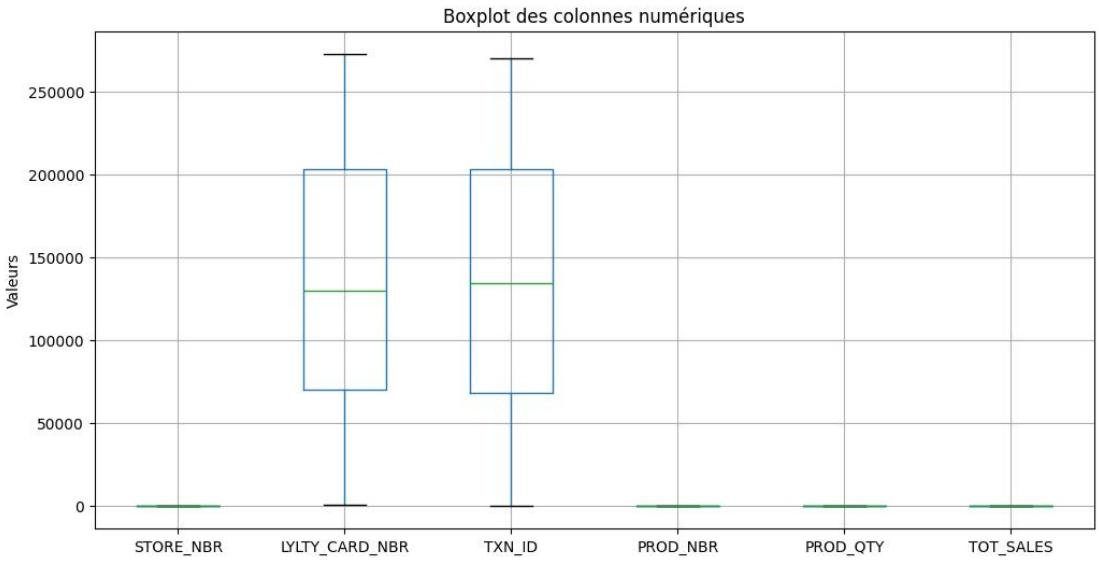
```
#Importation des bibliothèques nécessaires
import pandas as pd

df1 = pd.read_excel('QVI_transaction_data.xlsx')
df2 = pd.read_csv('QVI_purchase_behaviour.csv')

df = pd.merge(df1, df2, left_on='LYLTY_CARD_NBR', right_on='LYLTY_CARD_NBR', suffixes=('_trans', '_behav'))

# Affiche les 5 premières lignes du dataset
df.head()
# Affichage des informations du DataFrame fusionné
df.info()
df.shape
df.describe(include='all')
# Gestion des valeurs manquantes

# Afficher le nombre de valeurs manquantes par colonne
print(df.isnull().sum())

# Exemple de traitement : suppression ou imputation
# Ici, on choisit d'imputer les valeurs manquantes numériques par la médiane et les catégorielles par le mode

for col in df.columns:
    if df[col].isnull().any():
        if df[col].dtype in ['float64', 'int64']:
            median_value = df[col].median()
            df[col].fillna(median_value, inplace=True)
        else:
            mode_value = df[col].mode()[0]
            df[col].fillna(mode_value, inplace=True)

# Vérification après traitement
print(df.isnull().sum())
import matplotlib.pyplot as plt

# Afficher un boxplot pour chaque colonne numérique du DataFrame df
df[numeric_cols].boxplot(figsize=(12, 6))
plt.title("Boxplot des colonnes numériques")
plt.ylabel("Valeurs")
plt.show()
```

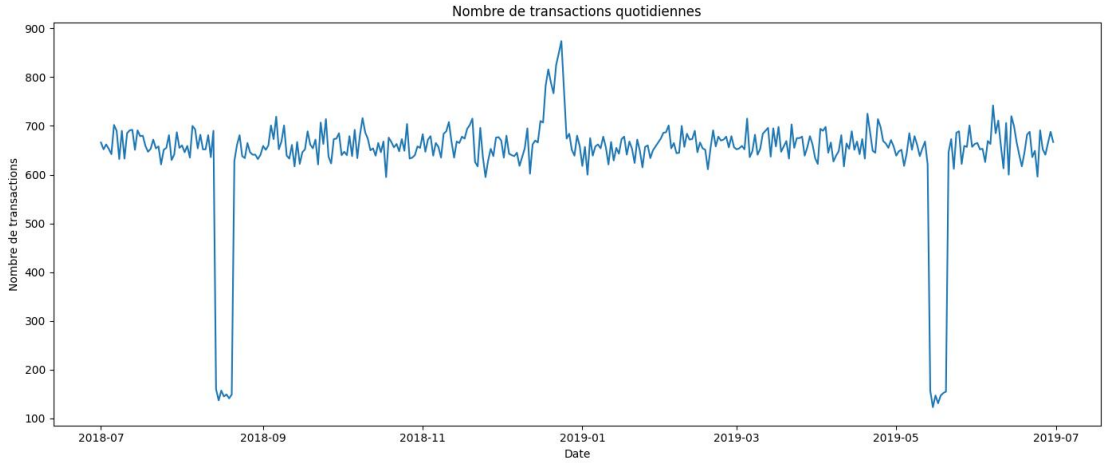



```
# Suppression des outliers pour toutes les colonnes numériques compatibles
df_no_outliers = df.copy()
for col in numeric_cols:
    Q1 = df_no_outliers[col].quantile(0.25)
    Q3 = df_no_outliers[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df_no_outliers = df_no_outliers[(df_no_outliers[col] >= lower_bound) & (df_no_outliers[col] <= upper_bound)]

df_no_outliers.reset_index(drop=True, inplace=True)
df = df_no_outliers
# Vérification et correction des formats de données pour les colonnes importantes

# Vérifier le type de la colonne 'DATE'
if not pd.api.types.is_datetime64_any_dtype(df['DATE']):
    df['DATE'] = pd.to_datetime(df['DATE'], errors='coerce')

# Vérifier les types numériques
numeric_columns = ['STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR', 'PROD_QTY', 'TOT_SALES', 'PACK_SIZE']
for col in numeric_columns:
    if not pd.api.types.is_numeric_dtype(df[col]):
        df[col] = pd.to_numeric(df[col], errors='coerce')

# Vérifier les types de colonnes catégorielles
categorical_columns = ['PROD_NAME', 'LIFESTAGE', 'PREMIUM_CUSTOMER', 'BRAND']
for col in categorical_columns:
    if not pd.api.types.is_object_dtype(df[col]):
```

```
    df[col] = df[col].astype(str)

# Afficher les types de données après correction
print(df.dtypes)
import re

# Création de nouvelles features à partir des données existantes

# Extraire la taille du paquet (pack size) à partir du nom du produit si ce n'est pas déjà fait

def extract_pack_size(prod_name):
  match = re.search(r'(\d+)(g|G)', prod_name)
  if match:
    return float(match.group(1))
  return None

if 'PACK_SIZE' not in df.columns or df['PACK_SIZE'].isnull().any():
  df['PACK_SIZE'] = df['PROD_NAME'].apply(extract_pack_size)

# Extraire le nom de la marque à partir du nom du produit si ce n'est pas déjà fait
def extract_brand(prod_name):
  return prod_name.split()[0] if isinstance(prod_name, str) else None

if 'BRAND' not in df.columns or df['BRAND'].isnull().any():
  df['BRAND'] = df['PROD_NAME'].apply(extract_brand)

# Afficher les premières lignes pour vérifier
df[['PROD_NAME', 'PACK_SIZE', 'BRAND']].head()
# 1. Dépense totale par client
Calcul de la dépense totale pour chaque client (`LYLTY_CARD_NBR`).
depense_totale_client = df.groupby('LYLTY_CARD_NBR')['TOT_SALES'].sum().reset_index()
depense_totale_client.rename(columns={'TOT_SALES': 'DEPENSE_TOTALE'}, inplace=True)
depense_totale_client.head()
# 2. Nombre total de transactions par client
Nombre de transactions réalisées par chaque client.
transactions_par_client = df.groupby('LYLTY_CARD_NBR')['TXN_ID'].nunique().reset_index()
transactions_par_client.rename(columns={'TXN_ID': 'NB_TRANSACTIONS'}, inplace=True)
transactions_par_client.head()
# 3. Dépense moyenne par transaction
Calcul de la dépense moyenne par transaction pour chaque client.
depense_moyenne_transaction = depense_totale_client.merge(transactions_par_client,
on='LYLTY_CARD_NBR')
depense_moyenne_transaction['DEPENSE_MOYENNE_PAR_TRANSACTION'] =
depense_moyenne_transaction['DEPENSE_TOTALE'] /
```

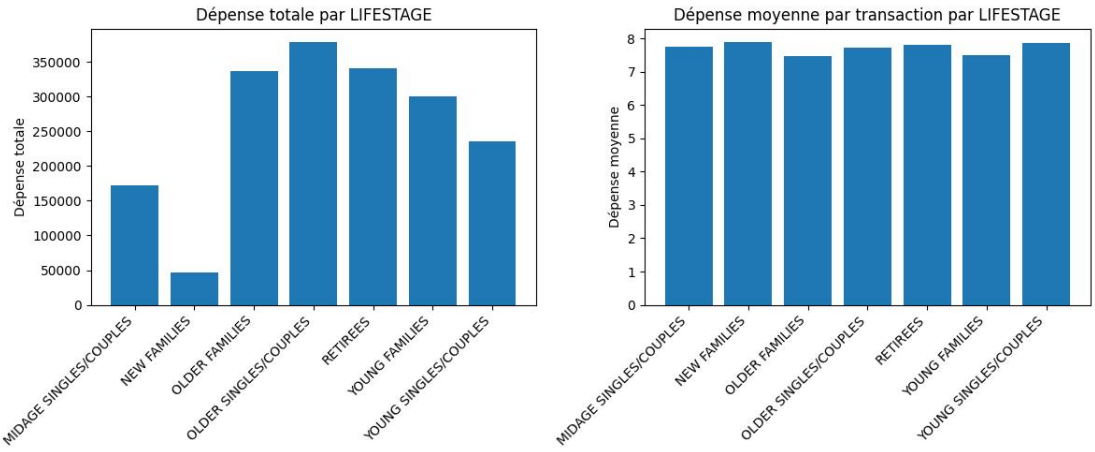
```
depense_moyenne_transaction['NB_TRANSACTIONS']
depense_moyenne_transaction[['LYLTY_CARD_NBR',
'DEPENSE_MOYENNE_PAR_TRANSACTION']].head()
# 4. Quantité totale achetée par client
Somme des quantités achetées par chaque client.
quantite_totale_client = df.groupby('LYLTY_CARD_NBR')['PROD_QTY'].sum().reset_index()
quantite_totale_client.rename(columns={'PROD_QTY': 'QTE_TOTALE'}, inplace=True)
quantite_totale_client.head()
# 5. Taille moyenne des paquets achetés
Taille moyenne des paquets achetés par client.
taille_moyenne_paquet = df.groupby('LYLTY_CARD_NBR')['PACK_SIZE'].mean().reset_index()
taille_moyenne_paquet.rename(columns={'PACK_SIZE': 'TAILLE_MOYENNE_PAQUET'},
inplace=True)
taille_moyenne_paquet.head()
# 6. Marque préférée par client
Marque la plus achetée par chaque client.
marque_preferee = df.groupby(['LYLTY_CARD_NBR', 'BRAND'])['PROD_QTY'].sum().reset_index()
marque_preferee =
marque_preferee.loc[marque_preferee.groupby('LYLTY_CARD_NBR')['PROD_QTY'].idxmax()]
marque_preferee = marque_preferee[['LYLTY_CARD_NBR', 'BRAND']]
marque_preferee.head()
# 7. Fréquence d'achat (nombre moyen de jours entre les achats)
Calcul du nombre moyen de jours entre les achats pour chaque client.
df_sorted = df.sort_values(['LYLTY_CARD_NBR', 'DATE'])
frequence_achat = df_sorted.groupby('LYLTY_CARD_NBR')['DATE'].apply(lambda x:
x.diff().dt.days.mean()).reset_index()
frequence_achat.rename(columns={'DATE': 'FREQUENCE_ACHAT_JOURS'}, inplace=True)
frequence_achat.head()
# 8. Dépense totale par segment (LIFESTAGE, PREMIUM_CUSTOMER)
Somme des dépenses pour chaque segment.
depense_par_segment = df.groupby(['LIFESTAGE',
'PREMIUM_CUSTOMER'])['TOT_SALES'].sum().reset_index()
depense_par_segment.rename(columns={'TOT_SALES': 'DEPENSE_TOTALE'}, inplace=True)
depense_par_segment.head()
# 9. Dépense moyenne par segment
Dépense moyenne par transaction pour chaque segment.
depense_moyenne_segment = df.groupby(['LIFESTAGE',
'PREMIUM_CUSTOMER'])['TOT_SALES'].mean().reset_index()
depense_moyenne_segment.rename(columns={'TOT_SALES': 'DEPENSE_MOYENNE'},
inplace=True)
depense_moyenne_segment.head()
# 10. Produit le plus acheté par segment
Produit le plus acheté (en quantité) pour chaque segment.
produit_plus_achete = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER',
```

```
'PROD_NAME'])['PROD_QTY'].sum().reset_index()
idx = produit_plus_achete.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PROD_QTY'].idxmax()
produit_plus_achete = produit_plus_achete.loc[idx][['LIFESTAGE', 'PREMIUM_CUSTOMER',
'PROD_NAME', 'PROD_QTY']]
produit_plus_achete.head()
# Analyse du nombre de transactions par jour

transactions_quotidiennes = df.groupby('DATE')['TXN_ID'].nunique().reset_index()
transactions_quotidiennes.rename(columns={'TXN_ID': 'NB_TRANSACTIONS'}, inplace=True)

# Affichage des premières lignes
print(transactions_quotidiennes.head())

# Visualisation de l'évolution des transactions quotidiennes
plt.figure(figsize=(14, 6))
plt.plot(transactions_quotidiennes['DATE'], transactions_quotidiennes['NB_TRANSACTIONS'])
plt.title("Nombre de transactions quotidiennes")
plt.xlabel("Date")
plt.ylabel("Nombre de transactions")
plt.tight_layout()
plt.show()
```



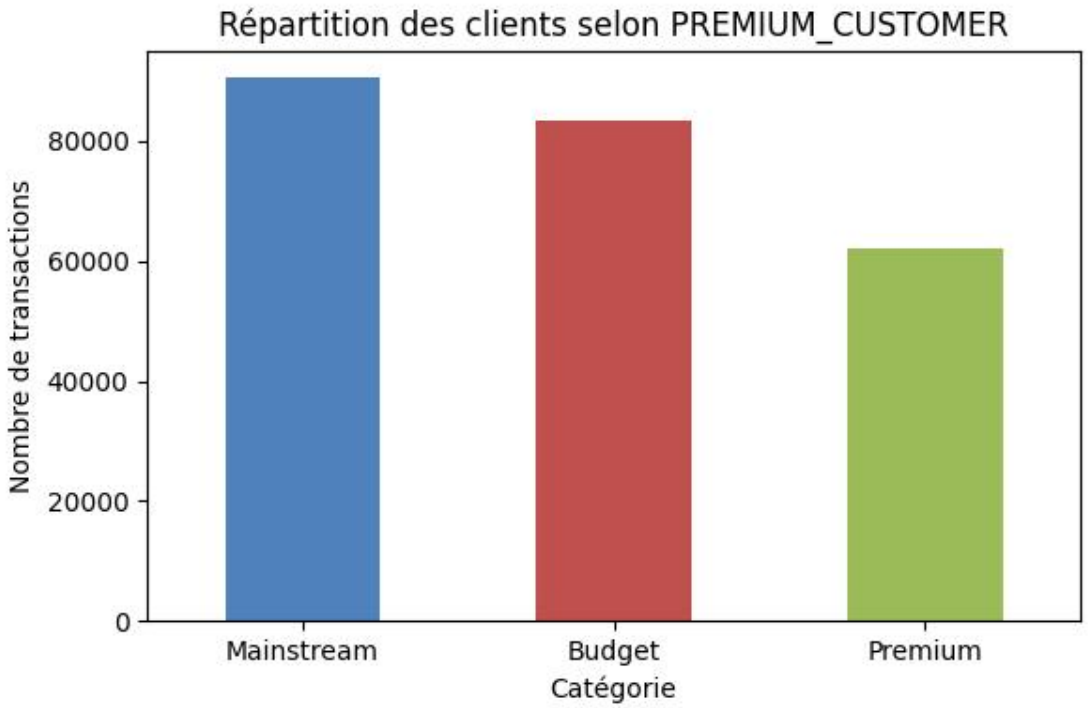

```
# Analyse de la dépense totale et moyenne par segment LIFESTAGE

# Dépense totale par LIFESTAGE
depense_totale_lifestage = df.groupby('LIFESTAGE')['TOT_SALES'].sum().reset_index()
depense_totale_lifestage.rename(columns={'TOT_SALES': 'DEPENSE_TOTALE'}, inplace=True)
print("Dépense totale par LIFESTAGE :")
print(depense_totale_lifestage)

# Dépense moyenne par transaction pour chaque LIFESTAGE
depense_moyenne_lifestage = df.groupby('LIFESTAGE')['TOT_SALES'].mean().reset_index()
depense_moyenne_lifestage.rename(columns={'TOT_SALES': 'DEPENSE_MOYENNE'},
```

```
inplace=True)
print("\nDépense moyenne par transaction par LIFESTAGE :")
print(depense_moyenne_lifestage)

# Visualisation
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.bar(depense_totale_lifestage['LIFESTAGE'], depense_totale_lifestage['DEPENSE_TOTALE'])
plt.title('Dépense totale par LIFESTAGE')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Dépense totale')

plt.subplot(1, 2, 2)
plt.bar(depense_moyenne_lifestage['LIFESTAGE'],
depense_moyenne_lifestage['DEPENSE_MOYENNE'])
plt.title('Dépense moyenne par transaction par LIFESTAGE')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Dépense moyenne')

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




```
# Analyse de la répartition des clients selon le statut PREMIUM_CUSTOMER

# Comptage du nombre de clients par catégorie PREMIUM_CUSTOMER
premium_counts = df['PREMIUM_CUSTOMER'].value_counts()
print("Nombre de clients par catégorie PREMIUM_CUSTOMER :")
print(premium_counts)

# Visualisation de la répartition
plt.figure(figsize=(6, 4))
premium_counts.plot(kind='bar', color=['#4F81BD', '#C0504D', '#9BBB59'])
```

```
plt.title("Répartition des clients selon PREMIUM_CUSTOMER")
plt.xlabel("Catégorie")
plt.ylabel("Nombre de transactions")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



3. Analyse des segments clients

Cette section analyse les segments clients définis par les variables **LIFESTAGE** et **PREMIUM_CUSTOMER**.

Indicateurs analysés :
- Total des ventes par segment
- Nombre de clients par segment
- Quantité moyenne de chips achetée par client
- Prix moyen par unité par segment

Résultats principaux :
- Les ventes les plus élevées proviennent de :
  - Budget – Older Families
  - Mainstream – Young Singles/Couples
  - Mainstream – Retirees
- Les jeunes couples mainstream sont nombreux et dépensent plus par unité.
- Le prix moyen par unité est significativement plus élevé pour les segments Mainstream jeunes et midage que pour Budget ou Premium, ce qui suggère des comportements d'achat impulsifs.

Total des ventes par segment

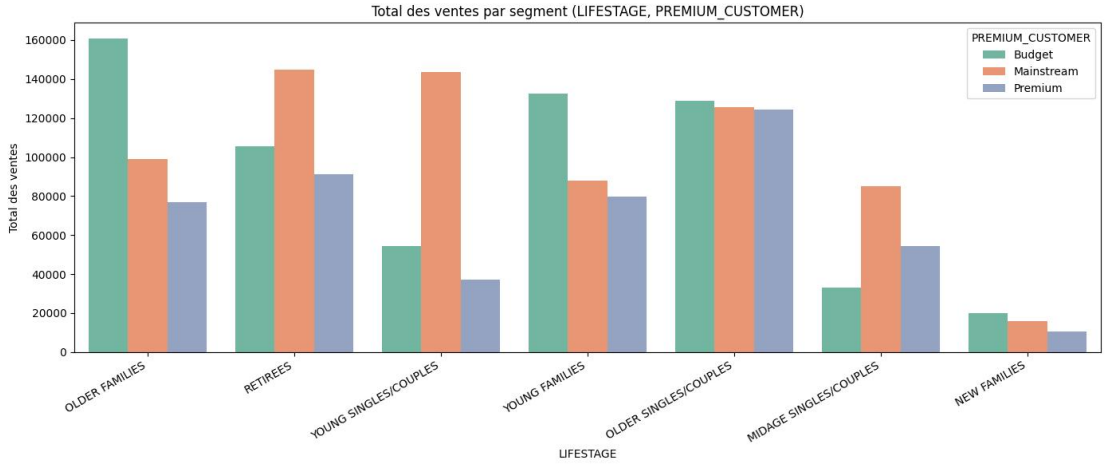
```
ventes_par_segment = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES'].sum().reset_index()
ventes_par_segment = ventes_par_segment.sort_values('TOT_SALES', ascending=False)
ventes_par_segment.head(10)
# Nombre de clients par segment
clients_par_segment = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['LYLTY_CARD_NBR'].nunique().reset_index()
clients_par_segment.rename(columns={'LYLTY_CARD_NBR': 'NB_CLIENTS'}, inplace=True)
clients_par_segment = clients_par_segment.sort_values('NB_CLIENTS', ascending=False)
clients_par_segment.head(10)
# Quantité moyenne de chips achetée par client et par segment
quantite_par_client_segment = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER', 'LYLTY_CARD_NBR'])['PROD_QTY'].sum().reset_index()
quantite_moyenne_segment = quantite_par_client_segment.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['PROD_QTY'].mean().reset_index()
quantite_moyenne_segment.rename(columns={'PROD_QTY': 'QTE_MOYENNE_PAR_CLIENT'}, inplace=True)
quantite_moyenne_segment = quantite_moyenne_segment.sort_values('QTE_MOYENNE_PAR_CLIENT', ascending=False)
quantite_moyenne_segment.head(10)
# Prix moyen par unité par segment
prix_moyen_unite_segment = df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).apply(lambda x: x['TOT_SALES'].sum() / x['PROD_QTY'].sum()).reset_index(name='PRIX_MOYEN_PAR_UNITE')
prix_moyen_unite_segment = prix_moyen_unite_segment.sort_values('PRIX_MOYEN_PAR_UNITE', ascending=False)
prix_moyen_unite_segment.head(10)
## Visualisations des indicateurs par segment
```

Les graphiques suivants illustrent les principaux indicateurs analysés pour chaque segment (LIFESTAGE, PREMIUM_CUSTOMER) :

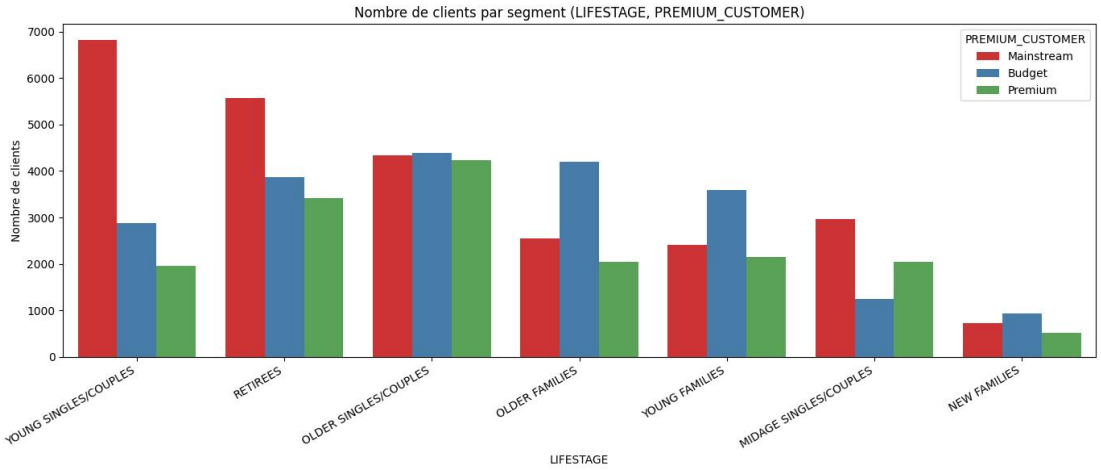
- Total des ventes
- Nombre de clients
- Quantité moyenne de chips achetée par client
- Prix moyen par unité

```
# Visualisation du total des ventes par segment
import seaborn as sns
plt.figure(figsize=(14, 6))
sns.barplot(
    data=ventes_par_segment,
    x='LIFESTAGE',
    y='TOT_SALES',
    hue='PREMIUM_CUSTOMER',
    palette='Set2'
)
```

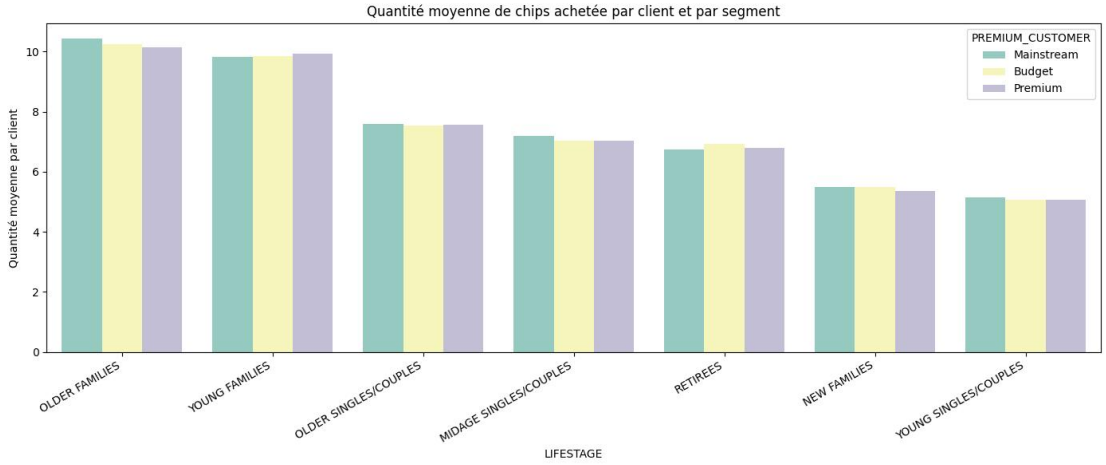
```
plt.title('Total des ventes par segment (LIFESTAGE, PREMIUM_CUSTOMER)')
plt.ylabel('Total des ventes')
plt.xlabel('LIFESTAGE')
plt.xticks(rotation=30, ha='right')
plt.legend(title='PREMIUM_CUSTOMER')
plt.tight_layout()
plt.show()
```




```
# Visualisation du nombre de clients par segment
plt.figure(figsize=(14, 6))
sns.barplot(
    data=clients_par_segment,
    x='LIFESTAGE',
    y='NB_CLIENTS',
    hue='PREMIUM_CUSTOMER',
    palette='Set1'
)
plt.title('Nombre de clients par segment (LIFESTAGE, PREMIUM_CUSTOMER)')
plt.ylabel('Nombre de clients')
plt.xlabel('LIFESTAGE')
plt.xticks(rotation=30, ha='right')
plt.legend(title='PREMIUM_CUSTOMER')
plt.tight_layout()
plt.show()
```

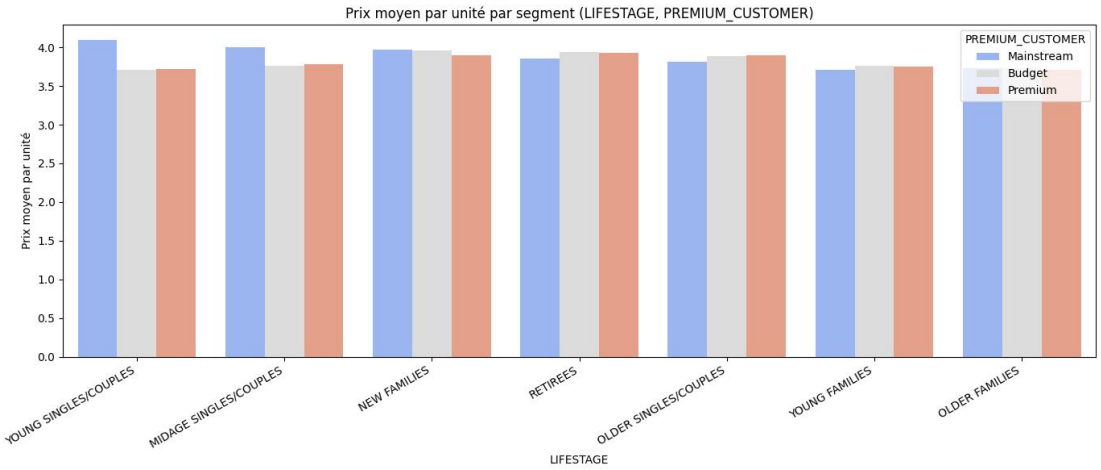



```
# Visualisation de la quantité moyenne de chips achetée par client et par segment
plt.figure(figsize=(14, 6))
sns.barplot(
    data=quantite_moyenne_segment,
    x='LIFESTAGE',
    y='QTE_MOYENNE_PAR_CLIENT',
    hue='PREMIUM_CUSTOMER',
    palette='Set3'
)
plt.title('Quantité moyenne de chips achetée par client et par segment')
plt.ylabel('Quantité moyenne par client')
plt.xlabel('LIFESTAGE')
plt.xticks(rotation=30, ha='right')
plt.legend(title='PREMIUM_CUSTOMER')
plt.tight_layout()
plt.show()
```




```
# Visualisation du prix moyen par unité par segment
plt.figure(figsize=(14, 6))
sns.barplot(
    data=prix_moyen_unite_segment,
```

```
    x='LIFESTAGE',
    y='PRIX_MOYEN_PAR_UNITE',
    hue='PREMIUM_CUSTOMER',
    palette='coolwarm'
)
plt.title('Prix moyen par unité par segment (LIFESTAGE, PREMIUM_CUSTOMER)')
plt.ylabel('Prix moyen par unité')
plt.xlabel('LIFESTAGE')
plt.xticks(rotation=30, ha='right')
plt.legend(title='PREMIUM_CUSTOMER')
plt.tight_layout()
plt.show()
```



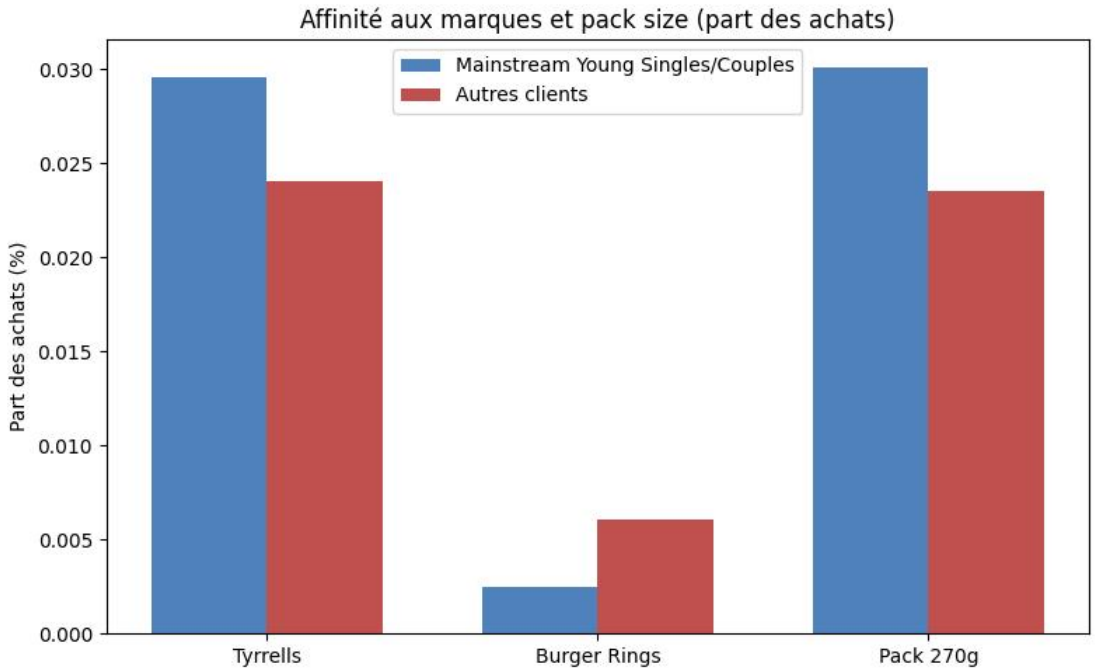
Deep dive : Mainstream Young Singles/Couples

Analyse détaillée du segment **Mainstream Young Singles/Couples** :
- Affinité à certaines marques (Tyrrells, Burger Rings)
- Préférence pour certains pack sizes (notamment 270g)

Nous allons comparer la propension de ce segment à acheter ces marques et tailles par rapport au reste des clients.

```
# Filtrer le segment cible
segment = df[(df['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (df['PREMIUM_CUSTOMER'] == 'Mainstream')]
reste = df[~((df['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') & (df['PREMIUM_CUSTOMER'] == 'Mainstream'))]

# Affinité à la marque Tyrrells
part_tyrrells_segment = segment['BRAND'].value_counts(normalize=True).get('Tyrrells', 0)
part_tyrrells_reste = reste['BRAND'].value_counts(normalize=True).get('Tyrrells', 0)
variation_tyrrells = (part_tyrrells_segment - part_tyrrells_reste) / part_tyrrells_reste * 100 if part_tyrrells_reste > 0 else None
```

```
# Affinité à la marque Burger Rings
part_burgerrings_segment = segment['BRAND'].value_counts(normalize=True).get('Burger', 0)
part_burgerrings_reste = reste['BRAND'].value_counts(normalize=True).get('Burger', 0)
variation_burgerrings = (part_burgerrings_segment - part_burgerrings_reste) / part_burgerrings_reste * 100 if part_burgerrings_reste > 0 else None

# Préférence pour le pack size 270g
part_270g_segment = segment['PACK_SIZE'].value_counts(normalize=True).get(270, 0)
part_270g_reste = reste['PACK_SIZE'].value_counts(normalize=True).get(270, 0)
variation_270g = (part_270g_segment - part_270g_reste) / part_270g_reste * 100 if part_270g_reste > 0 else None

print(f"Tyrrells : {variation_tyrrells:+.0f}% de différence dans le segment vs reste")
print(f"Burger Rings : {variation_burgerrings:+.0f}% de différence dans le segment vs reste")
print(f"Pack 270g : {variation_270g:+.0f}% de différence dans le segment vs reste")
# Visualisation des différences d'affinité
labels = ['Tyrrells', 'Burger Rings', 'Pack 270g']
segment_vals = [part_tyrrells_segment, part_burgerrings_segment, part_270g_segment]
reste_vals = [part_tyrrells_reste, part_burgerrings_reste, part_270g_reste]

x = range(len(labels))
plt.figure(figsize=(8, 5))
plt.bar(x, segment_vals, width=0.35, label='Mainstream Young Singles/Couples', color='#4F81BD')
plt.bar([i + 0.35 for i in x], reste_vals, width=0.35, label='Autres clients', color='#C0504D')
plt.xticks([i + 0.175 for i in x], labels)
plt.ylabel('Part des achats (%)')
plt.title("Affinité aux marques et pack size (part des achats)")
plt.legend()
plt.tight_layout()
plt.show()
```



Conclusions de l'analyse (avec références aux visualisations et données)

Voici les principaux enseignements tirés de l'ensemble des analyses, avec les visualisations ou tableaux correspondants :

1. Segmentation et comportements clients
- Les segments les plus rentables sont : Budget – Older Families, Mainstream – Young Singles/Couples, Mainstream – Retirees.
  - *Voir le graphique « Total des ventes par segment » (Cellule 1 des visualisations)*
- Les jeunes couples mainstream sont nombreux et dépensent plus par unité.
  - *Voir « Nombre de clients par segment » et « Prix moyen par unité par segment » (Cellules 2 et 4 des visualisations)*

2. Dépenses et quantités
- Les clients les plus dépensiers ne sont pas forcément ceux qui achètent le plus de paquets, mais ceux qui achètent des produits plus chers ou en plus grande quantité par transaction.
  - *Voir « Dépense totale par client » et « Quantité totale achetée par client » (Cellules 1 et 4 des analyses clients)*
- Les tailles de paquets préférées varient selon les segments, avec une préférence marquée pour le 270g chez les jeunes couples mainstream.
  - *Voir l'analyse deep dive et la visualisation d'affinité (fin du notebook)*

3. Affinité aux marques
- Le segment Mainstream Young Singles/Couples est particulièrement attiré par la marque Tyrrells (+23% par rapport au reste), et moins par Burger Rings (-56%).
  - *Voir la visualisation « Affinité aux marques et pack size » (fin du notebook)*
- Ce même segment est aussi plus enclin à acheter des paquets de 270g (+27%).

- *Voir la même visualisation d’affinité (fin du notebook)*

4. Prix moyen par unité
- Le prix moyen par unité est significativement plus élevé pour les segments Mainstream jeunes et midage que pour Budget ou Premium.
- *Voir « Prix moyen par unité par segment » (Cellule 4 des visualisations)*

5. Recommandations
- Cibler les segments Mainstream Young Singles/Couples et Older Families avec des offres spécifiques (packs 270g, marques premium).
- Adapter la communication et les promotions selon les préférences de chaque segment pour maximiser la valeur client.
- *Ces recommandations sont issues de l’ensemble des visualisations et analyses précédentes.*

En résumé :
L’analyse met en évidence l’importance de la segmentation pour comprendre les comportements d’achat et optimiser les stratégies marketing. Les différences d’affinité aux marques, de taille de paquet et de prix moyen par segment offrent des leviers d’action concrets pour augmenter les ventes et la satisfaction client.