In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:

```
# Loading required libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import datetime
import xlrd
import re
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from sklearn.preprocessing import OneHotEncoder
```

In [ ]:

```
import pandas as pd

# Example paths — adjust according to your folder structure
purchase_data_path = '/content/drive/MyDrive/Quantium/QVI_purchase_behaviour
transaction_data_path = '/content/drive/MyDrive/Quantium/QVI_transaction_dat

# Load the data
purchase_df = pd.read_csv(purchase_data_path)
transaction_df = pd.read_excel(transaction_data_path)
```

In [ ]:

```
# View basic structure
purchase_df.head()
```

Out[ ]:

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

In [ ]:

```
transaction_df.head()
```

Out[ ]:

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

In [ ]:

```
transaction_df['DATE'] = pd.to_datetime(transaction_df['DATE'], origin='1899
```

In [ ]:

```
transaction_df.head()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
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 [ ]:

```
# Null values
print("Purchase Data Nulls:\n", purchase_df.isnull().sum())
print("\nTransaction Data Nulls:\n", transaction_df.isnull().sum())

# Duplicates
print("\nDuplicate Transactions:", transaction_df.duplicated().sum())
print("Duplicate Customers:", purchase_df.duplicated().sum())
```

```
Purchase Data Nulls:
 LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64

Transaction Data Nulls:
 DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
dtype: int64

Duplicate Transactions: 1
Duplicate Customers: 0
```

In [ ]:

```
# View all unique entries in the product name column
transaction_df['PROD_NAME'].unique()
```

```
Out[ ]: 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 134g',
               '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 Salt',
               '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 300g',
               '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     165g',
               'Smith Crinkle Cut   Bolognese 150g', 'Doritos Salsa Mild  300g'],
              dtype=object)
```

We want to check that the products are only chips by counting the word frequencies in the product names. To make this process clearer, we can remove the digits and symbols from the names.

In [ ]:

```
# Remove digits from the product names
prod_name = transaction_df['PROD_NAME'].str.replace(r'[0-9]+[gG]','',regex=T

# Remove & characters from the product names and replace with a space to sep
prod_name = prod_name.str.replace(r'&',' ',regex=True);
```

In [ ]:

```
# Count the frequencies of words in product names and display counts in desc
word_counts = pd.Series(' '.join(prod_name).split()).value_counts()
```

```
with pd.option_context('display.max_rows', None): # show all rows
  display(word_counts)
```

	count
Chips	49770
Kettle	41288
Smiths	28860
Salt	27976
Cheese	27890
Pringles	25102
Doritos	24962
Crinkle	23960
Corn	22063
Original	21560
Cut	20754
Chip	18645
Chicken	18577
Salsa	18094
Chilli	15390
Sea	14145
Thins	14075
Sour	13882
Crisps	12607
Vinegar	12402
RRD	11894
Sweet	11060
Infuzions	11057
Supreme	10963
Chives	10951
Cream	10723
WW	10320
Cobs	9693
Popd	9693
Tortilla	9580
Tostitos	9471
Twisties	9454

	count
BBQ	9434
Sensations	9429
Lime	9347
Old	9324
Dip	9324
Paso	9324
El	9324
Tomato	7669
Thinly	7507
Tyrrells	6442
And	6373
Tangy	6332
SourCream	6296
Waves	6272
Grain	6272
Lightly	6248
Salted	6248
Soy	6121
Onion	6116
Natural	6050
Mild	6048
Deli	5885
Rock	5885
Red	5885
Thai	4737
Burger	4733
Swt	4718
Honey	4661
Nacho	4658
Potato	4647
Cheezels	4603
Garlic	4572

	count
CCs	4551
Woolworths	4437
Mozzarella	3304
Basil	3304
Pesto	3304
Jlpno	3296
ChpsHny	3296
Chili	3296
Swt/Chlli	3269
Sr/Cream	3269
Ched	3268
Pot	3257
Splash	3252
Of	3252
PotatoMix	3242
SweetChili	3242
Orgnl	3233
Crnkle	3233
Big	3233
Bag	3233
Spicy	3229
Hot	3229
Camembert	3219
Fig	3219
Barbeque	3210
Mexican	3204
Jalapeno	3204
Light	3188
Chp	3185
Dorito	3185
Spcy	3177
Rib	3174

	count
Prawn	3174
Crackers	3174
Southern	3172
Crm	3159
ChpsBtroot	3146
Ricotta	3146
Smoked	3145
Chipotle	3145
Gcamole	3144
Infzns	3144
Crn	3144
Crnchers	3144
ChpsFeta	3138
Strws	3134
Herbs	3134
Veg	3134
Siracha	3127
Chnky	3125
Ht	3125
Tom	3125
Mexicana	3115
Med	3114
Mystery	3114
Seasonedchicken	3114
Flavour	3114
Crips	3104
Vingar	3095
Slt	3095
Maple	3083
FriedChicken	3083
Sthrn	3083
Rings	3080

	count
ChipCo	3010
SR	2984
Smith	2963
Chs	2960
S/Cream	2934
Cheetos	2927
Medium	2879
French	2856
Mstrd	1576
Cheddr	1576
Whlgrn	1576
Snbts	1576
Co	1572
Hrb	1572
Tmato	1572
Spce	1572
Vinegr	1550
Tasty	1539
Belly	1526
Rst	1526
Slow	1526
Pork	1526
Roast	1519
Mac	1512
N	1512
Chutny	1507
Mango	1507
Papadums	1507
Coconut	1506
Sauce	1503
Snag	1503
Sp	1498

	count
Truffle	1498
Barbecue	1489
Stacked	1487
OnionStacked	1483
Bacon	1479
Balls	1479
Pepper	1473
D/Style	1469
Compny	1468
GrnWves	1468
Btroot	1468
SeaSalt	1468
Plus	1468
Jam	1468
Chli	1461
Hony	1460
Chckn	1460
Mzzrlla	1458
Chimuchurri	1455
Steak	1455
Box	1454
Bolognese	1451
Puffs	1448
Originl	1441
saltd	1441
CutSalt/Vinegr	1440
OnionDip	1438
Chikn	1434
Aioli	1434
Sunbites	1432
Frch/Onin	1432
Whlegrn	1432

	count
Pc	1431
NCC	1419
Garden	1419
Fries	1418

dtype: int64

In [ ]:
```
transaction_df.shape
```

Out[ ]:
```
(264836, 8)
```

In [ ]:
```
# Remove salsas from the dataset
transaction_df = transaction_df[transaction_df['PROD_NAME'].str.contains(r"[
transaction_df.shape # check for a reduction in no of rows
```

Out[ ]:
```
(246742, 8)
```

In [ ]:
```
transaction_df.describe()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD
count	246742	246742.000000	2.467420e+05	2.467420e+05	246742.00
mean	2018-12-30 01:19:01.211467520	135.051098	1.355310e+05	1.351311e+05	56.3
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.00
25%	2018-09-30 00:00:00	70.000000	7.001500e+04	6.756925e+04	26.00
50%	2018-12-30 00:00:00	130.000000	1.303670e+05	1.351830e+05	53.00
75%	2019-03-31 00:00:00	203.000000	2.030840e+05	2.026538e+05	87.00
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.00
std	NaN	76.787096	8.071528e+04	7.814772e+04	33.69

In [ ]:
```
# Check if there are any nans in the dataset
transaction_df.isnull().values.any()
```

Out[ ]:
```
np.False_
```

From the summary, there is at least one transaction with 200 packets. Let's investigate this purchase further.

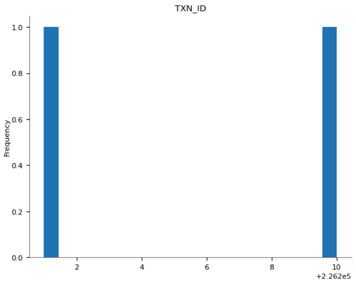
In [ ]:

```python
# Filter the entries that have 200 packets.
transaction_df.loc[transaction_df['PROD_QTY'] == 200.0]
```

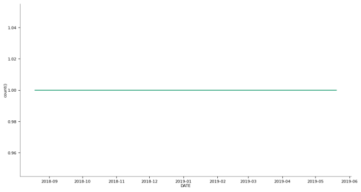
Out[ ]:

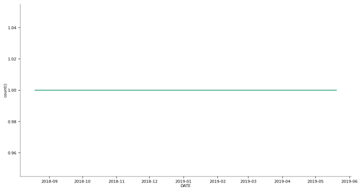
	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PR
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	

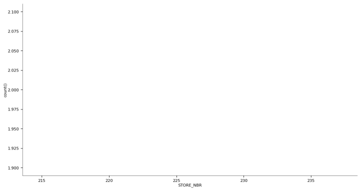
Distributions

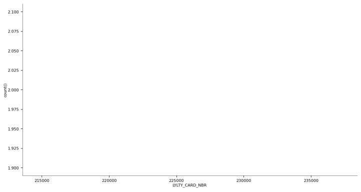


Time series

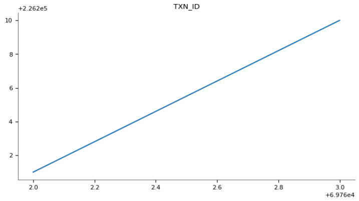








Values



In [ ]:

```
transaction_df.loc[transaction_df['LYLTY_CARD_NBR'] == 226000]
```

Out[ ]:

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

Looks like these are duplicate values made by the same customer. So we will remove them.

In [ ]:

```
# Remove the transactions
transaction_df = transaction_df[transaction_df['LYLTY_CARD_NBR'] != 226000]
```

In [ ]:

```
# Recheck the data summary
transaction_df.describe()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PRO
count	246740	246740.000000	2.467400e+05	2.467400e+05	246740.(
mean	2018-12-30 01:18:58.448569344	135.050361	1.355303e+05	1.351304e+05	56.
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.(
25%	2018-09-30 00:00:00	70.000000	7.001500e+04	6.756875e+04	26.(
50%	2018-12-30 00:00:00	130.000000	1.303670e+05	1.351815e+05	53.(
75%	2019-03-31 00:00:00	203.000000	2.030832e+05	2.026522e+05	87.(
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.(
std	NaN	76.786971	8.071520e+04	7.814760e+04	33.(

In [ ]:

```
# Add a new column to data with packet sizes and extract sizes from product
transaction_df.insert(8, "PACK_SIZE", transaction_df['PROD_NAME'].str.extrac
# Sort by packet sizes to check for outliers
transaction_df.sort_values(by='PACK_SIZE')
```

```
<>:2: DeprecationWarning: invalid escape sequence '\d'
<>:2: DeprecationWarning: invalid escape sequence '\d'
<ipython-input-20-cf72fe0547e9>:2: DeprecationWarning: invalid escape sequen
ce '\d'
  transaction_df.insert(8, "PACK_SIZE", transaction_df['PROD_NAME'].str.extr
act('(\d+)').astype(float), True)
```

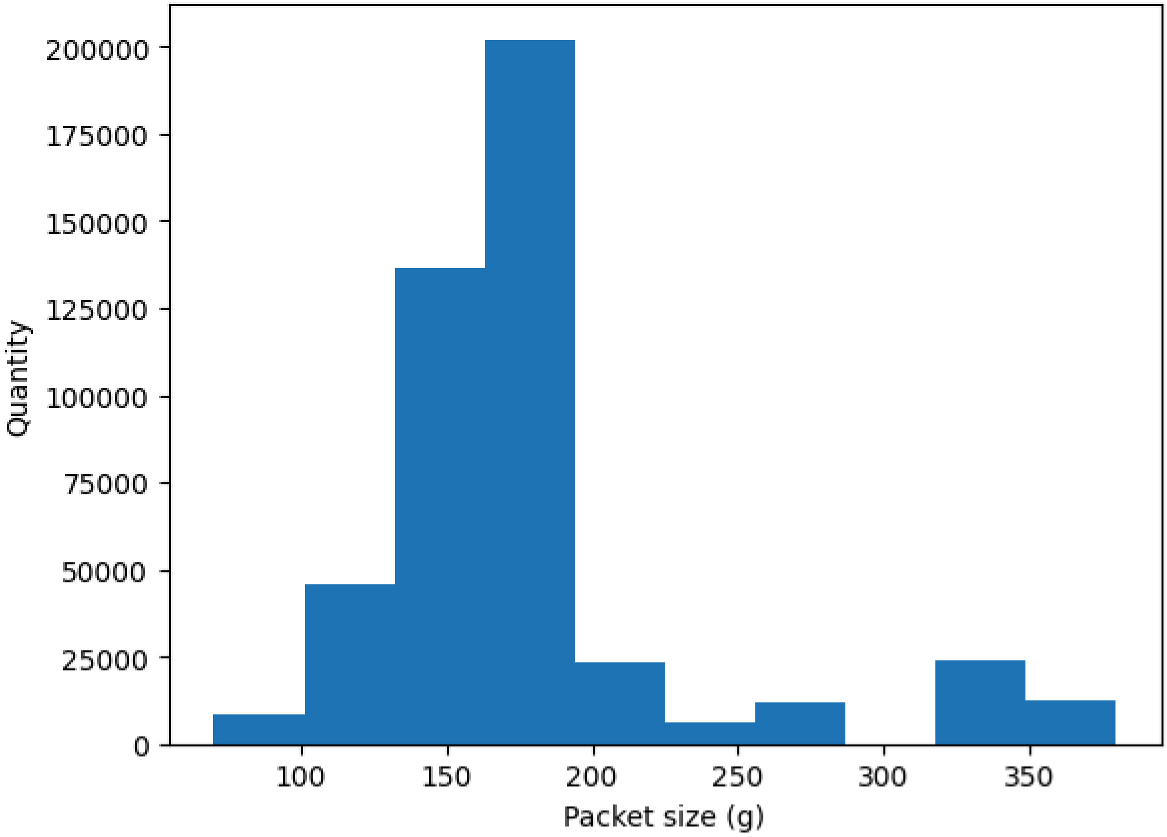
Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	P
259246	2019-03-08	20	20172	17144	38	Infuzions Mango Chutny Papadums 70g	
259135	2019-02-05	16	16458	14571	38	Infuzions Mango Chutny Papadums 70g	
258984	2019-02-23	9	9030	8391	38	Infuzions Mango Chutny Papadums 70g	
258943	2018-11-07	6	6473	6281	38	Infuzions Mango Chutny Papadums 70g	
259305	2018-09-24	22	22113	18261	38	Infuzions Mango Chutny Papadums 70g	
...	...	...	...	...	...	...	
145130	2019-01-25	175	175285	176546	14	Smiths Crnkle Chip Orgnl Big Bag 380g	
145165	2018-12-08	175	175375	176903	4	Dorito Corn Chp Supreme 380g	
145111	2019-05-10	175	175149	176010	14	Smiths Crnkle Chip Orgnl Big Bag 380g	
41	2019-05-20	55	55073	48887	4	Dorito Corn Chp Supreme 380g	
145073	2019-06-06	175	175048	175664	4	Dorito Corn Chp Supreme 380g	

246740 rows × 9 columns

In [ ]:

```
# Plot a histogram to visualise distribution of pack sizes.
plt.hist(transaction_df['PACK_SIZE'], weights=transaction_df['PROD_QTY']);
plt.xlabel('Packet size (g)');
plt.ylabel('Quantity');
```



In [ ]:

```
# Add a column to extract the first word of each product name to.
transaction_df.insert(9, "BRAND_NAME",transaction_df['PROD_NAME'].str.split(
transaction_df
```

Out[ ]:

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

246740 rows × 10 columns

Now we want to examine the customer data.

In [ ]:

```
# Now examine customer data
cust_df = purchase_df.copy()
cust_df.head()
```

Out[ ]:

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

In [ ]:

```
# Rename "PREMIUM_CUSTOMER" to "MEMBER_TYPE" for easier identification of th
cust_df = cust_df.rename(columns={'PREMIUM_CUSTOMER': 'MEMBER_TYPE'})
```

In [ ]:

```
# Check the summary of the customer data
cust_df.describe()
```

Out[ ]:

	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 [ ]:

```
# Join the customer and transaction datasets
merged_df = transaction_df.merge(cust_df, how='left', on='LYLTY_CARD_NBR')
merged_df.head()
```

Out[ ]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
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 [ ]:

```
#Sort transactons by date
full_df = merged_df.reset_index()
full_df = full_df.sort_values(by='DATE').reset_index(drop=True)
full_df
```

Out[ ]:

	index	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_
0	8572	2018-07-01	88	88140	86914	25	P Sour Onio
1	144861	2018-07-01	60	60276	57330	3	Sens Camem Fi
2	168994	2018-07-01	199	199014	197623	104	Inf Swe Pot
3	214152	2018-07-01	35	35052	31630	11	RRD Sal
4	97432	2018-07-01	72	72104	71038	20	D C Su
...	...	...	...	...	...	...	
246735	9601	2019-06-30	112	112141	114611	98	NC Cr C Chive
246736	105465	2019-06-30	207	207155	205513	99	P FriedC
246737	213436	2019-06-30	10	10140	9882	12	Natur Co Hrb
246738	213283	2019-06-30	6	6258	6047	29	Frenc Potatc
246739	244804	2019-06-30	183	183196	185975	22	Thins Origir

246740 rows × 13 columns

In [ ]:

```
# Check for nulls in the full dataset
full_df.isnull().values.any()
```

Out[ ]:

```
np.False_
```

Define Key Metrics per Segment. Data analysis on customer segments

In [ ]:

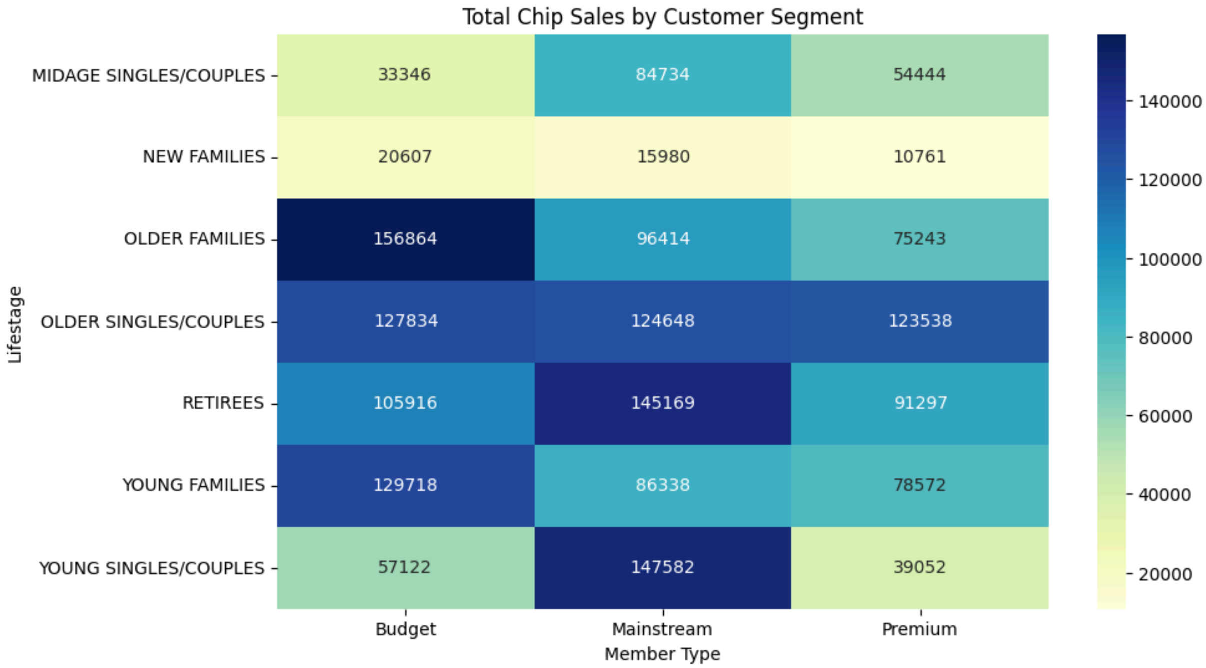
```
# Total sales by segment
segment_sales = full_df.groupby(['LIFESTAGE', 'MEMBER_TYPE'])['TOT_SALES'].s
segment_sales.sort_values(by = "TOT_SALES", ascending = False)
```

Out[ ]:

	LIFESTAGE	MEMBER_TYPE	TOT_SALES
6	OLDER FAMILIES	Budget	156863.75
19	YOUNG SINGLES/COUPLES	Mainstream	147582.20
13	RETIREES	Mainstream	145168.95
15	YOUNG FAMILIES	Budget	129717.95
9	OLDER SINGLES/COUPLES	Budget	127833.60
10	OLDER SINGLES/COUPLES	Mainstream	124648.50
11	OLDER SINGLES/COUPLES	Premium	123537.55
12	RETIREES	Budget	105916.30
7	OLDER FAMILIES	Mainstream	96413.55
14	RETIREES	Premium	91296.65
16	YOUNG FAMILIES	Mainstream	86338.25
1	MIDAGE SINGLES/COUPLES	Mainstream	84734.25
17	YOUNG FAMILIES	Premium	78571.70
8	OLDER FAMILIES	Premium	75242.60
18	YOUNG SINGLES/COUPLES	Budget	57122.10
2	MIDAGE SINGLES/COUPLES	Premium	54443.85
20	YOUNG SINGLES/COUPLES	Premium	39052.30
0	MIDAGE SINGLES/COUPLES	Budget	33345.70
3	NEW FAMILIES	Budget	20607.45
4	NEW FAMILIES	Mainstream	15979.70
5	NEW FAMILIES	Premium	10760.80

In [ ]:

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

# Total sales heatmap
pivot = segment_sales.pivot(index='LIFESTAGE', columns='MEMBER_TYPE', values
plt.figure(figsize=(10,6))
sns.heatmap(pivot, annot=True, fmt='.0f', cmap='YlGnBu')
plt.title('Total Chip Sales by Customer Segment')
plt.ylabel('Lifestage')
plt.xlabel('Member Type')
plt.show()
```



In [ ]:

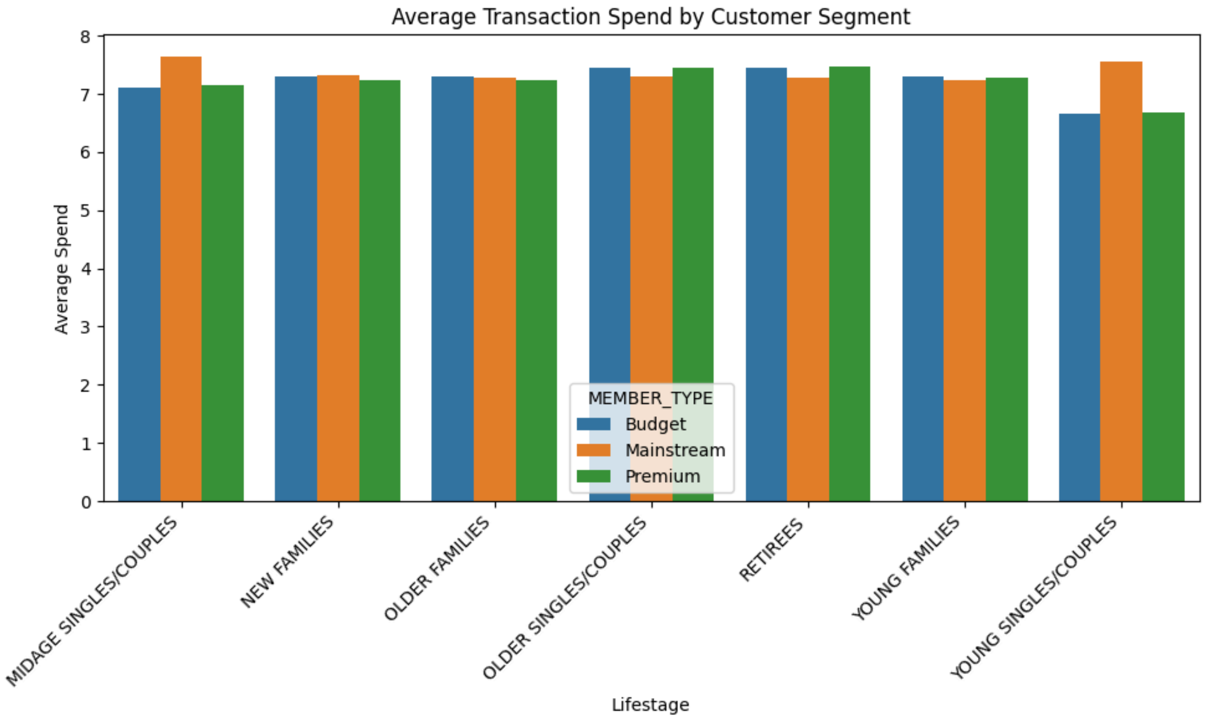
```
# Avg spend per transaction
avg_spend = full_df.groupby(['LIFESTAGE', 'MEMBER_TYPE'])['TOT_SALES'].mean(
avg_spend.sort_values(by = "AVG_SPEND", ascending = False)
```

Out[ ]:

	LIFESTAGE	MEMBER_TYPE	AVG_SPEND
1	MIDAGE SINGLES/COUPLES	Mainstream	7.637156
19	YOUNG SINGLES/COUPLES	Mainstream	7.551279
14	RETIREES	Premium	7.461315
11	OLDER SINGLES/COUPLES	Premium	7.459997
12	RETIREES	Budget	7.445786
9	OLDER SINGLES/COUPLES	Budget	7.444305
4	NEW FAMILIES	Mainstream	7.313364
10	OLDER SINGLES/COUPLES	Mainstream	7.306049
15	YOUNG FAMILIES	Budget	7.302705
3	NEW FAMILIES	Budget	7.297256
6	OLDER FAMILIES	Budget	7.291241
17	YOUNG FAMILIES	Premium	7.285951
7	OLDER FAMILIES	Mainstream	7.281440
13	RETIREES	Mainstream	7.269352
8	OLDER FAMILIES	Premium	7.232779
5	NEW FAMILIES	Premium	7.231720
16	YOUNG FAMILIES	Mainstream	7.226772
2	MIDAGE SINGLES/COUPLES	Premium	7.152371
0	MIDAGE SINGLES/COUPLES	Budget	7.108442
20	YOUNG SINGLES/COUPLES	Premium	6.673325
18	YOUNG SINGLES/COUPLES	Budget	6.663023

In [ ]:

```python
# Create the plot
plt.figure(figsize=(10, 6))
sns.barplot(x='LIFESTAGE', y='AVG_SPEND', hue='MEMBER_TYPE', data=avg_spend)
plt.title('Average Transaction Spend by Customer Segment')
plt.xlabel('Lifestage')
plt.ylabel('Average Spend')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better reada
plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```



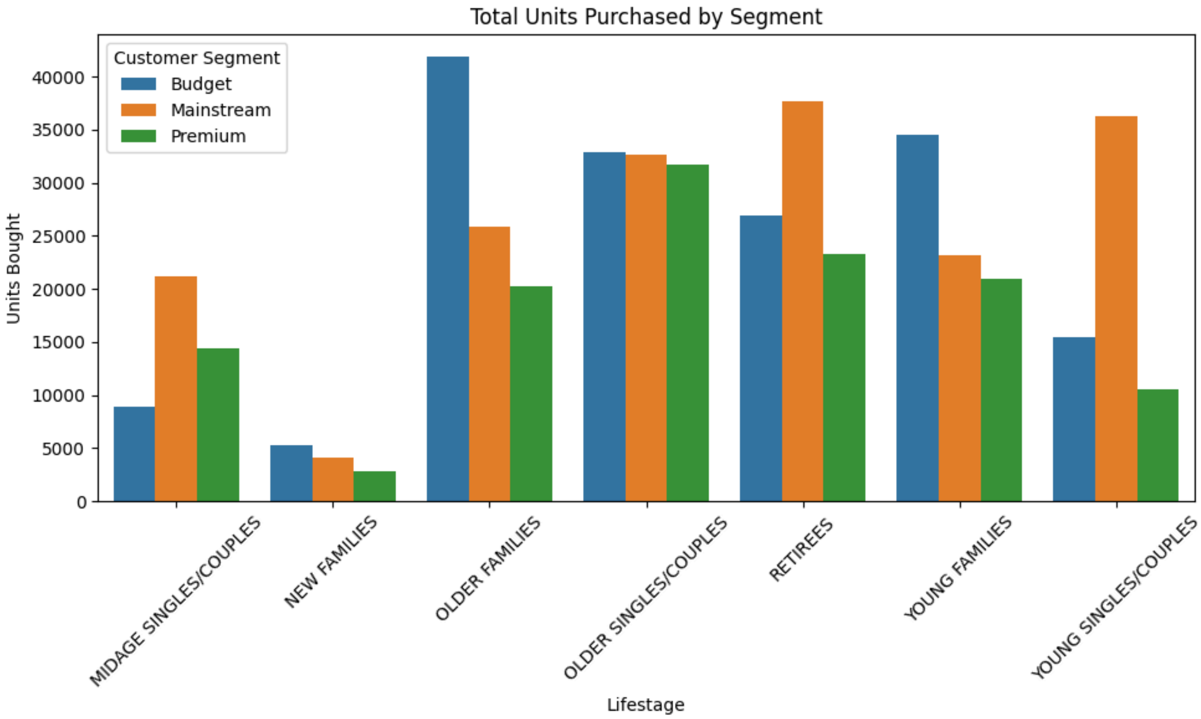
```
In [ ]: # Total units purchased
        total_qty = full_df.groupby(['LIFESTAGE', 'MEMBER_TYPE'])['PROD_QTY'].sum().
        total_qty.sort_values(by = "TOTAL_UNITS", ascending = False)
```

Out[ ]:

	LIFESTAGE	MEMBER_TYPE	TOTAL_UNITS
6	OLDER FAMILIES	Budget	41853
13	RETIREES	Mainstream	37677
19	YOUNG SINGLES/COUPLES	Mainstream	36225
15	YOUNG FAMILIES	Budget	34482
9	OLDER SINGLES/COUPLES	Budget	32883
10	OLDER SINGLES/COUPLES	Mainstream	32607
11	OLDER SINGLES/COUPLES	Premium	31695
12	RETIREES	Budget	26932
7	OLDER FAMILIES	Mainstream	25804
14	RETIREES	Premium	23266
16	YOUNG FAMILIES	Mainstream	23194
1	MIDAGE SINGLES/COUPLES	Mainstream	21213
17	YOUNG FAMILIES	Premium	20901
8	OLDER FAMILIES	Premium	20239
18	YOUNG SINGLES/COUPLES	Budget	15500
2	MIDAGE SINGLES/COUPLES	Premium	14400
20	YOUNG SINGLES/COUPLES	Premium	10575
0	MIDAGE SINGLES/COUPLES	Budget	8883
3	NEW FAMILIES	Budget	5241
4	NEW FAMILIES	Mainstream	4060
5	NEW FAMILIES	Premium	2769

In [ ]:

```
plt.figure(figsize=(10, 6))
sns.barplot(data=total_qty, x='LIFESTAGE', y='TOTAL_UNITS', hue='MEMBER_TYPE
plt.title('Total Units Purchased by Segment')
plt.xticks(rotation=45)
plt.ylabel('Units Bought')
plt.xlabel('Lifestage')
plt.legend(title='Customer Segment')
plt.tight_layout()
plt.show()
```



In [ ]:

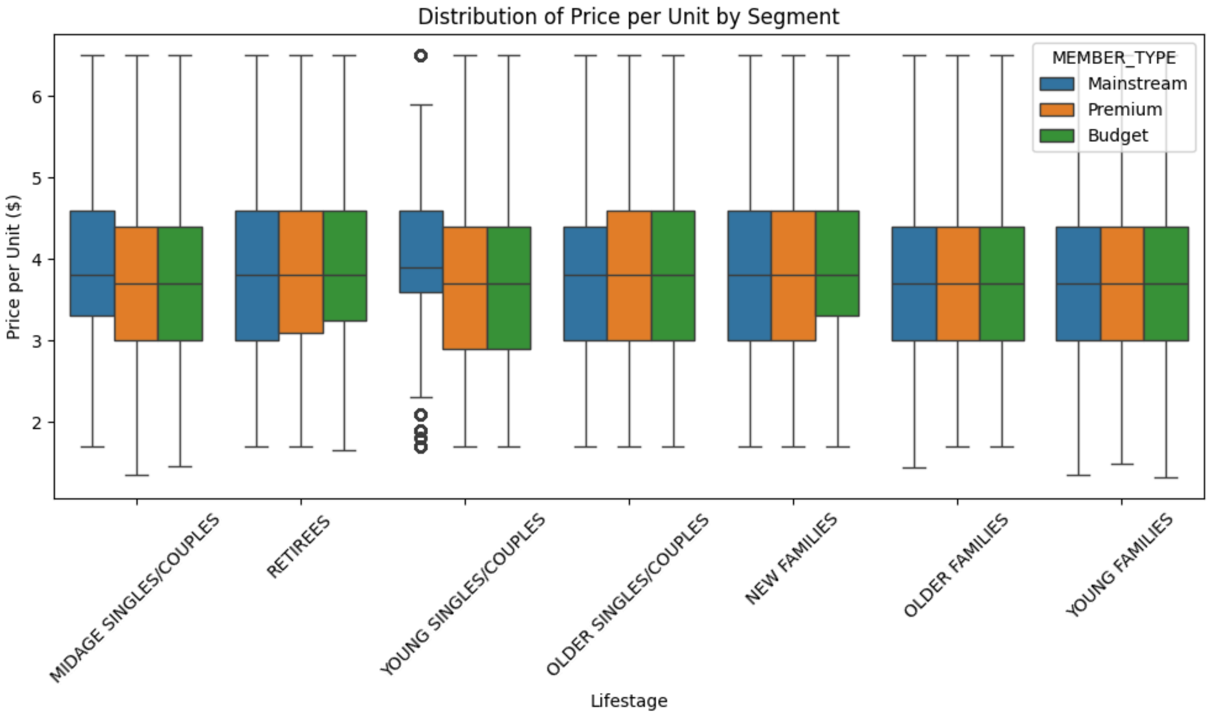
```
# Compute Average Price per Unit by Segment
# Calculate price per unit
full_df['PRICE_PER_UNIT'] = full_df['TOT_SALES'] / full_df['PROD_QTY']

# Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
avg_price_per_unit = (full_df.groupby(['LIFESTAGE', 'MEMBER_TYPE'])['PRICE_F

avg_price_per_unit.sort_values(by='PRICE_PER_UNIT', ascending=False)
```

Out[ ]:

	LIFESTAGE	MEMBER_TYPE	PRICE_PER_UNIT
19	YOUNG SINGLES/COUPLES	Mainstream	4.065642
1	MIDAGE SINGLES/COUPLES	Mainstream	3.994241
12	RETIREES	Budget	3.924404
14	RETIREES	Premium	3.920942
3	NEW FAMILIES	Budget	3.917688
4	NEW FAMILIES	Mainstream	3.916133
11	OLDER SINGLES/COUPLES	Premium	3.893182
9	OLDER SINGLES/COUPLES	Budget	3.882096
5	NEW FAMILIES	Premium	3.872110
13	RETIREES	Mainstream	3.844294
10	OLDER SINGLES/COUPLES	Mainstream	3.814665
2	MIDAGE SINGLES/COUPLES	Premium	3.770698
17	YOUNG FAMILIES	Premium	3.762150
15	YOUNG FAMILIES	Budget	3.760737
6	OLDER FAMILIES	Budget	3.745340
0	MIDAGE SINGLES/COUPLES	Budget	3.743328
7	OLDER FAMILIES	Mainstream	3.737077
16	YOUNG FAMILIES	Mainstream	3.724533
8	OLDER FAMILIES	Premium	3.717000
20	YOUNG SINGLES/COUPLES	Premium	3.665414
18	YOUNG SINGLES/COUPLES	Budget	3.657366

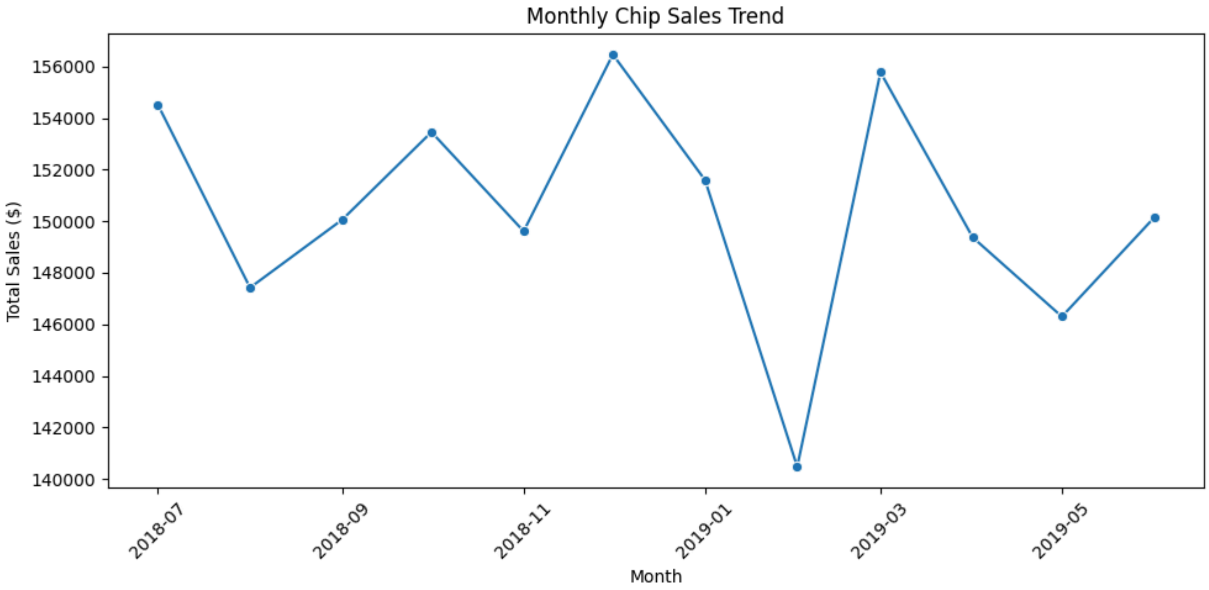
In [ ]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=full_df, x='LIFESTAGE', y='PRICE_PER_UNIT', hue='MEMBER_TYP
plt.title('Distribution of Price per Unit by Segment')
plt.xticks(rotation=45)
plt.ylabel('Price per Unit ($)')
plt.xlabel('Lifestage')
plt.tight_layout()
plt.show()
```



In [ ]:

```
#Time trends in chip sales
full_df['MONTH'] = full_df['DATE'].dt.to_period('M').dt.to_timestamp()
monthly_sales = full_df.groupby('MONTH')['TOT_SALES'].sum().reset_index()

plt.figure(figsize=(10, 5))
sns.lineplot(data=monthly_sales, x='MONTH', y='TOT_SALES', marker='o')
plt.title('Monthly Chip Sales Trend')
plt.xticks(rotation=45)
plt.ylabel('Total Sales ($)')
plt.xlabel('Month')
plt.tight_layout()
plt.show()
```



📅 Monthly Chip Sales Trend – Insight

The sales trend shows consistent chip demand throughout the year with two strong peaks in:

- **November 2018** — likely driven by festive season shopping
- **March 2019** — possibly aligned with major sporting events

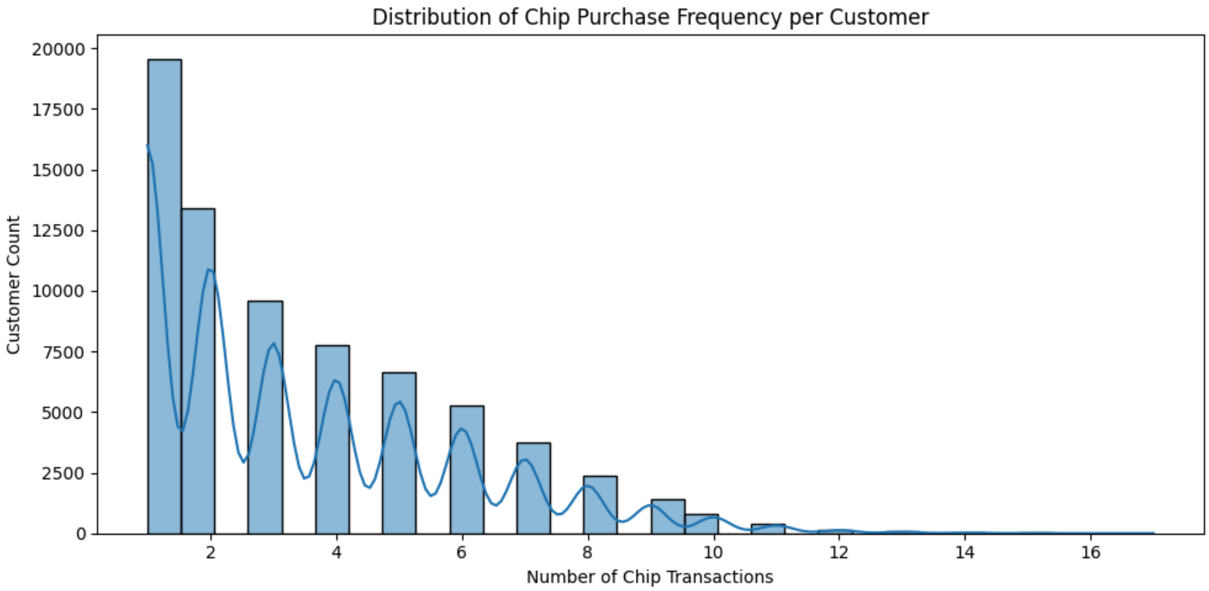
A notable dip in **February 2019** suggests a seasonal slowdown, potentially due to fewer days or post-holiday consumer fatigue.

Recommendation:

- Boost campaign spend and visibility during high-demand months like March and November.
- Launch counter-seasonal promotions or loyalty campaigns in February to smooth revenue and maintain customer engagement.

In [ ]:

```
#Customer Loyalty- Repeat Purchase Frequency
repeat_customers = full_df.groupby('LYLTY_CARD_NBR')['TXN_ID'].nunique().res

plt.figure(figsize=(10, 5))
sns.histplot(repeat_customers['NUM_TXNS'], bins=30, kde=True)
plt.title('Distribution of Chip Purchase Frequency per Customer')
plt.xlabel('Number of Chip Transactions')
plt.ylabel('Customer Count')
plt.tight_layout()
plt.show()
```



🔁 Repeat Purchase Frequency – Loyalty Insight

The purchase frequency histogram reveals that:

- **Most customers (~1 transaction)** are likely casual or trial buyers
- A **long tail of loyal customers (5+ transactions)** exists, offering high lifetime value

- Moderate-frequency buyers (2–4 transactions) present the biggest growth opportunity

Recommendation:

- **Re-engage one-time buyers** with “second purchase” incentives (e.g., email coupons or bundle discounts)
- **Reward high-frequency buyers** with exclusive loyalty perks or premium SKUs
- **Nurture mid-frequency buyers** to become loyalists through gamified promotions or personalized offers

🔍 INSIGHTS BY SEGMENT

💰 1. Total Sales

Mainstream Midage Singles/Couples and Mainstream Retirees are the biggest spenders on chips. Premium segments (especially older demographics) also contribute substantially.

💳 2. Average Spend per Transaction

Premium customers across all lifestages tend to spend more per visit — especially Young Singles/Couples. This indicates openness to higher-quality or branded chip products.

🧃 3. Quantity Purchased

Budget Midage and Retiree segments buy the most volume, possibly indicating bulk or price-conscious behavior. Suggests that these segments may respond well to value pack promotions.

💵 4. Price Sensitivity

Budget segments have the lowest price per unit, while Premium segments pay significantly more. Reinforces the opportunity to target Premium buyers with gourmet or limited-edition flavors.

🏷️ 5. Top Brands by Segment

Kettle, Smiths, and Doritos dominate sales across most segments. Kettle stands out among Premium customers, while Smiths appeals more broadly.

📌 Strategic Recommendations for Julia

- Target “Midage Singles/Couples” and “Retirees” — they are your highest revenue drivers.
- Offer premium product bundles to younger, higher-spending Premium segments.
- Promote large pack sizes to Budget segments in mid and older life stages.
- Use Kettle and Smiths as anchor brands — feature them prominently in promotions by segment.
- Consider product line extension for high-spend segments: e.g., health-conscious or artisan chips for Premium customers.