Doritos          6392
RRD              4427
Infuzions        3729
Thins            3708
Woolworths       3262
Cobs             2530
Tostitos         2498
Twisties         2440
GrnWves          2047
NCC              2027
Tyrrells         1664
Cheezels         1242
CCs              1241
Sunbites          820
Cheetos           765
Burger Rings      437
French Fries      372
Name: BRAND_NAME, dtype: int64
```



OBSERVATIONS:

- For all customer premium segments, the top 5 brands are all homogeneous, including order.
- The only observable difference is slight variation between ranks 3 & 4 (doritos & pringles), and 5 & 6 (rrd and infuzions).

Check Proportion of lifestage segments within premiums

In [62]:

```
data
```

Out[62]:

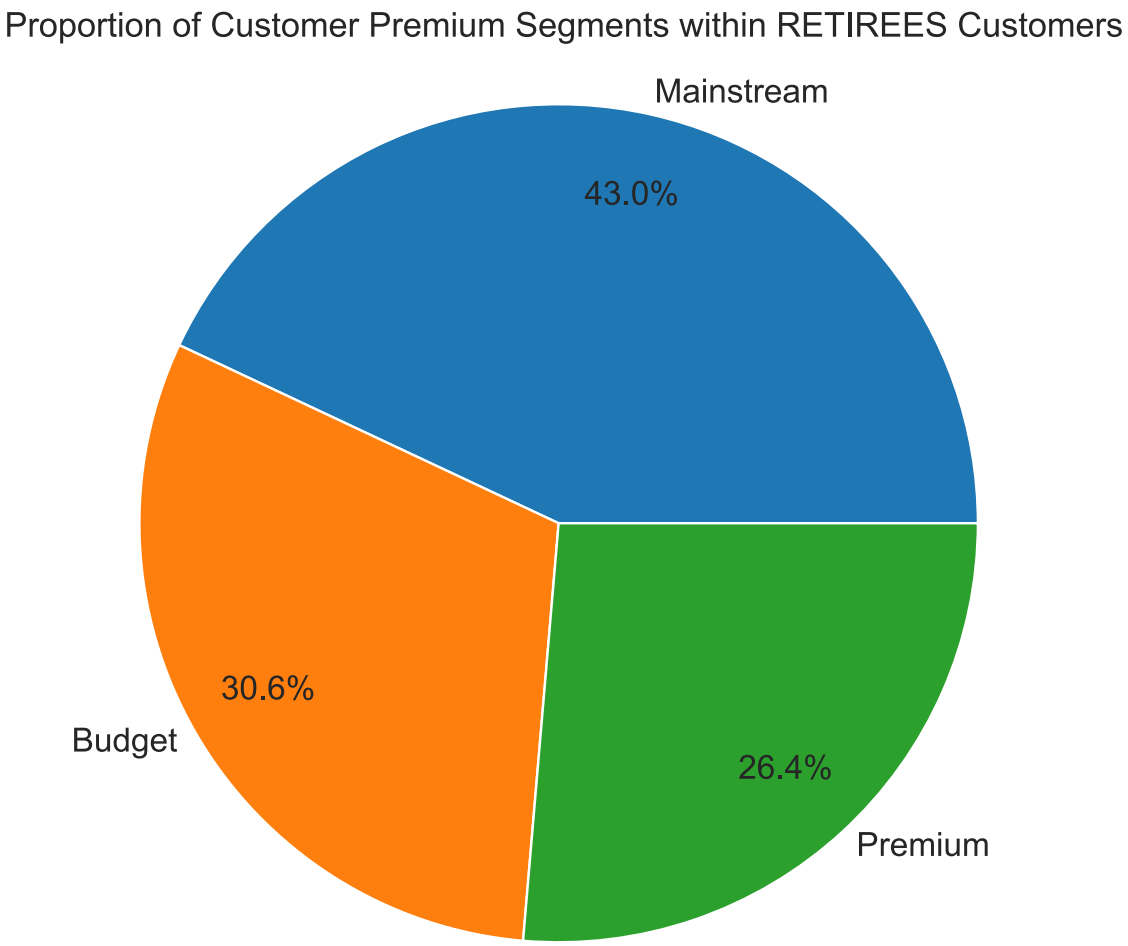
	real_date	STORE_NBR	TXN_ID	PROD_NAME	BRAND_NAME	PACKET_SIZE	PR
0	2018-07-01	9	8808	Smiths Thinly Cut Roast Chicken 175g	Smiths	175	
1	2018-07-01	86	84237	Red Rock Deli Sp Salt & Truffle 150G	RRD	150	
2	2018-07-01	129	132474	Smith Crinkle Cut Mac N Cheese 150g	Smiths	150	
3	2018-07-01	58	53145	Pringles Sthrn FriedChicken 134g	Pringles	134	
4	2018-07-01	97	97311	WW Crinkle Cut Chicken 175g	Woolworths	175	
...	...	...	...	...	...	...	
246735	2019-06-30	91	89519	Thins Chips Seasonedchicken 175g	Thins	175	
246736	2019-06-30	84	83704	Doritos Corn Chips Nacho Cheese 170g	Doritos	170	
246737	2019-06-30	24	20917	Smiths Crinkle Cut Chips Chs&Onion170g	Smiths	170	
246738	2019-06-30	199	198068	Doritos Corn Chips Nacho Cheese 170g	Doritos	170	
246739	2019-06-30	220	219497	Dorito Corn Chp Supreme 380g	Doritos	380	

246738 rows × 12 columns

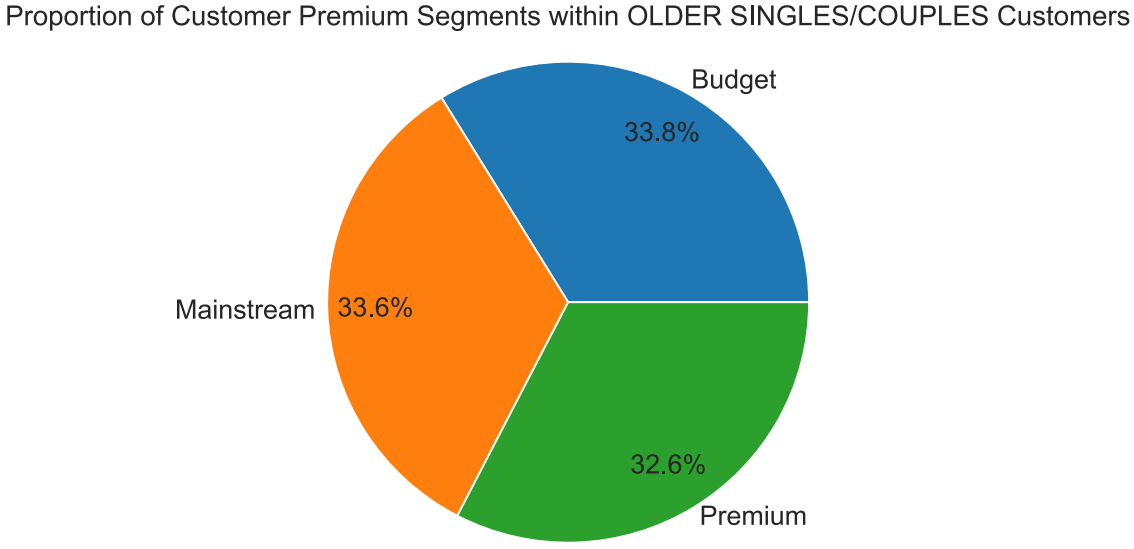
In [63]:

```
segments = ['RETIREES', 'OLDER SINGLES/COUPLES', 'YOUNG SINGLES/COUPLES', 'OLDER FAMILI
ES', 'YOUNG FAMILIES', 'MIDAGE SINGLES/COUPLES', 'NEW FAMILIES']

for i in segments:
    df = data[data['LIFESTAGE'] == i]
    dist = pd.DataFrame.from_dict(dict(df['PREMIUM_CUSTOMER'].value_counts()), orient =
'index', columns = ['Count'])
    print('------------------------------')
    print(i)
    print(dist)
    # Pie chart
    pie, ax = plt.subplots(figsize = (7,7))
    labels = dist.index
    plt.pie(x = dist['Count'], autopct = "%.1f%%", labels = labels, pctdistance = 0.8,
labeldistance = 1.05, textprops = {'fontsize':14}, radius = 0.5)
    plt.title(f'Proportion of Customer Premium Segments within {i} Customers')
    ax.axis('square')
    plt.tight_layout()
    plt.show()
```

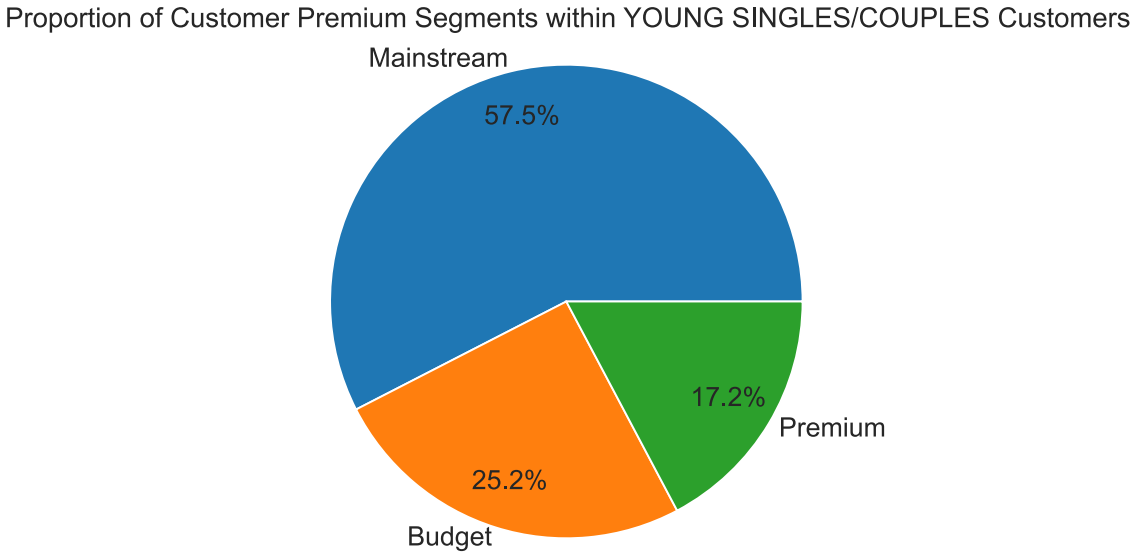
```
------------------------------
RETIREES
            Count
Mainstream  19970
Budget      14225
Premium     12236
```



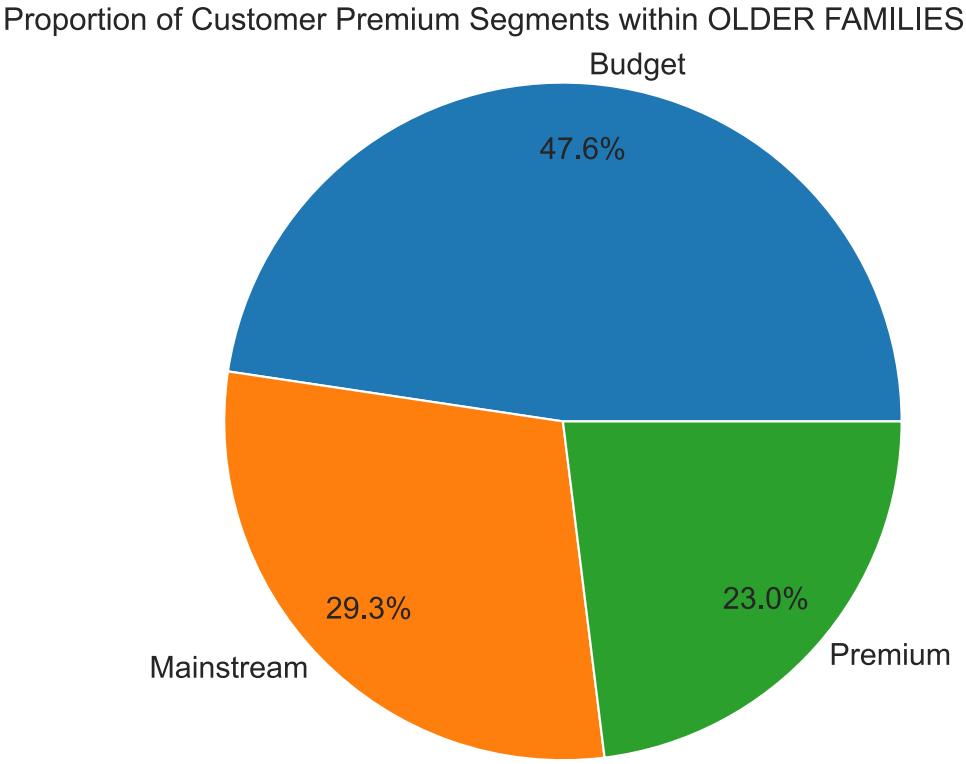

```
------------------------------
OLDER SINGLES/COUPLES
            Count
Budget      17172
Mainstream  17060
Premium     16560
```



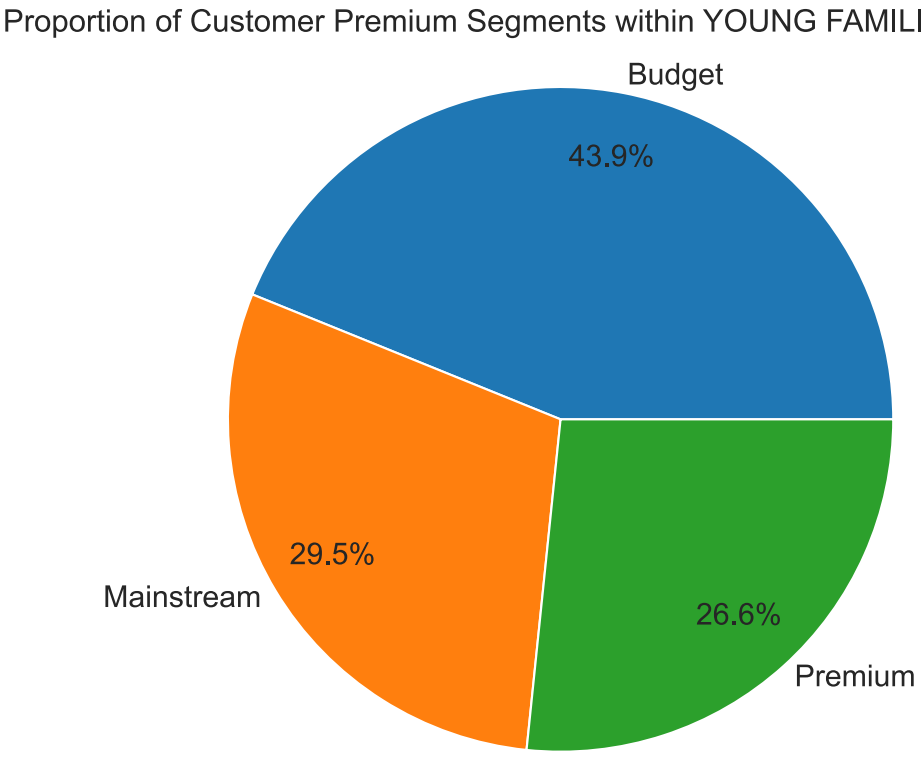

```
------------------------------
YOUNG SINGLES/COUPLES
            Count
Mainstream  19544
Budget       8572
Premium      5852
```



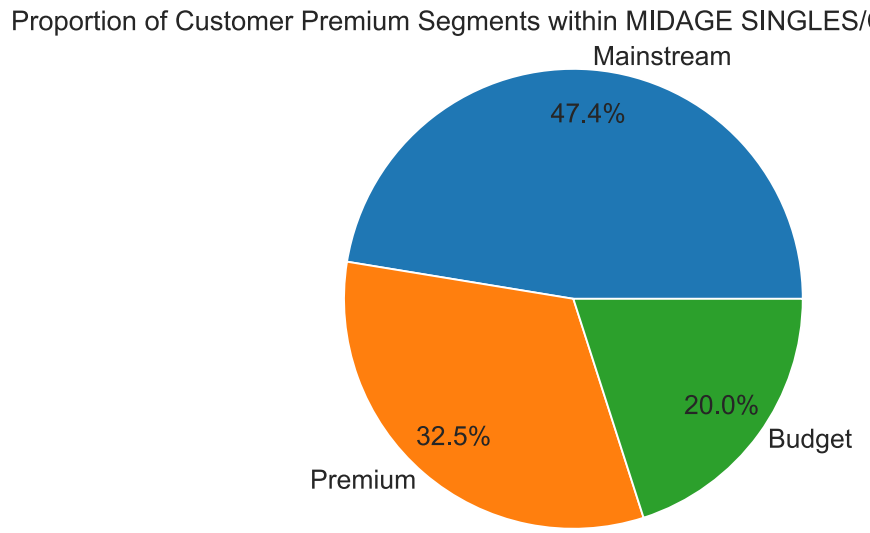

```
------------------------------
OLDER FAMILIES
            Count
Budget      21514
Mainstream  13241
Premium     10403
```



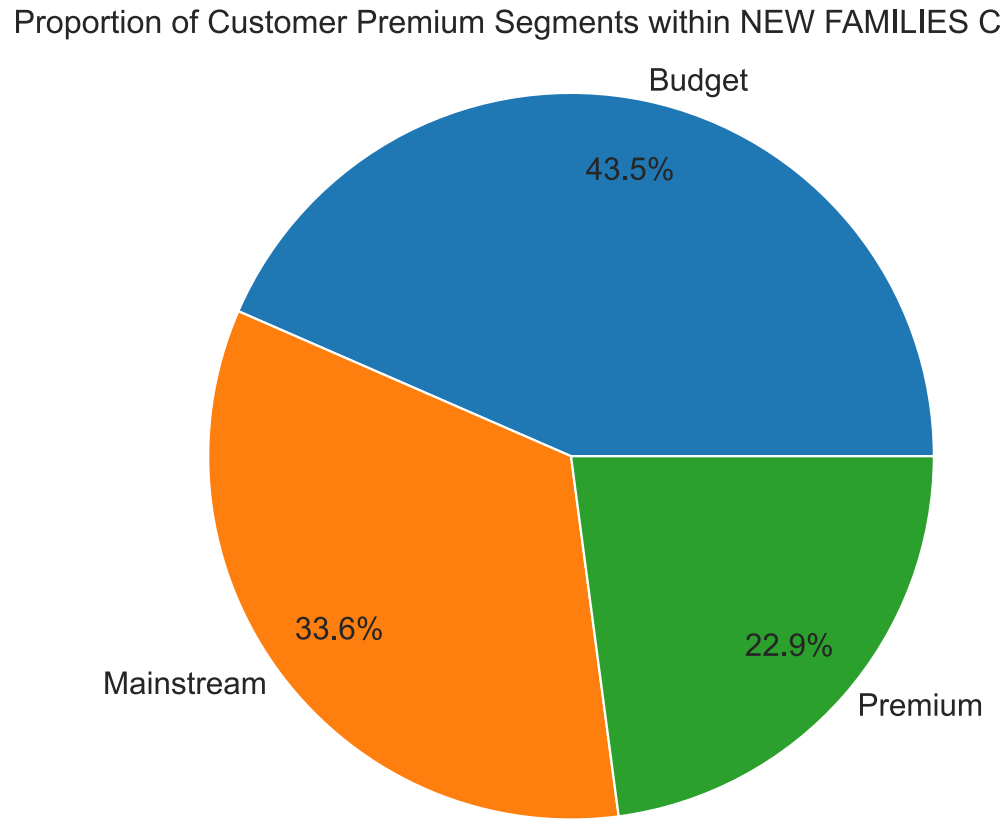

```
------------------------------
YOUNG FAMILIES
            Count
Budget      17763
Mainstream  11947
Premium     10784
```



```
------------------------------
MIDAGE SINGLES/COUPLES
            Count
Mainstream  11095
Premium      7612
Budget       4691
```



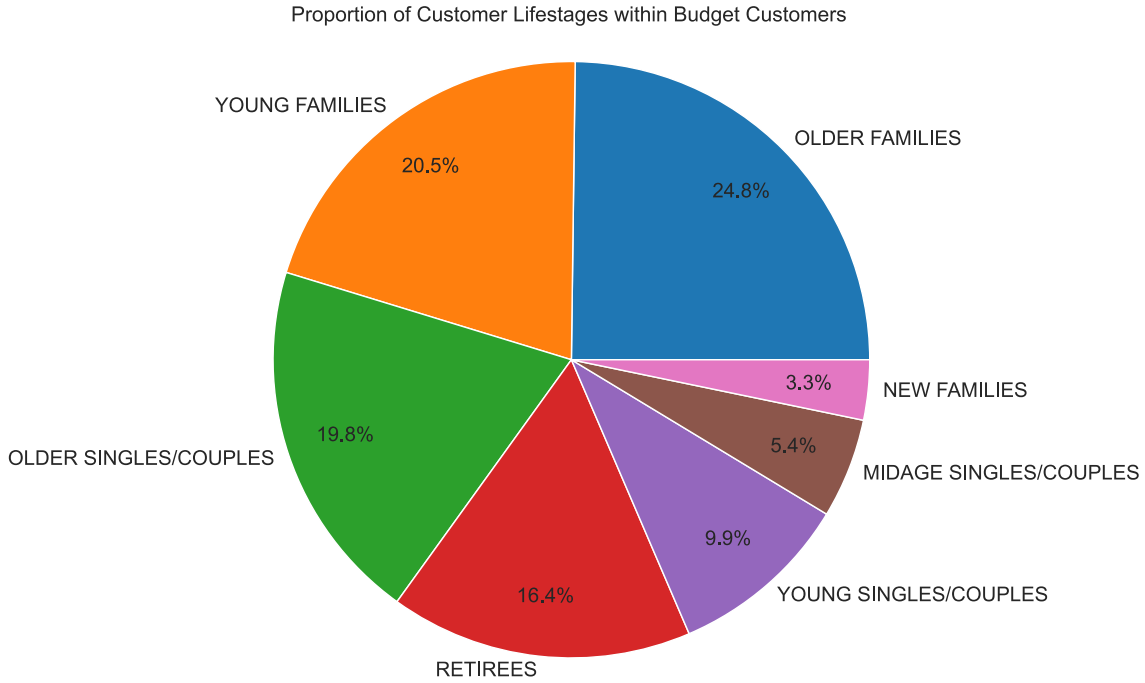

```
------------------------------
NEW FAMILIES
            Count
Budget       2824
Mainstream   2185
Premium      1488
```



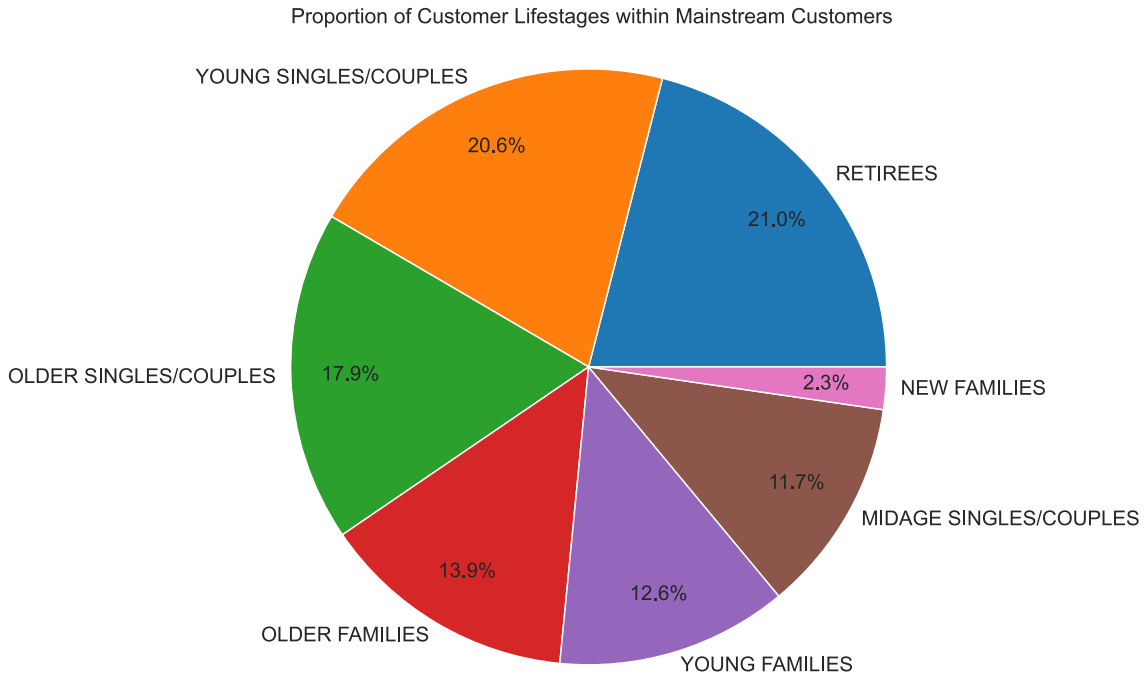
In [64]:

```
segments = ['Budget', 'Mainstream', 'Premium']

for i in segments:
    df = data[data['PREMIUM_CUSTOMER'] == i]
    dist = pd.DataFrame.from_dict(dict(df['LIFESTAGE'].value_counts()), orient = 'inde
x', columns = ['Count'])
    print('------------------------------')
    print(i)
    print(dist)
    # Pie chart
    pie, ax = plt.subplots(figsize = (12,7))
    labels = dist.index
    plt.pie(x = dist['Count'], autopct = "%.1f%%", labels = labels, pctdistance = 0.8,
labeldistance = 1.05, textprops = {'fontsize':14}, radius = 0.5)
    plt.title(f'Proportion of Customer Lifestages within {i} Customers')
    ax.axis('square')
    plt.tight_layout()
    plt.show()
```

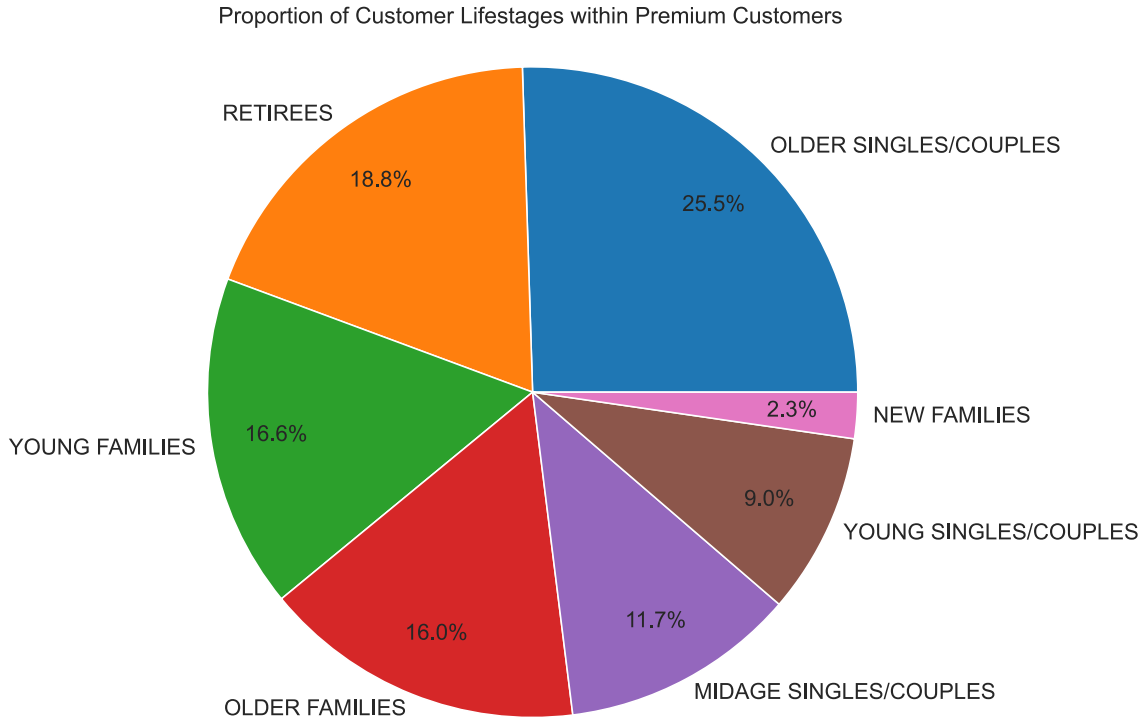
```
------------------------------
Budget
                        Count
OLDER FAMILIES          21514
YOUNG FAMILIES          17763
OLDER SINGLES/COUPLES   17172
RETIREES                14225
YOUNG SINGLES/COUPLES    8572
MIDAGE SINGLES/COUPLES   4691
NEW FAMILIES             2824
```




```
------------------------------
Mainstream
                        Count
RETIREES                19970
YOUNG SINGLES/COUPLES   19544
OLDER SINGLES/COUPLES   17060
OLDER FAMILIES          13241
YOUNG FAMILIES          11947
MIDAGE SINGLES/COUPLES  11095
NEW FAMILIES             2185
```




```
------------------------------
Premium
                         Count
OLDER SINGLES/COUPLES    16560
RETIREES                 12236
YOUNG FAMILIES           10784
OLDER FAMILIES           10403
MIDAGE SINGLES/COUPLES    7612
YOUNG SINGLES/COUPLES     5852
NEW FAMILIES              1488
```



Average Unit Price by Totals

In [65]:

```
# Check average unit price via total sales / total qty and NOT by transaction
lifestage
```

Out[65]:

	SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY
0	RETIREES	14805	14555	7.37	342381.90	1.89
1	OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91
2	YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83
3	OLDER FAMILIES	9780	9630	7.27	328519.90	1.95
4	YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94
5	MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90
6	NEW FAMILIES	2549	2492	7.29	47347.95	1.86

In [66]:

```
premium
```

Out[66]:

	Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	A
0	Mainstream	29245	28734	7.37	700859.70	1.90	180779	
1	Budget	24470	24006	7.28	631402.65	1.91	165772	
2	Premium	18922	18547	7.28	472905.45	1.91	123845	

In [67]:

```
# Check unit price per lifestage
lifestage['PER_UNIT_PRICE_TOTAL'] = lifestage['TOTAL_SPEND']/lifestage['TOT_QTY']
lifestage
```

Out[67]:

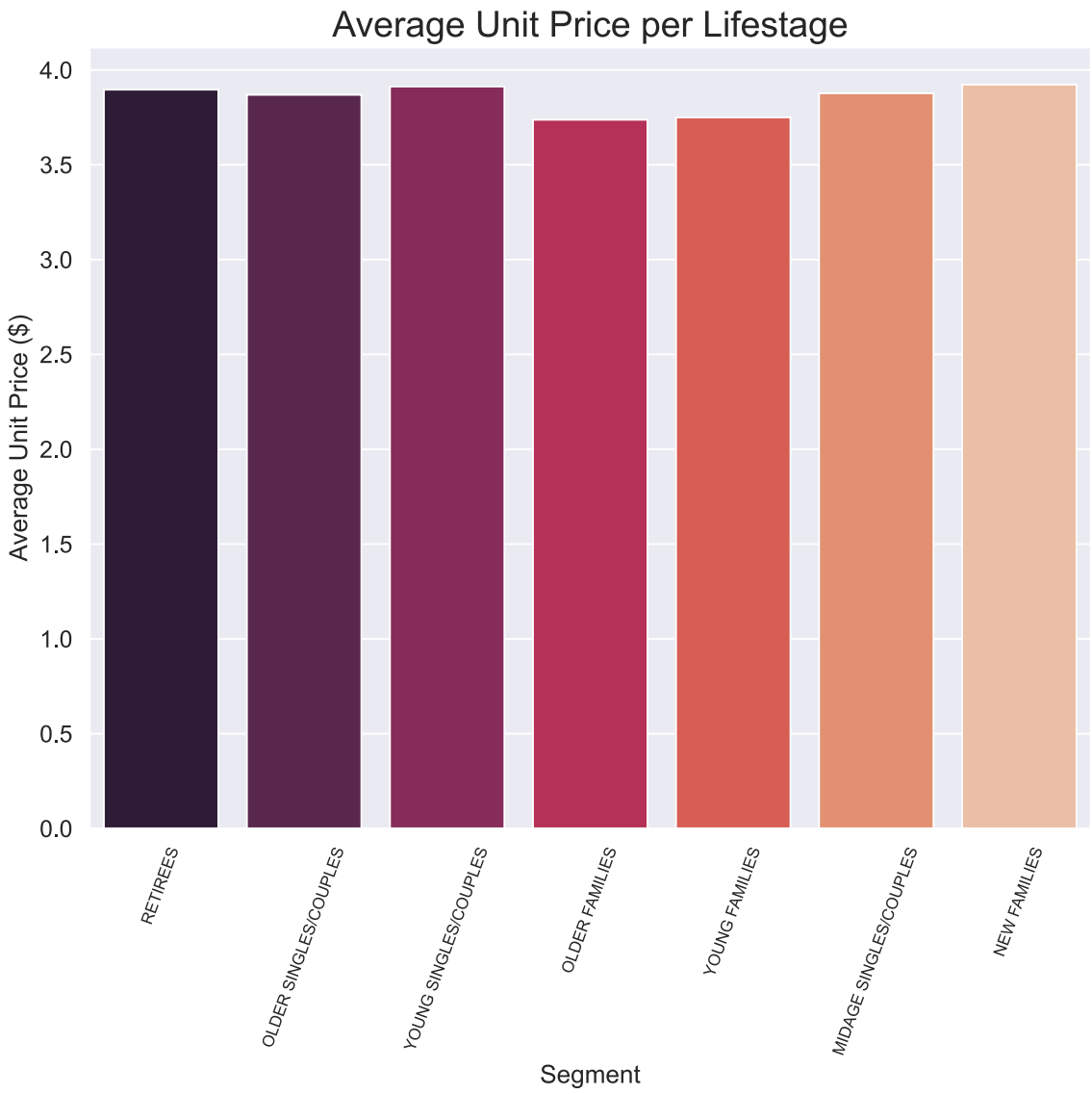
	SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY
0	RETIREES	14805	14555	7.37	342381.90	1.89
1	OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91
2	YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83
3	OLDER FAMILIES	9780	9630	7.27	328519.90	1.95
4	YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94
5	MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90
6	NEW FAMILIES	2549	2492	7.29	47347.95	1.86

In [68]:

```
# plot unit price totals
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'SEGMENT', y = 'PER_UNIT_PRICE_TOTAL', data = lifestage, palette =
'rocket')
ax.set_ylabel('Average Unit Price ($)', fontsize = 14)
ax.set_xlabel('Segment', fontsize = 14)
plt.xticks(fontsize = 9, rotation = 70)
plt.title('Average Unit Price per Lifestage', fontsize = 20)
```

Out[68]:

```
Text(0.5, 1.0, 'Average Unit Price per Lifestage')
```



In [69]:

```
premium['PER_UNIT_PRICE_TOTAL'] = premium['TOT_SPEND']/premium['TOT_QTY']
premium
```

Out[69]:

	Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	A
0	Mainstream	29245	28734	7.37	700859.70	1.90	180779	
1	Budget	24470	24006	7.28	631402.65	1.91	165772	
2	Premium	18922	18547	7.28	472905.45	1.91	123845	

In [70]:

```
# plot unit price totals
plt.figure(figsize = (10,8))
ax = sns.barplot(x = 'Segment', y = 'PER_UNIT_PRICE_TOTAL', data = premium, palette =
'rocket')
ax.set_ylabel('Average Unit Price ($)', fontsize = 16)
ax.set_xlabel('Segment', fontsize = 16)
plt.xticks(fontsize = 12)
plt.title('Average Unit Price per Customer Premium', fontsize = 20)
```

Out[70]:

```
Text(0.5, 1.0, 'Average Unit Price per Customer Premium')
```

OBSERVATIONS:

LIFESTAGE

- Younger singles/couples are more willing to spend on more premium ranges of snacks.

CUSTOMER PREMIUMS

- Mainstream customers are more willing to spend on more premium ranges of snacks.

Based on the data shown, young singles/couples who are registered as the mainstream premiums are willing to pay more per unit compared to other segments. Reasons for this could lie with a more health-oriented purchasing behaviour. This is further backed up by there being fewer purchases by premium middle aged and young singles/couples customers, compared to mainstream segments of the same kind.

- Must check if difference in unit price is statistically significant.

T-Test of significance for difference in average unit price.

In [71]:

```
# Gather Segment data into sepeate df's

# Mainstream young/midage singles/couples
query = """
        SELECT *
        FROM data
        WHERE (LIFESTAGE = 'YOUNG SINGLES/COUPLES') OR
        (LIFESTAGE = 'MIDAGE SINGLES/COUPLES')
        """
lifestage_df = ps.sqldf(query, locals())

query = """
        SELECT *
        FROM lifestage_df
        WHERE (PREMIUM_CUSTOMER = 'Mainstream')
        """
mainstream_data = ps.sqldf(query, locals())

# Premium young/midage singles/couples
query = """
        SELECT *
        FROM lifestage_df
        WHERE (PREMIUM_CUSTOMER = 'Premium') OR (PREMIUM_CUSTOMER = 'Budget')
        """
premium_budget_data = ps.sqldf(query, locals())
```

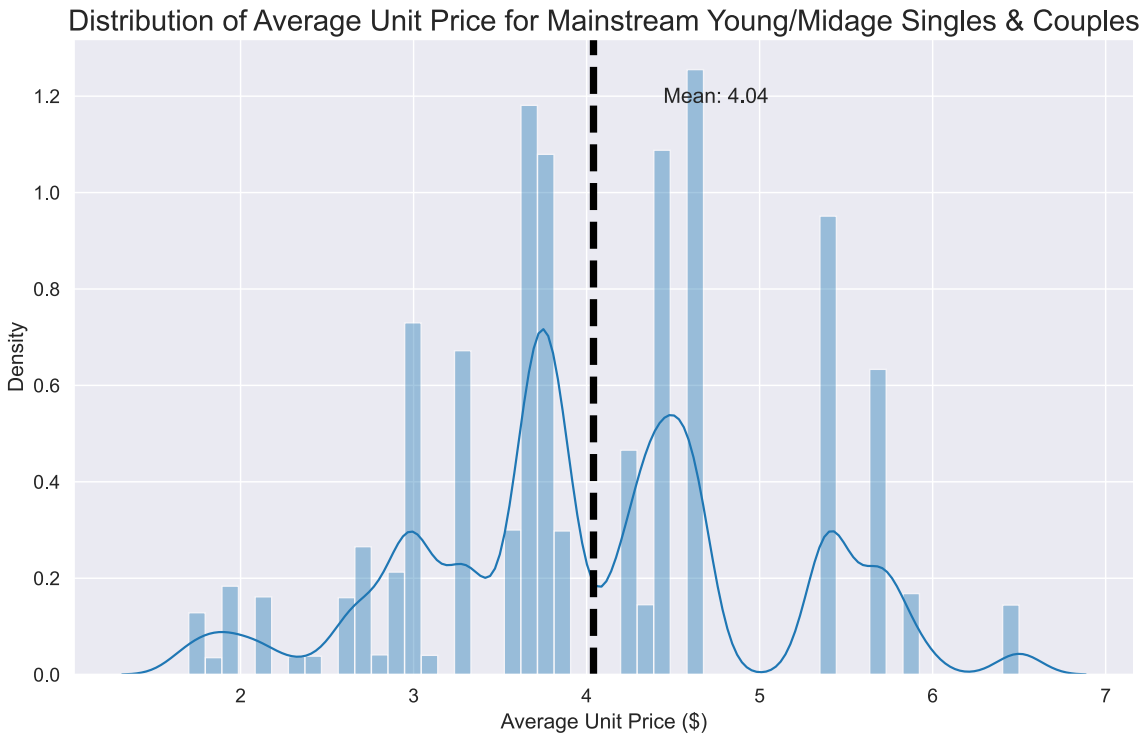
In [72]:

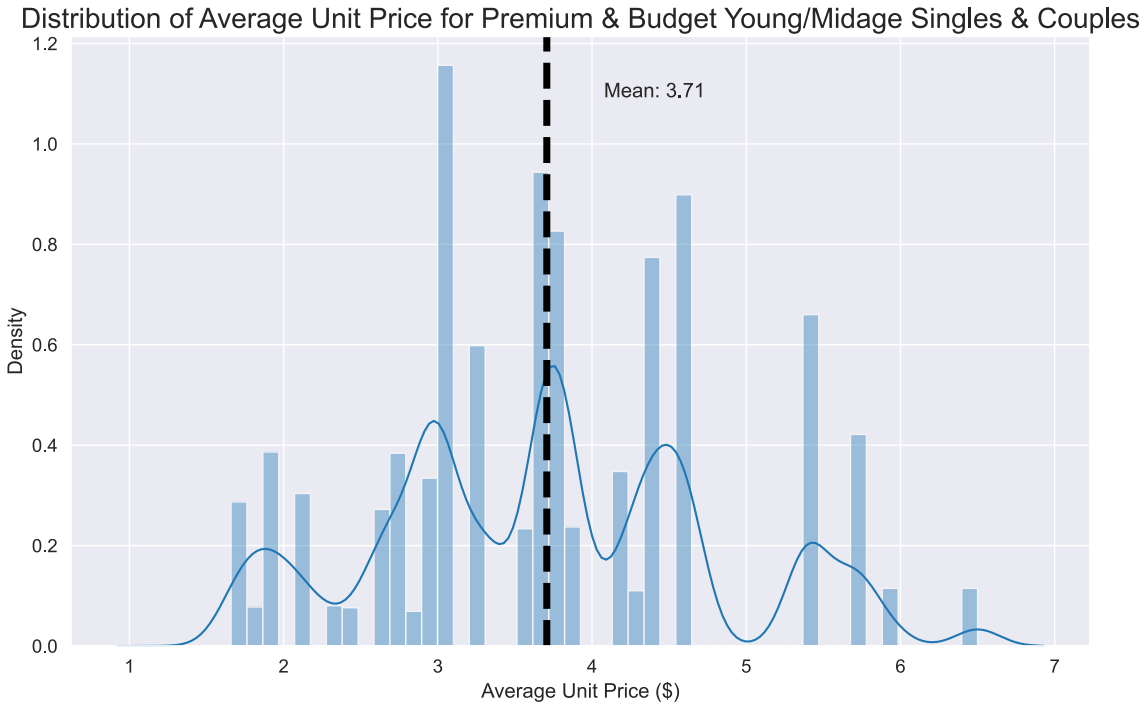
```
# Function for viz distribution
def plot_distribution(inp, subset):
    plt.figure(figsize = (13,8))
    ax = sns.distplot(inp)
    plt.axvline(np.mean(inp), color = 'k', linestyle = 'dashed', linewidth = 5)
    _, max_ = plt.ylim()
    plt.text(
        inp.mean() + inp.mean() / 10,
        max_ - max_ / 10,
        "Mean: {:.2f}".format(inp.mean())
    )
    plt.title(f'Distribution of Average Unit Price for {subset}', fontsize = 20)
    plt.xlabel('Average Unit Price ($)')
    return plt.figure
```

In [73]:

```
# Distributions of each segment's avg unit price
dataframes = [mainstream_data, premium_budget_data]
subset_names = ['Mainstream Young/Midage Singles & Couples',
                'Premium & Budget Young/Midage Singles & Couples']

for df, subset in zip(dataframes, subset_names):
    plot_distribution(df['AVG_CHIP_PRICE'], subset)
```

```
C:\Users\Joel\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-l
evel function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Joel\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-l
evel function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
```




Distribution of Average Unit Price for Premium & Budget Young/Midage Singles & Couples
Mean: 3.71
Density
Average Unit Price ($)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1
2
3
4
5
6
7

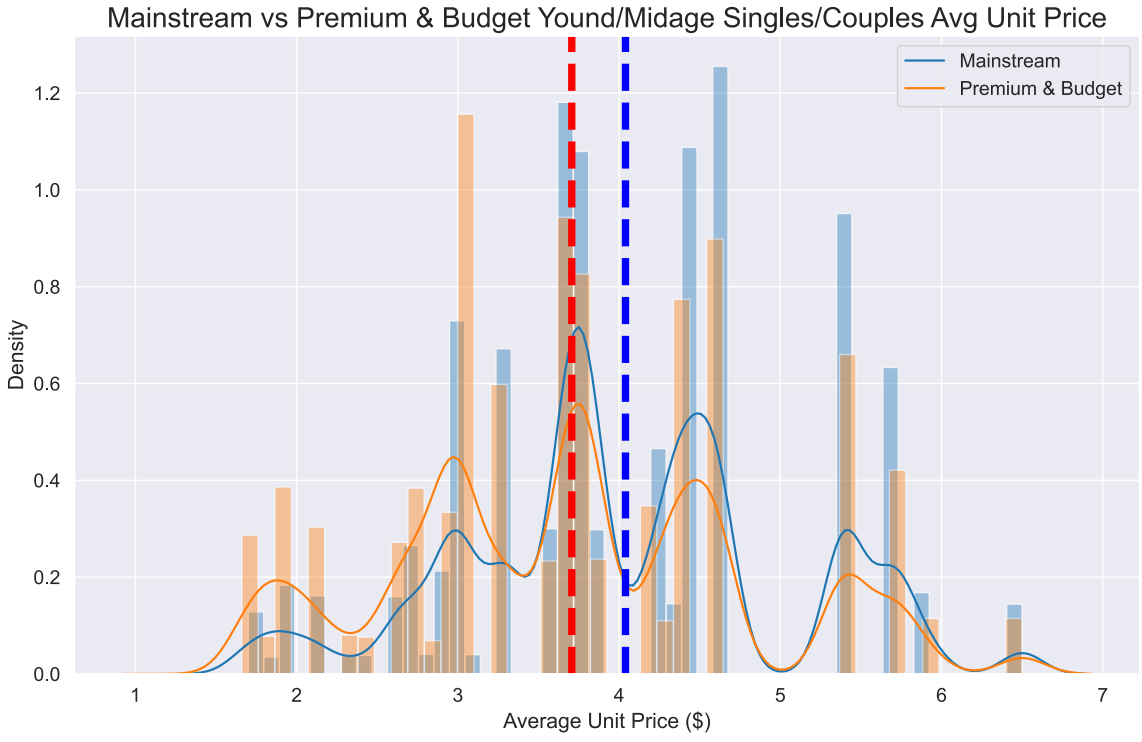
In [74]:

```
# Show Comparison distributions

#Mainstream vs Premium & Budget
plt.figure(figsize = (13,8))
ax1 = sns.distplot(mainstream_data['AVG_CHIP_PRICE'])
ax2 = sns.distplot(premium_budget_data['AVG_CHIP_PRICE'])
plt.axvline(np.mean(mainstream_data['AVG_CHIP_PRICE']), color = 'b', linestyle = 'dashe
d', linewidth = 5)
plt.axvline(np.mean(premium_budget_data['AVG_CHIP_PRICE']), color = 'red', linestyle =
'dashed', linewidth = 5)
plt.xlabel('Average Unit Price ($)')
plt.title('Mainstream vs Premium & Budget Yound/Midage Singles/Couples Avg Unit Price',
fontsize = 19)
plt.legend(['Mainstream','Premium & Budget'])
```

```
C:\Users\Joel\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-l
evel function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Joel\anaconda3\lib\site-packages\seaborn\distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-l
evel function with similar flexibility) or `histplot` (an axes-level funct
ion for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[74]:

```
<matplotlib.legend.Legend at 0x299d5e0fe50>
```



In [75]:

```
# compare 2 groups func
def compare_2_groups(arr_1, arr_2, alpha, sample_size):
    stat, p = ttest_ind(arr_1, arr_2)
    print('Statistics = %.3f, p=%.3f' % (stat, p))
    if p > alpha:
        print('Same distribution (fail to reject H0)')
    else:
        print('Different distributions (reject H0)')
```

In [89]:

```
# Perform T-Test
sample_size = 200
mainstream_sampled = np.random.choice(mainstream_data['AVG_CHIP_PRICE'], sample_size)
premium_budget_sampled = np.random.choice(premium_budget_data['AVG_CHIP_PRICE'], sample
_size)
compare_2_groups(mainstream_sampled, premium_budget_sampled, 0.05, sample_size)
```

```
Statistics = 2.445, p=0.015
Different distributions (reject H0)
```

OBSERVATIONS: The t-test results in a p-value of 0.015, i.e. the unit price for mainstream, young and mid-age singles and couples are significantly higher than that of budget or premium yound and midage singles and couples.

Further Insights for Segments

In [77]:

```
lifestage.sort_values(by = 'TOTAL_SPEND', ascending=False)
```

Out[77]:

	SEGMENT	TXN_COUNTS	UNIQUE_CUST	AVG_SPEND	TOTAL_SPEND	AVG_QTY
1	OLDER SINGLES/COUPLES	14609	14389	7.40	376013.95	1.91
0	RETIREES	14805	14555	7.37	342381.90	1.89
3	OLDER FAMILIES	9780	9630	7.27	328519.90	1.95
4	YOUNG FAMILIES	9178	9036	7.28	294627.90	1.94
2	YOUNG SINGLES/COUPLES	14441	14044	7.18	243752.40	1.83
5	MIDAGE SINGLES/COUPLES	7275	7141	7.37	172523.80	1.90
6	NEW FAMILIES	2549	2492	7.29	47347.95	1.86

In [78]:

```
premium.sort_values(by = 'TOT_SPEND', ascending=False)
```

Out[78]:

	Segment	Transactions	unique_cust	AVG_SPEND	TOT_SPEND	AVG_QTY	TOT_QTY	A
0	Mainstream	29245	28734	7.37	700859.70	1.90	180779	
1	Budget	24470	24006	7.28	631402.65	1.91	165772	
2	Premium	18922	18547	7.28	472905.45	1.91	123845	

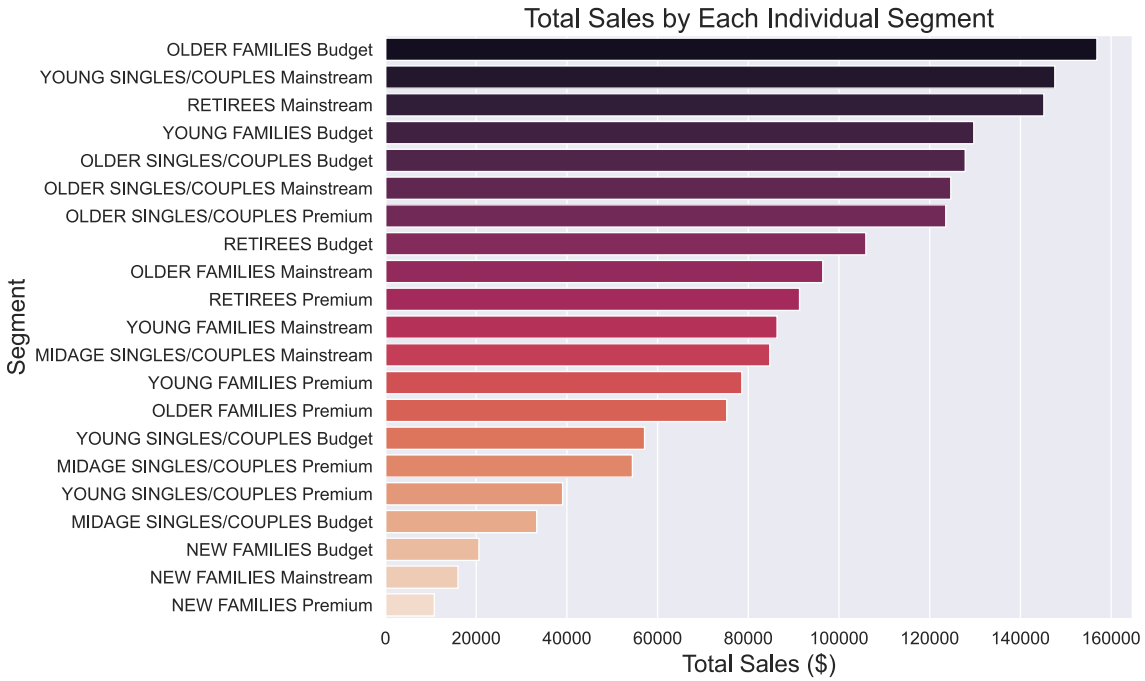
In [79]:

```
# Combined segments with most sales
sales = []
segs = []
for segment in lifestage['SEGMENT']:
    for prem in premium['Segment']:
        df = data[(data['LIFESTAGE'] == segment) & (data['PREMIUM_CUSTOMER'] == prem)]
        sales.append(round(np.sum(df['TOT_SALES']), 1))
        segs.append(f'{segment} {prem}')

# Make df for each individual segment
df_sales = pd.DataFrame(segs, columns = ['Segment'])
df_sales['Total_sales'] = sales
df_sales.sort_values(by = 'Total_sales', ascending = False, inplace = True)
print(df_sales)

# Plot segment sales
plt.figure(figsize = (10,8))
ax = sns.barplot(y = 'Segment', x = 'Total_sales', data = df_sales, palette = 'rocket')
ax.set_xlabel('Total Sales ($)', fontsize = 18)
ax.set_ylabel('Segment', fontsize = 18)
plt.title('Total Sales by Each Individual Segment', fontsize = 20)
```

	Segment	Total_sales
10	OLDER FAMILIES Budget	156863.8
6	YOUNG SINGLES/COUPLES Mainstream	147582.2
0	RETIREES Mainstream	145169.0
13	YOUNG FAMILIES Budget	129718.0
4	OLDER SINGLES/COUPLES Budget	127833.6
3	OLDER SINGLES/COUPLES Mainstream	124642.8
5	OLDER SINGLES/COUPLES Premium	123537.6
1	RETIREES Budget	105916.3
9	OLDER FAMILIES Mainstream	96413.6
2	RETIREES Premium	91296.6
12	YOUNG FAMILIES Mainstream	86338.2
15	MIDAGE SINGLES/COUPLES Mainstream	84734.2
14	YOUNG FAMILIES Premium	78571.7
11	OLDER FAMILIES Premium	75242.6
7	YOUNG SINGLES/COUPLES Budget	57117.9
17	MIDAGE SINGLES/COUPLES Premium	54443.8
8	YOUNG SINGLES/COUPLES Premium	39052.3
16	MIDAGE SINGLES/COUPLES Budget	33345.7
19	NEW FAMILIES Budget	20607.4
18	NEW FAMILIES Mainstream	15979.7
20	NEW FAMILIES Premium	10760.8

Out[79]:

```
Text(0.5, 1.0, 'Total Sales by Each Individual Segment')
```



OBSERVATIONS:

- The segments which spend the most include:
  - Older Families with Budget Premiums
  - Young singles/couples with Mainstream Premiums
  - Retirees with Mainstream Premiums

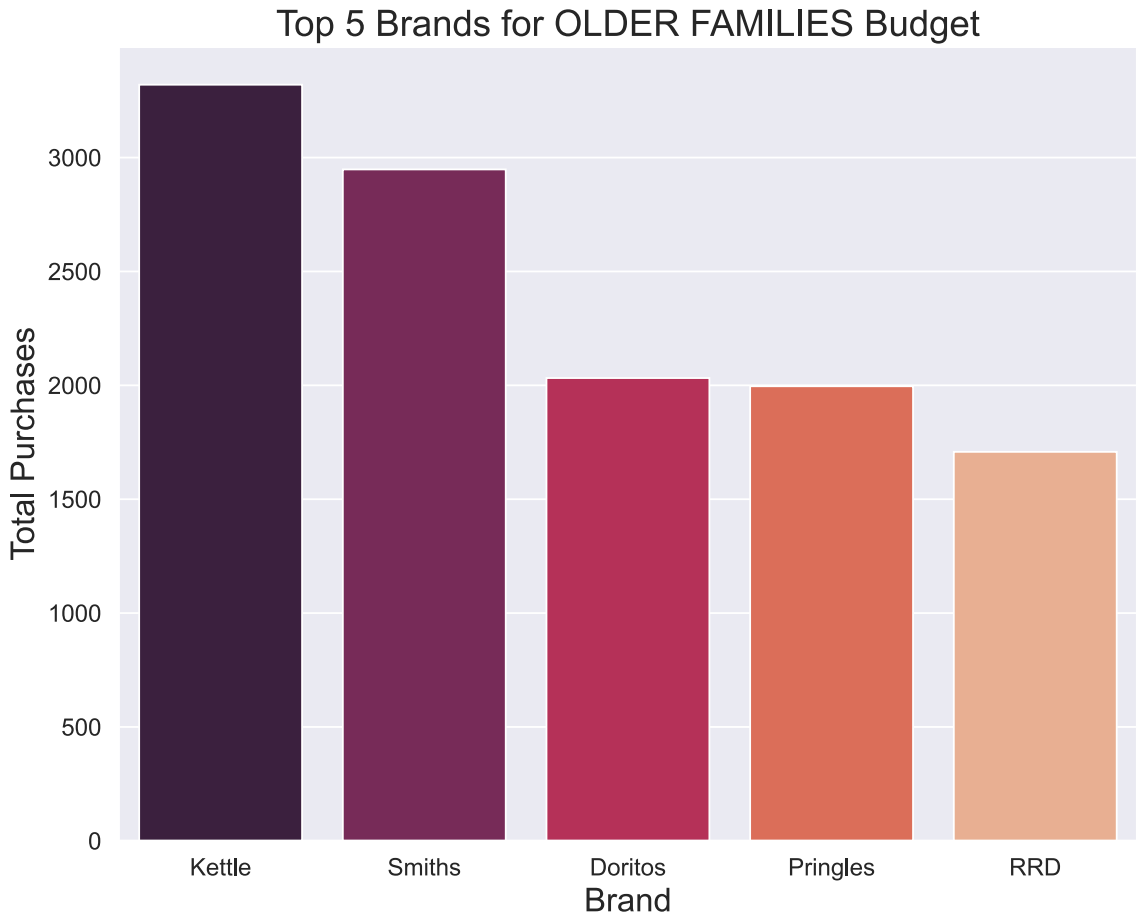
RECOMMENDATION:

- Target segments which provide the most sales.
- Find out their favorite brand of chips.
- Catering to these segments will further increase sales.

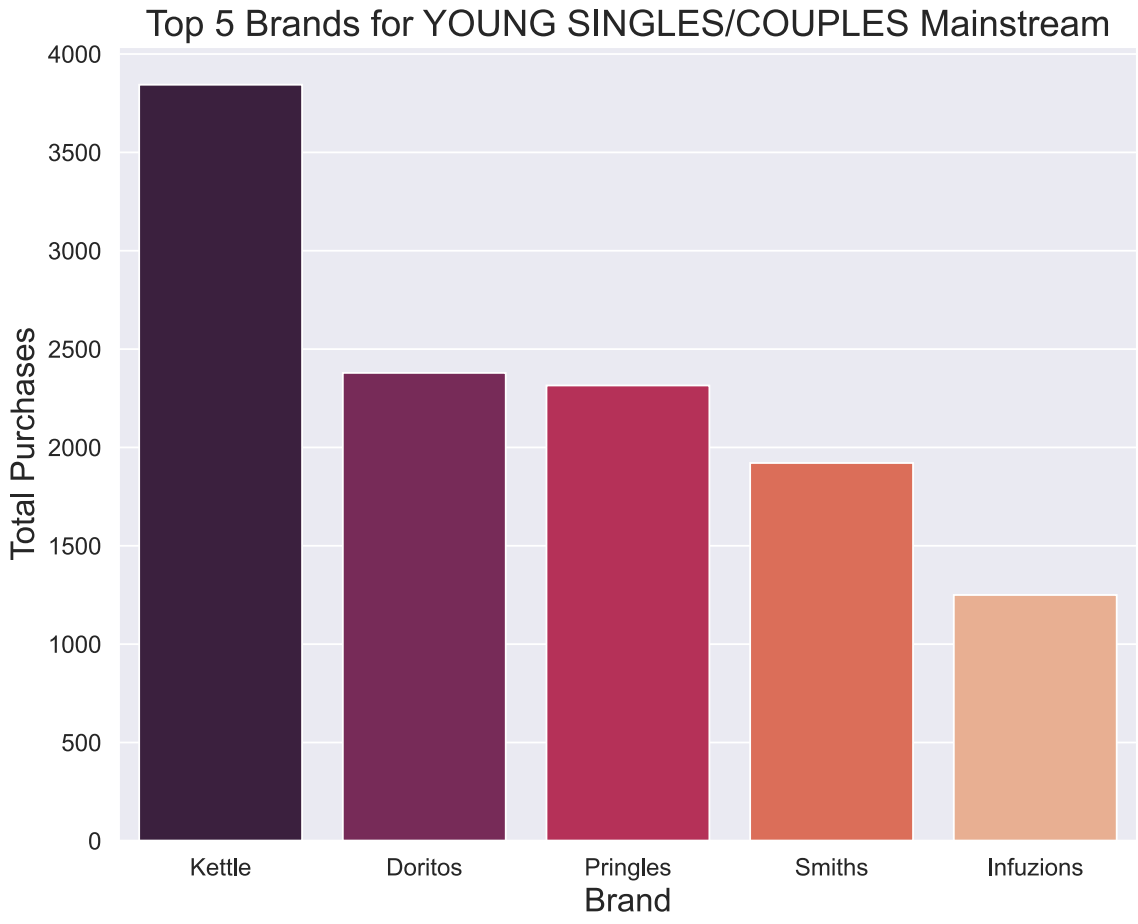
In [80]:

```
# top 3 segment premiums
prems = ['Budget', 'Mainstream', 'Mainstream']

# top 3 segment lifestages
lifestages = ['OLDER FAMILIES', 'YOUNG SINGLES/COUPLES', 'RETIREES']

# Visualization
for lifestage, prem in zip(lifestages, prems):
    df = data[(data['LIFESTAGE'] == lifestage) & (data['PREMIUM_CUSTOMER'] == prem)]
    print('----------------------------------------')
    segment = f'{lifestage} {prem}'
    print(lifestage, prem + ' Top Brands Ranked')
    print('----------------------------------------')
    print(df['BRAND_NAME'].value_counts())
    print('----------------------------------------')
    print(f'Plot: {segment}')
    print('----------------------------------------')
    viz = pd.DataFrame.from_dict(dict(df['BRAND_NAME'].value_counts()), orient= 'index'
, columns = ['Count'])
    # Show viz of top 5 brands
    values = viz['Count'][:5]
    labels = viz.index[:5]
    plt.figure(figsize = (10,8))
    ax = sns.barplot(x = labels, y = values, palette= 'rocket')
    ax.set_xlabel('Brand', fontsize = 18)
    ax.set_ylabel('Total Purchases', fontsize = 18)
    plt.title(f'Top 5 Brands for {segment}', fontsize = 20)
    plt.show()
```

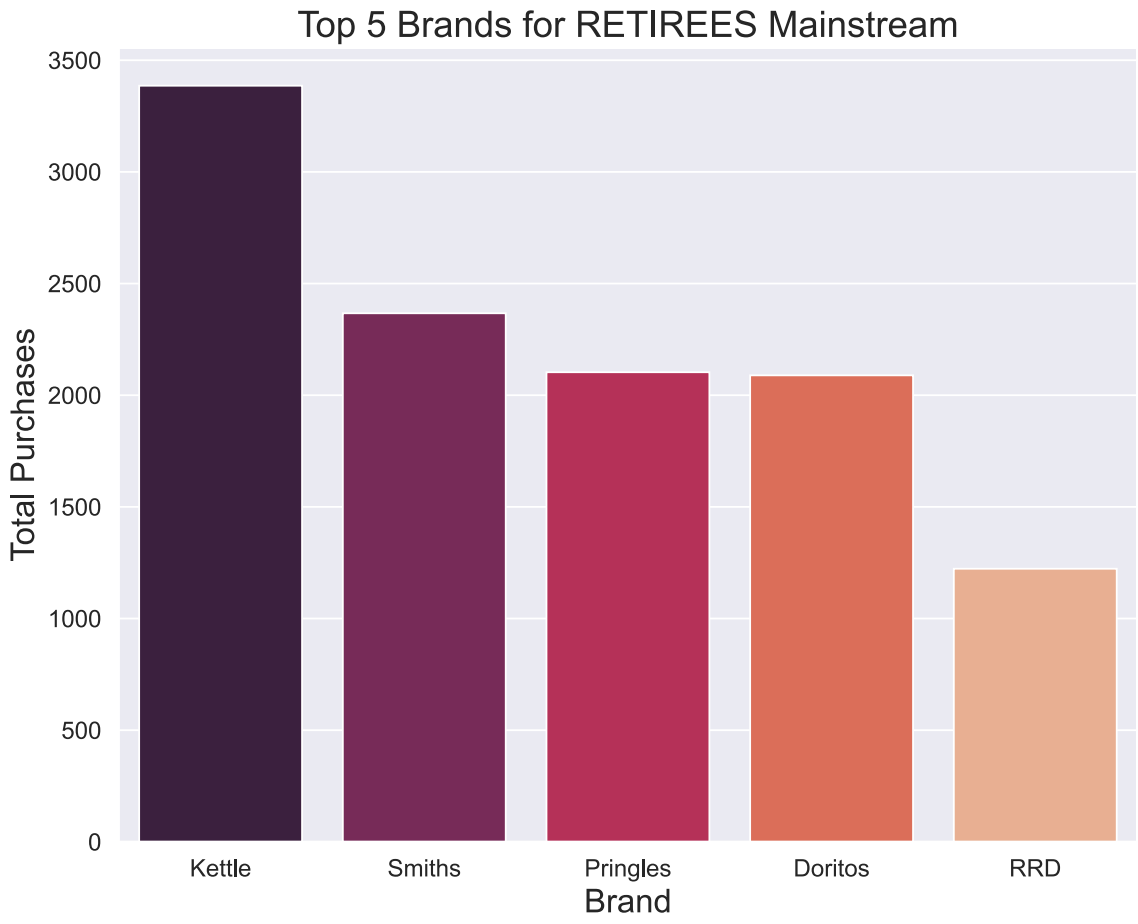
```
----------------------------------------
OLDER FAMILIES Budget Top Brands Ranked
----------------------------------------
Kettle          3320
Smiths          2948
Doritos         2032
Pringles        1996
RRD             1708
Woolworths      1213
Infuzions       1185
Thins           1171
Twisties         810
Cobs             760
NCC              741
Tostitos         705
GrnWves          671
Tyrrells         489
CCs              451
Cheezels         427
Sunbites         305
Cheetos          281
Burger Rings     159
French Fries     142
Name: BRAND_NAME, dtype: int64
----------------------------------------
Plot: OLDER FAMILIES Budget
----------------------------------------
```



```
----------------------------------------
YOUNG SINGLES/COUPLES Mainstream Top Brands Ranked
----------------------------------------
Kettle          3844
Doritos         2379
Pringles        2315
Smiths          1921
Infuzions       1250
Thins           1166
Twisties         900
Tostitos         890
RRD              875
Cobs             864
GrnWves          646
Tyrrells         619
Woolworths       479
NCC              394
Cheezels         346
CCs              222
Cheetos          166
Sunbites         128
French Fries      78
Burger Rings      62
Name: BRAND_NAME, dtype: int64
----------------------------------------
Plot: YOUNG SINGLES/COUPLES Mainstream
----------------------------------------
```



```
---------------------------------------
RETIREES Mainstream Top Brands Ranked
---------------------------------------
Kettle          3386
Smiths          2367
Pringles        2103
Doritos         2089
RRD             1223
Thins           1199
Infuzions       1182
Woolworths       902
Twisties         802
Cobs             776
Tostitos         739
GrnWves          667
NCC              587
Tyrrells         514
Cheezels         382
CCs              355
Sunbites         243
Cheetos          236
Burger Rings     122
French Fries      96
Name: BRAND_NAME, dtype: int64
---------------------------------------
Plot: RETIREES Mainstream
---------------------------------------
```



OBSERVATIONS:

OLDER FAMILIES Budget top brands:

- Kettle
- Smiths
- Doritos
- Pringles
- Red Rock Deli

YOUNG SINGLES/COUPLES Mainstream top brands:

- Kettle
- Smiths
- Doritos
- Pringles
- Red Rock Deli

RETIREES Mainstream Top Brands:

- Kettle
- Smiths
- Pringles
- Doritos
- Red Rock Deli

RECOMMENDATIONS:

- Promote these brands to increase sales.

In [81]:

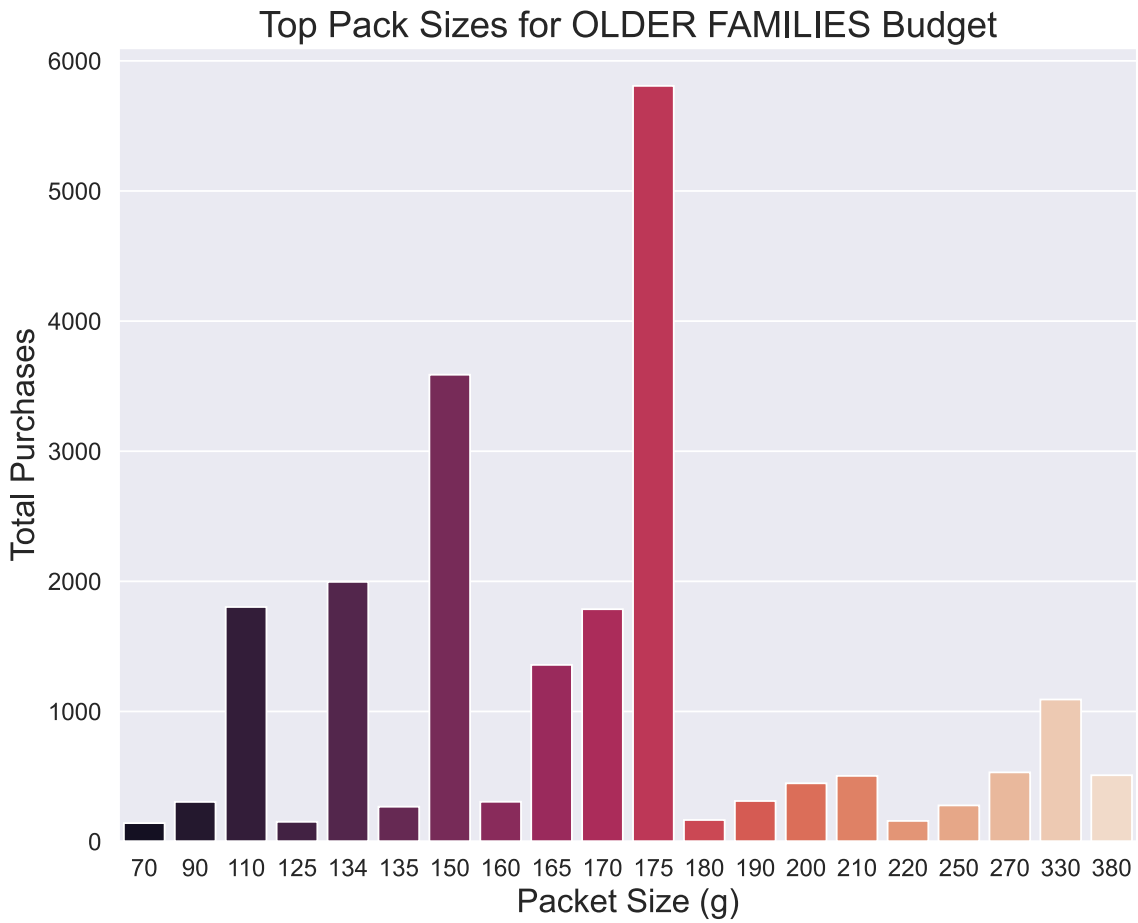
```
for i in data.columns:
    print(i)
```

```
real_date
STORE_NBR
TXN_ID
PROD_NAME
BRAND_NAME
PACKET_SIZE
PROD_QTY
TOT_SALES
LYLTY_CARD_NBR
LIFESTAGE
PREMIUM_CUSTOMER
AVG_CHIP_PRICE
```

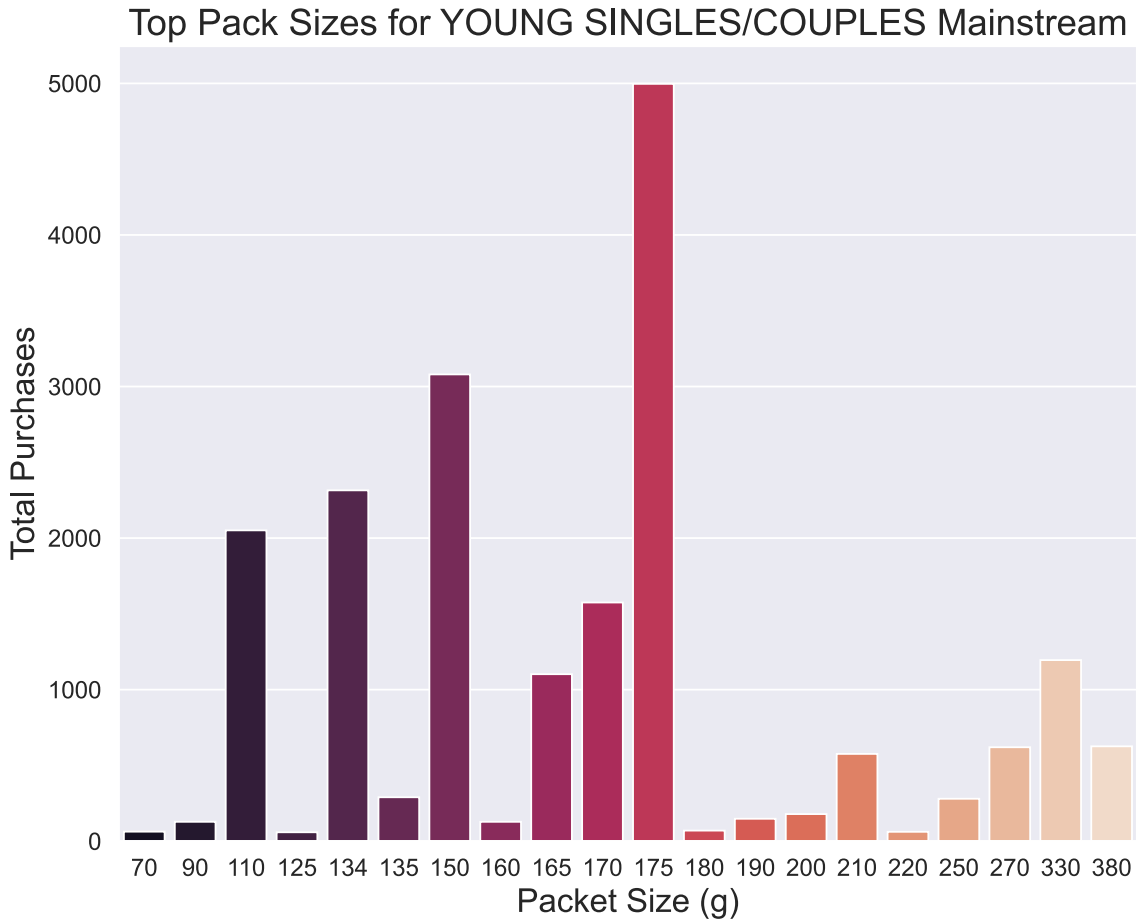
In [82]:

```
# Check Prefered Pack size per top ranked segments

# top 3 segment premiums
prems = ['Budget', 'Mainstream', 'Mainstream']

# top 3 segment lifestages
lifestages = ['OLDER FAMILIES', 'YOUNG SINGLES/COUPLES', 'RETIREES']

# Visualization
for lifestage, prem in zip(lifestages, prems):
    df = data[(data['LIFESTAGE'] == lifestage) & (data['PREMIUM_CUSTOMER'] == prem)]
    print('----------------------------------------')
    segment = f'{lifestage} {prem}'
    print(lifestage, prem + ' Favorite Chip Sizes')
    print('----------------------------------------')
    print(df['PACKET_SIZE'].value_counts())
    print('----------------------------------------')
    print(f'Plot: {segment}')
    print('----------------------------------------')
    viz = pd.DataFrame.from_dict(dict(df['PACKET_SIZE'].value_counts()), orient= 'inde
x', columns = ['Count'])
    # Show viz of top 5 brands
    values = viz['Count']
    labels = viz.index
    plt.figure(figsize = (10,8))
    ax = sns.barplot(x = labels, y = values, palette= 'rocket')
    ax.set_xlabel('Packet Size (g)', fontsize = 18)
    ax.set_ylabel('Total Purchases', fontsize = 18)
    plt.title(f'Top Pack Sizes for {segment}', fontsize = 20)
    plt.show()
```

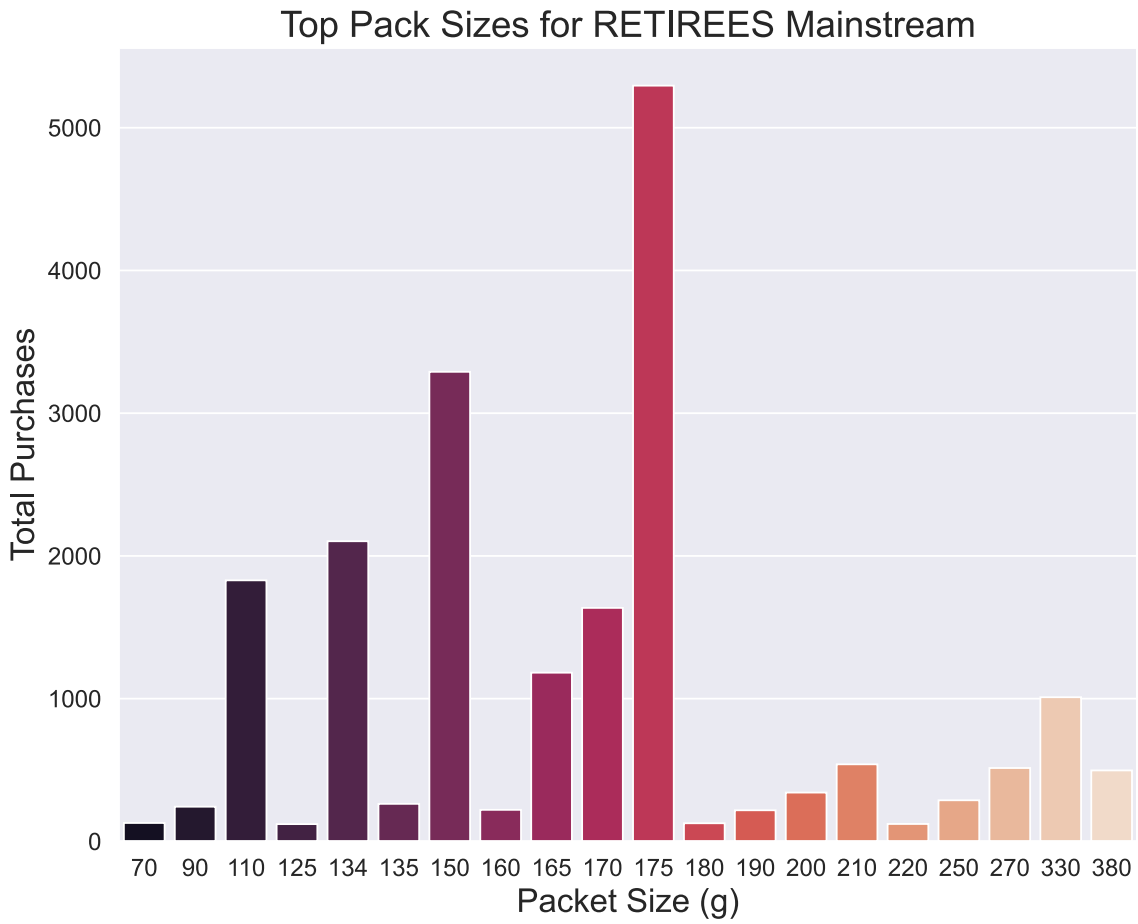
```
---------------------------------------
OLDER FAMILIES Budget Favorite Chip Sizes
---------------------------------------
175    5808
150    3588
134    1996
110    1803
170    1786
165    1358
330    1092
270     532
380     510
210     505
200     448
190     312
160     306
90      305
250     278
135     268
180     166
220     159
125     152
70      142
Name: PACKET_SIZE, dtype: int64
---------------------------------------
Plot: OLDER FAMILIES Budget
---------------------------------------
```


Top Pack Sizes for OLDER FAMILIES Budget
Total Purchases
0
1000
2000
3000
4000
5000
6000
70 90 110 125 134 135 150 160 165 170 175 180 190 200 210 220 250 270 330 380
Packet Size (g)

```
----------------------------------------
YOUNG SINGLES/COUPLES Mainstream Favorite Chip Sizes
----------------------------------------
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: PACKET_SIZE, dtype: int64
----------------------------------------
Plot: YOUNG SINGLES/COUPLES Mainstream
----------------------------------------
```



```
----------------------------------------
RETIREES Mainstream Favorite Chip Sizes
----------------------------------------
175    5295
150    3290
134    2103
110    1829
170    1636
165    1182
330    1010
210     540
270     514
380     497
200     342
250     288
135     263
90      243
160     221
190     218
70      129
180     127
220     122
125     121
Name: PACKET_SIZE, dtype: int64
----------------------------------------
Plot: RETIREES Mainstream
----------------------------------------
```



OBSERVATIONS:

- Across all 3 top segments, the most popular pack sizes are:
  - 175g
  - 150g
  - 134g

RECOMMENDATIONS:

- Prioritize these pack sizes when replenishing stock.

In [ ]: