```
#Importing the libaries + file upload
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
uploaded = files.upload()
```

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

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

```
#import datasets

df_transaction = pd.read_excel("QVI_transaction_data.xlsx")
df_behaviour = pd.read_csv("QVI_purchase_behaviour.csv")
```

```
#checking the null and data type formats for transaction
df_transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   DATE            264836 non-null  int64  
 1   STORE_NBR       264836 non-null  int64  
 2   LYLTY_CARD_NBR  264836 non-null  int64  
 3   TXN_ID          264836 non-null  int64  
 4   PROD_NBR        264836 non-null  int64  
 5   PROD_NAME       264836 non-null  object 
 6   PROD_QTY        264836 non-null  int64  
 7   TOT_SALES       264836 non-null  float64
dtypes: float64(1), int64(6), object(1)
memory usage: 16.2+ MB
```

```
#change excel int to datetime
df_transaction['DATE'] = pd.to_datetime(df_transaction["DATE"], unit="D", origin="1899-12-30")
```

```
#checking the null and data type formats for beahviour
df_behaviour.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72637 entries, 0 to 72636
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   LYLTY_CARD_NBR    72637 non-null  int64 
 1   LIFESTAGE         72637 non-null  object
 2   PREMIUM_CUSTOMER  72637 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.7+ MB
```

```
#Checking chip names
df_transaction["PROD_NAME"].unique()

import re
from collections import Counter

# Split all product names into words
all_words = " ".join(df_transaction["PROD_NAME"].astype(str)).lower().split()

# Remove words with digits or special characters
clean_words = [w for w in all_words if re.match(r'^[a-zA-Z]+$', w)]

# Count frequency
word_counts = Counter(clean_words)

# Top 20 most common words
print(word_counts.most_common(20))
```

```
[('chips', 49770), ('kettle', 41288), ('smiths', 28860), ('salt', 27976), ('cheese', 27890), ('pringles', 25102), ('doritos', 2
```

```
#flag salsa products and drop collumns that have them

df_transaction["SALSA"] = df_transaction["PROD_NAME"].str.lower().str.contains("salsa")
```

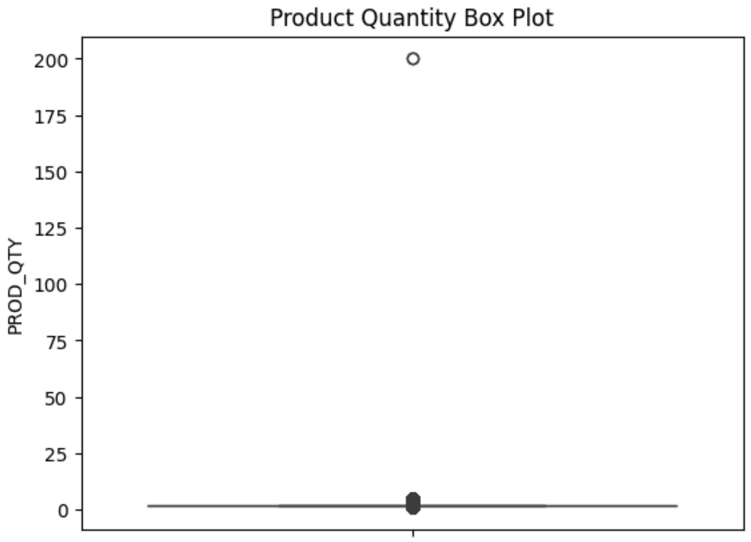
```
df_transaction = df_transaction[df_transaction["SALSA"] == False].drop(columns=["SALSA"])
```

```
#find any unsual values
df_transaction.describe()
```

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

```
# Box plot to visualize outliers

sns.boxplot(y=df_transaction['PROD_QTY'])
plt.title('Product Quantity Box Plot')
```

```
Text(0.5, 1.0, 'Product Quantity Box Plot')
```



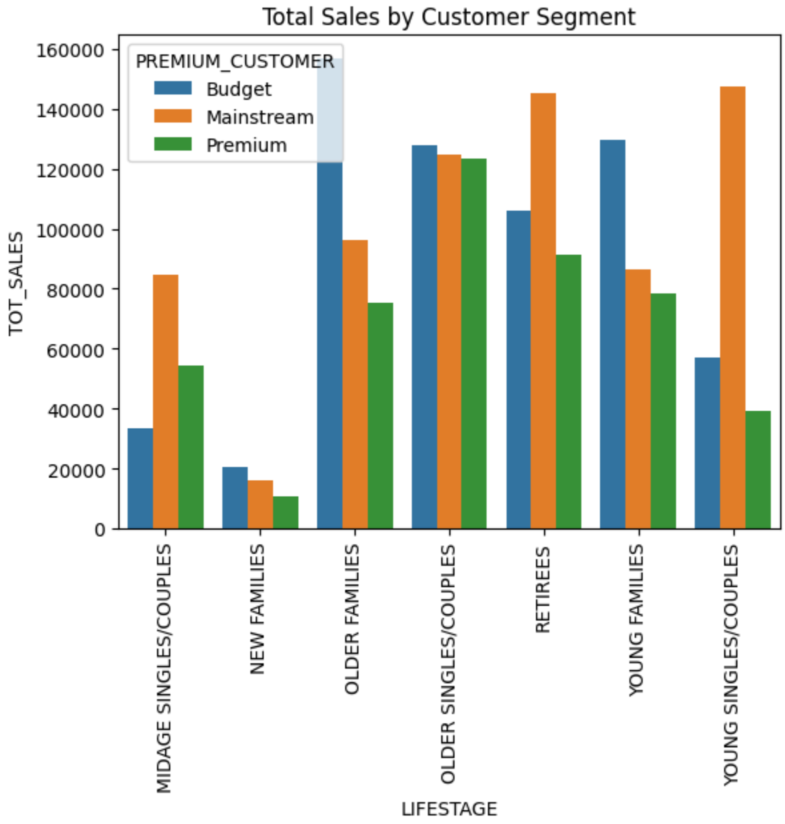
```
#remove outlier

df_transaction = df_transaction[df_transaction["PROD_QTY"] != 200]
```

```
#extracting new columns from data
# Extract pack size
df_transaction["PACK_SIZE"] = df_transaction["PROD_NAME"].str.extract(r'(\d+)(?=g)').astype(float)

# Extract brand (first word)
df_transaction["BRAND"] = df_transaction["PROD_NAME"].str.split().str[0]
```

```
/tmp/ipython-input-3365528442.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
  df_transaction["PACK_SIZE"] = df_transaction["PROD_NAME"].str.extract(r'(\d+)(?=g)').astype(float)
/tmp/ipython-input-3365528442.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
  df_transaction["BRAND"] = df_transaction["PROD_NAME"].str.split().str[0]
```

```
#with cleaning done now merging with behaviour data to make one data frame

df_merged = pd.merge(
    df_transaction,
    df_behaviour,
    how="left",
    left_on="LYLTY_CARD_NBR",
    right_on="LYLTY_CARD_NBR"
)
```

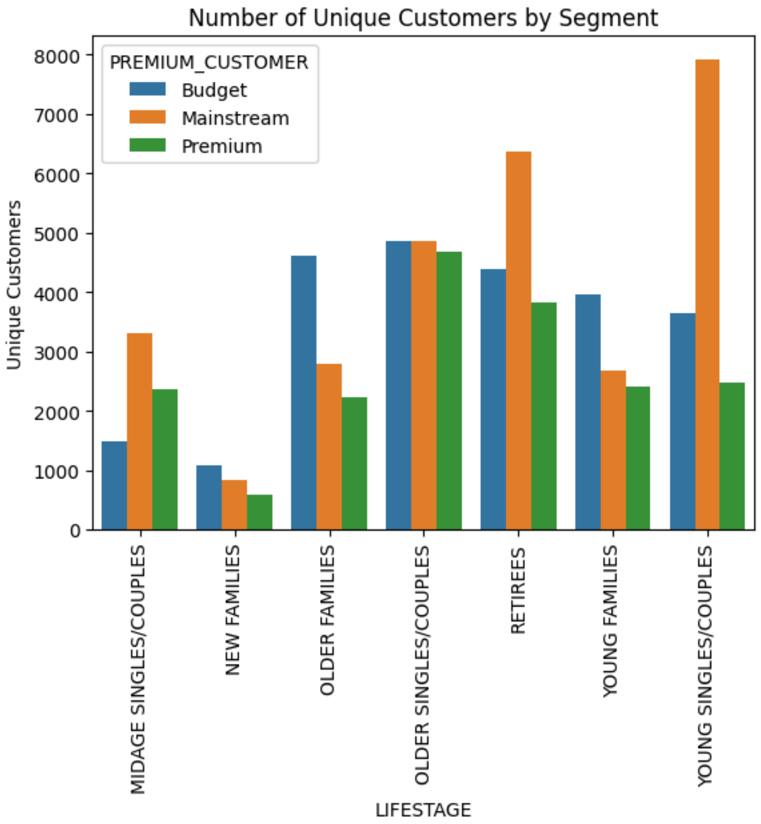
```
#now to segment sales and answer question : who spends the most on chips
segment_sales = df_merged.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"])["TOT_SALES"].sum().reset_index().copy()

sns.barplot(data=segment_sales, x="LIFESTAGE", y="TOT_SALES", hue="PREMIUM_CUSTOMER")
plt.xticks(rotation=90)
plt.title("Total Sales by Customer Segment")
plt.show()
```



```
#find out how many customers there are per segment
#"LYLTY_CARD_NBR" counts the unique amount of customers per segment

segment_customers = df_merged.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"])["LYLTY_CARD_NBR"].nunique().reset_index()

segment_customers
```

	LIFESTAGE	PREMIUM_CUSTOMER	LYLTY_CARD_NBR
0	MIDAGE SINGLES/COUPLES	Budget	1474
1	MIDAGE SINGLES/COUPLES	Mainstream	3298
2	MIDAGE SINGLES/COUPLES	Premium	2369
3	NEW FAMILIES	Budget	1087
4	NEW FAMILIES	Mainstream	830
5	NEW FAMILIES	Premium	575
6	OLDER FAMILIES	Budget	4611
7	OLDER FAMILIES	Mainstream	2788
8	OLDER FAMILIES	Premium	2231
9	OLDER SINGLES/COUPLES	Budget	4849
10	OLDER SINGLES/COUPLES	Mainstream	4858
11	OLDER SINGLES/COUPLES	Premium	4682
12	RETIREES	Budget	4385
13	RETIREES	Mainstream	6358
14	RETIREES	Premium	3812
15	YOUNG FAMILIES	Budget	3953
16	YOUNG FAMILIES	Mainstream	2685
17	YOUNG FAMILIES	Premium	2398
18	YOUNG SINGLES/COUPLES	Budget	3647
19	YOUNG SINGLES/COUPLES	Mainstream	7917
20	YOUNG SINGLES/COUPLES	Premium	2480

```
#plotting the data
sns.barplot(data=segment_customers, x="LIFESTAGE", y="LYLTY_CARD_NBR", hue="PREMIUM_CUSTOMER")
plt.title("Number of Unique Customers by Segment")
plt.xticks(rotation=90)
plt.ylabel("Unique Customers")
plt.show()
```



```
#avg units per customer, usually median is better, but mean tells a more accurate story in this case
```

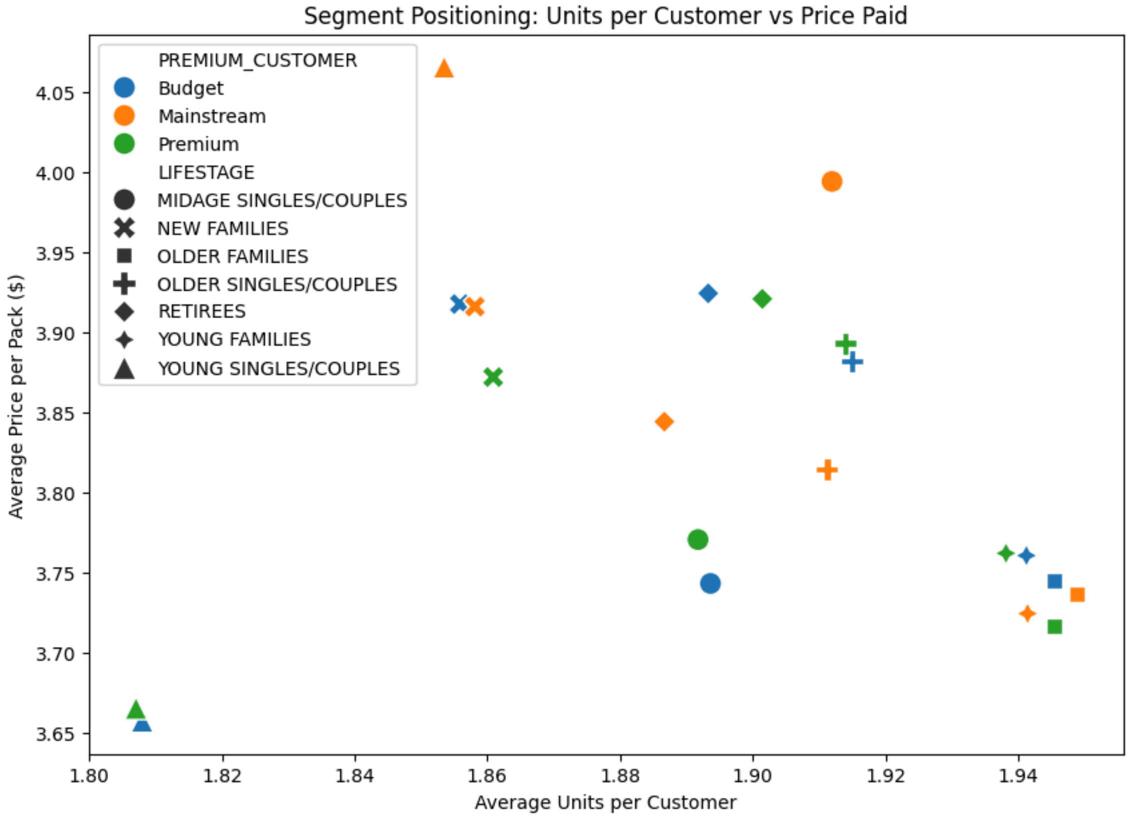
```
units_per_customer = df_merged.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"])["PROD_QTY"].mean().reset_index()

units_per_customer
```

	LIFESTAGE	PREMIUM_CUSTOMER	PROD_QTY
0	MIDAGE SINGLES/COUPLES	Budget	1.893626
1	MIDAGE SINGLES/COUPLES	Mainstream	1.911942
2	MIDAGE SINGLES/COUPLES	Premium	1.891750
3	NEW FAMILIES	Budget	1.855878
4	NEW FAMILIES	Mainstream	1.858124
5	NEW FAMILIES	Premium	1.860887
6	OLDER FAMILIES	Budget	1.945384
7	OLDER FAMILIES	Mainstream	1.948795
8	OLDER FAMILIES	Premium	1.945496
9	OLDER SINGLES/COUPLES	Budget	1.914920
10	OLDER SINGLES/COUPLES	Mainstream	1.911201
11	OLDER SINGLES/COUPLES	Premium	1.913949
12	RETIREES	Budget	1.893286
13	RETIREES	Mainstream	1.886680
14	RETIREES	Premium	1.901438
15	YOUNG FAMILIES	Budget	1.941226
16	YOUNG FAMILIES	Mainstream	1.941408
17	YOUNG FAMILIES	Premium	1.938149
18	YOUNG SINGLES/COUPLES	Budget	1.808002
19	YOUNG SINGLES/COUPLES	Mainstream	1.853510
20	YOUNG SINGLES/COUPLES	Premium	1.807075

```
#now to find the average unit price per segment

df_merged["UNIT_PRICE"] = df_merged["TOT_SALES"] / df_merged["PROD_QTY"]
avg_price_segment = df_merged.groupby(["LIFESTAGE","PREMIUM_CUSTOMER"])["UNIT_PRICE"].mean().reset_index()

avg_price_segment
```

	LIFESTAGE	PREMIUM_CUSTOMER	UNIT_PRICE
0	MIDAGE SINGLES/COUPLES	Budget	3.743328
1	MIDAGE SINGLES/COUPLES	Mainstream	3.994241

```
#combine segments to plot
segment_combined = pd.merge(units_per_customer, avg_price_segment, on=["LIFESTAGE","PREMIUM_CUSTOMER"])

plt.figure(figsize=(10,7))
sns.scatterplot(
    data=segment_combined,
    x="PROD_QTY", y="UNIT_PRICE",
    hue="PREMIUM_CUSTOMER", style="LIFESTAGE", s=150)
plt.title("Segment Positioning: Units per Customer vs Price Paid")
plt.xlabel("Average Units per Customer")
plt.ylabel("Average Price per Pack ($)")
plt.show()
```

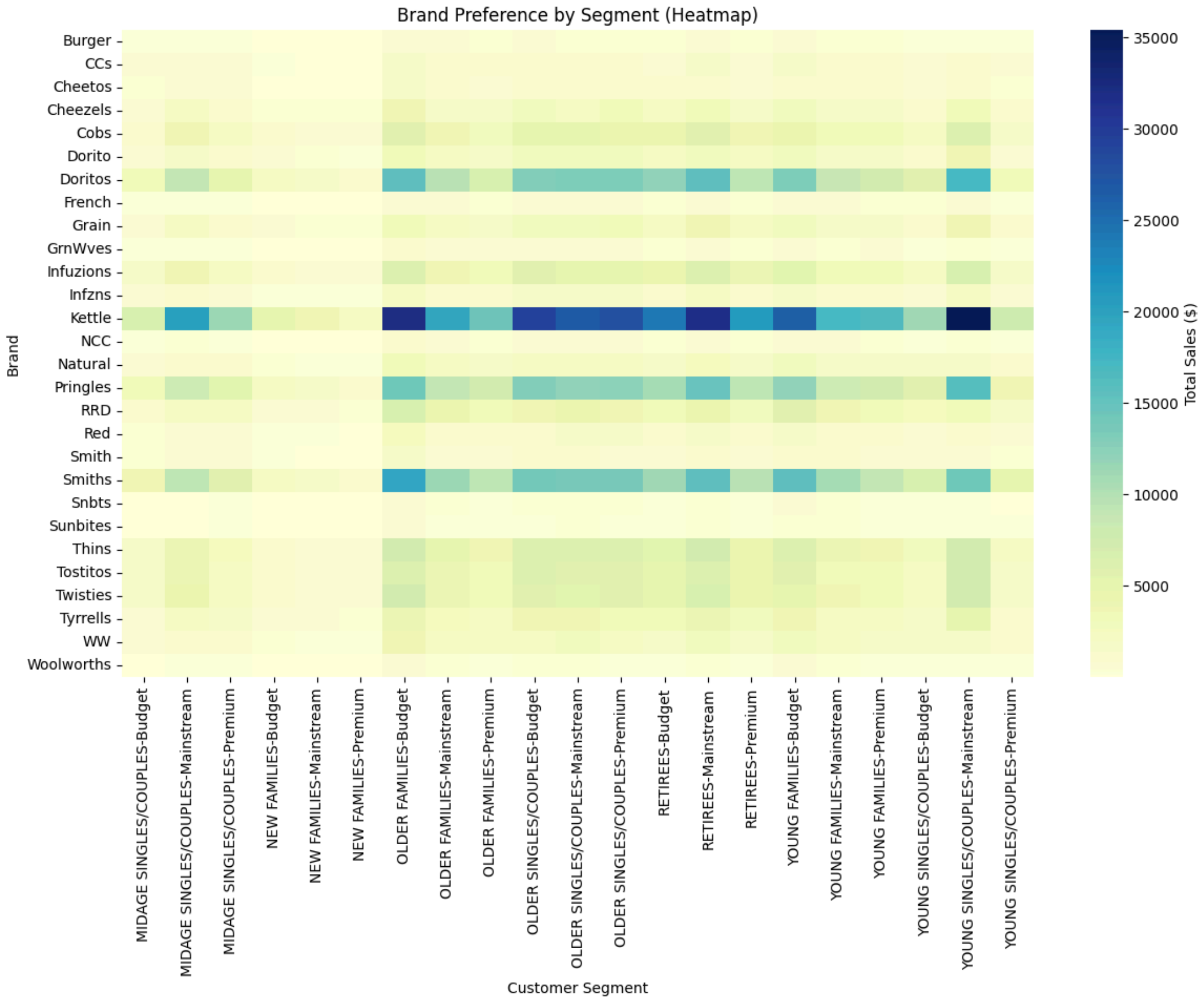


```
#now locating the top brand based on segments

top_brands = df_merged.groupby(["LIFESTAGE","PREMIUM_CUSTOMER","BRAND"])["TOT_SALES"].sum().reset_index()
top_brands
```

	LIFESTAGE	PREMIUM_CUSTOMER	BRAND	TOT_SALES
0	MIDAGE SINGLES/COUPLES	Budget	Burger	193.2
1	MIDAGE SINGLES/COUPLES	Budget	CCs	430.5
2	MIDAGE SINGLES/COUPLES	Budget	Cheetos	337.6
3	MIDAGE SINGLES/COUPLES	Budget	Cheezels	612.3
4	MIDAGE SINGLES/COUPLES	Budget	Cobs	1311.0
...	...	...	...	...
583	YOUNG SINGLES/COUPLES	Premium	Tostitos	1698.4
584	YOUNG SINGLES/COUPLES	Premium	Twisties	1619.0
585	YOUNG SINGLES/COUPLES	Premium	Tyrrells	991.2
586	YOUNG SINGLES/COUPLES	Premium	WW	1105.9
587	YOUNG SINGLES/COUPLES	Premium	Woolworths	163.8

588 rows × 4 columns

```
#convert to pivot table and plot

brand_pivot = top_brands.pivot_table(
    values="TOT_SALES",
    index="BRAND",
    columns=["LIFESTAGE","PREMIUM_CUSTOMER"],
    fill_value=0
)

plt.figure(figsize=(14,8))
sns.heatmap(brand_pivot, cmap="YlGnBu", cbar_kws={'label': 'Total Sales ($)'})
plt.title("Brand Preference by Segment (Heatmap)")
plt.ylabel("Brand")
plt.xlabel("Customer Segment")
plt.show()
```



```
#additional pack size preference segments

pack_size_pref = df_merged.groupby(["LIFESTAGE","PREMIUM_CUSTOMER","PACK_SIZE"])["TOT_SALES"].sum().reset_index()

pack_size_pref
```

```
                   LIFESTAGE  PREMIUM_CUSTOMER  PACK_SIZE  TOT_SALES
  0   MIDAGE SINGLES/COUPLES            Budget       70.0      122.4
  1   MIDAGE SINGLES/COUPLES            Budget       90.0      222.7
  2   MIDAGE SINGLES/COUPLES            Budget      110.0     3146.4
  3   MIDAGE SINGLES/COUPLES            Budget      125.0      105.0
  4   MIDAGE SINGLES/COUPLES            Budget      134.0     3159.8
 ...                     ...               ...        ...        ...
415    YOUNG SINGLES/COUPLES           Premium      220.0      234.6
416    YOUNG SINGLES/COUPLES           Premium      250.0      464.4
417    YOUNG SINGLES/COUPLES           Premium      270.0     1154.6
418    YOUNG SINGLES/COUPLES           Premium      330.0     2627.7
419    YOUNG SINGLES/COUPLES           Premium      380.0     1640.3

420 rows × 4 columns
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```