In [129…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from matplotlib.ticker import StrMethodFormatter
from matplotlib.colors import to_rgba
```

Transaction data

In [130…
```
transaction_data = pd.read_excel(r'C:\Users\BOSS\Desktop\DATA\python_folder\poject_1\QVI_transaction_data.xlsx'
```

DATE — Дата транзакції. Вказує на день, коли була здійснена покупка.

STORE_NBR — Номер магазину. Унікальний ідентифікатор магазину, де була здійснена транзакція.

LYLTY_CARD_NBR — Номер картки лояльності. Унікальний номер картки клієнта, зареєстрованого в програмі лояльності.

TXN_ID — Ідентифікатор транзакції. Унікальний номер для кожної покупки, який дозволяє відслідковувати конкретну операцію.

PROD_NBR — Номер продукту. Унікальний ідентифікатор товару, який був куплений.

PROD_NAME — Назва продукту. Назва товару, який було куплено.

PROD_QTY — Кількість продукту. Відображає кількість одиниць товару, яку купив клієнт.

TOT_SALES — Загальна сума продажу. Вартість транзакції для купленого товару або декількох товарів.

In [131…
```
#Виводимо типи даних
transaction_data.info()
#Відображаємо перші 5 рядків
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
```

Out[131]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

In [132…
```
# Змінюємо тип даних в стовпці DATE на datetime
transaction_data['DATE']=pd.to_datetime(transaction_data['DATE'], unit='D', origin='1899-12-30')
```

In [ ]:
```
## Базовий аналіз тексту
# Узагальнюючи окремі слова в назві товару, щоб переконатися що в нащопу наборі даних лише чіпси.
# #Створюєм дата фрейм з унікальними словами з стовпця 'PROD_NAME'
# unique_prod_name=transaction_data['PROD_NAME'].unique()
# prod_word=list(set([word for prod_name in unique_prod_name for word in prod_name.split()]))
# word_table= pd.DataFrame(prod_word, columns=['words'])

# # Функція(маска), яка видаляє всі  цифрами та спеціальними символами
# def contains_only_letters(w):
#     return bool(re.match(r'^[A-Za-z]+$',w))

# word_table=word_table[word_table['words'].apply(contains_only_letters)]

# #Кількість повторень кожного слова в 'PROD_NAME'
# counts=transaction_data['PROD_NAME'].str.split(expand=True).stack().value_counts()
# word_table['count']=word_table['words'].map(counts).fillna(0).astype(int)
# #Переглянемо перші топ20 слів, які повторюються в наборі даних
# word_table.sort_values('count', ascending=False).head(100)#[word_table['words'].str.contains('Salsa')]
```

Набір даних містить також категорію товарів 'Salsa'

In [134…
```
transaction_data['PROD_NAME'][transaction_data['PROD_NAME'].str.contains('Salsa')].unique()
```

Out[134]:
```
array(['Old El Paso Salsa   Dip Tomato Mild 300g',
       'Red Rock Deli SR    Salsa & Mzzrlla 150g',
       'Smiths Crinkle Cut  Tomato Salsa 150g',
       'Doritos Salsa       Medium 300g',
       'Old El Paso Salsa   Dip Chnky Tom Ht300g',
       'Woolworths Mild     Salsa 300g',
       'Old El Paso Salsa   Dip Tomato Med 300g',
       'Woolworths Medium   Salsa 300g', 'Doritos Salsa Mild  300g'],
      dtype=object)
```

In [154…
```
#Фільтруємо набір даних залишаємо тільки чіпси
transaction_data=transaction_data[~transaction_data['PROD_NAME'].str.contains('Salsa')]
```

In [136…
```
# Переглянем загальну статистику
transaction_data.describe()
```

Out[136]:

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

PROD_QTY максималь 200 одиниць. Якщо поглянути на середнє значення 1.9 та стандартне відхилення 0.65. Можна припустити що це "викид".
Розглянемо випадок, коли 200 пакети чіпсів купуються за одну операцію.

In [137…
```
#Фільтруєм набір даних, щоб побачити викид
transaction_data[transaction_data['PROD_QTY']==200]
```

Out[137]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

Маємо дві транзакції і обидві ці транзакції були здійснені одним клієнтом.
Поглянемо на інші транзакції цього клієнта.

In [138…
```
transaction_data[transaction_data['LYLTY_CARD_NBR']==226000]
```

Out[138]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

Має лише дві транзакції від даного клієнта, що є не типовим для нашого набору даних. Тому вилумо їх з подальшого аналізу.

In [139…
```
#Фільтруємо викиди в наборі даних.
transaction_data=transaction_data[transaction_data['LYLTY_CARD_NBR']!=226000]
```

In [140…
```
# Перевіряємо пропущенні дати
start = transaction_data['DATE'].min()
end = transaction_data['DATE'].max()
date_line= pd.date_range(start=start, end=end)
miss_date= date_line.difference(transaction_data['DATE'])
print(miss_date)
```

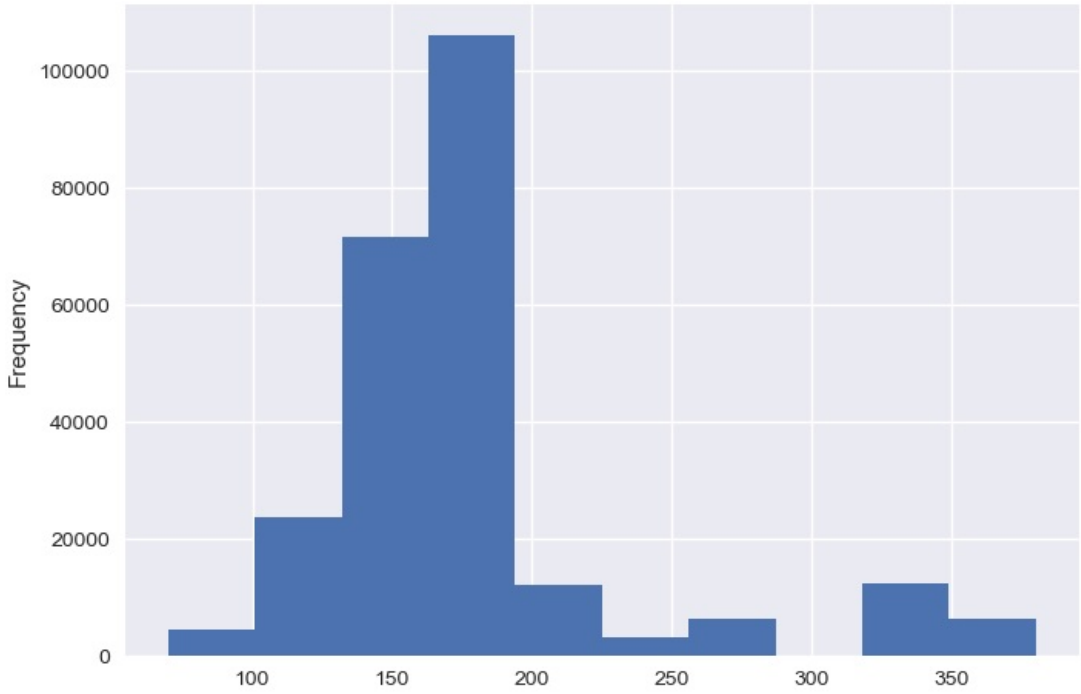
```
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq=None)
```

In [141…
```
grouped = transaction_data[['DATE', 'PROD_QTY']].groupby('DATE').sum() #кількість проданих чіпсів кожного дня

#Додаєм пропущену дату та даємо значення "0"
grouped.loc[miss_date[0]]=[0]
grouped=grouped.sort_index()

#Будуємо графік
plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots(figsize=(15,5))
ax.plot(grouped, label='Кількість проданих упаковок')
plt.title('Динаміка продажу чіпсів за рік', size=14)
plt.legend(loc='upper right', bbox_to_anchor=(1.21, 1))
ax.annotate( 'Пропущена дата', xy=((miss_date[0]), 0), xytext=((miss_date[0])+pd.Timedelta(days=30), 1), arrowp
plt.show()
```

Бачимо, що зростання продажів відбувається напередодні Різдва на сам день Різдва немає розпродажів. Можемо припустити, що це через закриття магазинів Різдвяний день.

In [142…
```
#Визначаємо розмір упаковки
# Додаєм стовпець 'PACKET_SIZE'
transaction_data['PACKET_SIZE']=transaction_data['PROD_NAME'].str.extract(r'(\d+)').astype(int)
#Гістограма розподілу 'PACKET_SIZE'
transaction_data['PACKET_SIZE'].plot(kind='hist')
```

Out[142]:
```
<Axes: ylabel='Frequency'>
```



In [143…
```
#Відокремлюємо перше слово в назві товару як бренд
#Додаєм стопець'BRAND'
transaction_data['BRAND']=transaction_data['PROD_NAME'].str.extract(r'(\S+)')
#Замінюємо 'Red' ( Red Rock Deli) на RRD
transaction_data['BRAND']=transaction_data['BRAND'].str.replace('Red', 'RRD')
#Замінюємо 'WW' на  'Woolworths'
transaction_data['BRAND']=transaction_data['BRAND'].str.replace('WW', 'Woolworths')
#Замінюємо 'Natural' (Natural Chip Compny) на 'NCC'
transaction_data['BRAND']=transaction_data['BRAND'].str.replace('Natural', 'NCC')
#Замінюємо 'Smith'  на 'Smiths'
transaction_data['BRAND']=transaction_data['BRAND'].str.replace(r'(\bSmith\b)', 'Smiths', regex= True)
#Замінюємо  'Snbts ' на 'Sunbites'
transaction_data['BRAND']=transaction_data['BRAND'].str.replace('Snbts ', 'Sunbites')
```

Customer data

In [144…
```
customer=pd.read_csv(r'C:\Users\BOSS\Desktop\DATA\python_folder\poject_1\QVI_purchase_behaviour.csv')
```

In [145…
```
customer.info()
customer.head()
```

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

Out[145]:

	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

Дані customer не потребують очищення

In [ ]:
```
# Обєднуємо ransaction_data та customer
df=pd.merge(transaction_data, customer , on='LYLTY_CARD_NBR', how='left')
df.info()
```

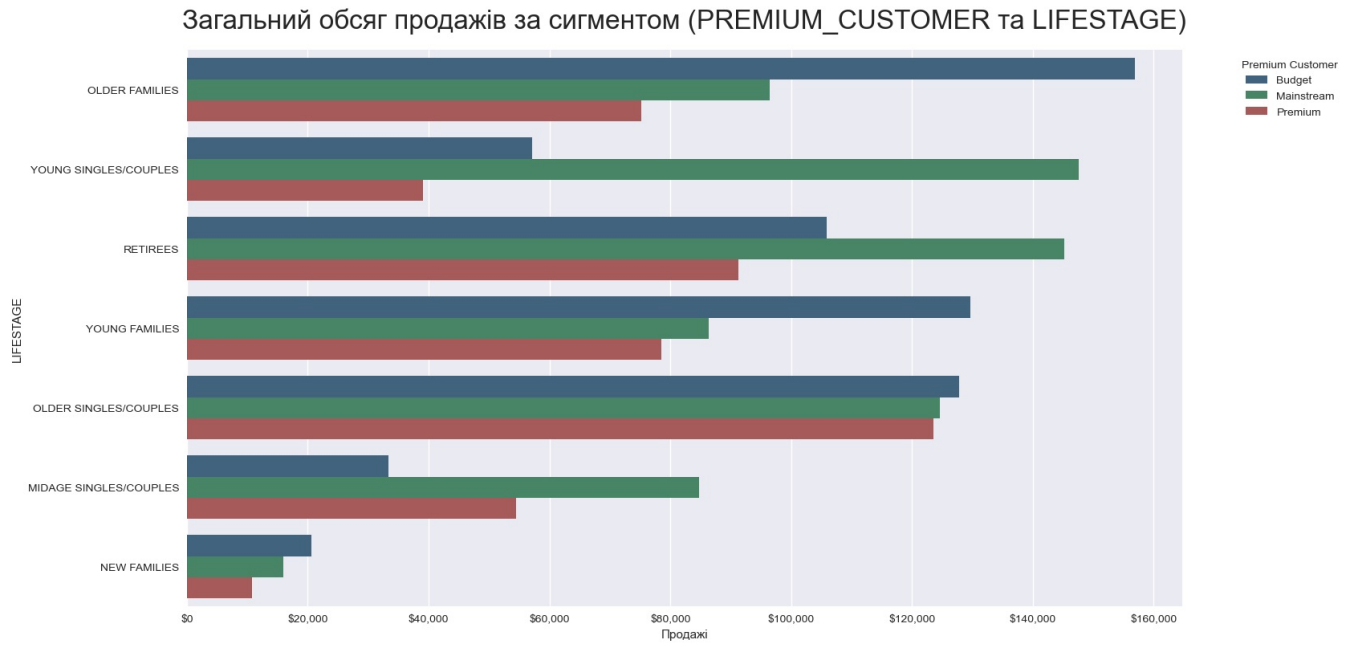
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246740 entries, 0 to 246739
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   DATE              246740 non-null  datetime64[ns]
 1   STORE_NBR         246740 non-null  int64         
 2   LYLTY_CARD_NBR    246740 non-null  int64         
 3   TXN_ID            246740 non-null  int64         
 4   PROD_NBR          246740 non-null  int64         
 5   PROD_NAME         246740 non-null  object        
 6   PROD_QTY          246740 non-null  int64         
 7   TOT_SALES         246740 non-null  float64       
 8   PACKET_SIZE       246740 non-null  int32         
 9   BRAND             246740 non-null  object        
 10  LIFESTAGE         246740 non-null  object        
 11  PREMIUM_CUSTOMER  246740 non-null  object        
dtypes: datetime64[ns](1), float64(1), int32(1), int64(5), object(4)
memory usage: 21.6+ MB
```

In [147…
```
# Збереження df( transaction_data + customer)
# df.to_csv(r'C:\Users\BOSS\Desktop\DATA\python_folder\poject_1\QVI_data.csv')
```

Аналіз

- Який сегмент клієнтів витрачає найбільше на чіпси (враховуючи загальний обсяг продажів), і які характеристики цих клієнтів за категоріями LIFESTAGE та PREMIUM_CUSTOMER?
- Скільки клієнтів належить до кожного сегмента?
- Скільки упаковок чіпсів в середньому купує один клієнт у кожному сегменті?
- Яка середня ціна упаковки чіпсів у кожному сегменті?

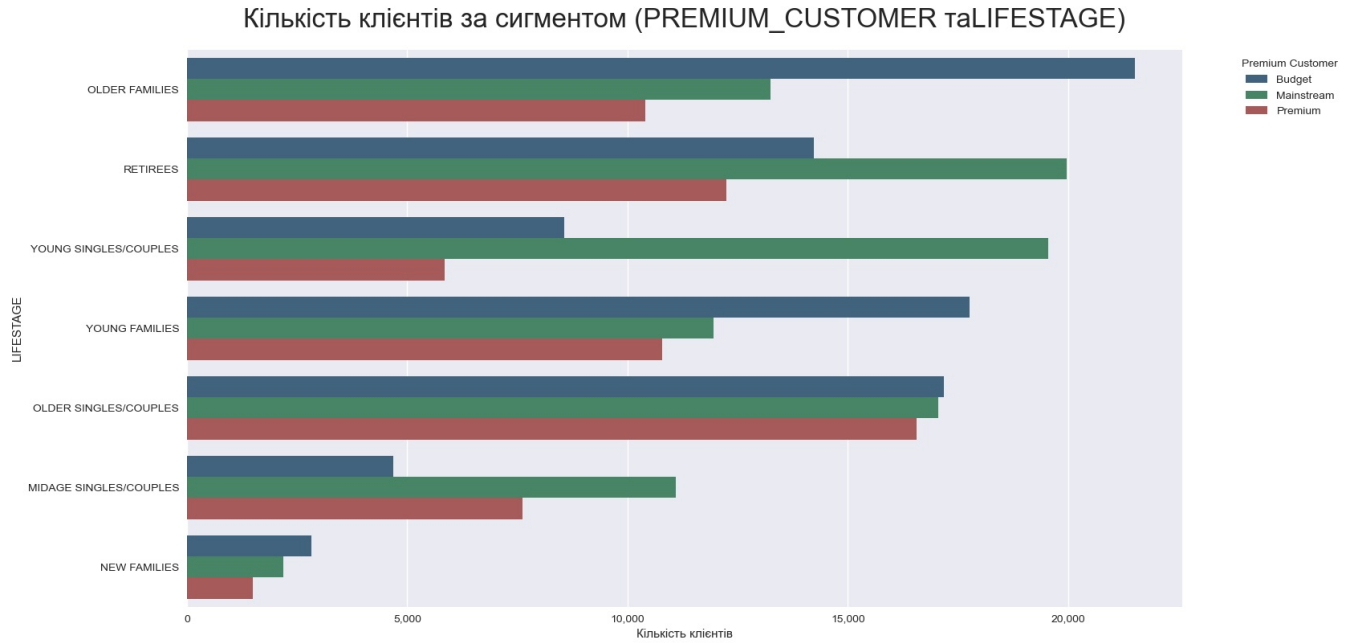
In [ ]:
```
#Групуван даних
grouped_df=df.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE'])['TOT_SALES'].sum().reset_index().sort_values(by='TOT_S

# Визначення кольорів для кожної групи PREMIUM_CUSTOMER
custom_palette = {'Budget': '#386487','Mainstream': '#3d8f66','Premium': '#b34d4d'}

# Створення графіка
plt.figure(figsize=(16, 9))
g=sns.barplot(data=grouped_df, y='LIFESTAGE', x='TOT_SALES', hue='PREMIUM_CUSTOMER', palette=custom_palette)

# Налаштування графіка
plt.title('Загальний обсяг продажів за сигментом (PREMIUM_CUSTOMER та LIFESTAGE)', size=24, pad=20)
plt.xlabel('Продажі')
plt.ylabel('LIFESTAGE')
plt.legend(title='Premium Customer', bbox_to_anchor=(1.05, 1), loc='upper left')  # Позиція легенди
g.xaxis.set_major_formatter(StrMethodFormatter('${x:,.0f}'))
plt.show()
```



Продажі надходять в основному від Budget - older families, Mainstream - young singles/couples, та Mainstream - retirees.

In [159…

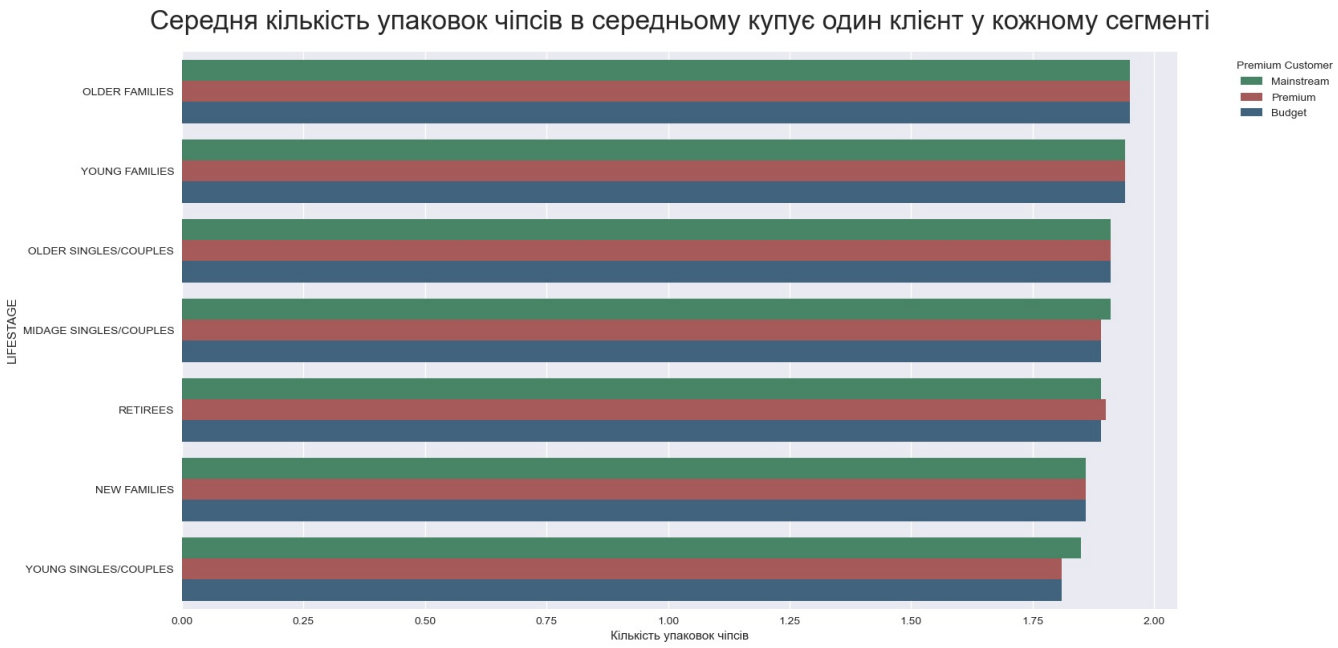
```
# Групування даних
customer_group_count=df.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE'])['LYLTY_CARD_NBR'].count().reset_index().sort

# Побудова графіка
plt.figure(figsize=(16,9))
g=sns.barplot(data= customer_group_count, y='LIFESTAGE', x='LYLTY_CARD_NBR', hue='PREMIUM_CUSTOMER',palette=cus

# Налаштування графіка
plt.title('Кількість клієнтів за сигментом (PREMIUM_CUSTOMER таLIFESTAGE)', size=24, pad=20)
plt.xlabel('Кількість клієнтів')
plt.ylabel('LIFESTAGE')
plt.legend(title='Premium Customer', bbox_to_anchor=(1.05, 1), loc='upper left')
g.xaxis.set_major_formatter(StrMethodFormatter('{x:,.0f}'))
plt.show()
```



Перевіремо скільки в середньому купують упаковок чіпсів з сегментом покупця.

In [160…

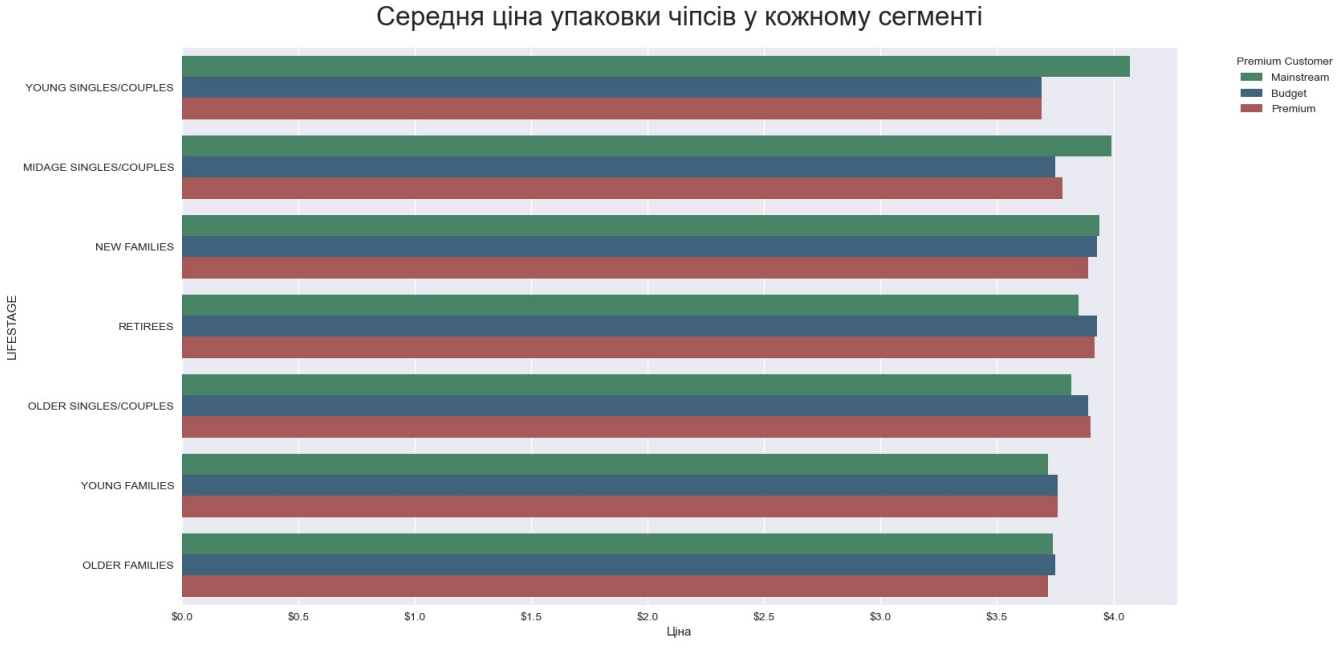
```
# Групування даних
customer_group_avg=df.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE'])['PROD_QTY'].mean().reset_index().sort_values(b
customer_group_avg['PROD_QTY']=customer_group_avg['PROD_QTY'].round(2)

# Побудова графіка
plt.figure(figsize=(16,9))
g=sns.barplot(data= customer_group_avg, y='LIFESTAGE', x='PROD_QTY', hue='PREMIUM_CUSTOMER', palette=custom_pal

# Налаштування графіка
plt.title('Середня кількість упаковок чіпсів в середньому купує один клієнт у кожному сегменті', size=24, pad=2
plt.xlabel('Кількість упаковок чіпсів')
plt.ylabel('LIFESTAGE')
plt.legend(title='Premium Customer', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



Older families та young families купують найбільше чіпсів

In [161…

```python
# Групування данихю
price_group=df.groupby(['PREMIUM_CUSTOMER', 'LIFESTAGE'])[['TOT_SALES','PROD_QTY']].sum()
price_group['AVG_Price']= (price_group['TOT_SALES']/price_group['PROD_QTY']).round(2) # Середня ціна продажу
price_group=price_group.reset_index().sort_values(by='AVG_Price', ascending=False)

# Побудова графіка
plt.figure(figsize=(16,9))
g=sns.barplot(data= price_group, y='LIFESTAGE', x='AVG_Price', hue='PREMIUM_CUSTOMER',palette=custom_palette)

# Налаштування графіка
plt.title('Середня ціна упаковки чіпсів у кожному сегменті', size=24, pad=20)
plt.xlabel('Ціна')
plt.ylabel('LIFESTAGE')
plt.legend(title='Premium Customer', bbox_to_anchor=(1.05, 1), loc='upper left')
g.xaxis.set_major_formatter(StrMethodFormatter('${x:.1f}'))
plt.show()
```



Групи Mainstream Young Singles/Couples та Midage Single/Couples готові витрачати більше на дорогі чіпси. Перевіримо, чи існує статистично значуща різниця між цими двома групами та всіма іншими покупцями.

Використаємо t-тест для порівняння середніх значень витрат, та перепірки нашого припущення.

- H0 (нульова гіпотеза): Середні витрати на чіпси для груп Mainstream Young Singles/Couples та Midage Single/Couples не відрізняються від середніх витрат решти груп.
- H1 (альтернативна гіпотеза): Середні витрати на чіпси для груп Mainstream Young Singles/Couples та Midage Single/Couples статистично значущо відрізняються від середніх витрат решти груп..

In [162…

```python
from scipy import stats
df['SALES_PER_QTY']= df['TOT_SALES']/df['PROD_QTY']
```

```
#група 1 Mainstream Young Singles/Couples + Mainstream Midage Singles/Cuopeles
group_1=df[(df['PREMIUM_CUSTOMER'] == 'Mainstream') & (df['LIFESTAGE'].isin(['YOUNG SINGLES/COUPLES', 'MIDAGE S

# група 2
group_2=df[~(df['PREMIUM_CUSTOMER'] == 'Mainstream') & (df['LIFESTAGE'].isin(['YOUNG SINGLES/COUPLES', 'MIDAGE

# Обчислення t та p
t_stat, p_value = stats.ttest_ind(group_1,group_2)
# Результат
print(f"t-stat : {t_stat}, p-value: {p_value}")
```

```
t-stat : 37.83196107667815, p-value: 2.235645611549355e-309
```

p-value < 0.05 та t-stat = 37.83 означає, що є статистично значуща різниця між середніми значеннями витрат двох груп, і можна стверджувати, що ці групи мають різні середні витрати на чіпси. Отже, Mainstream Young Singles/Couples та Midage Single/Couples дійсно купувати дорощі товари категорії чіпси.

Підсумок

Сегмент Budget Older Families приносить найбільший дохід від продажів чіпсів, оскільки вони купують найбільше упаковок. Однак, ці споживачі витрачають найменше на кожну упаковку, віддаючи перевагу бюджетним брендам.

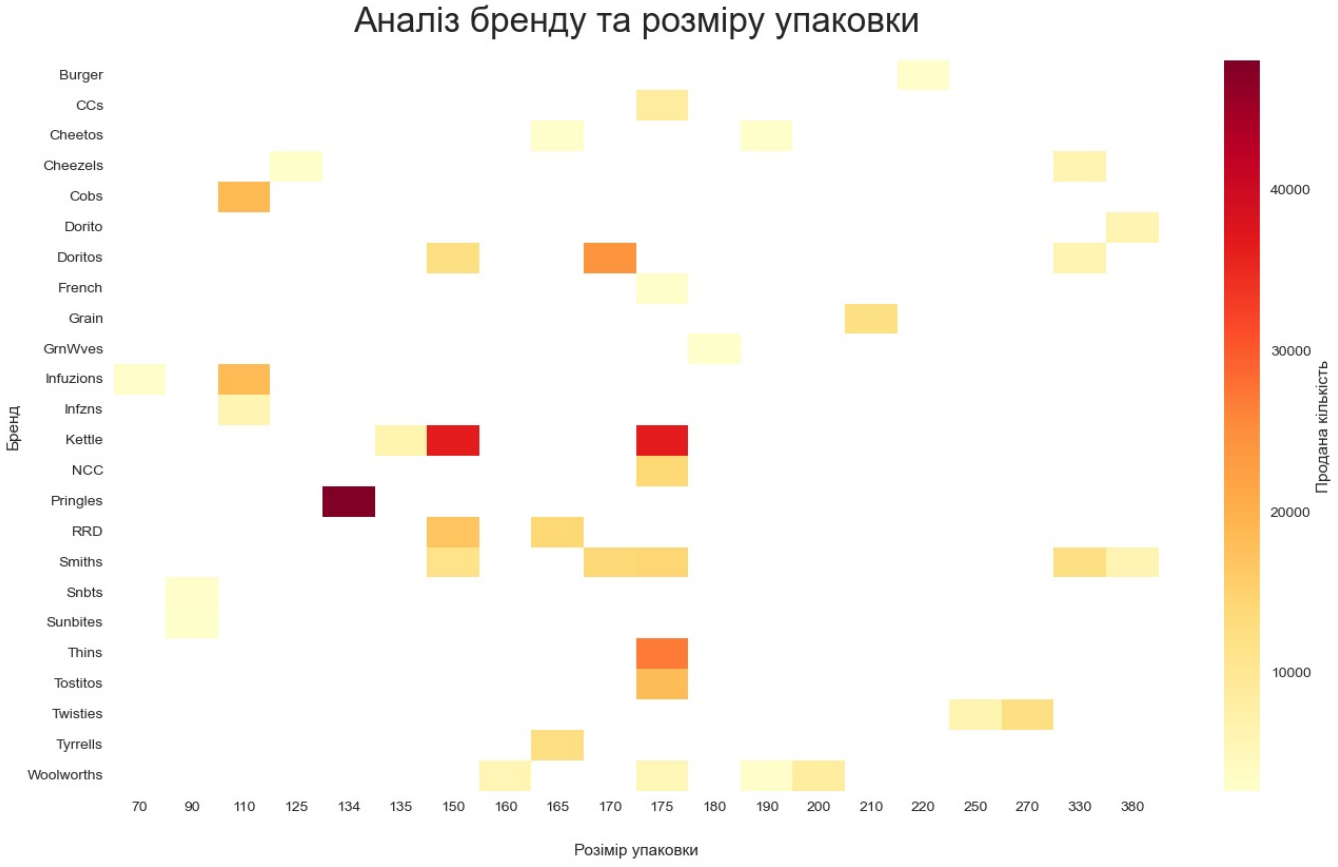
Сегмент Mainstream Young Singles/Couples також робить значний внесок у дохід, купуючи дорожчі упаковки чіпсів, що збільшує витрати на кожну одиницю.

Сегмент Mainstream Retirees приносить значну частку доходу не через кількість покупок, а через відносно невелику популяцію в сегменті, що призводить до вищої середньої вартості на покупця.

Сегмент Mainstream Midage Singles/Couples купує багато упаковок чіпсів і обирає дорожчі бренди, однак мала чисельність цього сегменту призводить до нижчої частки доходу загалом.

Розглянемо, які бренди та розміри упаковок користуються найбільшою популярністю серед споживачів.

In [ ]:
```
size_pivot = df.pivot_table(index='BRAND', columns='PACKET_SIZE', values='PROD_QTY', aggfunc='sum')
fig, ax = plt.subplots(figsize=(16,9))
sns.heatmap(size_pivot, cmap='YlOrRd', annot=False, ax=ax, cbar_kws={'label':'Продана кількість'})
ax.set_facecolor('white')
plt.title('Аналіз бренду та розміру упаковки', size=24, pad=20)
plt.ylabel('Бренд')
plt.xlabel('Розімір упаковки', labelpad=20)
plt.show()
```



Найбільшою популярністю користуються чіпси Pringles в упаковці 134 г.
Також покупці віддають перевагу чіпсам Kettle в упаковках 150 г та 175 г, тоді як упаковка 135 г виявилася менш популярною —

ймовірно, споживачі схиляються до вибору більшого об’єму.

Крім того, як популярні бренди можна виділити Thins у форматі 175 г та Doritos в упаковці 170 г.