In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
customer_data = pd.read_csv('QVI_purchase_behaviour.csv')
```

In [3]:
```
transaction_data = pd.read_excel('QVI_transaction_data.xlsx')
```

In [4]:
```
customer_data.head()
```

Out[4]:

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

In [5]:
```
customer_data.shape
```

Out[5]:
```
(72637, 3)
```

In [6]:
```
transaction_data.head()
```

Out[6]:

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

In [7]:
```
transaction_data.shape
```

Out[7]: (264836, 8)

In [8]:
```
df = pd.merge(customer_data, transaction_data, on = 'LYLTY_CARD_NBR', how = 'outer'
```

In [9]:
```
df.shape
```

Out[9]: (264836, 10)

In [10]:
```
df.head(20)
```

Out[10]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID
0	1000	YOUNG SINGLES/COUPLES	Premium	2018-10-17	1	1
1	1002	YOUNG SINGLES/COUPLES	Mainstream	2018-09-16	1	2
2	1003	YOUNG FAMILIES	Budget	2019-03-07	1	3
3	1003	YOUNG FAMILIES	Budget	2019-03-08	1	4
4	1004	OLDER SINGLES/COUPLES	Mainstream	2018-11-02	1	5
5	1005	MIDAGE SINGLES/COUPLES	Mainstream	2018-12-28	1	6
6	1007	YOUNG SINGLES/COUPLES	Budget	2018-12-04	1	7
7	1007	YOUNG SINGLES/COUPLES	Budget	2018-12-05	1	8
8	1009	NEW FAMILIES	Premium	2018-11-20	1	9
9	1010	YOUNG SINGLES/COUPLES	Mainstream	2018-09-09	1	10
10	1010	YOUNG SINGLES/COUPLES	Mainstream	2018-12-14	1	11
11	1011	OLDER SINGLES/COUPLES	Mainstream	2018-07-29	1	12
12	1011	OLDER SINGLES/COUPLES	Mainstream	2018-11-08	1	13
13	1011	OLDER SINGLES/COUPLES	Mainstream	2018-12-01	1	14
14	1011	OLDER SINGLES/COUPLES	Mainstream	2018-12-19	1	15

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID
15	1012	OLDER FAMILIES	Mainstream	2019-03-15	1	16
16	1012	OLDER FAMILIES	Mainstream	2019-06-19	1	17
17	1013	RETIREES	Budget	2019-03-04	1	18
18	1013	RETIREES	Budget	2019-03-07	1	19
19	1016	OLDER FAMILIES	Mainstream	2019-04-19	1	20

In [11]:
```
df.info()
```

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

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
DATE                0
STORE_NBR           0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
dtype: int64
```

In [13]:
```
df.duplicated().sum()
```

Out[13]: 1

In [14]:
```
df.drop_duplicates(keep='last', inplace = True)
```

In [15]:
```
df.shape
```

Out[15]: (264835, 10)

In [16]:
```
import re

# Define regex pattern to extract numbers at the end of each product name
pattern = r'(\d+)[gG]' # Matches one or more digits followed by 'g' at the end of t

# Extract numbers at the end of each product name using regex and create a new colu
df['PROD_WEIGHT'] = df['PROD_NAME'].apply(lambda x: re.search(pattern, x).group(1)

# Convert the 'Product_Weight' column to numeric type
df['PROD_WEIGHT'] = pd.to_numeric(df['PROD_WEIGHT'])
```

In [17]:
```
df.head()
```

Out[17]:

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

In [18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 264835 entries, 0 to 264835
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264835 non-null  int64
 1   LIFESTAGE         264835 non-null  object
 2   PREMIUM_CUSTOMER  264835 non-null  object
 3   DATE              264835 non-null  datetime64[ns]
 4   STORE_NBR         264835 non-null  int64
 5   TXN_ID            264835 non-null  int64
 6   PROD_NBR          264835 non-null  int64
 7   PROD_NAME         264835 non-null  object
 8   PROD_QTY          264835 non-null  int64
 9   TOT_SALES         264835 non-null  float64
 10  PROD_WEIGHT       264835 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(6), object(3)
memory usage: 24.2+ MB
```

In [19]:
```
df.isnull().sum()
```

Out[19]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
DATE                0
STORE_NBR           0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PROD_WEIGHT         0
dtype: int64
```

In [20]:
```
df.to_csv('output.csv') #download the file to add the brand name feature that has a
```

In [21]:
```
df = pd.read_csv('chips.csv') #read the file with a new feature
```

In [22]:
```
df.head()
```

Out[22]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TX
0	1000	YOUNG SINGLES/COUPLES	Premium	17/10/2018	1	
1	1002	YOUNG SINGLES/COUPLES	Mainstream	16/09/2018	1	
2	1003	YOUNG FAMILIES	Budget	7/3/2019	1	
3	1003	YOUNG FAMILIES	Budget	8/3/2019	1	
4	1004	OLDER SINGLES/COUPLES	Mainstream	2/11/2018	1	

In [23]:
```
df['DATE'] = pd.to_datetime(df['DATE'])
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9452\2541906660.py:1: UserWarning: Parsing
dates in %d/%m/%Y format when dayfirst=False (the default) was specified. Pass `dayf
irst=True` or specify a format to silence this warning.
  df['DATE'] = pd.to_datetime(df['DATE'])
```

In [24]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264835 entries, 0 to 264834
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264835 non-null  int64
 1   LIFESTAGE         264835 non-null  object
 2   PREMIUM_CUSTOMER  264835 non-null  object
 3   DATE              264835 non-null  datetime64[ns]
 4   STORE_NBR         264835 non-null  int64
 5   TXN_ID            264835 non-null  int64
 6   PROD_NBR          264835 non-null  int64
 7   PROD_NAME         264835 non-null  object
 8   PROD_QTY          264835 non-null  int64
 9   TOT_SALES         264835 non-null  float64
 10  BRAND_NAME        264835 non-null  object
 11  PROD_WEIGHT       264835 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(6), object(4)
memory usage: 24.2+ MB
```

Data Cleaning is completed

- i merged the 2 tables together
- created 2 new columns (Brand name & Product weight)
- no missing values
- one duplicated, droped it
- changed the data type of the date column to datetime

Exploratory Data Analysis

- univariate analysis
- bivariate analysis
- coorelation
-

In [25]:
```
df.describe().T
```

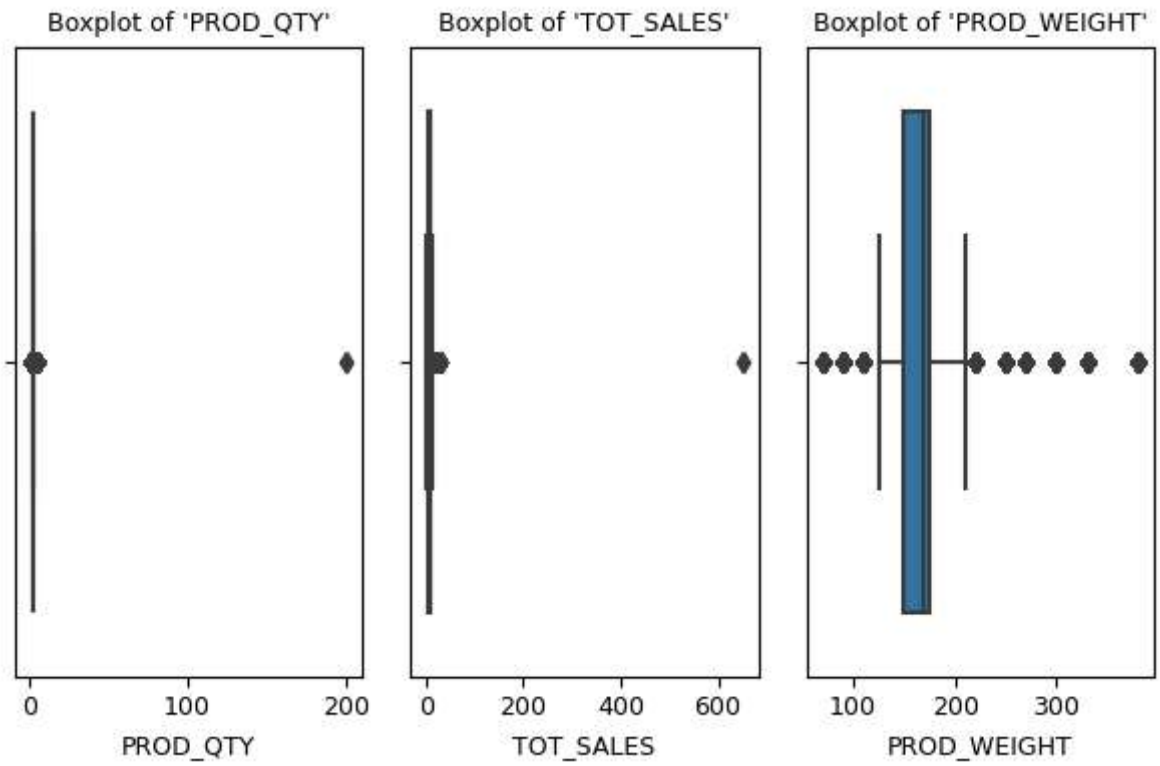
Out[25]:

	count	mean	min	25%	50%	75%	
LYLTY_CARD_NBR	264835.0	135549.584115	1000.0	70021.0	130358.0	203094.5	2373
DATE	264835	2018-12-30 00:52:42.252723200	2018-07-01 00:00:00	2018-09-30 00:00:00	2018-12-30 00:00:00	2019-03-31 00:00:00	201 00:
STORE_NBR	264835.0	135.080216	1.0	70.0	130.0	203.0	
TXN_ID	264835.0	135158.411619	1.0	67601.0	135138.0	202701.5	2415
PROD_NBR	264835.0	56.583201	1.0	28.0	56.0	85.0	
PROD_QTY	264835.0	1.907308	1.0	2.0	2.0	2.0	
TOT_SALES	264835.0	7.304205	1.5	5.4	7.4	9.2	
PROD_WEIGHT	264835.0	182.427032	70.0	150.0	170.0	175.0	

total sales looks like it might have an outlier because of the difference between the 75th% and the max

In [26]:
```
# Plotting boxplots for depth, table, x, y, and z
variables = ['PROD_QTY', 'TOT_SALES', 'PROD_WEIGHT']

plt.rc('font', size=9) #to change the font size

# Create subplots
fig, axes = plt.subplots(1, 3, figsize=(6, 4))

# Flatten the axes array
axes = axes.flatten()

# Iterate over variables and plot boxplots
```

```
for i, var in enumerate(variables):
    sns.boxplot(x=var, data=df, ax=axes[i])
    axes[i].set_title(f"Boxplot of '{var}'", fontsize = 9)
    axes[i].set_xlabel(var)

# Adjust layout
plt.tight_layout()
plt.show()
```



In [27]:
```
filtered_df = df[df['TOT_SALES'] == 650.0]
filtered_df
```

Out[27]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER	DATE	STORE_NBR	TXN_ID
221624	226000	OLDER FAMILIES	Premium	2018-08-19	226	226201
221625	226000	OLDER FAMILIES	Premium	2019-05-20	226	226210

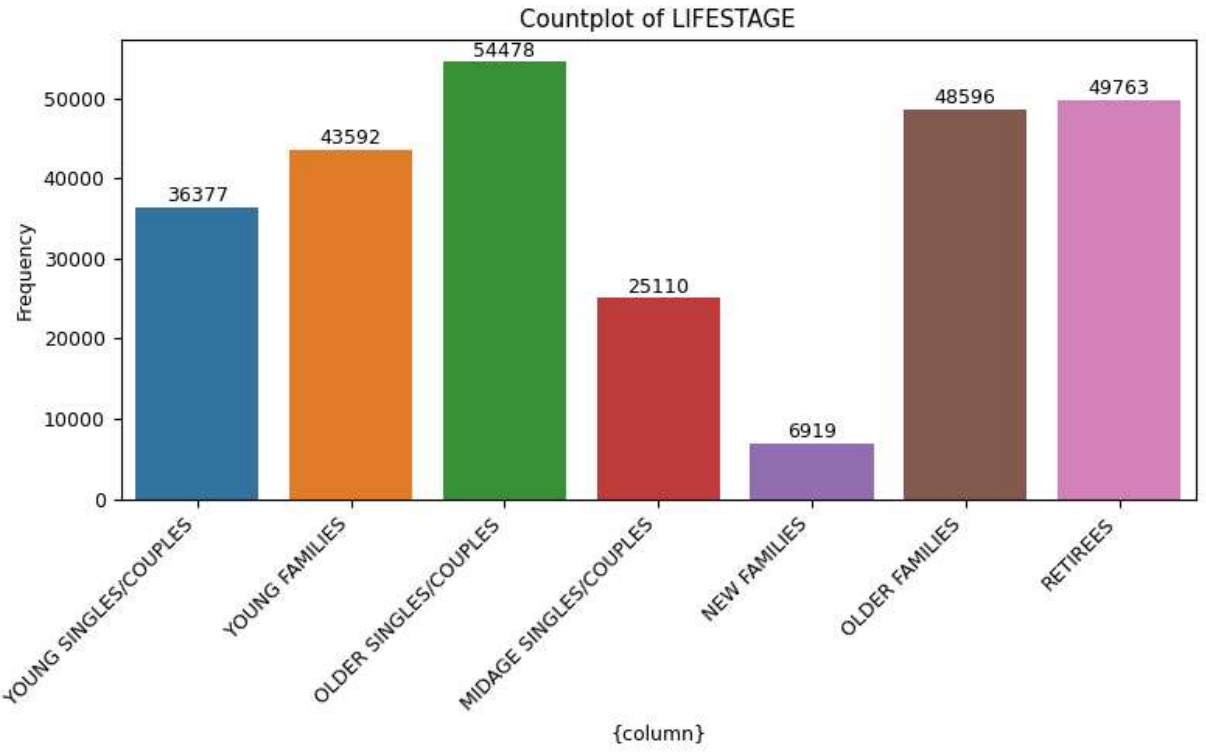
i looked into the outlier in total sales and it corresponds to the one in product quantity

- so, it makes sense that the higher the quantity the higher the sales
- therefore,i will leave the ouitliers to see the trends

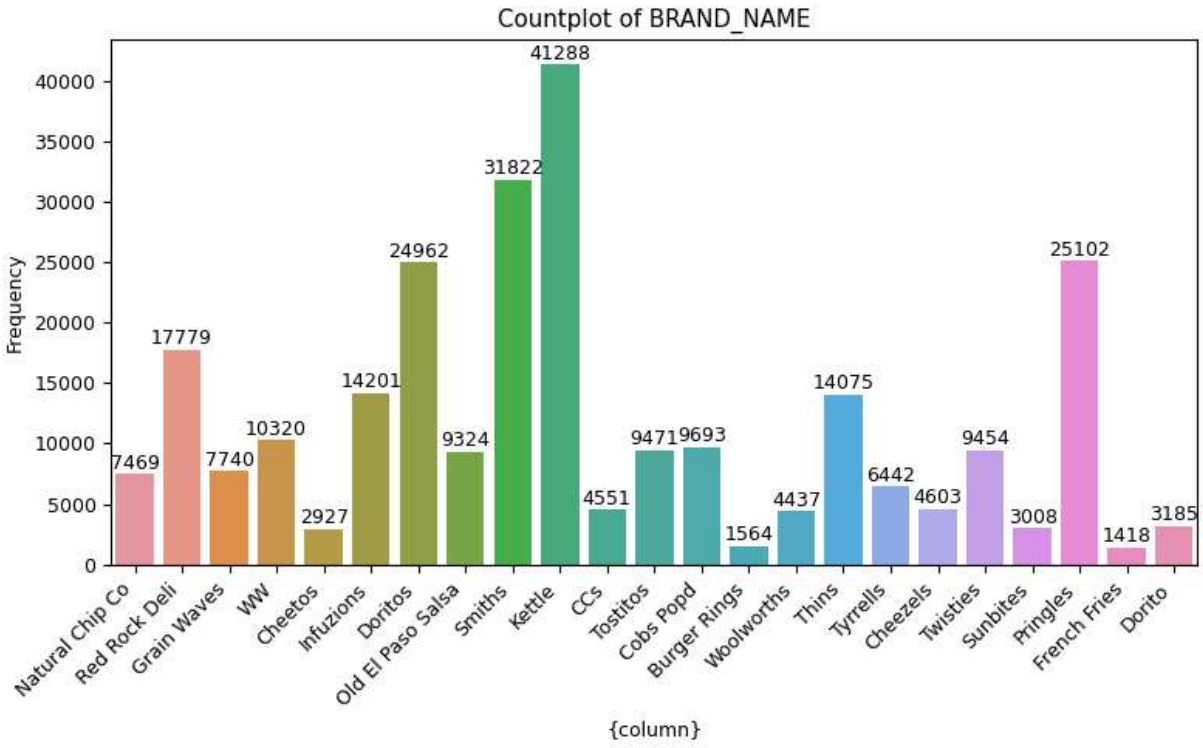
In [ ]:

In [28]:
```
def count_plot(df, column):
    plt.figure(figsize=(8, 5))
    ax = sns.countplot(x=column, data=df)
    plt.title(f'Countplot of {column}')
    plt.xlabel('{column}')
    plt.ylabel('Frequency')
    plt.xticks(rotation=45, ha ='right')  # Rotate x-axis labels for better readabi

    # Add count labels on top of each bar
    for p in ax.patches:
        ax.annotate(format(p.get_height(), '.0f'),
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha = 'center', va = 'center',
                    xytext = (0, 5),
                    textcoords = 'offset points')

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

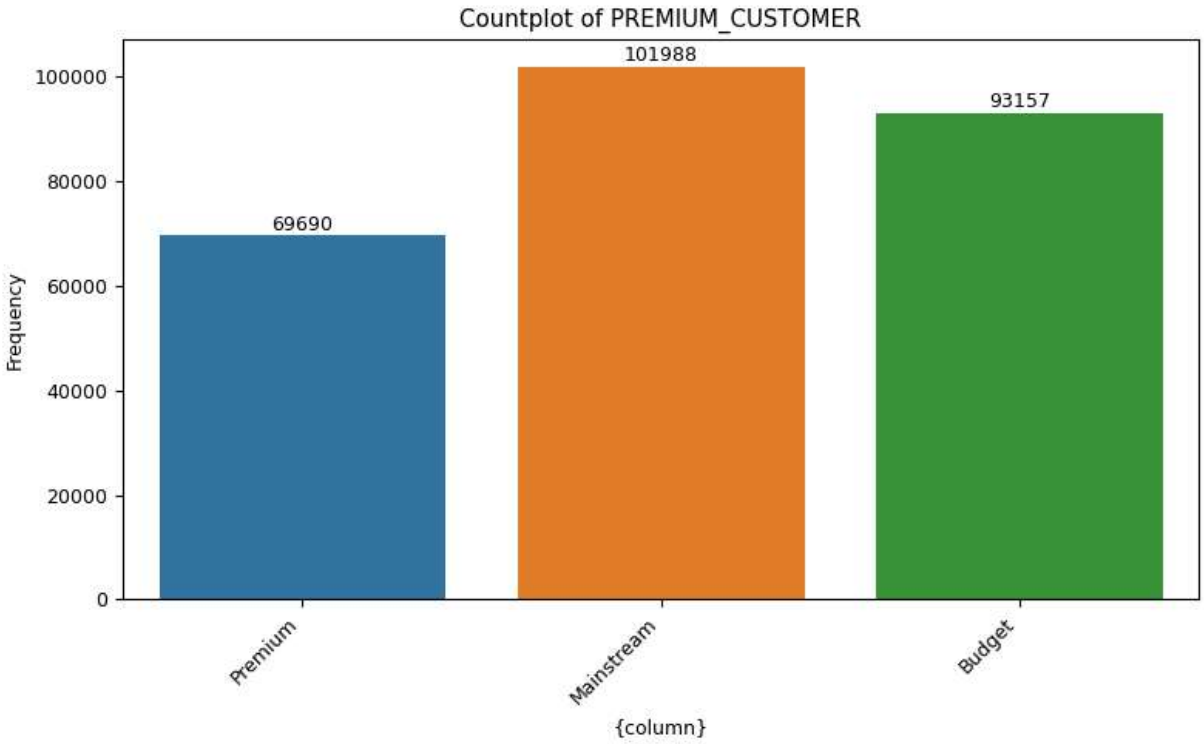
In [29]:
```
count_plot(df, 'LIFESTAGE')
```



In [30]:
```
count_plot(df, 'BRAND_NAME')
```



In [31]:
```
count_plot(df, 'PREMIUM_CUSTOMER')
```



The older singles/couples have the most transactions, followed by the retirees and older families

kettle had the most transaction and salefollowed by smiths, pringles and doritos

mainstream has the most transactions followed by budget then premium

In [ ]:

Customer Segmentation:

1. Define Customer Segments:

Total Sales by LIFESTAGE and PREMIUM_CUSTOMER

In [32]:
```
# Define metrics of interest
customer_segment = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).agg(
    unique_customers=('LYLTY_CARD_NBR', 'nunique'),
    total_sales=('TOT_SALES', 'sum'),
    total_quantity=('PROD_QTY', 'sum'),
    ave_quantity=('PROD_QTY', 'mean'),
    mode_pack_size=('PROD_WEIGHT', lambda x: x.mode().iat[0]),
    ave_pack_size=('PROD_WEIGHT', 'mean')
    ).reset_index()

customer_segments = customer_segment.sort_values(by = 'total_sales', ascending = Fa
customer_segments
```

Out[32]:

	LIFESTAGE	PREMIUM_CUSTOMER	unique_customers	total_sales	total_quantity
6	OLDER FAMILIES	Budget	4675	168363.25	45065
19	YOUNG SINGLES/COUPLES	Mainstream	8088	157621.60	38632
13	RETIREES	Mainstream	6479	155677.05	40518
15	YOUNG FAMILIES	Budget	4017	139345.85	37111
9	OLDER SINGLES/COUPLES	Budget	4929	136769.80	35220
10	OLDER SINGLES/COUPLES	Mainstream	4930	133393.80	34997
11	OLDER SINGLES/COUPLES	Premium	4750	132257.15	33984
12	RETIREES	Budget	4454	113147.80	28764
7	OLDER FAMILIES	Mainstream	2831	103445.55	27756
14	RETIREES	Premium	3872	97646.05	24884
16	YOUNG FAMILIES	Mainstream	2728	92788.75	25044
1	MIDAGE SINGLES/COUPLES	Mainstream	3340	90803.85	22699
17	YOUNG FAMILIES	Premium	2433	84025.50	22406
8	OLDER FAMILIES	Premium	2274	81958.40	22171
18	YOUNG SINGLES/COUPLES	Budget	3779	61141.60	16671
2	MIDAGE SINGLES/COUPLES	Premium	2431	58432.65	15526
20	YOUNG SINGLES/COUPLES	Premium	2574	41642.10	11331
0	MIDAGE SINGLES/COUPLES	Budget	1504	35514.80	9496
3	NEW FAMILIES	Budget	1112	21928.45	5571
4	NEW FAMILIES	Mainstream	849	17013.90	4319
5	NEW FAMILIES	Premium	588	11491.10	2957

Customer Segments: The data includes various customer segments such as “MIDAGE SINGLES/COUPLES,” “NEW FAMILIES,” “OLDER FAMILIES,” “OLDER SINGLES/COUPLES,” “RETIREES,” and “YOUNG FAMILIES.”

Each segment has different purchasing behaviors and preferences.

The data is categorized into three premium categories: “Budget,” “Mainstream,” and “Premium.”

Sales Analysis:

- “OLDER FAMILIES” in the “Budget” category have the highest total sales ($168,363.25) and the highest total quantity of products bought,
- followed by young singles/couples in the “Mainstream” category ($157621.60).
- “NEW FAMILIES” in the “Premium” category have the lowest total sales ($11491.10).

from this we can say there is a positive correlation between total sales and total quantity bought

Average Quantity Analysis:

- “OLDER FAMILIES” in the “Premium” category have the highest average quantity (1.98), indicating they purchase more items per transaction.
- “YOUNG SINGLES/COUPLES” in the “Budget” category have the lowest average quantity (1.80).

Unique Customers Analysis:

- “YOUNG SINGLES/COUPLES” in the “Mainstream” category have the highest number of unique customers (8,088).
- “NEW FAMILIES” in the “Premium” category have the lowest number of unique customers (588).

Pack Size Preferences: The most frequently purchased pack size (mode) is 175g across most segments.

However, the average pack size (ave_pack_size) shows some variation, suggesting some segments might occasionally purchase larger packs while still favoring the 175g option.

Overall Conclusion:

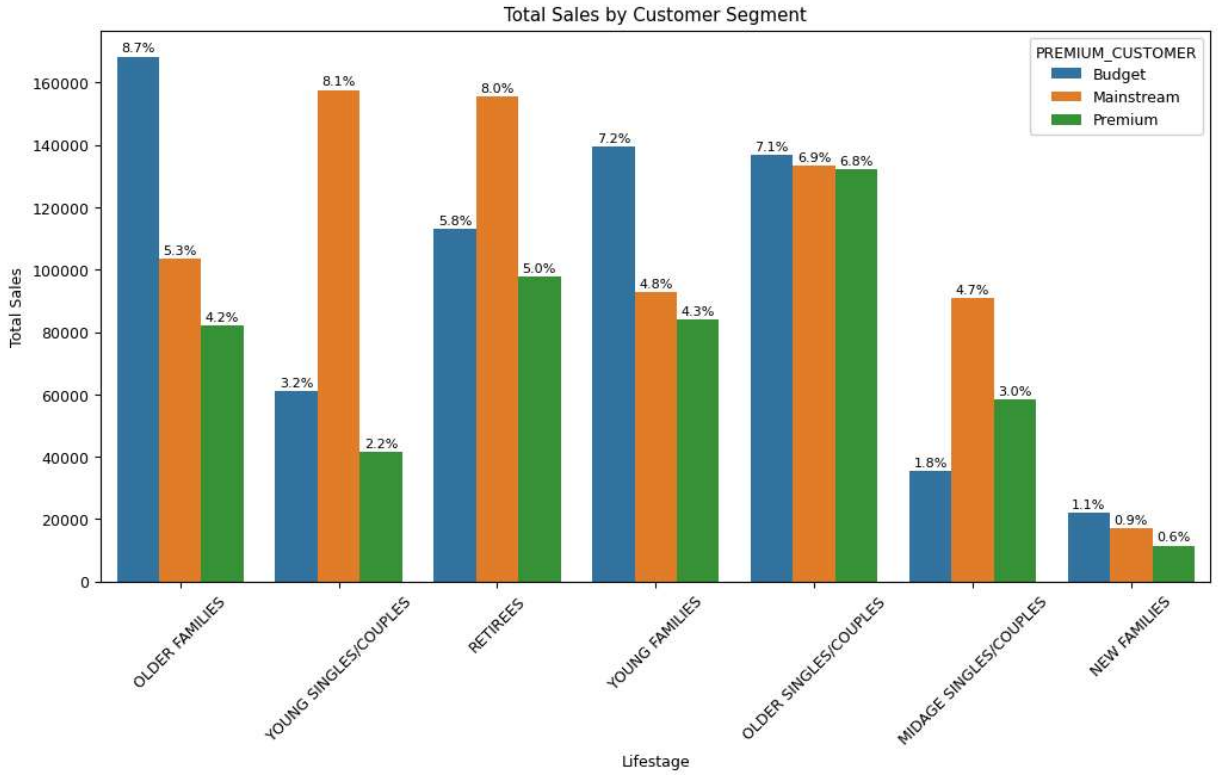
- "OLDER FAMILIES" and "OLDER SINGLES/COUPLES" tend to have higher total sales compared to segments like "NEW FAMILIES" or "YOUNG SINGLES/COUPLES".

- Segments with a higher number of unique customers, such as "YOUNG SINGLES/COUPLES" in the Mainstream category, represent larger market opportunities compared to segments with fewer unique customers.

They are also the 2nd highest spenders

In [33]:
```
#lets visualize the metrics
```

```
plt.figure(figsize=(12, 6))
ax = sns.barplot(x='LIFESTAGE', y='total_sales', hue='PREMIUM_CUSTOMER', data=custo
plt.title('Total Sales by Customer Segment')
plt.xlabel('Lifestage')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)

for p in ax.patches:
    ax.annotate('{:.1f}%'.format(p.get_height() / customer_segments['total_sales'].
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center',
                xytext=(0, 5),
                fontsize = 8,
                textcoords='offset points')


plt.show()
```



PRODUCTS

In [ ]:

In [34]:
```
def most_popular(brand_series):
  """
  This function finds the mode (most frequent value) of a brand series.
  If there's a tie for most frequent brands, it returns all tied brands.
  """
  return brand_series.mode().tolist()
```

In [35]:
```
segment_top_brands = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['BRAND_NAME'].ap
segment_top_brands = segment_top_brands.rename(columns={'BRAND_NAME': 'Most Popular

segment_top_brands
```

Out[35]:

	LIFESTAGE	PREMIUM_CUSTOMER	Most Popular Brand
0	MIDAGE SINGLES/COUPLES	Budget	[Kettle]
1	MIDAGE SINGLES/COUPLES	Mainstream	[Kettle]
2	MIDAGE SINGLES/COUPLES	Premium	[Kettle]
3	NEW FAMILIES	Budget	[Kettle]
4	NEW FAMILIES	Mainstream	[Kettle]
5	NEW FAMILIES	Premium	[Kettle]
6	OLDER FAMILIES	Budget	[Kettle]
7	OLDER FAMILIES	Mainstream	[Kettle]
8	OLDER FAMILIES	Premium	[Smiths]
9	OLDER SINGLES/COUPLES	Budget	[Kettle]
10	OLDER SINGLES/COUPLES	Mainstream	[Kettle]
11	OLDER SINGLES/COUPLES	Premium	[Kettle]
12	RETIREES	Budget	[Kettle]
13	RETIREES	Mainstream	[Kettle]
14	RETIREES	Premium	[Kettle]
15	YOUNG FAMILIES	Budget	[Kettle]
16	YOUNG FAMILIES	Mainstream	[Kettle]
17	YOUNG FAMILIES	Premium	[Kettle]
18	YOUNG SINGLES/COUPLES	Budget	[Smiths]
19	YOUNG SINGLES/COUPLES	Mainstream	[Kettle]
20	YOUNG SINGLES/COUPLES	Premium	[Kettle]

Across most customer segments, "Kettle" appears as the most popular brand. This suggests "Kettle" might have a broad appeal across different demographics and spending habits in your data.

Exceptions: There are a few segments with exceptions:

- **Older Families (Premium):** "Smiths" is the most popular brand for Older Families in the Premium category.

- **Young Singles/Couples (Budget):** "Smiths" is also the most popular brand for Young Singles/Couples in the Budget category.

In [36]:
```
segment_top_product = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PROD_NAME'].ap
segment_top_product = segment_top_product.rename(columns={'PROD_NAME': 'Most Popula

segment_top_product
```

Out[36]:

	LIFESTAGE	PREMIUM_CUSTOMER	Most Popular product
0	MIDAGE SINGLES/COUPLES	Budget	[Infzns Crn Crnchers Tangy Gcamole 110g]
1	MIDAGE SINGLES/COUPLES	Mainstream	[Smiths Crinkle Chips Salt & Vinegar 330g]
2	MIDAGE SINGLES/COUPLES	Premium	[Pringles Sweet&Spcy BBQ 134g]
3	NEW FAMILIES	Budget	[Kettle Honey Soy Chicken 175g]
4	NEW FAMILIES	Mainstream	[Kettle 135g Swt Pot Sea Salt, Pringles Slt Vi...
5	NEW FAMILIES	Premium	[Grain Waves Sweet Chilli 210g]
6	OLDER FAMILIES	Budget	[Smiths Crinkle Chips Salt & Vinegar 330g]
7	OLDER FAMILIES	Mainstream	[Old El Paso Salsa Dip Chnky Tom Ht300g]
8	OLDER FAMILIES	Premium	[Pringles Chicken Salt Crips 134g]
9	OLDER SINGLES/COUPLES	Budget	[Cobs Popd Sea Salt Chips 110g]
10	OLDER SINGLES/COUPLES	Mainstream	[Kettle 135g Swt Pot Sea Salt]
11	OLDER SINGLES/COUPLES	Premium	[Doritos Corn Chip Southern Chicken 150g]
12	RETIREES	Budget	[Kettle Chilli 175g]
13	RETIREES	Mainstream	[Grain Waves Sweet Chilli 210g]
14	RETIREES	Premium	[Infuzions Thai SweetChili PotatoMix 110g]
15	YOUNG FAMILIES	Budget	[Old El Paso Salsa Dip Tomato Med 300g]
16	YOUNG FAMILIES	Mainstream	[Pringles Sthrn FriedChicken 134g]
17	YOUNG FAMILIES	Premium	[Cobs Popd Swt/Chlli &Sr/Cream Chips 110g]
18	YOUNG SINGLES/COUPLES	Budget	[Pringles Original Crisps 134g]
19	YOUNG SINGLES/COUPLES	Mainstream	[Tostitos Splash Of Lime 175g]
20	YOUNG SINGLES/COUPLES	Premium	[Cobs Popd Sour Crm &Chives Chips 110g]

Brand Trends: Some brands seem more popular within specific segments:

- Smiths: Popular among Budget and Mainstream customers in some segments (e.g., Midage Singles/Couples, Older Families).
- Kettle: Popular across various segments, but not always the most popular.
- Pringles: Appears popular in some Premium segments (e.g., Midage Singles/Couples, Older Families).

Specific Product Insights: The results reveal specific popular products within segments, potentially providing clues about buying behavior:

- Budget - Young Families: "Old El Paso Salsa Dip Tomato Med 300g" might be a popular choice for families looking for a larger, budget-friendly dip.
- Premium - Young Families: "Cobs Popd Swt/Chlli &Sr/Cream Chips 110g" suggests a preference for combination flavors among some premium young families.

In [ ]:

In [37]:
```
# Calculate total sales, average purchase amount, purchase frequency, etc.

average_purchase = df['TOT_SALES'].mean()
purchase_frequency = df['TXN_ID'].nunique()  # Count unique transaction IDs

# Analyze popular brands and pack sizes
popular_brands = df['BRAND_NAME'].mode().iloc[0]  # Assuming single most popular br
popular_pack_size = df['PROD_WEIGHT'].mode().iloc[0]

print(f"Average purchase amount: ${average_purchase:.2f}")
print(f"Purchase frequency: {purchase_frequency}")
print(f"Popular brands: {popular_brands}")
print(f"Popular pack size: {popular_pack_size}g")
```

```
Average purchase amount: $7.30
Purchase frequency: 263127
Popular brands: Kettle
Popular pack size: 175g
```

TOP 5 BRANDS BY SALES

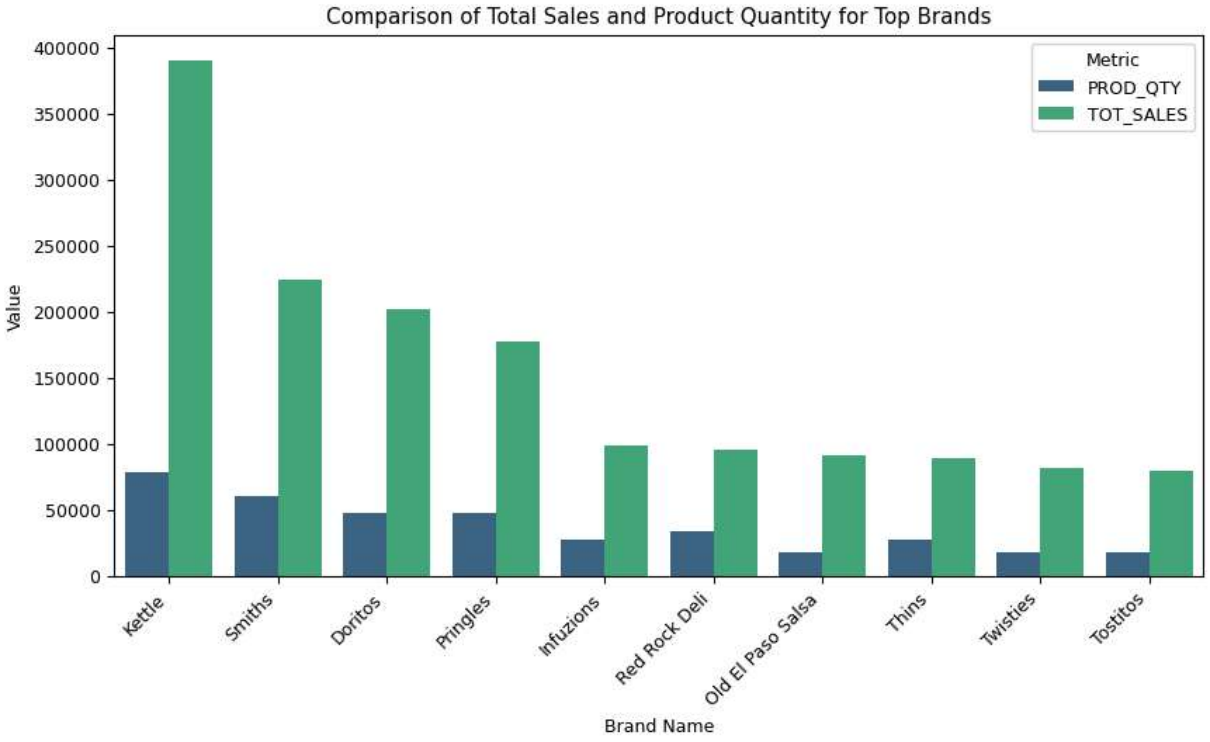
In [38]:
```
# Group by brand and sum the quantity sold and total sales
brand_sales_qty = df.groupby(['BRAND_NAME']).agg(
                                            {'PROD_QTY': 'sum', 'TOT_SALES': 'sum'}

# Sort in descending order and get the top 5 brands
top_brands = brand_sales_qty.sort_values(by='TOT_SALES', ascending=False).head(10)

top_brands
```

Out[38]:

	BRAND_NAME	PROD_QTY	TOT_SALES
10	Kettle	79051	390239.8
15	Smiths	60337	224654.2
6	Doritos	47707	201538.9
13	Pringles	48019	177655.5
9	Infuzions	27119	99047.6
14	Red Rock Deli	33646	95046.0
12	Old El Paso Salsa	17805	90785.1
17	Thins	26929	88852.5
19	Twisties	18118	81522.1
18	Tostitos	18134	79789.6

In [39]:

```
# Melt the DataFrame to long format for easier plotting with seaborn
melted_df = top_brands.melt(id_vars='BRAND_NAME', value_vars=['PROD_QTY', 'TOT_SALE
                            var_name='Metric', value_name='Value')

# Create the grouped bar plot
plt.figure(figsize=(10, 5))
bar_plot = sns.barplot(x='BRAND_NAME', y='Value', hue='Metric', data=melted_df, pal

# Add labels and title
plt.ylabel('Value')
plt.xlabel('Brand Name')
plt.title('Comparison of Total Sales and Product Quantity for Top Brands')
plt.xticks(rotation = 45, ha = 'right')

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



the highest sale by brand is kettle, then smiths, doritos, pringle

recommendations: increase quantity of top 3 brands to bring more sales especially for kettle chips

Top Popular Products by Total Sales

In [40]:
```
# Group by product and sum the quantity sold
product_sales = df.groupby(['PROD_NAME', 'PROD_NBR'])['TOT_SALES'].sum().reset_inde

top_sales = product_sales.sort_values(by = 'TOT_SALES', ascending = False).head(10)
top_sales
```

Out[40]:

PROD_NBR	PROD_NAME	TOT_SALES
4	Dorito Corn Chp Supreme 380g	40352.0
14	Smiths Crnkle Chip Orgnl Big Bag 380g	36367.6
16	Smiths Crinkle Chips Salt & Vinegar 330g	34804.2
102	Kettle Mozzarella Basil & Pesto 175g	34457.4
7	Smiths Crinkle Original 330g	34302.6
23	Cheezels Cheese 330g	34296.9
20	Doritos Cheese Supreme 330g	33390.6
89	Kettle Sweet Chilli And Sour Cream 175g	33031.8
46	Kettle Original 175g	32740.2
32	Kettle Sea Salt And Vinegar 175g	32589.0

Top-Selling Product: Dorito Corn Chip Supreme 380g has the highest sales. **Top Brands by Sales:** The top ten products by sales belong to Doritos, Smiths, Kettle, and Cheezels brands. **Popular Pack Sizes:** The 380g, 330g, and 175g pack sizes are the most popular in terms of top products by sales.

TOP SELLING PRODUCT BY TOTAL QUANTITY

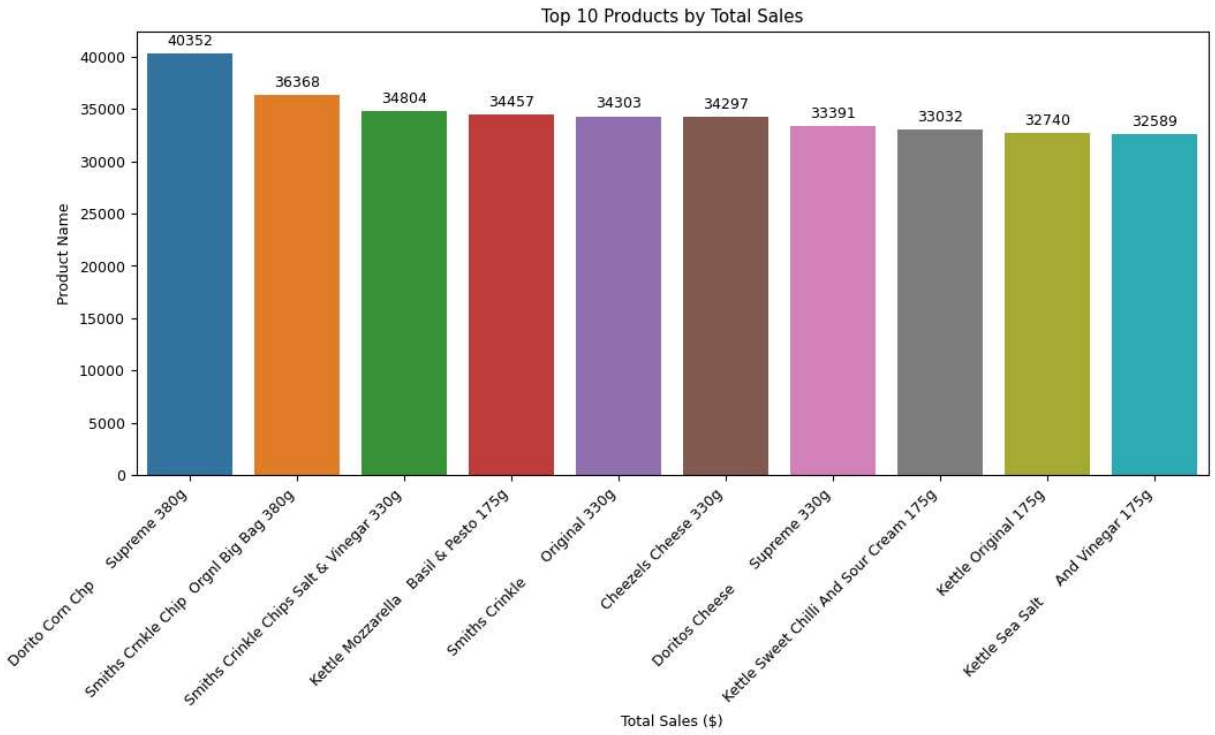
In [41]:
```
# Group by product name and product number, then sum the total sales and quantity s
product_sales = df.groupby(['PROD_NAME', 'PROD_NBR']).agg({'PROD_QTY': 'sum', 'TOT_

# Sort by total quantity sold in descending order and get the top 10 products
top_products = product_sales.sort_values(by='PROD_QTY', ascending=False).head(10)

top_products
```

Out[41]:

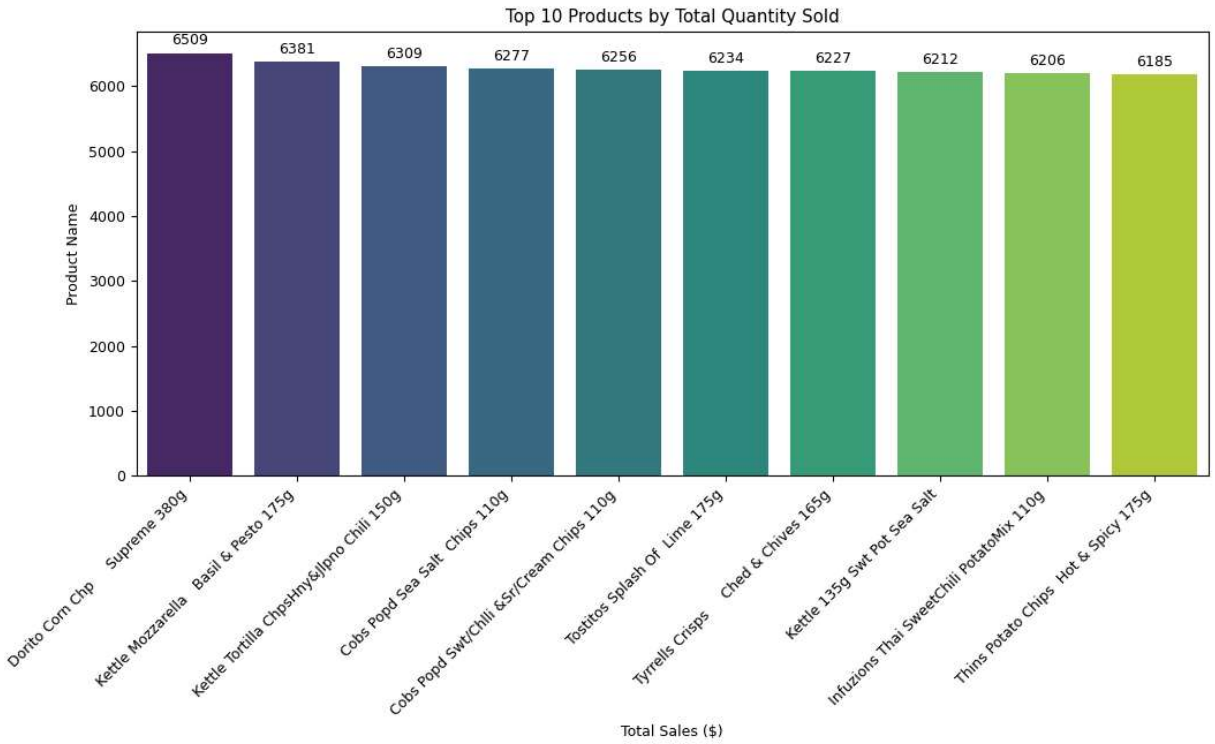
PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
4	Dorito Corn Chp Supreme 380g	6509	40352.0
102	Kettle Mozzarella Basil & Pesto 175g	6381	34457.4
108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	6309	29021.4
75	Cobs Popd Sea Salt Chips 110g	6277	23852.6
33	Cobs Popd Swt/Chlli &Sr/Cream Chips 110g	6256	23772.8
74	Tostitos Splash Of Lime 175g	6234	27429.6
112	Tyrrells Crisps Ched & Chives 165g	6227	26149.2
63	Kettle 135g Swt Pot Sea Salt	6212	26090.4
104	Infuzions Thai SweetChili PotatoMix 110g	6206	23582.8
28	Thins Potato Chips Hot & Spicy 175g	6185	20410.5

In [42]:

```
# Create the bar plot
plt.figure(figsize=(12, 5))
bar_plot = sns.barplot(x='PROD_NAME', y='TOT_SALES', data=top_sales)

# Add labels and title
plt.xlabel('Total Sales ($)')
plt.ylabel('Product Name')
plt.title('Top 10 Products by Total Sales')
plt.xticks(rotation = 45, ha = 'right')

# Annotate the bars with sales values
for container in bar_plot.containers:
    bar_plot.bar_label(container, fmt='%.0f', padding=3)

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



In [43]:
```
# Create the bar plot
plt.figure(figsize=(12, 5))
bar_plot = sns.barplot(x='PROD_NAME', y='PROD_QTY', data=top_products, palette='vir

# Add labels and title
plt.xlabel('Total Sales ($)')
plt.ylabel('Product Name')
plt.title('Top 10 Products by Total Quantity Sold')
plt.xticks(rotation = 45, ha = 'right')

# Annotate the bars with sales values
for container in bar_plot.containers:
    bar_plot.bar_label(container, fmt='%.0f', padding=3)

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



Based on the observations, here are the insights derived from the data:

Top-Selling Product:

- Dorito Corn Chip Supreme 380g has the highest quantity sold and the highest total sales, indicating its popularity and high demand.

High Quantity Brands:

- Kettle and Cobs are brands with the next highest quantity sold, suggesting that these brands are also popular among customers.

Popular Pack Sizes in quanity:

- The 175g, 110g, 150g, and 165g pack sizes are the most popular in terms of top products by quantity sold.
- this is quite different from the popular pack sizes by product sales
- lets analyze pack sizes on its own

Sales vs Quantity:

- The top two highest-selling products by quantity, which are Dorito Corn Chip Supreme 380g and Kettle Mozzarella Basil & Pesto 175g, are also in the top 10 products by total sales.

However, the remaining products in the top 10 by quantity sold do not appear in the top 10 by total sales. This indicates that while these products are frequently purchased, their individual sale values might be lower compared to other products.

In [ ]:

In [ ]:

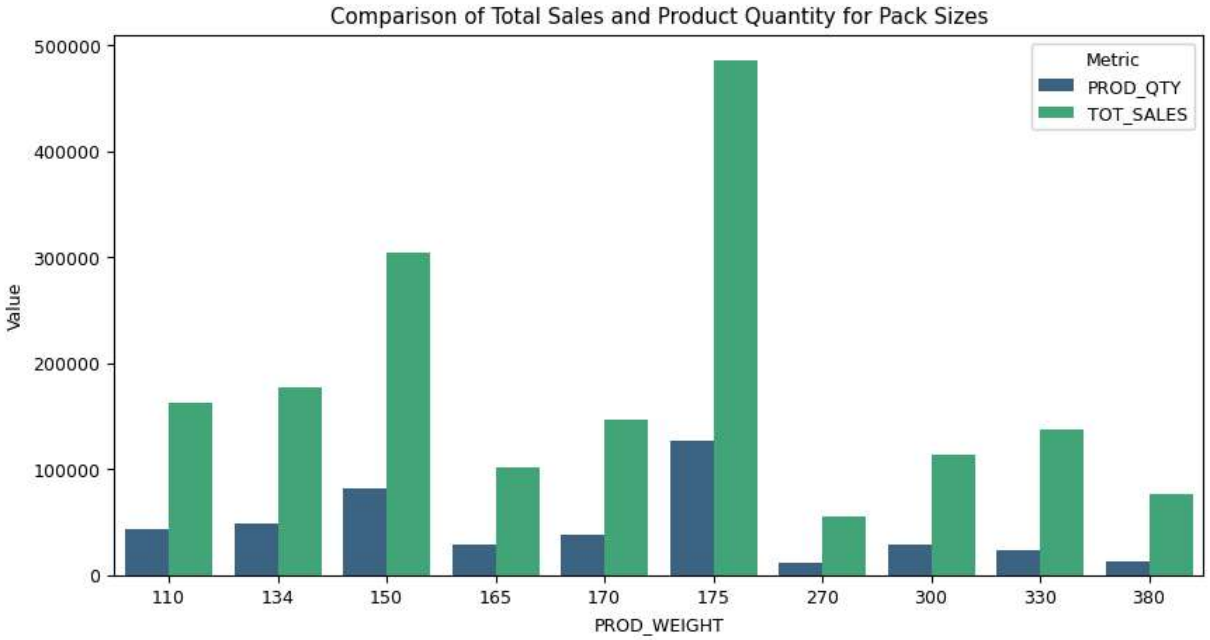
PACK SIZE ANALYSIS

In [44]:
```
# Define metrics of interest
pck_size_sales = df.groupby(['PROD_WEIGHT']).agg(
    {'TOT_SALES': 'sum', 'PROD_QTY': 'sum', 'LYLTY_CARD_NBR': 'nunique'}).reset_ind

pck_sizes = pck_size_sales.sort_values(by='TOT_SALES', ascending=False).head(10)

pck_sizes
```

Out[44]:

	PROD_WEIGHT	TOT_SALES	PROD_QTY	LYLTY_CARD_NBR
10	175	485431.4	126465	40736
6	150	304288.5	82174	30676
4	134	177655.5	48019	20657
2	110	162765.4	42835	18678
9	170	146673.0	38088	16916
19	330	136794.3	23999	11282
18	300	113330.6	28813	13310
8	165	101360.6	29051	13229
20	380	76719.6	12673	6096
17	270	55425.4	12049	5924

In [45]:
```
# Melt the DataFrame to long format for easier plotting with seaborn
melted_pck_sizes = pck_sizes.melt(id_vars='PROD_WEIGHT', value_vars=['PROD_QTY', 'T
                                  var_name='Metric', value_name='Value')

# Create the grouped bar plot
plt.figure(figsize=(10, 5))
bar_plot = sns.barplot(x='PROD_WEIGHT', y='Value', hue='Metric', data=melted_pck_si

# Add labels and title
plt.ylabel('Value')
plt.xlabel('PROD_WEIGHT')
plt.title('Comparison of Total Sales and Product Quantity for Pack Sizes')



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



The product with the highest total sales, highest quantity and highest number of unique custiomers is the one with a weight of 175g

This suggests that this pack size may be more appealing to a wider range of customers who are purchasing smaller quantities more frequently.

Products with higher weights do not necessarily translate to higher efficiency in terms of sales per customer or quantity per customer. Mid-weight products (e.g., 175 units) show better efficiency.

In [ ]:

sales over time

In [46]:
```
# Add columns for year and month
df['YEAR'] = df['DATE'].dt.year
df['MONTH'] = df['DATE'].dt.month
```

In [47]:
```
# Group by year, month, and brand, then sum the total sales
df.groupby(['YEAR'])['TOT_SALES'].sum().reset_index().sort_values(by='TOT_SALES', a
```

Out[47]:

	YEAR	TOT_SALES
0	2018	977085.15
1	2019	957323.85

In [51]:
```
# Group by year, month, and brand, then sum the total sales
monthly_sales = df.groupby(['YEAR', 'MONTH'])['TOT_SALES'].sum().reset_index().sort
monthly_sales
```

Out[51]:

	YEAR	MONTH	TOT_SALES
5	2018	12	167913.40
8	2019	3	166265.20
0	2018	7	165275.30
3	2018	10	164409.70
6	2019	1	162642.30
11	2019	6	160538.60
2	2018	9	160522.00
4	2018	11	160233.70
9	2019	4	159845.10
1	2018	8	158731.05
10	2019	5	157367.65
7	2019	2	150665.00

In [52]:
```
# Group by year, month, and brand, then sum the total sales
monthly_year_sales = df.groupby(['YEAR', 'BRAND_NAME']).agg(
    {'TOT_SALES': 'sum'}).reset_index().sort_values(by='TOT_SALES', ascending=False
monthly_year_sales.head(10)
```

Out[52]:

	YEAR	BRAND_NAME	TOT_SALES
10	2018	Kettle	197532.4
33	2019	Kettle	192707.4
15	2018	Smiths	112461.9
38	2019	Smiths	112192.3
6	2018	Doritos	102479.1
29	2019	Doritos	99059.8
13	2018	Pringles	90453.9
36	2019	Pringles	87201.6
9	2018	Infuzions	49923.4
32	2019	Infuzions	49124.2

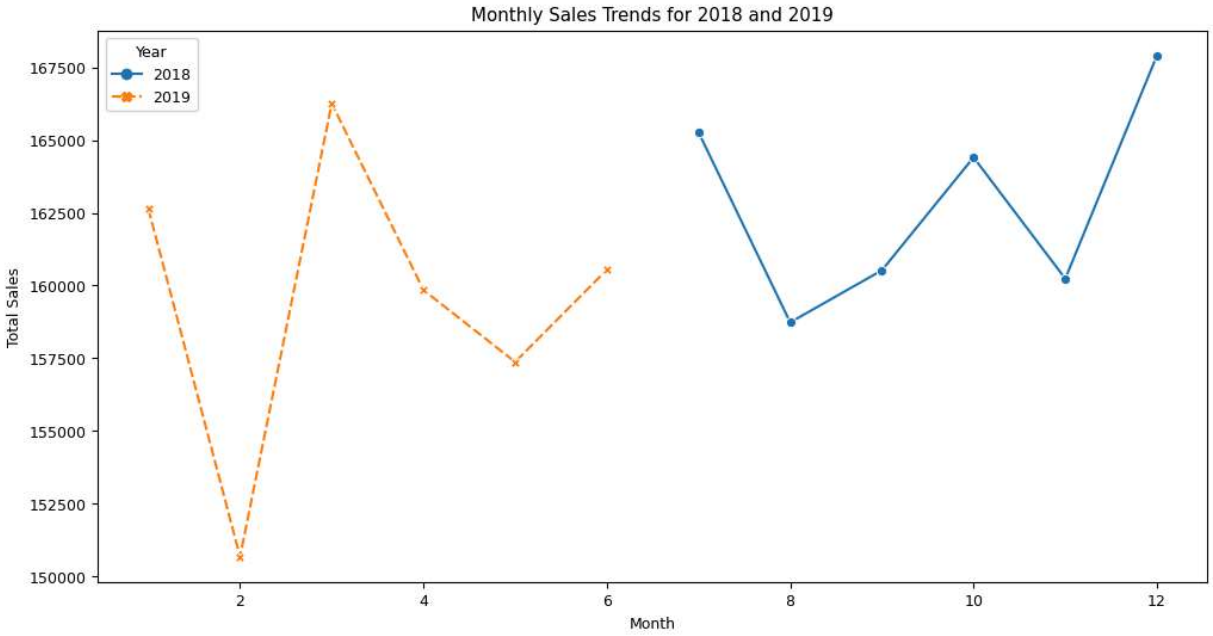
In [53]:
```
monthly_sales_pivot = monthly_sales.pivot(index='MONTH', columns='YEAR', values='TO
monthly_sales_pivot
```

Out[53]:

YEAR	2018	2019
MONTH		
1	NaN	162642.30
2	NaN	150665.00
3	NaN	166265.20
4	NaN	159845.10
5	NaN	157367.65
6	NaN	160538.60
7	165275.30	NaN
8	158731.05	NaN
9	160522.00	NaN
10	164409.70	NaN
11	160233.70	NaN
12	167913.40	NaN

In [54]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_sales_pivot, markers=True)
plt.title('Monthly Sales Trends for 2018 and 2019')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.legend(title='Year')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



The analysis indicates that **2018** experienced higher sales figures compared to **2019**. Notably, Kettle maintained its position as the top-selling brand in both years, showcasing its consistent popularity among consumers.

Moreover, the top 5 best-selling brands remained consistent across both years, suggesting a stable market landscape and the enduring appeal of these brands to consumers.

In [ ]:

TOP STORES BY SALES

In [55]:
```
store_sales = df.groupby(['STORE_NBR']).agg(
    {'PROD_QTY': 'sum', 'TOT_SALES': 'sum'}).reset_index().sort_values(by = 'TOT_SA
store_sales
```

Out[55]:

	STORE_NBR	PROD_QTY	TOT_SALES
225	226	4401	18905.45
87	88	3718	16333.25
164	165	3602	15973.75
39	40	3499	15559.50
236	237	3515	15539.50
57	58	3463	15251.45
198	199	3343	14797.00
3	4	3316	14647.65
202	203	3287	14551.60
25	26	3256	14469.30

- The store with store number 226 also has the highest total sales, amounting to $18905.45.
- The top 10 stores by total sales range from $14469.30 to 18905.45.

TOP STORES BY QUANTITY

In [56]:
```
store_qty = df.groupby('STORE_NBR')['PROD_QTY'].sum().reset_index().sort_values(by
store_qty
```

Out[56]:

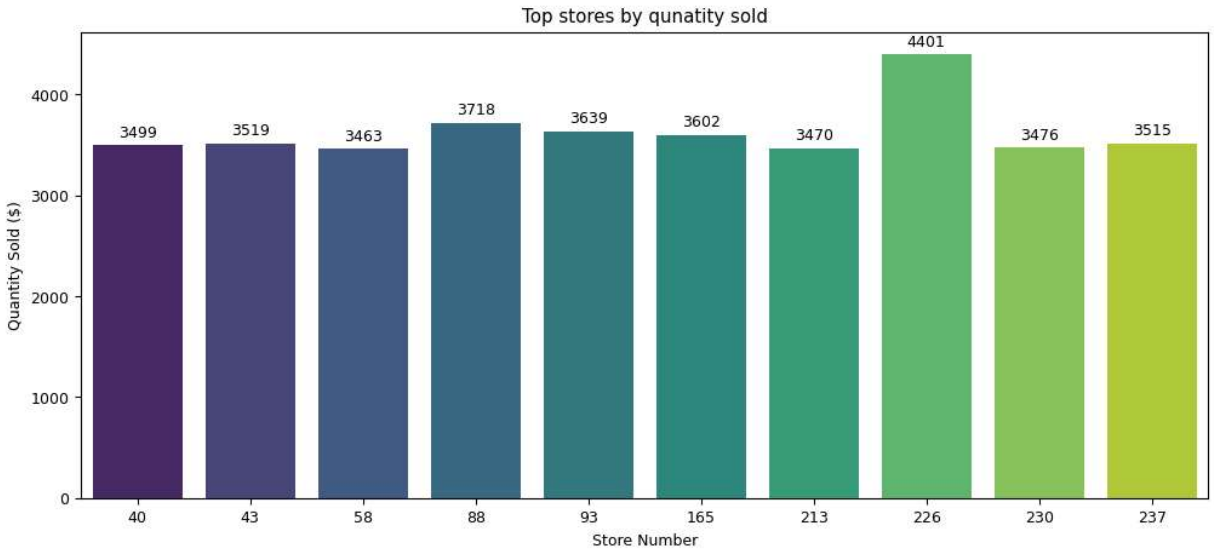
	STORE_NBR	PROD_QTY
225	226	4401
87	88	3718
92	93	3639
164	165	3602
42	43	3519
236	237	3515
39	40	3499
229	230	3476
212	213	3470
57	58	3463

- The store with store number 226 has the highest quantity sold, with 4401 units.

Store number 226 stands out as the top performer in both quantity sold and total sales, indicating strong performance across both metrics.

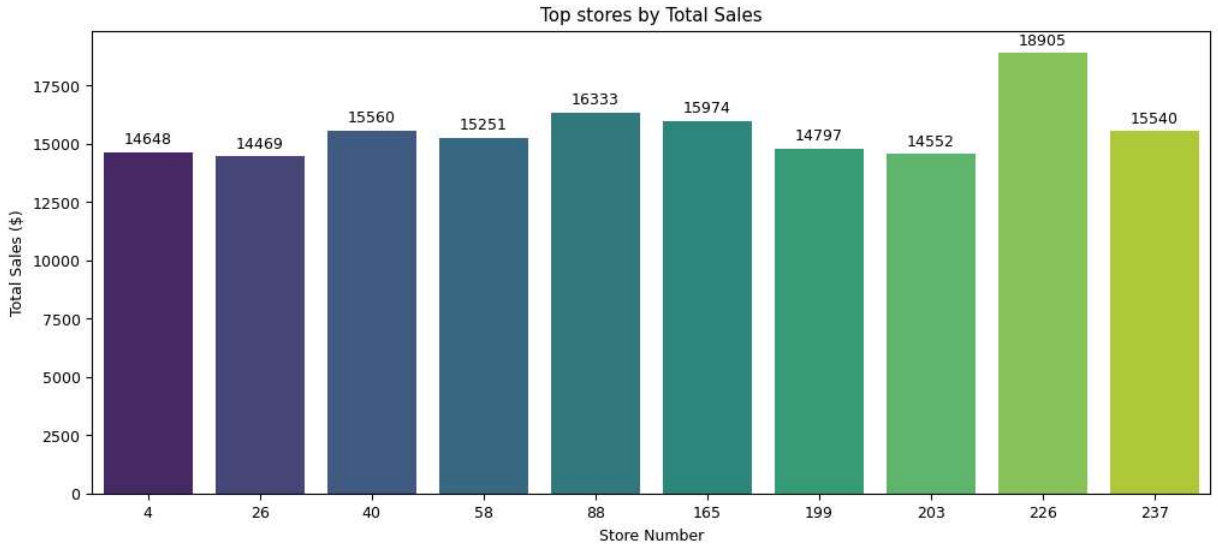
Overall, there seems to be a correlation between quantity sold and total sales, as some stores with higher quantities sold also tend to have higher total sales.

In [57]:
```
# Create the bar plot
plt.figure(figsize=(12, 5))
bar_plot = sns.barplot(x='STORE_NBR', y='PROD_QTY', data=store_qty, palette='viridi

# Add labels and title
plt.ylabel('Quantity Sold ($)')
plt.xlabel('Store Number')
plt.title('Top stores by qunatity sold')


# Annotate the bars with sales values
for container in bar_plot.containers:
    bar_plot.bar_label(container, fmt='%.0f', padding=3)

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



In [58]:
```
# Create the bar plot
plt.figure(figsize=(12, 5))
bar_plot = sns.barplot(x='STORE_NBR', y='TOT_SALES', data=store_sales, palette='vir

# Add labels and title
plt.ylabel('Total Sales ($)')
plt.xlabel('Store Number')
plt.title('Top stores by Total Sales')


# Annotate the bars with sales values
for container in bar_plot.containers:
    bar_plot.bar_label(container, fmt='%.0f', padding=3)
```

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



In [ ]:

Key Insights Recap Customer Segments:

- Older Singles/Couples have the highest number of transactions.
- Young Singles/Couples in the "Mainstream" category have the highest number of unique customers.
- Older Families in the "Budget" category have the highest total sales.
- New Families in the "Premium" category have the lowest total sales and unique customers.

Brand Performance:

- Kettle is the leading brand in terms of both transactions and sales.
- Doritos Corn Chp Supreme 380g has the highest sales and quantity among all products.

Sales Trends:

- 2018 saw higher sales compared to 2019.
- Consistency in top-performing brands across both years.

Product Preferences:

- The 175g pack size is particularly popular, indicating a preference for mid-sized products.
- Higher weights do not necessarily lead to better sales performance, suggesting mid-weight products (e.g., 175g) are more efficient in terms of sales per customer.

Recommendations Inventory Management:

- Increase Stock for Top Brands: Focus on increasing the inventory of Kettle, Smiths, and Doritos to meet demand.
- Optimize Pack Sizes: Given the popularity of 175g packs, ensure these are adequately stocked. Consider promoting other mid-weight products.

Targeted Marketing:

Older Families: Develop marketing campaigns targeting "Older Families" in the "Budget" category, as they are the highest spenders.

Young Singles/Couples: Create promotions for "Young Singles/Couples" in the "Mainstream" category to leverage their large customer base and spending power.

Product Development:

Focus on Mid-Weight Products: Since 175g products are highly efficient, consider developing and marketing more products in this weight category.

Brand Diversification: While Kettle and other top brands perform well, explore ways to boost the sales of underperforming brands through promotions or product innovation.

Sales Strategy:

Year-End Promotions: Since sales were higher in 2018 than in 2019, consider year-end promotions or discounts to boost sales in 2019.

Customer Loyalty Programs: Implement loyalty programs targeting high-value segments like "Older Families" and "Young Singles/Couples" to encourage repeat purchases.

Market Analysis:

Monitor Outliers: Keep an eye on products with unusually high quantities and sales to understand and replicate their success.

Analyze Low Performers: Investigate why "New Families" in the "Premium" category have low sales and unique customers. Tailor strategies to better engage this segment.

In [ ]: