In [ ]:
```
"""
Task:
Analyze transaction and customer data to identify trends and inconsistenci
Develop metrics and examine sales drivers to gain insights into overall sa
Create visualizations and prepare findings to formulate a clear recommenda
"""
```

In [2]:
```
#importing the required libraries for this task

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# For statistical testing
from scipy import stats
```

In [6]:
```
import chardet
with open(tdatafilepath, 'rb') as rawdata:
    result = chardet.detect(rawdata.read(100000))
result
```

Out[6]:
```
{'encoding': None, 'confidence': 0.0, 'language': None}
```

In [7]:
```
customer_data = pd.read_csv("QVI_purchase_behaviour.csv")
```

In [9]:
```
transaction_data = pd.read_excel("QVI_transaction_data.xlsx", engine='oper
```

In [11]:
```
print(transaction_data.info())
print(transaction_data.describe())
print(transaction_data.head())
```

```
<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
None
                DATE     STORE_NBR  LYLTY_CARD_NBR        TXN_ID  \
count  264836.000000  264836.00000    2.648360e+05  2.648360e+05   
mean    43464.036260     135.08011    1.355495e+05  1.351583e+05   
std       105.389282      76.78418    8.057998e+04  7.813303e+04   
min     43282.000000       1.00000    1.000000e+03  1.000000e+00   
25%     43373.000000      70.00000    7.002100e+04  6.760150e+04   
50%     43464.000000     130.00000    1.303575e+05  1.351375e+05   
75%     43555.000000     203.00000    2.030942e+05  2.027012e+05   
max     43646.000000     272.00000    2.373711e+06  2.415841e+06   

            PROD_NBR       PROD_QTY      TOT_SALES  
count  264836.000000  264836.000000  264836.000000  
mean       56.583157       1.907309       7.304200  
std        32.826638       0.643654       3.083226  
min         1.000000       1.000000       1.500000  
25%        28.000000       2.000000       5.400000  
50%        56.000000       2.000000       7.400000  
75%        85.000000       2.000000       9.200000  
max       114.000000     200.000000     650.000000  
    DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0  43390          1            1000       1         5   
1  43599          1            1307     348        66   
2  43605          1            1343     383        61   
3  43329          2            2373     974        69   
4  43330          2            2426    1038       108   

                                PROD_NAME  PROD_QTY  TOT_SALES  
0    Natural Chip        Compny SeaSalt175g         2        6.0  
1                  CCs Nacho Cheese    175g         3        6.3  
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9  
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0  
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8  
```

In [12]:
```
# Convert DATE column to datetime
transaction_data['DATE'] = pd.to_datetime(transaction_data['DATE'], origin
```

In [13]:
```
transaction_data.head()
```

Out[13]:

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

In [94]:
```
#get the details of the product name column
product_summary = transaction_data['PROD_NAME'].describe()
print(product_summary)
```

```
count                                 246742
unique                                   105
top       Kettle Mozzarella   Basil & Pesto 175g
freq                                    3304
Name: PROD_NAME, dtype: object
```

In [32]:
```
product_name_counts = transaction_data['PROD_NAME'].value_counts()
print(product_name_counts)
```

```
PROD_NAME
Kettle Mozzarella   Basil & Pesto 175g      3304
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3296
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3269
Tyrrells Crisps     Ched & Chives 165g      3268
Cobs Popd Sea Salt  Chips 110g              3265
                                            ...
RRD Pc Sea Salt     165g                    1431
Woolworths Medium   Salsa 300g              1430
NCC Sour Cream &    Garden Chives 175g      1419
French Fries Potato Chips 175g              1418
WW Crinkle Cut      Original 175g           1410
Name: count, Length: 114, dtype: int64
```

In [34]:

```
#Get the unique items in the product name column
transaction_data['PROD_NAME'].unique()
```

Out[34]:
```
array(['Natural Chip        Compny SeaSalt175g',
       'CCs Nacho Cheese    175g',
       'Smiths Crinkle Cut  Chips Chicken 170g',
       'Smiths Chip Thinly  S/Cream&Onion 175g',
       'Kettle Tortilla ChpsHny&Jlpno Chili 150g',
       'Old El Paso Salsa   Dip Tomato Mild 300g',
       'Smiths Crinkle Chips Salt & Vinegar 330g',
       'Grain Waves         Sweet Chilli 210g',
       'Doritos Corn Chip Mexican Jalapeno 150g',
       'Grain Waves Sour    Cream&Chives 210G',
       'Kettle Sensations   Siracha Lime 150g',
       'Twisties Cheese     270g', 'WW Crinkle Cut      Chicken 175g',
       'Thins Chips Light&  Tangy 175g', 'CCs Original 175g',
       'Burger Rings 220g', 'NCC Sour Cream &    Garden Chives 175g',
       'Doritos Corn Chip Southern Chicken 150g',
       'Cheezels Cheese Box 125g', 'Smiths Crinkle      Original 330g',
       'Infzns Crn Crnchers Tangy Gcamole 110g',
       'Kettle Sea Salt     And Vinegar 175g',
       'Smiths Chip Thinly  Cut Original 175g', 'Kettle Original 175g',
       'Red Rock Deli Thai  Chilli&Lime 150g',
       'Pringles Sthrn FriedChicken 134g', 'Pringles Sweet&Spcy BBQ 134
g',
       'Red Rock Deli SR    Salsa & Mzzrlla 150g',
       'Thins Chips         Originl saltd 175g',
       'Red Rock Deli Sp    Salt & Truffle 150G',
       'Smiths Thinly       Swt Chli&S/Cream175G', 'Kettle Chilli 175g',
       'Doritos Mexicana    170g',
       'Smiths Crinkle Cut  French OnionDip 150g',
       'Natural ChipCo      Hony Soy Chckn175g',
       'Dorito Corn Chp     Supreme 380g', 'Twisties Chicken270g',
       'Smiths Thinly Cut   Roast Chicken 175g',
       'Smiths Crinkle Cut  Tomato Salsa 150g',
       'Kettle Mozzarella   Basil & Pesto 175g',
       'Infuzions Thai SweetChili PotatoMix 110g',
       'Kettle Sensations   Camembert & Fig 150g',
       'Smith Crinkle Cut   Mac N Cheese 150g',
       'Kettle Honey Soy    Chicken 175g',
       'Thins Chips Seasonedchicken 175g',
       'Smiths Crinkle Cut  Salt & Vinegar 170g',
       'Infuzions BBQ Rib   Prawn Crackers 110g',
       'GrnWves Plus Btroot & Chilli Jam 180g',
       'Tyrrells Crisps     Lightly Salted 165g',
       'Kettle Sweet Chilli And Sour Cream 175g',
       'Doritos Salsa       Medium 300g', 'Kettle 135g Swt Pot Sea Sal
t',
       'Pringles SourCream  Onion 134g',
       'Doritos Corn Chips  Original 170g',
       'Twisties Cheese     Burger 250g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
       'Woolworths Mild     Salsa 300g',
       'Natural Chip Co     Tmato Hrb&Spce 175g',
       'Smiths Crinkle Cut  Chips Original 170g',
       'Cobs Popd Sea Salt  Chips 110g',
       'Smiths Crinkle Cut  Chips Chs&Onion170g',
       'French Fries Potato Chips 175g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
```

```
       'Doritos Corn Chips  Cheese Supreme 170g',
       'Pringles Original   Crisps 134g',
       'RRD Chilli&         Coconut 150g',
       'WW Original Corn    Chips 200g',
       'Thins Potato Chips  Hot & Spicy 175g',
       'Cobs Popd Sour Crm  &Chives Chips 110g',
       'Smiths Crnkle Chip  Orgnl Big Bag 380g',
       'Doritos Corn Chips  Nacho Cheese 170g',
       'Kettle Sensations   BBQ&Maple 150g',
       'WW D/Style Chip     Sea Salt 200g',
       'Pringles Chicken    Salt Crips 134g',
       'WW Original Stacked Chips 160g',
       'Smiths Chip Thinly  CutSalt/Vinegr175g', 'Cheezels Cheese 330g',
       'Tostitos Lightly    Salted 175g',
       'Thins Chips Salt &  Vinegar 175g',
       'Smiths Crinkle Cut  Chips Barbecue 170g', 'Cheetos Puffs 165g',
       'RRD Sweet Chilli &  Sour Cream 165g',
       'WW Crinkle Cut      Original 175g',
       'Tostitos Splash Of  Lime 175g', 'Woolworths Medium   Salsa 300
g',
       'Kettle Tortilla ChpsBtroot&Ricotta 150g',
       'CCs Tasty Cheese    175g', 'Woolworths Cheese   Rings 190g',
       'Tostitos Smoked     Chipotle 175g', 'Pringles Barbeque   134g',
       'WW Supreme Cheese   Corn Chips 200g',
       'Pringles Mystery    Flavour 134g',
       'Tyrrells Crisps     Ched & Chives 165g',
       'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
       'Cheetos Chs & Bacon Balls 190g', 'Pringles Slt Vingar 134g',
       'Infuzions SourCream&Herbs Veg Strws 110g',
       'Kettle Tortilla ChpsFeta&Garlic 150g',
       'Infuzions Mango     Chutny Papadums 70g',
       'RRD Steak &         Chimuchurri 150g',
       'RRD Honey Soy       Chicken 165g',
       'Sunbites Whlegrn    Crisps Frch/Onin 90g',
       'RRD Salt & Vinegar  165g', 'Doritos Cheese      Supreme 330g',
       'Smiths Crinkle Cut  Snag&Sauce 150g',
       'WW Sour Cream &OnionStacked Chips 160g',
       'RRD Lime & Pepper   165g',
       'Natural ChipCo Sea  Salt & Vinegr 175g',
       'Red Rock Deli Chikn&Garlic Aioli 150g',
       'RRD SR Slow Rst     Pork Belly 150g', 'RRD Pc Sea Salt     165
g',
       'Smith Crinkle Cut   Bolognese 150g', 'Doritos Salsa Mild  300
g'],
      dtype=object)
```

In [53]:

```
#Clean product names
import re
from collections import Counter


product_names = transaction_data['PROD_NAME'].unique()

# Function to clean product names
def clean_product_name(name):
    # Remove digits
    name = re.sub(r'\d+', '', name)
    # Remove special characters
    name = re.sub(r'[^a-zA-Z\s]', '', name)
    # Remove any remaining single characters
    name = re.sub(r'\b\w\b', '', name)
    # Remove extra spaces
    name = re.sub(r'\s+', ' ', name).strip()
    return name

# Clean product names
cleaned_product_names = [clean_product_name(name) for name in product_name

# Split the names into words and count frequency
words = []
for name in cleaned_product_names:
    words.extend(name.split())

# Count frequency of each word
word_counts = Counter(words)

# Convert to DataFrame for easier analysis
word_freq_df = pd.DataFrame(word_counts.items(), columns=['Word', 'Frequer

# Sort by frequency
word_freq_df = word_freq_df.sort_values(by='Frequency', ascending=False)

print(word_freq_df.head(20))  # Display the top 20 most frequent words
```

	Word	Frequency
10	Chips	21
7	Smiths	15
9	Cut	13
14	Kettle	13
8	Crinkle	13
6	Cheese	12
18	Salt	12
38	Original	10
1	Chip	9
118	RRD	8
25	Corn	8
59	Pringles	8
11	Chicken	7
34	WW	7
24	Doritos	7
23	Chilli	6
52	Sea	6
28	Sour	6
19	Vinegar	5
102	Crisps	5

In [47]:
```
#Get a pack size column from product name
transaction_data['PACK_SIZE'] = transaction_data['PROD_NAME'].str.extract(
    .fillna(0).astype(int)  # Replace missing values with -1 (or any defau
```

In [48]:
```
# Apply the cleaning function to the PROD_NAME column
transaction_data['CLEANED_PROD_NAME'] = transaction_data['PROD_NAME'].appl
```

In [49]:
```
transaction_data.head()
```

Out[49]:

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

In [50]:
```
# Identify salsa products which should not be part of the analysis of Chip
transaction_data['SALSA'] = transaction_data['PROD_NAME'].str.contains('sa

# Remove salsa products
transaction_data = transaction_data[transaction_data['SALSA'] == False]

# Drop the 'SALSA' column
transaction_data = transaction_data.drop(columns=['SALSA'])
```

In [54]:
```
word_freq_df.head(20)
```

Out[54]:

	Word	Frequency
10	Chips	21
7	Smiths	15
9	Cut	13
14	Kettle	13
8	Crinkle	13
6	Cheese	12
18	Salt	12
38	Original	10
1	Chip	9
118	RRD	8
25	Corn	8
59	Pringles	8
11	Chicken	7
34	WW	7
24	Doritos	7
23	Chilli	6
52	Sea	6
28	Sour	6
19	Vinegar	5
102	Crisps	5

In [55]:

```
# Get summary statistics
summary_stats = transaction_data.describe(include='all')

# Check for null values
null_counts = transaction_data.isnull().sum()

print("\nSummary Statistics:")
print(summary_stats)

print("\nNull Counts:")
print(null_counts)
```

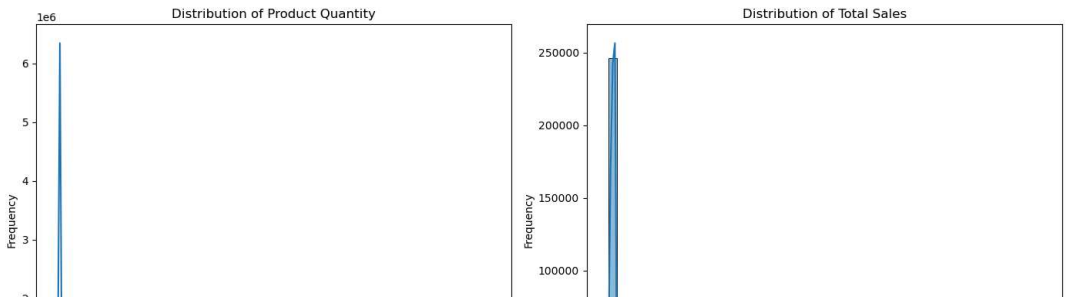
```
Summary Statistics:
                                DATE      STORE_NBR  LYLTY_CARD_NBR  \
count                         246742  246742.000000    2.467420e+05
unique                           NaN            NaN             NaN
top                              NaN            NaN             NaN
freq                             NaN            NaN             NaN
mean    2018-12-30 01:19:01.211467520     135.051098    1.355310e+05
min               2018-07-01 00:00:00       1.000000    1.000000e+03
25%               2018-09-30 00:00:00      70.000000    7.001500e+04
50%               2018-12-30 00:00:00     130.000000    1.303670e+05
75%               2019-03-31 00:00:00     203.000000    2.030840e+05
max               2019-06-30 00:00:00     272.000000    2.373711e+06
std                              NaN      76.787096    8.071528e+04

              TXN_ID       PROD_NBR                                PROD_N
AME  \
count   2.467420e+05  246742.000000                                   246
742
unique           NaN            NaN
105
top              NaN            NaN  Kettle Mozzarella   Basil & Pesto 1
75g
freq             NaN            NaN                                     3
304
mean    1.351311e+05      56.351789
NaN
min     1.000000e+00       1.000000
NaN
25%     6.756925e+04      26.000000
NaN
50%     1.351830e+05      53.000000
NaN
75%     2.026538e+05      87.000000
NaN
max     2.415841e+06     114.000000
NaN
std     7.814772e+04      33.695428
NaN

             PROD_QTY      TOT_SALES      PACK_SIZE  \
count   246742.000000  246742.000000  246742.000000
unique            NaN            NaN            NaN
top               NaN            NaN            NaN
freq              NaN            NaN            NaN
mean         1.908062       7.321322     175.585178
min          1.000000       1.700000      70.000000
25%          2.000000       5.800000     150.000000
50%          2.000000       7.400000     170.000000
75%          2.000000       8.800000     175.000000
max        200.000000     650.000000     380.000000
std          0.659831       3.077828      59.434727

                  CLEANED_PROD_NAME
count                        246742
unique                          105
top     Kettle Mozzarella Basil Pesto
```

```
freq                          3304
mean                           NaN
min                            NaN
25%                            NaN
50%                            NaN
75%                            NaN
max                            NaN
std                            NaN

Null Counts:
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PACK_SIZE           0
CLEANED_PROD_NAME   0
dtype: int64
```

In [56]:

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

# Plot histograms for PROD_QTY and TOT_SALES
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Histogram for PROD_QTY
sns.histplot(transaction_data['PROD_QTY'], bins=50, kde=True, ax=ax[0])
ax[0].set_title('Distribution of Product Quantity')
ax[0].set_xlabel('Product Quantity')
ax[0].set_ylabel('Frequency')

# Histogram for TOT_SALES
sns.histplot(transaction_data['TOT_SALES'], bins=50, kde=True, ax=ax[1])
ax[1].set_title('Distribution of Total Sales')
ax[1].set_xlabel('Total Sales ($)')
ax[1].set_ylabel('Frequency')

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

```
C:\Users\Tonye\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 inste
ad.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Tonye\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 inste
ad.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [57]:
```
#Investigating for outliers
# Investigate transactions with high product quantity
high_qty = transaction_data[transaction_data['PROD_QTY'] > 50]
print("High Quantity Transactions:")
print(high_qty)

# Investigate transactions with high total sales
high_sales = transaction_data[transaction_data['TOT_SALES'] > 100]
print("High Sales Transactions:")
print(high_sales)
```

```
High Quantity Transactions:
            DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
69762 2018-08-19        226          226000  226201         4
69763 2019-05-20        226          226000  226210         4

                              PROD_NAME  PROD_QTY  TOT_SALES  PACK_SIZE
\
69762  Dorito Corn Chp     Supreme 380g       200      650.0        380
69763  Dorito Corn Chp     Supreme 380g       200      650.0        380

             CLEANED_PROD_NAME
69762  Dorito Corn Chp Supreme
69763  Dorito Corn Chp Supreme
High Sales Transactions:
            DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
69762 2018-08-19        226          226000  226201         4
69763 2019-05-20        226          226000  226210         4

                              PROD_NAME  PROD_QTY  TOT_SALES  PACK_SIZE
\
69762  Dorito Corn Chp     Supreme 380g       200      650.0        380
69763  Dorito Corn Chp     Supreme 380g       200      650.0        380

             CLEANED_PROD_NAME
69762  Dorito Corn Chp Supreme
69763  Dorito Corn Chp Supreme
```

In [58]:
```
# Remove transactions with PROD_QTY > 50 or TOT_SALES > 100
transaction_data_cleaned = transaction_data[(transaction_data['PROD_QTY']

# Check the shape of the cleaned data
print("Shape of cleaned data:", transaction_data_cleaned.shape)
```

```
Shape of cleaned data: (246740, 10)
```

In [60]:

```
# Identify the loyalty card number of the customer with high quantity trar
loyalty_card_number_to_remove = transaction_data[transaction_data['PROD_Q1

# Filter out this customer's transactions
transaction_data_filtered = transaction_data[~transaction_data['LYLTY_CARD

# Check the shape of the filtered data
print("Shape of filtered data:", transaction_data_filtered.shape)
```

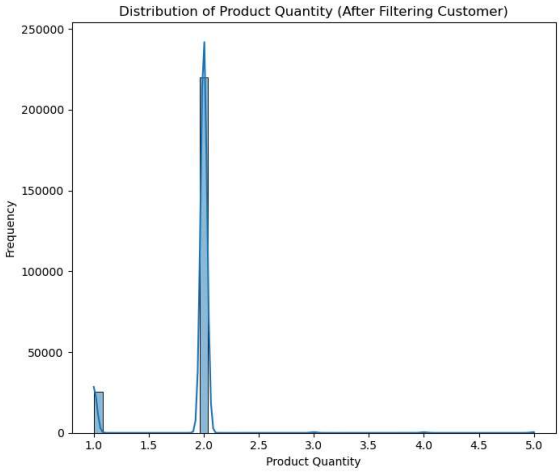
```
Shape of filtered data: (246740, 10)
```

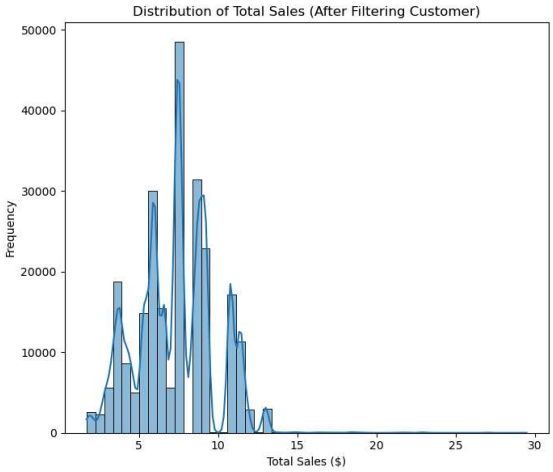
In [61]:

```
# Re-check summary statistics
summary_stats_filtered = transaction_data_filtered.describe(include='all')
print("Updated Summary Statistics (After Filtering Customer):")
print(summary_stats_filtered)

# Re-check null counts
null_counts_filtered = transaction_data_filtered.isnull().sum()
print("Updated Null Counts (After Filtering Customer):")
print(null_counts_filtered)

# Visualize data distribution again
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Histogram for PROD_QTY
sns.histplot(transaction_data_filtered['PROD_QTY'], bins=50, kde=True, ax=
ax[0].set_title('Distribution of Product Quantity (After Filtering Custome
ax[0].set_xlabel('Product Quantity')
ax[0].set_ylabel('Frequency')

# Histogram for TOT_SALES
sns.histplot(transaction_data_filtered['TOT_SALES'], bins=50, kde=True, ax
ax[1].set_title('Distribution of Total Sales (After Filtering Customer)')
ax[1].set_xlabel('Total Sales ($)')
ax[1].set_ylabel('Frequency')

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

```
Updated Summary Statistics (After Filtering Customer):
                                DATE      STORE_NBR  LYLTY_CARD_NBR  \
count                         246740  246740.000000    2.467400e+05
unique                           NaN            NaN             NaN
top                              NaN            NaN             NaN
freq                             NaN            NaN             NaN
mean    2018-12-30 01:18:58.448569344     135.050361    1.355303e+05
min               2018-07-01 00:00:00       1.000000    1.000000e+03
25%               2018-09-30 00:00:00      70.000000    7.001500e+04
50%               2018-12-30 00:00:00     130.000000    1.303670e+05
75%               2019-03-31 00:00:00     203.000000    2.030832e+05
max               2019-06-30 00:00:00     272.000000    2.373711e+06
std                              NaN      76.786971    8.071520e+04

              TXN_ID       PROD_NBR                                PROD_N
AME  \
count   2.467400e+05  246740.000000                                   246
740
unique           NaN            NaN
105
top              NaN            NaN  Kettle Mozzarella   Basil & Pesto 1
75g
freq             NaN            NaN                                     3
304
mean    1.351304e+05      56.352213
NaN
min     1.000000e+00       1.000000
NaN
25%     6.756875e+04      26.000000
NaN
50%     1.351815e+05      53.000000
NaN
75%     2.026522e+05      87.000000
NaN
max     2.415841e+06     114.000000
NaN
std     7.814760e+04      33.695235
NaN

             PROD_QTY      TOT_SALES      PACK_SIZE  \
count   246740.000000  246740.000000  246740.000000
unique            NaN            NaN            NaN
top               NaN            NaN            NaN
freq              NaN            NaN            NaN
mean         1.906456       7.316113     175.583521
min          1.000000       1.700000      70.000000
25%          2.000000       5.800000     150.000000
50%          2.000000       7.400000     170.000000
75%          2.000000       8.800000     175.000000
max          5.000000      29.500000     380.000000
std          0.342499       2.474897      59.432118

                   CLEANED_PROD_NAME
count                         246740
unique                           105
top     Kettle Mozzarella Basil Pesto
freq                            3304
```

```
mean                                NaN
min                                 NaN
25%                                 NaN
50%                                 NaN
75%                                 NaN
max                                 NaN
std                                 NaN
Updated Null Counts (After Filtering Customer):
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
PACK_SIZE           0
CLEANED_PROD_NAME   0
dtype: int64

C:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: 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:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: 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):
```

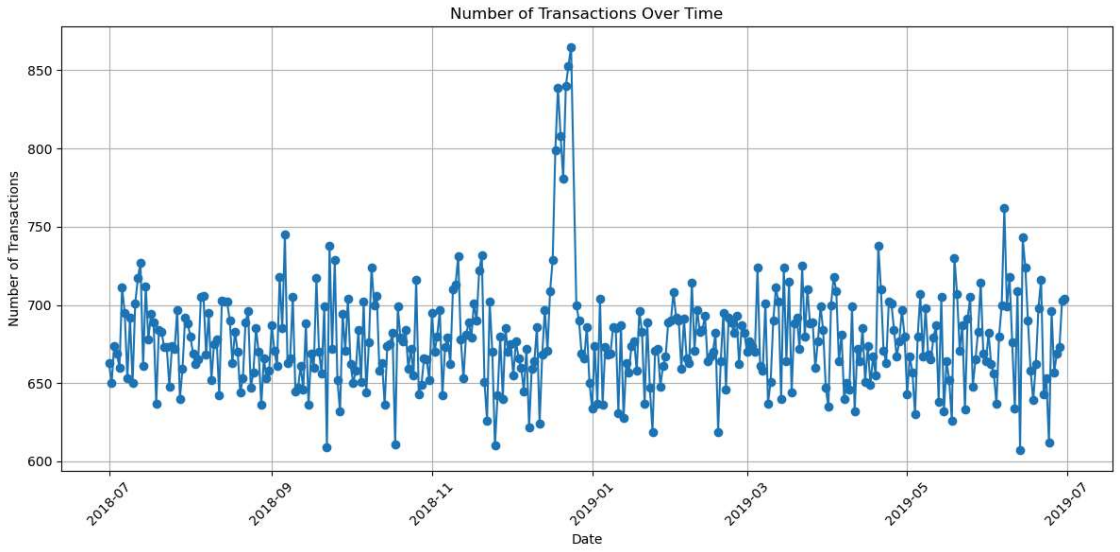




In [62]:
```
# Group by date and count the number of transactions
transaction_counts_by_date = transaction_data_filtered.groupby('DATE').siz

# Display the first few rows to check the result
print(transaction_counts_by_date.head())
```

```
        DATE  transaction_count
0 2018-07-01                663
1 2018-07-02                650
2 2018-07-03                674
3 2018-07-04                669
4 2018-07-05                660
```

In [63]:
```
import matplotlib.pyplot as plt

# Plot the number of transactions over time
plt.figure(figsize=(12, 6))
plt.plot(transaction_counts_by_date['DATE'], transaction_counts_by_date['t
plt.title('Number of Transactions Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [65]:
```
transaction_counts_by_date.describe()
```

Out[65]:

	DATE	transaction_count
count	364	364.000000
mean	2018-12-30 00:19:46.813186816	677.857143
min	2018-07-01 00:00:00	607.000000
25%	2018-09-29 18:00:00	658.000000
50%	2018-12-30 12:00:00	674.000000
75%	2019-03-31 06:00:00	694.250000
max	2019-06-30 00:00:00	865.000000
std	NaN	33.687536

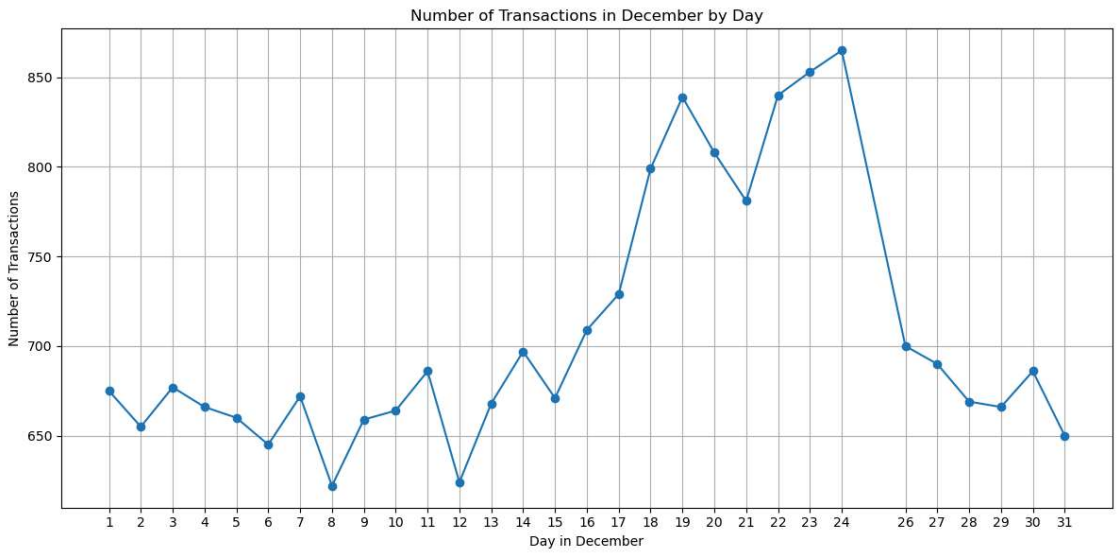
In [66]:
```
# Convert 'DATE' column to datetime
transaction_data_filtered['DATE'] = pd.to_datetime(transaction_data_filter

# Filter to include only transactions from December
december_transactions = transaction_data_filtered[transaction_data_filtere

# Display the first few rows to check the result
print(december_transactions.head())
```

```
          DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
224 2018-12-14          2            2256     866        55
232 2018-12-17          2            2454    1071        34
265 2018-12-12          4            4074    2980         4
284 2018-12-02          4            4371    4315       108
285 2018-12-24          4            4371    4316        78

                                    PROD_NAME  PROD_QTY  TOT_SALES  PACK
_SIZE  \
224      Snbts Whlgrn Crisps Cheddr&Mstrd 90g         1        1.7
90
232                  Pringles Slt Vingar 134g         1        3.7
134
265          Dorito Corn Chp     Supreme 380g         2       13.0
380
284  Kettle Tortilla ChpsHny&Jlpno Chili 150g         2        9.2
150
285          Thins Chips Salt &  Vinegar 175g         2        6.6
175

                     CLEANED_PROD_NAME
224     Snbts Whlgrn Crisps CheddrMstrd
232                Pringles Slt Vingar
265            Dorito Corn Chp Supreme
284  Kettle Tortilla ChpsHnyJlpno Chili
285            Thins Chips Salt Vinegar

C:\Users\Tonye\AppData\Local\Temp\ipykernel_21600\219299048.py:2: Settin
gWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  transaction_data_filtered['DATE'] = pd.to_datetime(transaction_data_fi
ltered['DATE'])
```

In [67]:
```
# Group by day and count the number of transactions
december_transaction_counts = december_transactions.groupby(december_trans

# Display the first few rows to check the result
print(december_transaction_counts.head())
```

```
   DATE  transaction_count
0     1                675
1     2                655
2     3                677
3     4                666
4     5                660
```

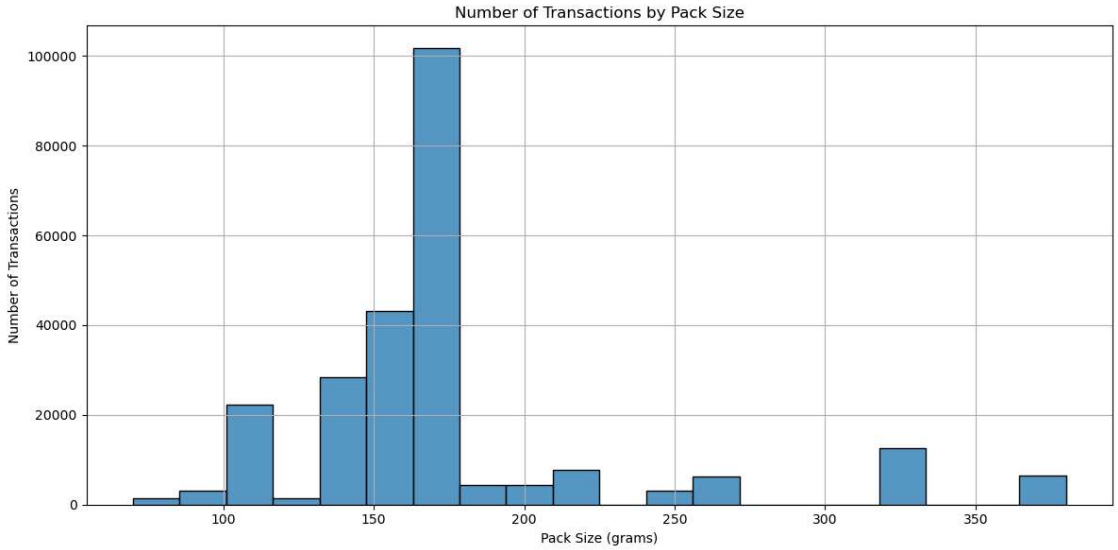
In [68]:
```
# Plot the number of transactions for each day in December
plt.figure(figsize=(12, 6))
plt.plot(december_transaction_counts['DATE'], december_transaction_counts[
plt.title('Number of Transactions in December by Day')
plt.xlabel('Day in December')
plt.ylabel('Number of Transactions')
plt.xticks(december_transaction_counts['DATE'])  # Ensuring each day is sh
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [69]:

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

# Plot a histogram of PACK_SIZE
plt.figure(figsize=(12, 6))
sns.histplot(data=transaction_data_filtered, x='PACK_SIZE', bins=20, kde=F
plt.title('Number of Transactions by Pack Size')
plt.xlabel('Pack Size (grams)')
plt.ylabel('Number of Transactions')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
C:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: 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):
```



In [74]:

```
# Create a column which contains the brand of the product by extracting it
transaction_data_filtered.loc[:, 'BRAND'] = transaction_data_filtered['PRO

# Check the results
print(transaction_data_filtered[['PROD_NAME', 'BRAND']].head())

# Clean brand names
transaction_data_filtered.loc[:, 'BRAND'] = transaction_data_filtered['BRA
    'RED': 'RRD',
    'Red': 'RRD',  # Ensure both uppercase and title case are handled
    'Infzns': 'Infuzions',
    'Dorito': 'Doritos',
    'GrnWves': 'GrainWaves',
    'WW': 'Woolworths',
    'Snbts': 'Sunbites',
    'Smith': 'Smiths',  # Combine Smith and Smiths
    'Grain': 'GrainWaves',  # Combine Grain and GrainWaves
    'NCC': 'Natural',  # Combine NCC and Natural
})

# Check the cleaned brand names
print(transaction_data_filtered['BRAND'].unique())
```

```
                                PROD_NAME    BRAND
0    Natural Chip        Compny SeaSalt175g  Natural
1                  CCs Nacho Cheese    175g      CCs
2    Smiths Crinkle Cut  Chips Chicken 170g   Smiths
3    Smiths Chip Thinly  S/Cream&Onion 175g   Smiths
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g   Kettle
['Natural' 'CCs' 'Smiths' 'Kettle' 'GrainWaves' 'Doritos' 'Twisties'
 'Woolworths' 'Thins' 'Burger' 'Cheezels' 'Infuzions' 'RRD' 'Pringles'
 'Tyrrells' 'Cobs' 'French' 'Tostitos' 'Cheetos' 'Sunbites']
```

In [75]:
```
customer_data.head(20)
```

Out[75]:

	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
5	1007	YOUNG SINGLES/COUPLES	Budget
6	1009	NEW FAMILIES	Premium
7	1010	YOUNG SINGLES/COUPLES	Mainstream
8	1011	OLDER SINGLES/COUPLES	Mainstream
9	1012	OLDER FAMILIES	Mainstream
10	1013	RETIREES	Budget
11	1016	OLDER FAMILIES	Mainstream
12	1018	YOUNG SINGLES/COUPLES	Mainstream
13	1019	OLDER SINGLES/COUPLES	Premium
14	1020	YOUNG SINGLES/COUPLES	Mainstream
15	1022	OLDER FAMILIES	Budget
16	1023	MIDAGE SINGLES/COUPLES	Premium
17	1024	YOUNG SINGLES/COUPLES	Premium
18	1025	YOUNG FAMILIES	Budget
19	1026	MIDAGE SINGLES/COUPLES	Premium

In [77]:
```
customer_data.shape
```

Out[77]: (72637, 3)

In [78]:
```
customer_data.describe()
```

Out[78]:

	LYLTY_CARD_NBR
count	7.263700e+04
mean	1.361859e+05
std	8.989293e+04
min	1.000000e+03
25%	6.620200e+04
50%	1.340400e+05
75%	2.033750e+05
max	2.373711e+06

In [79]:
```
# transaction_data_filtered is your transaction data DataFrame
merged_data = pd.merge(transaction_data_filtered, customer_data, on='LYLTY

# Display the first few rows of the merged DataFrame
print(merged_data.head())
```

```
        DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2018-10-17          1            1000       1         5
1 2019-05-14          1            1307     348        66
2 2019-05-20          1            1343     383        61
3 2018-08-17          2            2373     974        69
4 2018-08-18          2            2426    1038       108

                                PROD_NAME  PROD_QTY  TOT_SALES  PACK_S
IZE  \
0    Natural Chip        Compny SeaSalt175g         2        6.0
175
1                  CCs Nacho Cheese    175g         3        6.3
175
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
170
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
175
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
150

                   CLEANED_PROD_NAME    BRAND               LIFESTAGE
\
0       Natural Chip Compny SeaSaltg  Natural   YOUNG SINGLES/COUPLES
1                    CCs Nacho Cheese      CCs  MIDAGE SINGLES/COUPLES
2    Smiths Crinkle Cut Chips Chicken   Smiths  MIDAGE SINGLES/COUPLES
3      Smiths Chip Thinly SCreamOnion   Smiths  MIDAGE SINGLES/COUPLES
4  Kettle Tortilla ChpsHnyJlpno Chili   Kettle  MIDAGE SINGLES/COUPLES

  PREMIUM_CUSTOMER
0          Premium
1           Budget
2           Budget
3           Budget
4           Budget
```

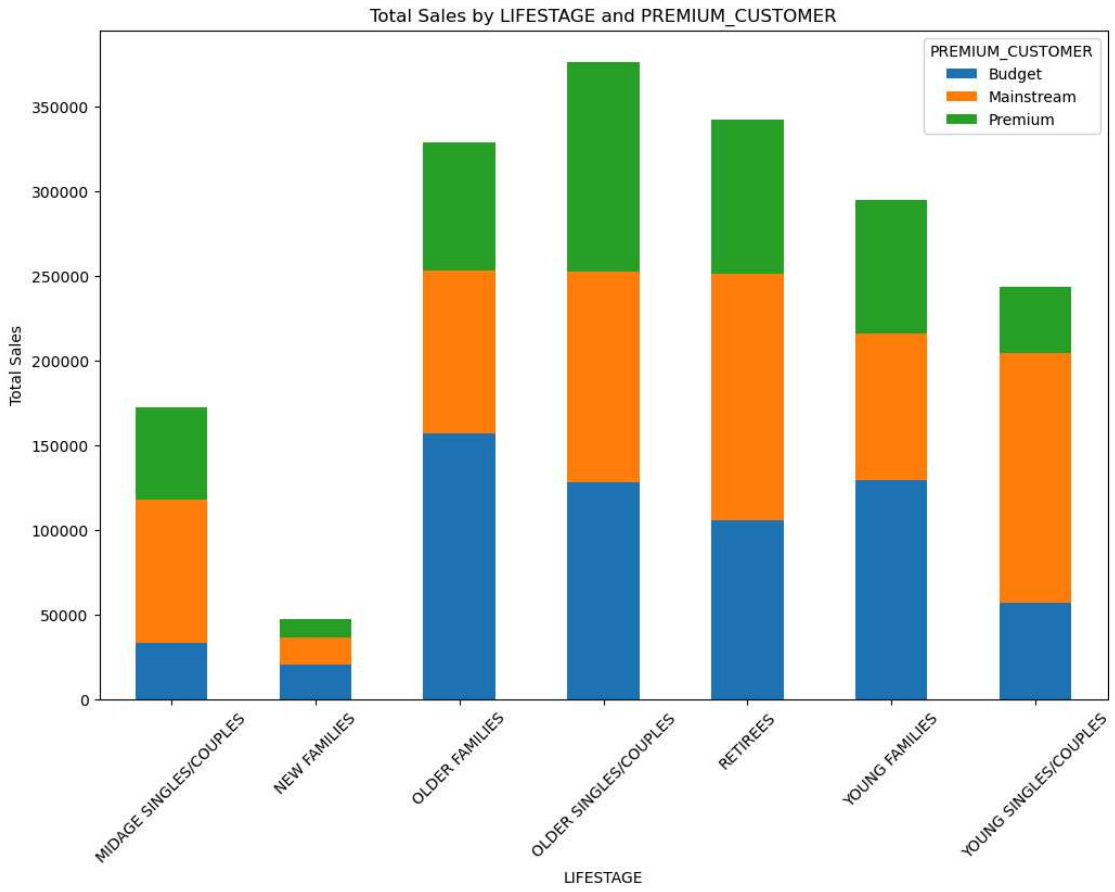
In [80]:

```
# Check for null values in the merged DataFrame
null_counts = merged_data.isnull().sum()

# Display the null counts
print(null_counts)
```

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

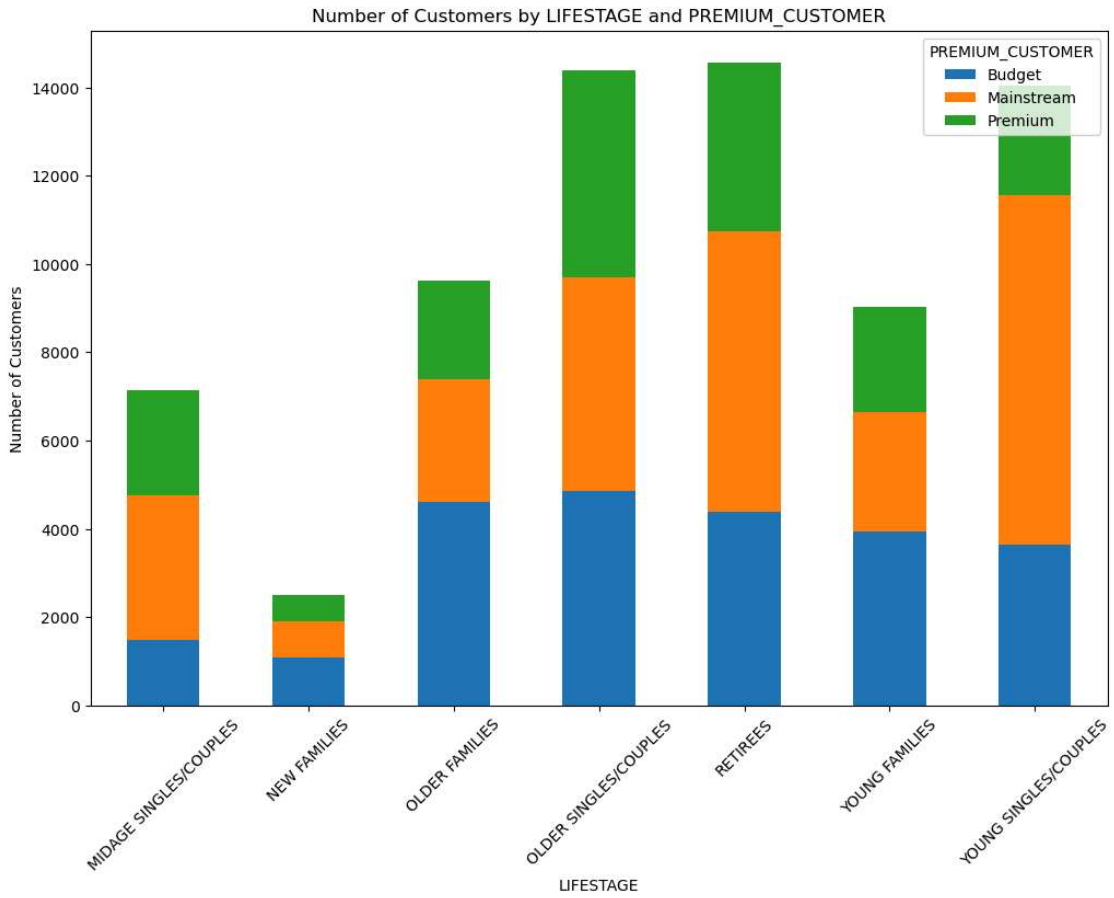
In [81]:

```
# Group by LIFESTAGE and PREMIUM_CUSTOMER and sum the total sales
sales_by_segment = merged_data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])

# Plot the total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales_by_segment.plot(kind='bar', stacked=True, figsize=(12, 8))

# Adding titles and labels
plt.title('Total Sales by LIFESTAGE and PREMIUM_CUSTOMER')
plt.xlabel('LIFESTAGE')
plt.ylabel('Total Sales')
plt.legend(title='PREMIUM_CUSTOMER', loc='upper right')
plt.xticks(rotation=45)

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



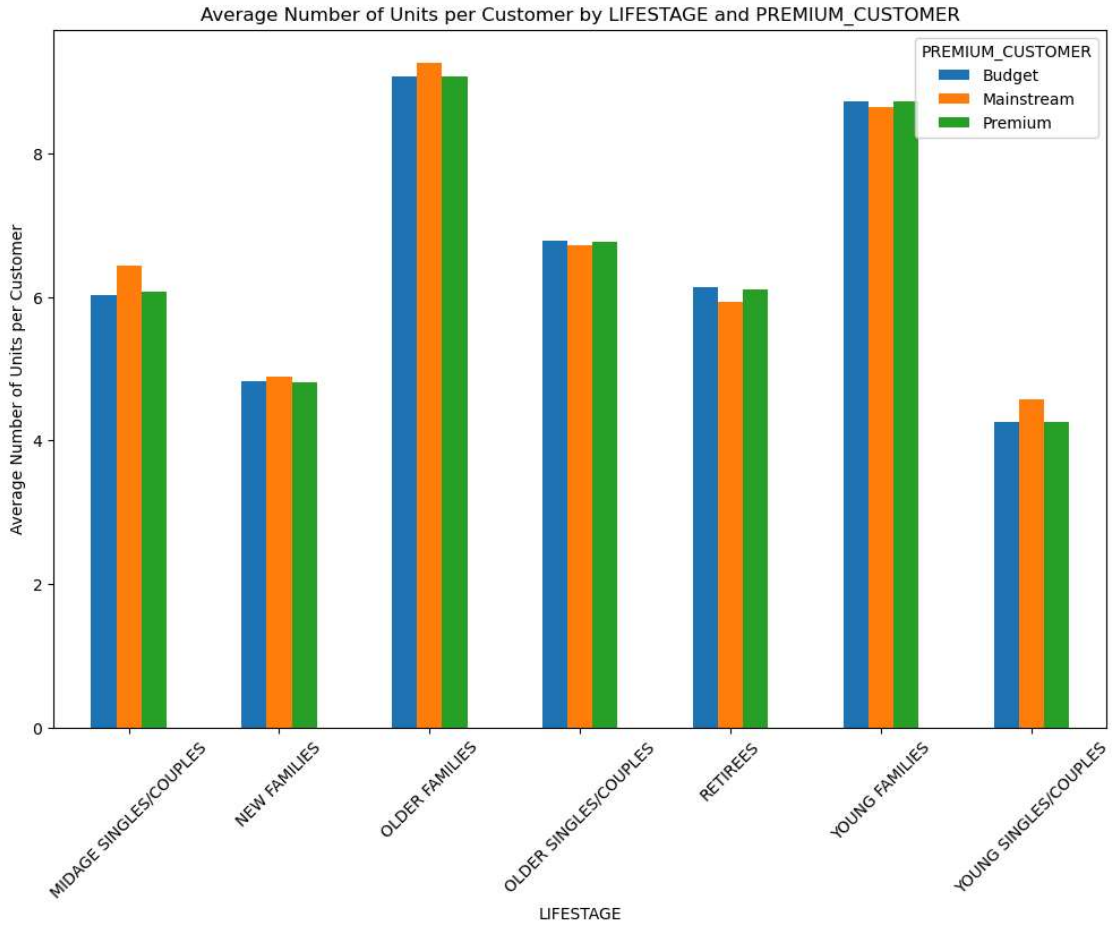
In [82]:

```
# Group by LIFESTAGE and PREMIUM_CUSTOMER and count the unique loyalty car
customers_by_segment = merged_data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER

# Plot the number of customers by LIFESTAGE and PREMIUM_CUSTOMER
customers_by_segment.plot(kind='bar', stacked=True, figsize=(12, 8))

# Adding titles and labels
plt.title('Number of Customers by LIFESTAGE and PREMIUM_CUSTOMER')
plt.xlabel('LIFESTAGE')
plt.ylabel('Number of Customers')
plt.legend(title='PREMIUM_CUSTOMER', loc='upper right')
plt.xticks(rotation=45)

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



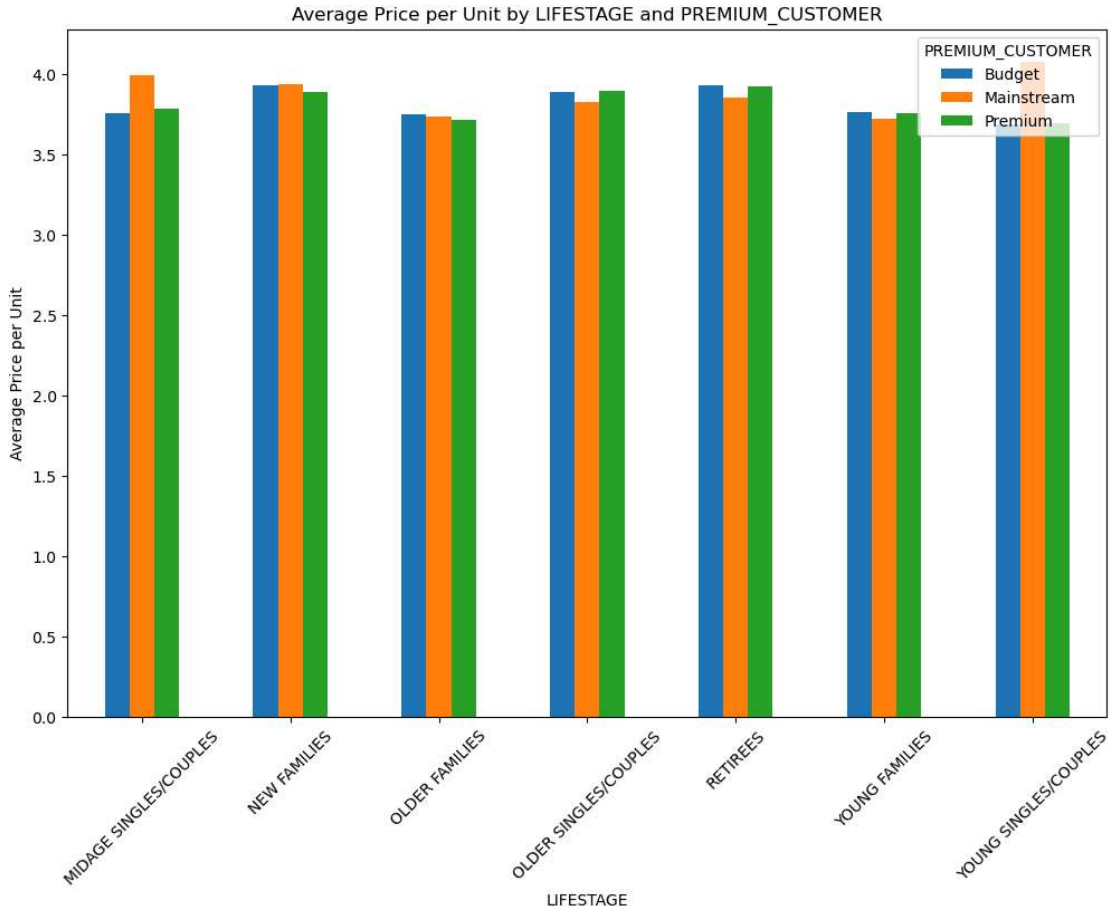
In [83]:

```
# Group by LIFESTAGE and PREMIUM_CUSTOMER and calculate total units and ur
grouped_data = merged_data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).agg(

# Calculate the average number of units per customer
grouped_data['Avg_Units_Per_Customer'] = grouped_data['PROD_QTY'] / groupe

# Unstack the data to get PREMIUM_CUSTOMER as columns
avg_units_per_customer = grouped_data['Avg_Units_Per_Customer'].unstack()

# Plot the average number of units per customer by LIFESTAGE and PREMIUM_C
avg_units_per_customer.plot(kind='bar', stacked=False, figsize=(12, 8))

# Adding titles and labels
plt.title('Average Number of Units per Customer by LIFESTAGE and PREMIUM_C
plt.xlabel('LIFESTAGE')
plt.ylabel('Average Number of Units per Customer')
plt.legend(title='PREMIUM_CUSTOMER', loc='upper right')
plt.xticks(rotation=45)

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



In [84]:

```
# Group by LIFESTAGE and PREMIUM_CUSTOMER and calculate total sales and to
grouped_data = merged_data.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).agg(

# Calculate the average price per unit
grouped_data['Avg_Price_Per_Unit'] = grouped_data['TOT_SALES'] / grouped_d

# Unstack the data to get PREMIUM_CUSTOMER as columns
avg_price_per_unit = grouped_data['Avg_Price_Per_Unit'].unstack()

# Plot the average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
avg_price_per_unit.plot(kind='bar', stacked=False, figsize=(12, 8))

# Adding titles and labels
plt.title('Average Price per Unit by LIFESTAGE and PREMIUM_CUSTOMER')
plt.xlabel('LIFESTAGE')
plt.ylabel('Average Price per Unit')
plt.legend(title='PREMIUM_CUSTOMER', loc='upper right')
plt.xticks(rotation=45)

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



In [85]:
```
from scipy.stats import ttest_ind

# Assuming merged_data is already defined

# Calculate the average price per unit for each transaction
merged_data['Avg_Price_Per_Unit'] = merged_data['TOT_SALES'] / merged_data

# Filter data for mainstream vs premium
mainstream = merged_data[(merged_data['PREMIUM_CUSTOMER'] == 'Mainstream')
premium = merged_data[(merged_data['PREMIUM_CUSTOMER'] == 'Premium')]

# Perform t-test between mainstream and premium
t_stat, p_value = ttest_ind(mainstream['Avg_Price_Per_Unit'], premium['Avg
print(f"T-test between Mainstream and Premium customers:\nT-statistic: {t_

# Filter data for budget midage and young singles/couples
budget_midage_young = merged_data[(merged_data['PREMIUM_CUSTOMER'] == 'Bud
                                  (merged_data['LIFESTAGE'].isin(['MIDAGE

# Split into midage and young singles/couples
midage_singles_couples = budget_midage_young[budget_midage_young['LIFESTAG
young_singles_couples = budget_midage_young[budget_midage_young['LIFESTAGE

# Perform t-test between midage and young singles/couples
t_stat, p_value = ttest_ind(midage_singles_couples['Avg_Price_Per_Unit'],
                            young_singles_couples['Avg_Price_Per_Unit'],
                            equal_var=False)
print(f"T-test between Budget Midage and Young Singles/Couples:\nT-statist
```

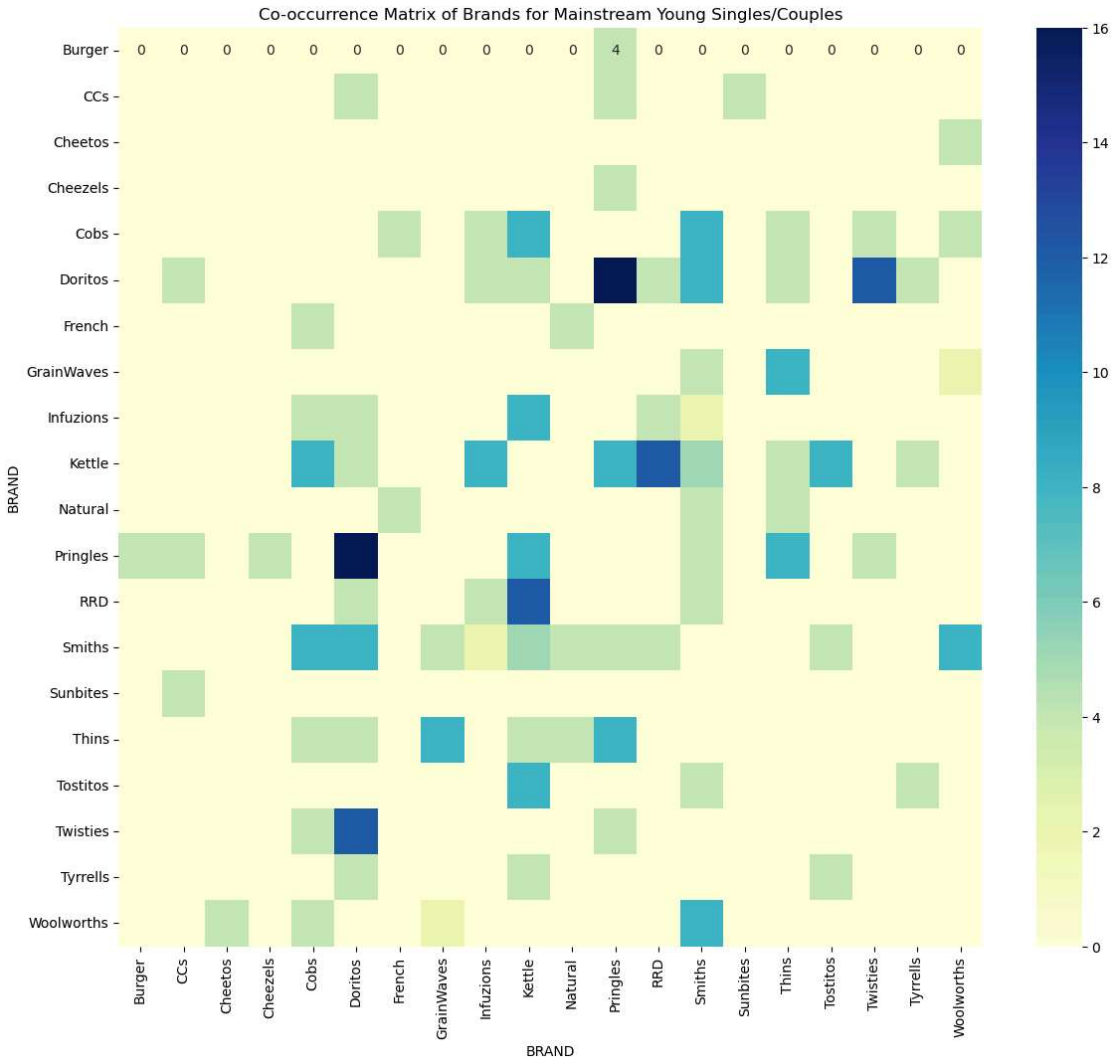
```
T-test between Mainstream and Premium customers:
T-statistic: 11.05723574336507, P-value: 2.0788364041187993e-28
T-test between Budget Midage and Young Singles/Couples:
T-statistic: 4.343021896976378, P-value: 1.4196195274595408e-05
```

In [87]:
```
pip install mlxtend
```

```
Collecting mlxtendNote: you may need to restart the kernel to use upd
ated packages.

  Downloading mlxtend-0.23.1-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in c:\users\tonye\anacond
a3\lib\site-packages (from mlxtend) (1.11.4)
Requirement already satisfied: numpy>=1.16.2 in c:\users\tonye\anacon
da3\lib\site-packages (from mlxtend) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in c:\users\tonye\anaco
nda3\lib\site-packages (from mlxtend) (2.1.4)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\tonye
\anaconda3\lib\site-packages (from mlxtend) (1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\tonye\an
aconda3\lib\site-packages (from mlxtend) (3.8.0)
Requirement already satisfied: joblib>=0.13.2 in c:\users\tonye\anaco
nda3\lib\site-packages (from mlxtend) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\tonye\ana
conda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\tonye\anacond
```

In [91]:

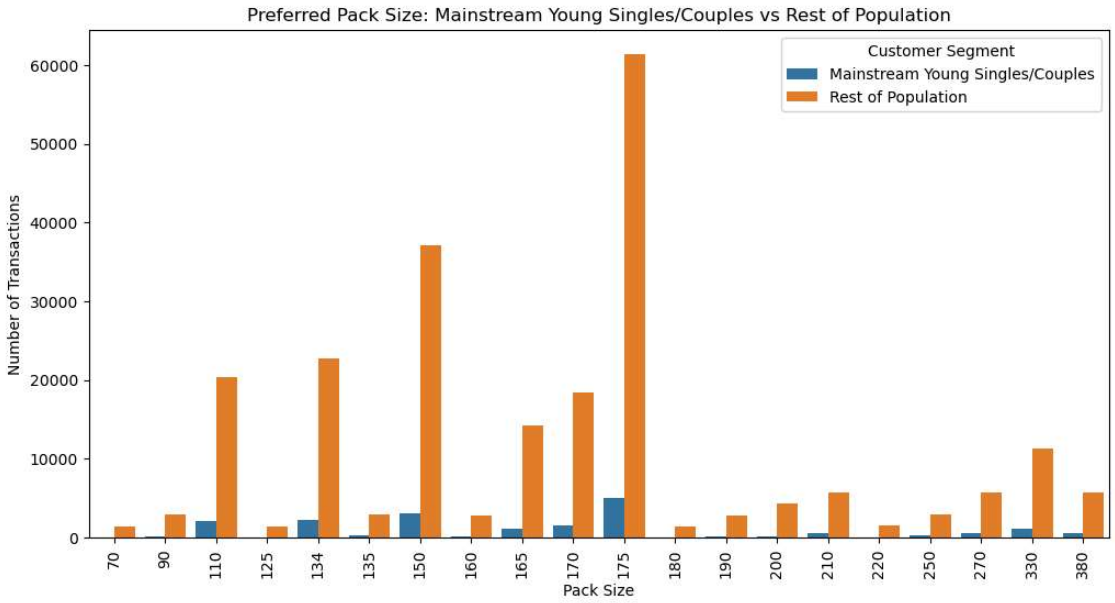
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Filter the data
mainstream_young_singles_couples = merged_data[
    (merged_data['PREMIUM_CUSTOMER'] == 'Mainstream') &
    (merged_data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES')
]

# Step 2: Transform the data
basket = (mainstream_young_singles_couples
          .groupby(['TXN_ID', 'BRAND'])['PROD_QTY']
          .sum().unstack().reset_index().fillna(0)
          .set_index('TXN_ID'))

# Create a co-occurrence matrix
cooccurrence_matrix = basket.T.dot(basket)
# Set the diagonal to 0 to ignore self-co-occurrence
for i in range(len(cooccurrence_matrix)):
    cooccurrence_matrix.iat[i, i] = 0

# Step 3: Visualize the co-occurrence matrix
plt.figure(figsize=(14, 12))
sns.heatmap(cooccurrence_matrix, annot=True, cmap="YlGnBu")
plt.title('Co-occurrence Matrix of Brands for Mainstream Young Singles/Cou
plt.show()
```


Co-occurrence Matrix of Brands for Mainstream Young Singles/Couples
BRAND
Burger
CCs
Cheetos
Cheezels
Cobs
Doritos
French
GrainWaves
Infuzions
Kettle
Natural
Pringles
RRD
Smiths
Sunbites
Thins
Tostitos
Twisties
Tyrrells
Woolworths
0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0 0
16
14
12
10
8
6
4
2
0

In [92]:

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

# Filter the data for mainstream young singles/couples
mainstream_young_singles_couples = merged_data[
    (merged_data['PREMIUM_CUSTOMER'] == 'Mainstream') &
    (merged_data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES')
]

# Calculate the average pack size for the target segment
avg_pack_size_target = mainstream_young_singles_couples.groupby('PACK_SIZE
avg_pack_size_target['segment'] = 'Mainstream Young Singles/Couples'

# Calculate the average pack size for the rest of the population
rest_of_population = merged_data[
    ~((merged_data['PREMIUM_CUSTOMER'] == 'Mainstream') &
      (merged_data['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES'))
]
avg_pack_size_rest = rest_of_population.groupby('PACK_SIZE').size().reset_
avg_pack_size_rest['segment'] = 'Rest of Population'

# Combine both dataframes
avg_pack_size = pd.concat([avg_pack_size_target, avg_pack_size_rest])

# Plotting the comparison
plt.figure(figsize=(12, 6))
sns.barplot(data=avg_pack_size, x='PACK_SIZE', y='count', hue='segment')
plt.title('Preferred Pack Size: Mainstream Young Singles/Couples vs Rest o
plt.xlabel('Pack Size')
plt.ylabel('Number of Transactions')
plt.legend(title='Customer Segment')
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Dep
recationWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Dep
recationWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Dep
recationWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\Tonye\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Dep
recationWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, pd.CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```


Preferred Pack Size: Mainstream Young Singles/Couples vs Rest of Population
Customer Segment
Mainstream Young Singles/Couples
Rest of Population
Number of Transactions
0
10000
20000
30000
40000
50000
60000
70
90
110
125
134
135
150
160
165
170
175
180
190
200
210
220
250
270
330
380
Pack Size

In [ ]: