# Zbakh_Jason_1_notebook_052024

July 16, 2024

Projet 9 : Analysez les ventes d'une librairie avec R ou Python

### 0.0.1 Index


- Etape 1 - Importation des librairies et chargement des fichiers
  - 1.1 - Importation des librairies
  - 1.2 - Chargements des fichiers
- Etape 2 - Exploration des données
  - 2.1 - Données df_clients
  - 2.2 - Données df_produits
  - 2.3 - Données df_transactions
- Etape 4 - Jonction des fichiers
  - 4.1 - Réalisation de la jointure entre df_clients et df_transactions
  - 4.2 - Réalisation de la jointure entre df_merge et df_produits
- Etape 5 - Études du chiffre d'affaires
  - 5.1 - Evolution du chiffre d'affaires journalier
  - 5.2 - Evolution du chiffre d'affaires mensuel et détails par catégories
  - 5.3 - Evolution du chiffre d'affaires cumulés
  - 5.4 - Total chiffres d'affaires par catégories
  - 5.5 - Courbe de Lorenz et coeff de gini
- Etape 6 - Chiffres d'affaires par clients et definition des clients B2B
  - 6.1 - Distribution du CA par clients
  - 6.2 - Courbe de Lorenz et coeff de gini
  - 6.3 - Top 10 des clients par chiffre d'affaires total et détails pour chaques catégories
  - 6.4 - Definition des clients Outlier par le CA (methode IQR)
  - 6.5 - Definition des clients Outlier par le CA (methode Zscore)
  - 6.6 - Exclusion des 4 clients B2B
  - 6.7 - Répartition du chiffre d'affaires entre clients et B2B et détails des catégories pour les B2B
  - 6.8 - Top 10 des clients par chiffre d'affaires total et détails pour chaques catégories (clients B2B exclus)
  - 6.9 - 80/20 en Chiffre d'Affaires (Analyse de Pareto)

- Etape 7 - Indicateurs de ventes (KPI's)
  - 7.1 - Nombre de transactions par clients et par categories
  - 7.2 - Evolution du nombre de transactions(mensuel)
  - 7.3 - Nombre de Sessions par clients
  - 7.4 - Panier_Moyen par clients
  - 7.5 - Evolution du nombre de clients (mensuel)
  - 7.6 - Nombre de références vendus par mois
  - 7.7 - Tops et Flops Références Vendues par An
- Etape 8 - Etudes des relations demandées
  - 8.1 Préparation de la matrice pour les études des relations
  - 8.2 Lien entre le genre d'un client et les catégories des livres achetés
  - 8.3 Lien entre l'âge des clients et le montant total des achats
  - 8.4 Lien entre l'âge des clients et la fréquence d'achats
  - 8.5 Lien entre l'âge des clients et la taille du panier moyen
  - 8.6 Lien entre l'âge des clients et la catégorie des livres achetés

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

[ ]:
```
#Import de la bibliothèque Pandas
import pandas as pd
# Ajuster les options d'affichage
pd.set_option('display.max_rows', 200)
pd.set_option('display.max_columns', None)
```

[ ]:
```
#Import de la bibliothèque plotly express
import plotly.express as px
```

[ ]:
```
#Import de la bibliothèque matplotlib
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
```

[ ]:
```
# Import bibliothèque Numpy
import numpy as np
```

[ ]:
```
# Import de la bibliothèque Seaborn
import seaborn as sns
```

[ ]:
```
# Import de la bibliothèque Datetime
from datetime import datetime
```

[ ]:
```
#Import de la bibliothèque scipy et import des tests statistiques
import scipy
from scipy.stats import chi2_contingency, pearsonr, levene, shapiro, chi2,␣
↪f_oneway
```

[ ]:
```
#Import de la bibliothèque statsmodels
from statsmodels.formula.api import ols
import statsmodels.api as sm
import statsmodels.stats.anova as anova
import statsmodels.graphics.gofplots as gofplots
```

1.2 - Chargements des fichiers

[ ]:
```
#Import du fichier customers.csv
df_clients = pd.read_csv('C:\\Users\\WadJa\\Desktop\\Openclassroom\\Projet␣
 ↪9\\Data\\customers.csv', sep=';')

#Import du fichier products.csv
df_produits = pd.read_csv('C:\\Users\\WadJa\\Desktop\\Openclassroom\\Projet␣
 ↪9\\Data\\products.csv', sep=';', dtype={'categ': object})

#Import du fichier Transactions.csv
df_transactions = pd.read_csv(
    'C:\\Users\\WadJa\\Desktop\\Openclassroom\\Projet 9\\Data\\Transactions.
 ↪csv', sep=';',dtype={
        'id_prod': 'object',
        'session_id': 'object',
        'client_id': 'object'},
    parse_dates=['date']
)
```

Etape 2 - Exploration et Analyse

2.1 - Données df_clients

[ ]:
```
df_clients.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8621 entries, 0 to 8620
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   client_id  8621 non-null   object
 1   sex        8621 non-null   object
 2   birth      8621 non-null   int64
dtypes: int64(1), object(2)
memory usage: 202.2+ KB
```

[ ]:
```
df_clients.describe(include = 'all')
```

[ ]:

| | client_id | sex | birth |
|---|---|---|---|
| count | 8621 | 8621 | 8621.000000 |
| unique | 8621 | 2 | NaN |
| top | c_4410 | f | NaN |

```
freq           1  4490          NaN
mean         NaN   NaN  1978.275606
std          NaN   NaN    16.917958
min          NaN   NaN  1929.000000
25%          NaN   NaN  1966.000000
50%          NaN   NaN  1979.000000
75%          NaN   NaN  1992.000000
max          NaN   NaN  2004.000000
```

[ ]:
```
# Remplacer 'm' par 'Homme' et 'f' par 'Femme' dans la colonne 'sex'
df_clients['sex'] = df_clients['sex'].replace({'m': 'Homme', 'f': 'Femme'})
```

[ ]:
```
# Calculer l'âge
df_clients['age'] = datetime.now().year - df_clients['birth']
```

[ ]:
```
df_clients.head()
```

[ ]:
```
  client_id    sex  birth  age
0    c_4410  Femme   1967   57
1    c_7839  Femme   1975   49
2    c_1699  Femme   1984   40
3    c_5961  Femme   1962   62
4    c_5320  Homme   1943   81
```

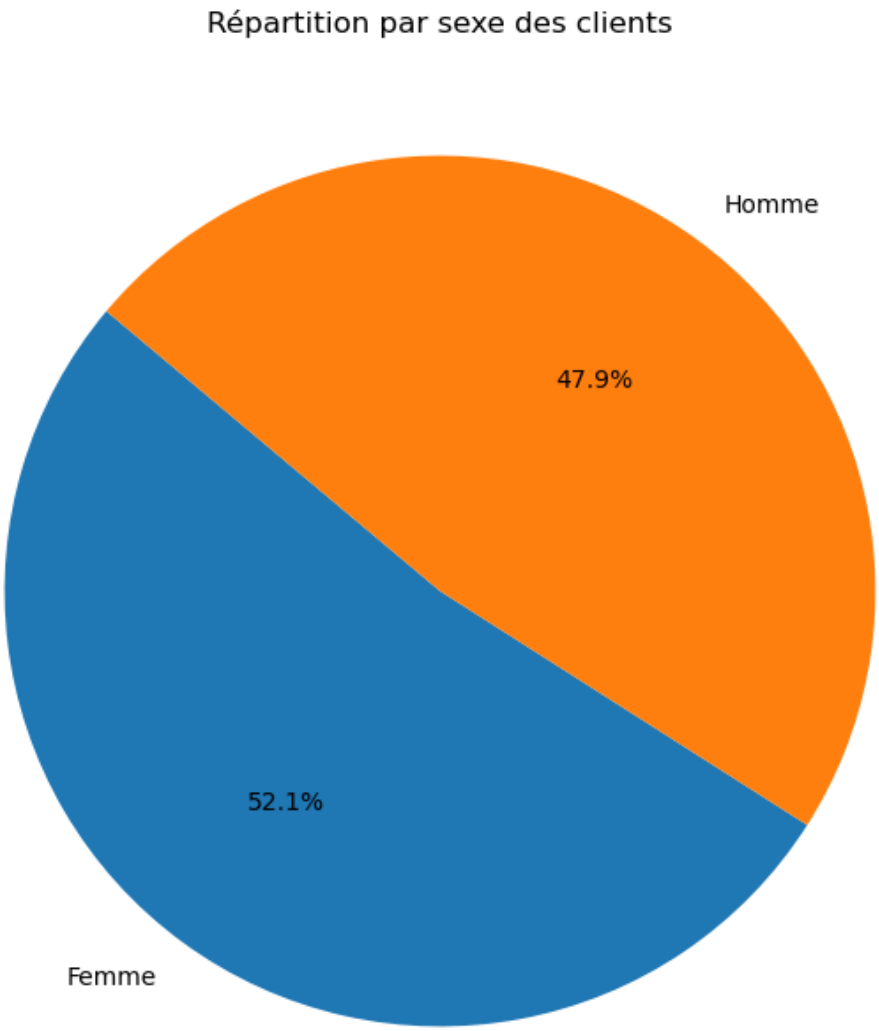
[ ]:
```
#Suppression de la colonne 'birth'
df_clients = df_clients.drop('birth', axis=1)
```

[ ]:
```
df_clients
```

[ ]:
```
     client_id    sex  age
0       c_4410  Femme   57
1       c_7839  Femme   49
2       c_1699  Femme   40
3       c_5961  Femme   62
4       c_5320  Homme   81
…          …    …    …
8616    c_7920  Homme   68
8617    c_7403  Femme   54
8618    c_5119  Homme   50
8619    c_5643  Femme   56
8620      c_84  Femme   42

[8621 rows x 3 columns]
```

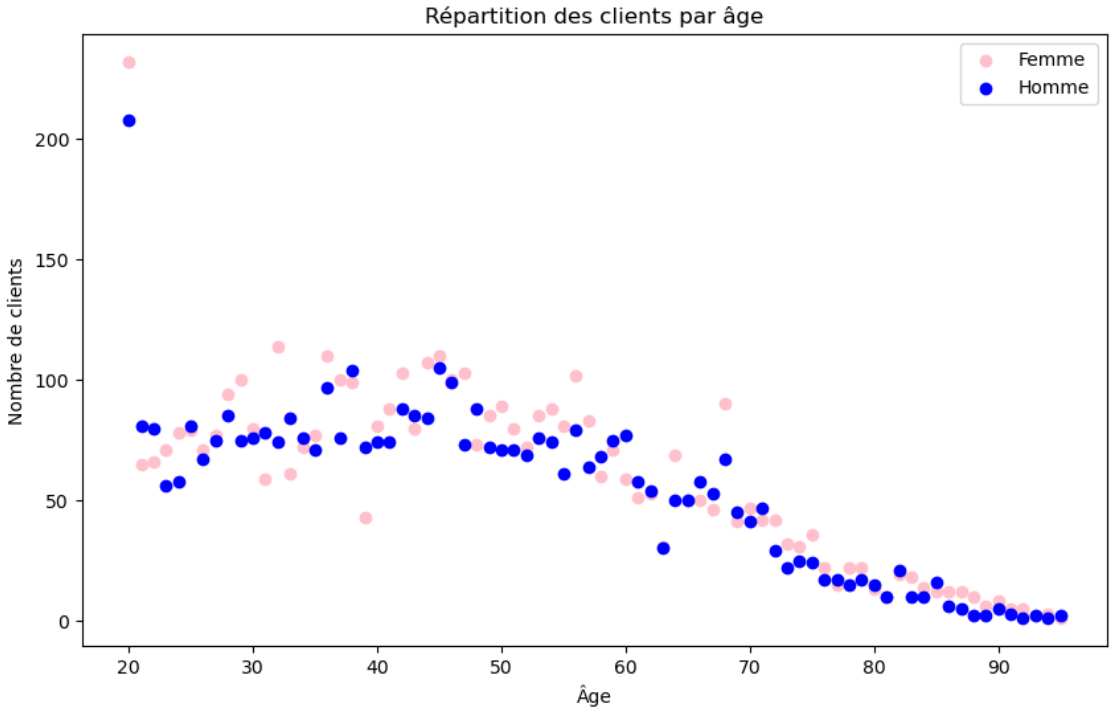
[ ]:
```
# Analyse de la répartition du sex

distribution_sex = df_clients['sex'].value_counts()
```

```
plt.figure(figsize=(8, 8))
plt.pie(distribution_sex, labels=distribution_sex.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Répartition par sexe des clients')
plt.show()
```




[ ]:
```
# Calculer le nombre de clients par âge et sexe
grouped_data = df_clients.groupby(['age', 'sex']).size().
 ↪reset_index(name='count')

# Séparation des données par sexe
```

```
data_femme = grouped_data[grouped_data['sex'] == 'Femme']
data_homme = grouped_data[grouped_data['sex'] == 'Homme']

# Création du nuage de points
plt.figure(figsize=(10, 6))
plt.scatter(data_femme['age'], data_femme['count'], color='pink', label='Femme')
plt.scatter(data_homme['age'], data_homme['count'], color='blue', label='Homme')

plt.xlabel('Âge')
plt.ylabel('Nombre de clients')
plt.title('Répartition des clients par âge')
plt.legend()
plt.show()
```




2.2 - Données df_produits

[ ]:
```
df_produits.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3286 entries, 0 to 3285
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id_prod  3286 non-null   object
 1   price    3286 non-null   float64
```

```
 2   categ    3286 non-null   object
dtypes: float64(1), object(2)
memory usage: 77.1+ KB
```

[ ]:
```
df_produits.describe(include = 'all')
```

[ ]:
```
        id_prod        price categ
count      3286  3286.000000  3286
unique     3286          NaN     3
top     0_1421          NaN     0
freq          1          NaN  2308
mean        NaN    21.863597   NaN
std         NaN    29.849786   NaN
min         NaN     0.620000   NaN
25%         NaN     6.990000   NaN
50%         NaN    13.075000   NaN
75%         NaN    22.990000   NaN
max         NaN   300.000000   NaN
```

[ ]:
```
# Analyse de la répartition des prix produits

fig = px.histogram(df_produits, x='price', title='Distribution des prix des␣
↪produits')
fig.update_layout(bargap=0.2)

fig.show()
```

[ ]:
```
# Création d'un boxplot de la distribution des prix des produits par catégories
fig = px.box(df_produits, x='categ', y='price', color='categ',
             title='Distribution des prix par catégories')
fig.update_layout(xaxis_title='Catégorie', yaxis_title='Prix')

fig.show()
```

2.3 - Données df_transactions

[ ]:
```
df_transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 4 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   id_prod     687534 non-null   object
 1   date        687534 non-null   datetime64[ns]
 2   session_id  687534 non-null   object
 3   client_id   687534 non-null   object
dtypes: datetime64[ns](1), object(3)
memory usage: 32.0+ MB
```

[ ]:
```
#RangeIndex: 1048575 entries / 687534 non-nul
#Suppression des lignes où toutes les colonnes sont nulles
df_transactions = df_transactions.dropna(how='all')
```

[ ]:
```
#Vérification
df_transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 687534 entries, 0 to 687533
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   id_prod     687534 non-null  object
 1   date        687534 non-null  datetime64[ns]
 2   session_id  687534 non-null  object
 3   client_id   687534 non-null  object
dtypes: datetime64[ns](1), object(3)
memory usage: 26.2+ MB
```

[ ]:
```
df_transactions.describe(include = 'all')
```

[ ]:

| | id_prod | date | session_id | client_id |
|---|---|---|---|---|
| count | 687534 | 687534 | 687534 | 687534 |
| unique | 3265 | NaN | 345505 | 8600 |
| top | 1_369 | NaN | s_118668 | c_1609 |
| freq | 2340 | NaN | 14 | 25586 |
| mean | NaN | 2022-03-01 21:24:00.618519296 | NaN | NaN |
| min | NaN | 2021-03-01 00:01:07.843138 | NaN | NaN |
| 25% | NaN | 2021-09-10 10:35:20.642323456 | NaN | NaN |
| 50% | NaN | 2022-02-27 06:50:25.400120064 | NaN | NaN |
| 75% | NaN | 2022-08-28 22:16:49.841665536 | NaN | NaN |
| max | NaN | 2023-02-28 23:58:30.792755 | NaN | NaN |

[ ]:
```
df_transactions
```

[ ]:

| | id_prod | date | session_id | client_id |
|---|---|---|---|---|
| 0 | 0_1259 | 2021-03-01 00:01:07.843138 | s_1 | c_329 |
| 1 | 0_1390 | 2021-03-01 00:02:26.047414 | s_2 | c_664 |
| 2 | 0_1352 | 2021-03-01 00:02:38.311413 | s_3 | c_580 |
| 3 | 0_1458 | 2021-03-01 00:04:54.559692 | s_4 | c_7912 |
| 4 | 0_1358 | 2021-03-01 00:05:18.801198 | s_5 | c_2033 |
| … | … | … | … | … |
| 687529 | 1_508 | 2023-02-28 23:49:03.148402 | s_348444 | c_3573 |
| 687530 | 2_37 | 2023-02-28 23:51:29.318531 | s_348445 | c_50 |
| 687531 | 1_695 | 2023-02-28 23:53:18.929676 | s_348446 | c_488 |
| 687532 | 0_1547 | 2023-02-28 23:58:00.107815 | s_348447 | c_4848 |
| 687533 | 0_1398 | 2023-02-28 23:58:30.792755 | s_348435 | c_3575 |

```
[687534 rows x 4 columns]
```

Etape 4 - Jonction des fichiers

4.1 Réalisation de la jointure entre df_clients et df_transactions

[ ]:
```
df_merge= pd.merge(df_clients, df_transactions, how="inner", on='client_id')

# Affichage du DataFrame résultant pour vérification
df_merge.head(10)
```

[ ]:

| | client_id | sex | age | id_prod | date | session_id |
|---|---|---|---|---|---|---|
| 0 | c_4410 | Femme | 57 | 1_483 | 2021-03-13 21:35:55.949042 | s_5913 |
| 1 | c_4410 | Femme | 57 | 0_1111 | 2021-03-22 01:27:49.480137 | s_9707 |
| 2 | c_4410 | Femme | 57 | 1_385 | 2021-03-22 01:40:22.782925 | s_9707 |
| 3 | c_4410 | Femme | 57 | 0_1455 | 2021-03-22 14:29:25.189266 | s_9942 |
| 4 | c_4410 | Femme | 57 | 0_1420 | 2021-03-22 22:31:25.825764 | s_10092 |
| 5 | c_4410 | Femme | 57 | 0_521 | 2021-03-23 17:11:46.158290 | s_10454 |
| 6 | c_4410 | Femme | 57 | 1_407 | 2021-03-24 18:30:13.156928 | s_10922 |
| 7 | c_4410 | Femme | 57 | 0_1587 | 2021-04-25 09:06:27.525631 | s_25473 |
| 8 | c_4410 | Femme | 57 | 1_584 | 2021-05-28 05:19:16.367701 | s_40563 |
| 9 | c_4410 | Femme | 57 | 1_91 | 2021-05-28 05:29:10.024293 | s_40563 |

[ ]:
```
df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 687534 entries, 0 to 687533
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   client_id   687534 non-null  object
 1   sex         687534 non-null  object
 2   age         687534 non-null  int64
 3   id_prod     687534 non-null  object
 4   date        687534 non-null  datetime64[ns]
 5   session_id  687534 non-null  object
dtypes: datetime64[ns](1), int64(1), object(4)
memory usage: 31.5+ MB
```

[ ]:
```
# Vérifier le nombre d'entrées uniques de `client_id` dans df_merge
print("Nombre id_client uniques dans df_merge:", df_merge['client_id'].
 ↪nunique())

# Vérifier le nombre d'entrées uniques de `client_id` dans df_clients
print("Nombre id_client uniques dans df_clients:", df_clients['client_id'].
 ↪nunique())

# Vérifier le nombre d'entrées uniques de `client_id` dans df_transactions
```

```
print("Nombre id_client uniques dans df_transactions:",␣
 ↪df_transactions['client_id'].nunique())
```

```
Nombre id_client uniques dans df_merge: 8600
Nombre id_client uniques dans df_clients: 8621
Nombre id_client uniques dans df_transactions: 8600
```

[ ]:
```
print((df_clients['client_id'].nunique())-(df_transactions['client_id'].
 ↪nunique()),"clients n'ont jamais passés de commande")
```

```
21 clients n'ont jamais passés de commande
```

4.2 Réalisation de la jointure entre df_merge et df_produits

[ ]:
```
df_lapage = pd.merge(df_merge, df_produits,how="inner", on='id_prod')

df_lapage
```

[ ]:
```
       client_id    sex  age id_prod                       date session_id  \
0         c_4410  Femme   57   1_483 2021-03-13 21:35:55.949042     s_5913
1         c_4410  Femme   57  0_1111 2021-03-22 01:27:49.480137     s_9707
2         c_4410  Femme   57   1_385 2021-03-22 01:40:22.782925     s_9707
3         c_4410  Femme   57  0_1455 2021-03-22 14:29:25.189266     s_9942
4         c_4410  Femme   57  0_1420 2021-03-22 22:31:25.825764    s_10092
…            …    …  …     …                             …        …
687529      c_84  Femme   42  0_1472 2022-05-14 00:24:49.391917   s_208110
687530      c_84  Femme   42  0_1438 2022-05-29 06:11:50.316631   s_215697
687531      c_84  Femme   42   1_459 2022-12-17 00:16:56.629536   s_313173
687532      c_84  Femme   42  0_1104 2022-12-17 00:24:14.357525   s_313173
687533      c_84  Femme   42   1_688 2022-12-17 00:36:24.148027   s_313173

        price categ
0       15.99     1
1       19.99     0
2       25.99     1
3        8.99     0
4       11.53     0
…       …   …
687529  12.49     0
687530   9.31     0
687531  15.99     1
687532  13.21     0
687533  18.19     1

[687534 rows x 8 columns]
```

[ ]:
```
# Vérifier le nombre d'entrées uniques de `client_id` dans df_merge
print("Nombre d'entrées uniques de `id_prod` dans df_merge:",␣
 ↪df_lapage['id_prod'].nunique())
```

```
# Vérifier le nombre d'entrées uniques de `client_id` dans df_lapage
print("Nombre d'entrées uniques de `id_prod` df_lapage:", df_merge['id_prod'].
  ↪nunique())
```

```
Nombre d'entrées uniques de `id_prod` dans df_merge: 3265
Nombre d'entrées uniques de `id_prod` df_lapage: 3265
```

Chaques références produits a été achetées au moins une fois.

[ ]:
```
df_lapage.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 687534 entries, 0 to 687533
Data columns (total 8 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   client_id   687534 non-null  object
 1   sex         687534 non-null  object
 2   age         687534 non-null  int64
 3   id_prod     687534 non-null  object
 4   date        687534 non-null  datetime64[ns]
 5   session_id  687534 non-null  object
 6   price       687534 non-null  float64
 7   categ       687534 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 42.0+ MB
```

[ ]:
```
df_lapage.describe(include='all')
```

[ ]:

| | client_id | sex | age | id_prod | \ |
|---|---|---|---|---|---|
| count | 687534 | 687534 | 687534.000000 | 687534 | |
| unique | 8600 | 2 | NaN | 3265 | |
| top | c_1609 | Homme | NaN | 1_369 | |
| freq | 25586 | 344841 | NaN | 2340 | |
| mean | NaN | NaN | 46.182609 | NaN | |
| min | NaN | NaN | 20.000000 | NaN | |
| 25% | NaN | NaN | 37.000000 | NaN | |
| 50% | NaN | NaN | 44.000000 | NaN | |
| 75% | NaN | NaN | 54.000000 | NaN | |
| max | NaN | NaN | 95.000000 | NaN | |
| std | NaN | NaN | 13.607935 | NaN | |

| | date | session_id | price | categ |
|---|---|---|---|---|
| count | 687534 | 687534 | 687534.000000 | 687534 |
| unique | NaN | 345505 | NaN | 3 |
| top | NaN | s_118668 | NaN | 0 |
| freq | NaN | 14 | NaN | 415459 |
| mean | 2022-03-01 21:24:00.618518528 | NaN | 17.493918 | NaN |

```
min        2021-03-01 00:01:07.843138        NaN       0.620000     NaN
25%     2021-09-10 10:35:20.642323456        NaN       8.990000     NaN
50%     2022-02-27 06:50:25.400120064        NaN      13.990000     NaN
75%     2022-08-28 22:16:49.841665536        NaN      19.080000     NaN
max        2023-02-28 23:58:30.792755        NaN     300.000000     NaN
std                                NaN        NaN      18.238337     NaN
```

Etape 5 - Études du chiffre d'affaires

5.1 - Evolution du chiffre d'affaires journalier

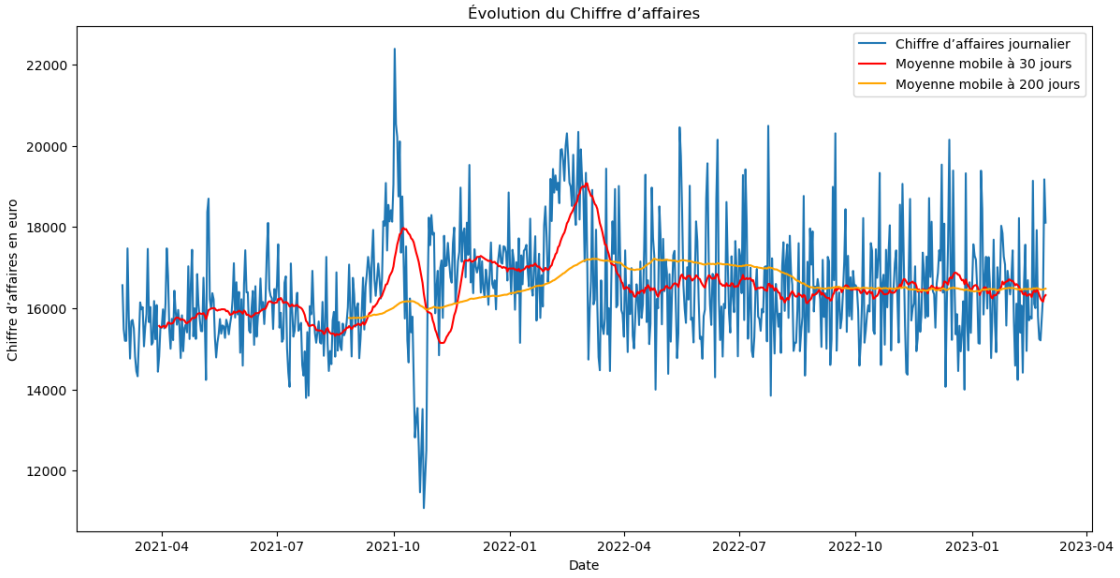
[ ]:
```
#Création d'une copie du df avec la colonne 'date' comme index

df_Dateindex = df_lapage.copy()

df_Dateindex.set_index('date', inplace=True)
df_Dateindex
```

[ ]:
```
                           client_id    sex  age id_prod session_id  price  \
date
2021-03-13 21:35:55.949042    c_4410  Femme   57   1_483     s_5913  15.99
2021-03-22 01:27:49.480137    c_4410  Femme   57  0_1111     s_9707  19.99
2021-03-22 01:40:22.782925    c_4410  Femme   57   1_385     s_9707  25.99
2021-03-22 14:29:25.189266    c_4410  Femme   57  0_1455     s_9942   8.99
2021-03-22 22:31:25.825764    c_4410  Femme   57  0_1420    s_10092  11.53
…                                …      …  …      …          …     …
2022-05-14 00:24:49.391917      c_84  Femme   42  0_1472   s_208110  12.49
2022-05-29 06:11:50.316631      c_84  Femme   42  0_1438   s_215697   9.31
2022-12-17 00:16:56.629536      c_84  Femme   42   1_459   s_313173  15.99
2022-12-17 00:24:14.357525      c_84  Femme   42  0_1104   s_313173  13.21
2022-12-17 00:36:24.148027      c_84  Femme   42   1_688   s_313173  18.19

                           categ
date
2021-03-13 21:35:55.949042     1
2021-03-22 01:27:49.480137     0
2021-03-22 01:40:22.782925     1
2021-03-22 14:29:25.189266     0
2021-03-22 22:31:25.825764     0
…                            …
2022-05-14 00:24:49.391917     0
2022-05-29 06:11:50.316631     0
2022-12-17 00:16:56.629536     1
2022-12-17 00:24:14.357525     0
2022-12-17 00:36:24.148027     1

[687534 rows x 7 columns]
```

[ ]:
```
df_CA_journalier = df_Dateindex.groupby(df_Dateindex.index.date)['price'].sum()
df_CA_journalier
```

[ ]:
```
2021-03-01    16565.22
2021-03-02    15486.45
2021-03-03    15198.69
2021-03-04    15196.07
2021-03-05    17471.37
                 ...
2023-02-24    15207.89
2023-02-25    15761.25
2023-02-26    16304.72
2023-02-27    19170.81
2023-02-28    18105.15
Name: price, Length: 730, dtype: float64
```

[ ]:
```
# Calculer les moyennes mobiles
mm_30 = df_CA_journalier.rolling(window=30).mean()
mm_200 = df_CA_journalier.rolling(window=180).mean()

# Tracer les moyennes mobiles
plt.figure(figsize=(14, 7))
plt.plot(df_CA_journalier, label='Chiffre d'affaires journalier')
plt.plot(mm_30, label='Moyenne mobile à 30 jours', color='red')
plt.plot(mm_200, label='Moyenne mobile à 200 jours', color='orange')
plt.title('Évolution du Chiffre d'affaires')
plt.xlabel('Date')
plt.ylabel('Chiffre d'affaires en euro')
plt.legend()
plt.show()
```



5.2 - Evolution du chiffre d'affaires mensuel et détails par categories

[ ]:
```
df_CA_journalier = df_Dateindex.groupby(df_Dateindex.index.date)['price'].sum()
df_CA_journalier
```

[ ]:
```
2021-03-01    16565.22
2021-03-02    15486.45
2021-03-03    15198.69
2021-03-04    15196.07
2021-03-05    17471.37
                 …
2023-02-24    15207.89
2023-02-25    15761.25
2023-02-26    16304.72
2023-02-27    19170.81
2023-02-28    18105.15
Name: price, Length: 730, dtype: float64
```

[ ]:
```
df_Dateindex['Date'] = df_Dateindex.index.to_period('M')
df_Dateindex
```

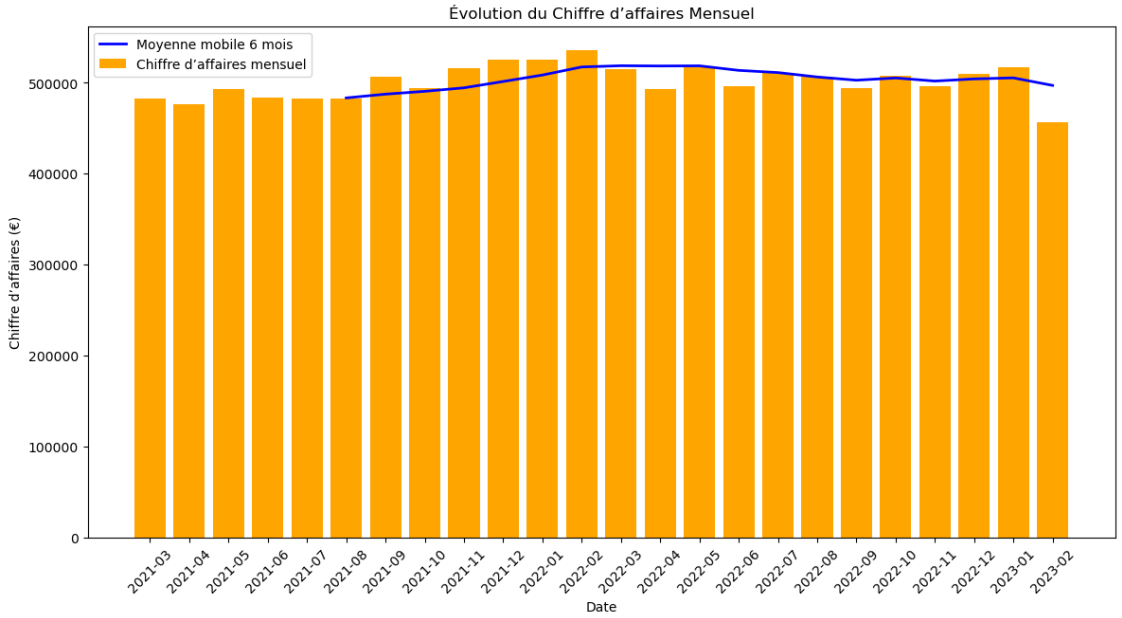
[ ]:
```
                           client_id    sex  age id_prod session_id  price  \
date
2021-03-13 21:35:55.949042    c_4410  Femme   57   1_483     s_5913  15.99
2021-03-22 01:27:49.480137    c_4410  Femme   57  0_1111     s_9707  19.99
2021-03-22 01:40:22.782925    c_4410  Femme   57   1_385     s_9707  25.99
2021-03-22 14:29:25.189266    c_4410  Femme   57  0_1455     s_9942   8.99
2021-03-22 22:31:25.825764    c_4410  Femme   57  0_1420    s_10092  11.53
…                                …      …  …      …          …      …
2022-05-14 00:24:49.391917      c_84  Femme   42  0_1472   s_208110  12.49
2022-05-29 06:11:50.316631      c_84  Femme   42  0_1438   s_215697   9.31
2022-12-17 00:16:56.629536      c_84  Femme   42   1_459   s_313173  15.99
2022-12-17 00:24:14.357525      c_84  Femme   42  0_1104   s_313173  13.21
2022-12-17 00:36:24.148027      c_84  Femme   42   1_688   s_313173  18.19

                           categ     Date
date
2021-03-13 21:35:55.949042     1  2021-03
2021-03-22 01:27:49.480137     0  2021-03
2021-03-22 01:40:22.782925     1  2021-03
2021-03-22 14:29:25.189266     0  2021-03
2021-03-22 22:31:25.825764     0  2021-03
…                            …       …
2022-05-14 00:24:49.391917     0  2022-05
2022-05-29 06:11:50.316631     0  2022-05
```

```
2022-12-17 00:16:56.629536     1  2022-12
2022-12-17 00:24:14.357525     0  2022-12
2022-12-17 00:36:24.148027     1  2022-12

[687534 rows x 8 columns]
```

[ ]:
```
df_CA_mensuel = df_Dateindex.groupby('Date')['price'].sum()
```

[ ]:
```
# Calculer la moyenne mobile
mm_6_mensuel = round(df_CA_mensuel.rolling(window=6).mean(),2)
```

[ ]:
```
plt.figure(figsize=(14, 7))

df_CA_mensuel.index = df_CA_mensuel.index.astype(str)
mm_6_mensuel.index = mm_6_mensuel.index.astype(str)

# Création du barplot pour le chiffre d'affaires mensuel
plt.bar(df_CA_mensuel.index, df_CA_mensuel.values, label="Chiffre d'affaires␣
 ↪mensuel", color='orange')

# Superposition de la moyenne mobile de 6 mois avec une ligne
plt.plot(mm_6_mensuel.index, mm_6_mensuel.values, label='Moyenne mobile 6␣
 ↪mois', color='blue', linewidth=2)

# Titres et de légendes
plt.title("Évolution du Chiffre d'affaires Mensuel")
plt.xlabel('Date')
plt.xticks(rotation=45)
plt.ylabel("Chiffre d'affaires (€)")
plt.legend()

plt.show()
```




[ ]:
```
df_Dateindex
```

[ ]:
```
                           client_id    sex  age id_prod session_id  price  \
date
2021-03-13 21:35:55.949042    c_4410  Femme   57   1_483     s_5913  15.99
2021-03-22 01:27:49.480137    c_4410  Femme   57  0_1111     s_9707  19.99
2021-03-22 01:40:22.782925    c_4410  Femme   57   1_385     s_9707  25.99
2021-03-22 14:29:25.189266    c_4410  Femme   57  0_1455     s_9942   8.99
2021-03-22 22:31:25.825764    c_4410  Femme   57  0_1420    s_10092  11.53
…                                …      …  …      …          …     …
2022-05-14 00:24:49.391917      c_84  Femme   42  0_1472    s_208110  12.49
2022-05-29 06:11:50.316631      c_84  Femme   42  0_1438    s_215697   9.31
2022-12-17 00:16:56.629536      c_84  Femme   42   1_459    s_313173  15.99
2022-12-17 00:24:14.357525      c_84  Femme   42  0_1104    s_313173  13.21
2022-12-17 00:36:24.148027      c_84  Femme   42   1_688    s_313173  18.19

                           categ     Date
date
2021-03-13 21:35:55.949042     1  2021-03
2021-03-22 01:27:49.480137     0  2021-03
2021-03-22 01:40:22.782925     1  2021-03
2021-03-22 14:29:25.189266     0  2021-03
2021-03-22 22:31:25.825764     0  2021-03
…                            …        …
2022-05-14 00:24:49.391917     0  2022-05
2022-05-29 06:11:50.316631     0  2022-05
```

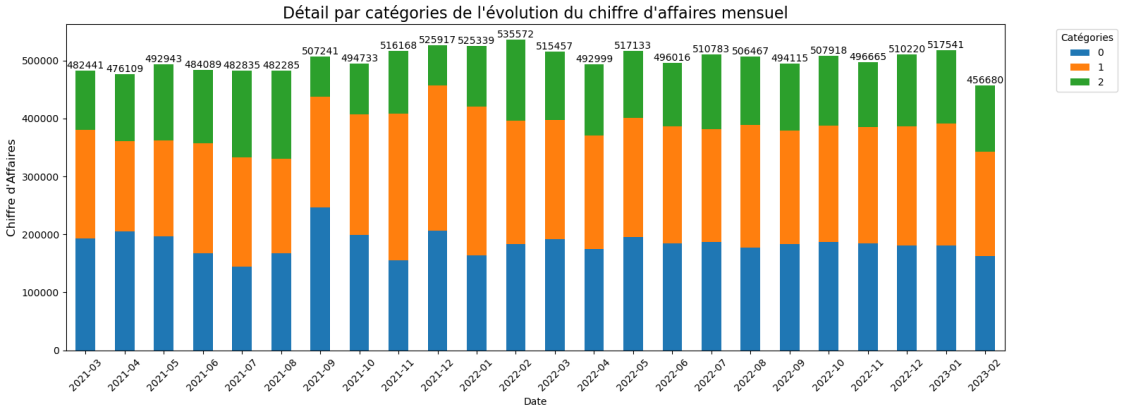
```
2022-12-17 00:16:56.629536     1  2022-12
2022-12-17 00:24:14.357525     0  2022-12
2022-12-17 00:36:24.148027     1  2022-12

[687534 rows x 8 columns]
```

[ ]:
```
# Grouper les données par période mensuelle et catégorie pour calculer la somme
 ↪du chiffre d'affaires
df_grouper = df_Dateindex.groupby(['Date', 'categ'])['price'].sum().
 ↪unstack(fill_value=0)

df_grouper.index.astype(str)

df_grouper
```

[ ]:
```
categ            0          1          2
Date
2021-03  193629.17  186974.17  101837.27
2021-04  205222.46  156138.35  114748.49
2021-05  196186.72  165893.40  130863.35
2021-06  167943.15  189162.04  126983.37
2021-07  144750.79  188523.27  149561.34
2021-08  167737.62  162991.38  151555.79
2021-09  246353.91  190613.78   70272.99
2021-10  199250.83  207696.74   87785.59
2021-11  155909.56  252910.39  107347.78
2021-12  206036.24  251026.75   68854.29
2022-01  164210.51  256267.92  104860.56
2022-02  183197.33  213120.64  139253.53
2022-03  191464.94  206485.26  117506.33
2022-04  174997.26  195263.97  122737.71
2022-05  194872.34  205532.63  116727.63
2022-06  183934.86  201912.06  110169.20
2022-07  187097.00  193969.72  129716.40
2022-08  177372.76  211360.09  117734.42
2022-09  183329.24  195379.54  115405.75
2022-10  187429.17  199609.66  120878.94
2022-11  184594.35  200427.99  111642.60
2022-12  180470.70  205945.71  123803.09
2023-01  181283.06  210104.41  126153.08
2023-02  162457.00  180347.24  113875.52
```

[ ]:
```
# Création du graphique
fig, ax = plt.subplots(figsize=(16, 6))

df_grouper.plot(kind='bar', stacked=True, ax=ax)
```

```
# Personnalisation du graphique
plt.title("Détail par catégories de l'évolution du chiffre d'affaires mensuel",␣
 ↪fontsize=16)
plt.ylabel('Chiffre d\'Affaires', fontsize=12)
plt.legend(title='Catégories', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45)

# Ajout des valeurs totales au-dessus de chaque barre
for i, total in enumerate(df_grouper.sum(axis=1)):
    ax.text(i, total, f'{total:.0f}', ha='center', va='bottom')

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

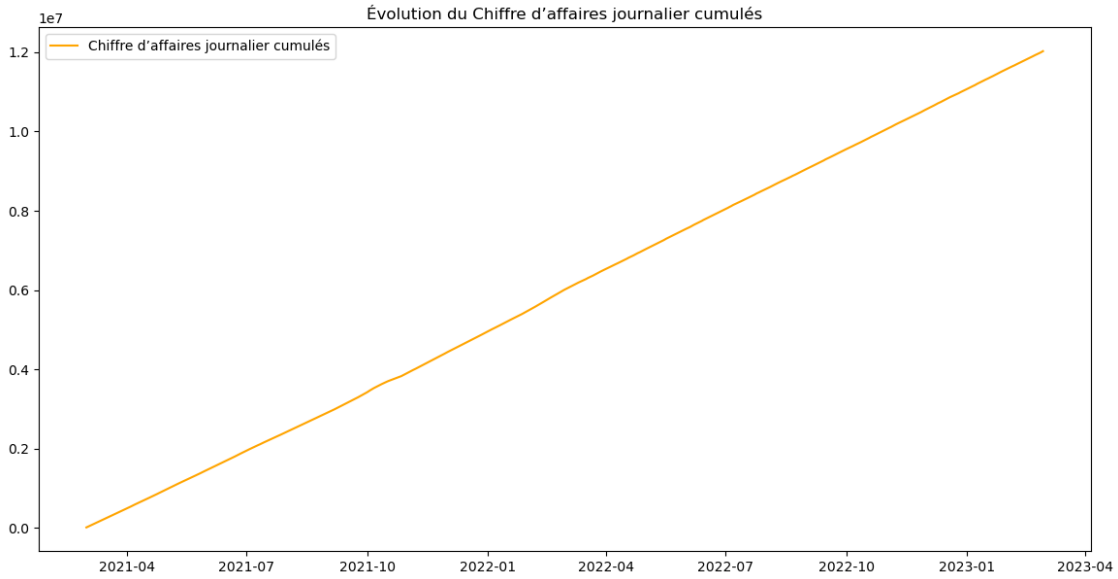



5.3 - Evolution du chiffre d'affaires cumulés

[ ]:
```
# Calculer le chiffre d'affaires cumulés
df_CA_cumulé = df_CA_journalier.cumsum()
df_CA_cumulé.head()
```

[ ]:
```
2021-03-01    16565.22
2021-03-02    32051.67
2021-03-03    47250.36
2021-03-04    62446.43
2021-03-05    79917.80
Name: price, dtype: float64
```

[ ]:
```
plt.figure(figsize=(14, 7))
df_CA_cumulé.plot(label='Chiffre d'affaires journalier cumulés', color='Orange')
plt.title('Évolution du Chiffre d'affaires journalier cumulés')
plt.legend()
plt.show()
```

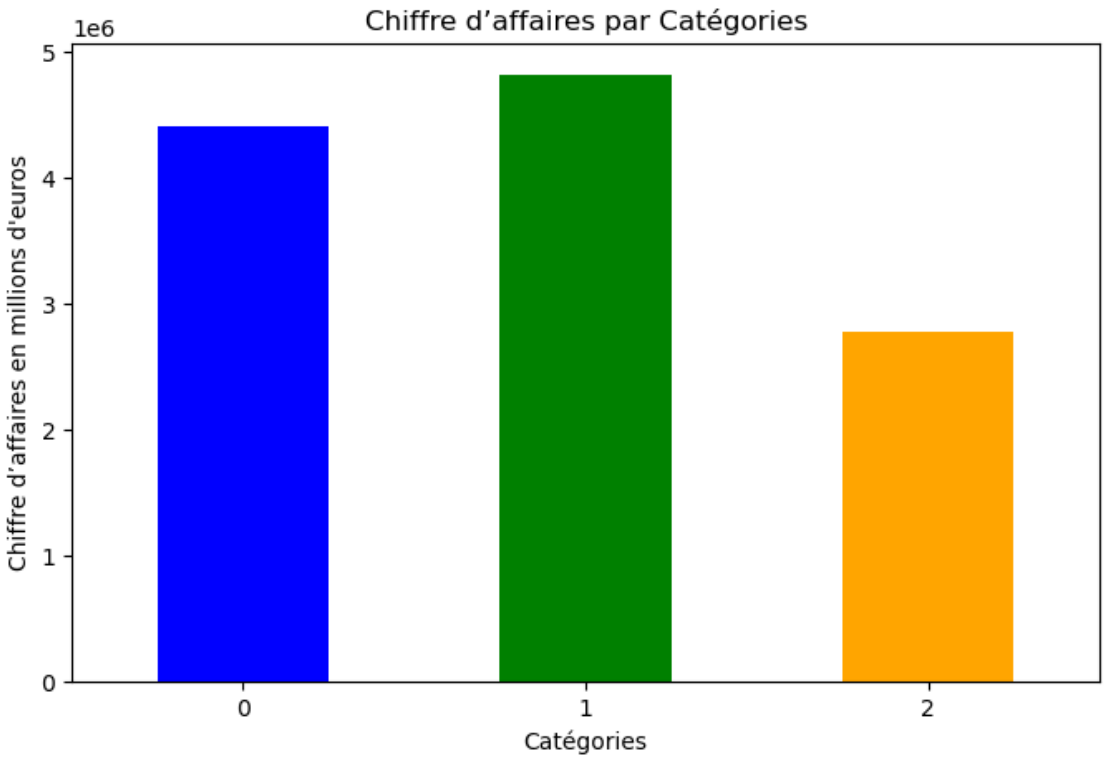



5.4 - Total du chiffre d'affaires par catégories

[ ]:
```
df_CA_par_cat = df_lapage.groupby('categ')['price'].sum()
df_CA_par_cat
```

[ ]:
```
categ
0    4419730.97
1    4827657.11
2    2780275.02
Name: price, dtype: float64
```

[ ]:
```
colors = ['blue', 'green', 'orange']

plt.figure(figsize=(8, 5))
df_CA_par_cat.plot(kind='bar', color=colors )
plt.title("Chiffre d'affaires par Catégories")
plt.xlabel('Catégories')
plt.ylabel("Chiffre d'affaires en millions d'euros")
plt.xticks(rotation=0)
plt.show()
```




Etape 6 - Chiffres d'affaires par clients et definition des clients B2B

6.1 - Distribution du CA par clients

[ ]:
```
# Calcul CA total par clients
df_CA_par_clients = df_lapage.groupby('client_id')['price'].sum().
 ↪reset_index(name='CA_total')
# Trier le df par CA
df_CA_par_clients_tri = df_CA_par_clients.sort_values(by='CA_total')
```

[ ]:
```
# Merge sur 'client_id' pour ajouter la colonne au df_lapage
df_lapage = df_lapage.merge(df_CA_par_clients, on='client_id', how='left')

df_lapage
```

[ ]:
```
       client_id    sex  age id_prod                       date session_id  \
0         c_4410  Femme   57   1_483 2021-03-13 21:35:55.949042     s_5913
1         c_4410  Femme   57  0_1111 2021-03-22 01:27:49.480137     s_9707
2         c_4410  Femme   57   1_385 2021-03-22 01:40:22.782925     s_9707
3         c_4410  Femme   57  0_1455 2021-03-22 14:29:25.189266     s_9942
4         c_4410  Femme   57  0_1420 2021-03-22 22:31:25.825764    s_10092
…            …    …  …      …                                 …          …
687529      c_84  Femme   42  0_1472 2022-05-14 00:24:49.391917   s_208110
```

```
687530      c_84  Femme   42  0_1438 2022-05-29 06:11:50.316631   s_215697
687531      c_84  Femme   42   1_459 2022-12-17 00:16:56.629536   s_313173
687532      c_84  Femme   42  0_1104 2022-12-17 00:24:14.357525   s_313173
687533      c_84  Femme   42   1_688 2022-12-17 00:36:24.148027   s_313173

        price categ  CA_total
0       15.99     1   1376.82
1       19.99     0   1376.82
2       25.99     1   1376.82
3        8.99     0   1376.82
4       11.53     0   1376.82
…       …   …       …
687529  12.49     0    427.51
687530   9.31     0    427.51
687531  15.99     1    427.51
687532  13.21     0    427.51
687533  18.19     1    427.51

[687534 rows x 9 columns]
```

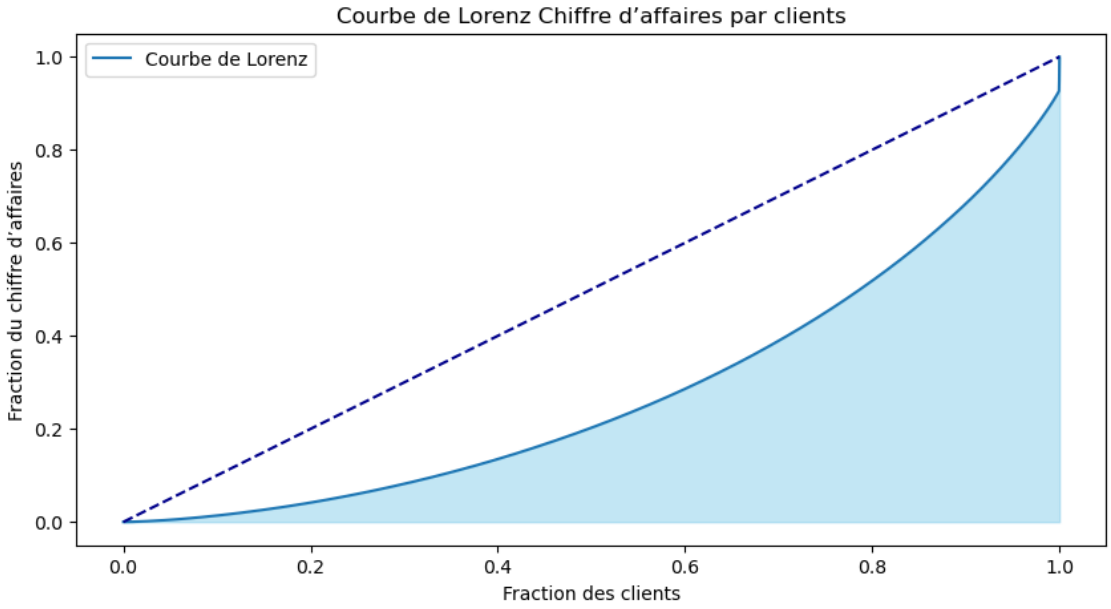
[ ]:
```
# Création du graphique de distribution des transactions par client

fig = px.box(df_CA_par_clients_tri, y='CA_total', title='Distribution du CA par␣
 ↪clients')
fig.show()
```

[ ]:
```
# Distribution du C.A par clients : Le graphique C.A par clients montre une␣
 ↪distribution avec quelques valeurs très élevées,
```

6.2 - Courbe de Lorenz et coeff de gini

[ ]:
```
income = df_CA_par_clients_tri['CA_total'].values

# Tri des valeurs et calcul de la courbe de Lorenz
sorted_income = np.sort(income)
lorenz_curve = np.cumsum(sorted_income) / np.sum(sorted_income)
lorenz_curve = np.insert(lorenz_curve, 0, 0)  # Ajouter le point (0,0)


# Création de la courbe de Lorenz
plt.figure(figsize=(10, 5))
plt.plot(np.linspace(0.0, 1.0, lorenz_curve.size), lorenz_curve, label="Courbe␣
 ↪de Lorenz")
plt.fill_between(np.linspace(0.0, 1.0, lorenz_curve.size), lorenz_curve,␣
 ↪color='skyblue', alpha=0.5)
plt.title('Courbe de Lorenz Chiffre d'affaires par clients')
plt.xlabel('Fraction des clients')
```

```
plt.ylabel("Fraction du chiffre d'affaires")
plt.plot([0,1], [0,1], color='darkblue', linestyle='--')  # Ligne d'égalité␣
↪parfaite
plt.legend()
plt.show()
```




[ ]:
```
#definir une function pour calculaler le coeff de Gini
def gini(x):
    total = 0
    for i, xi in enumerate(x[:-1], 1):
        total += np.sum(np.abs(xi - x[i:]))
    return total / (len(x)**2 * np.mean(x))

gini_lorenz = gini(income)

print("Indice de Gini :", round(gini_lorenz, 3))
```

```
Indice de Gini : 0.442
```

[ ]:
```
def gini_coefficient(x):
    diffsum = 0
    for i, xi in enumerate(x[:-1], 1):
        diffsum += np.sum(np.abs(xi - x[i:]))
    return diffsum / (len(x)**2 * np.mean(x))

gini_brown = gini_coefficient(income)
print("Indice de Gini (méthode de Brown) :", round(gini_brown, 3))
```

```
Indice de Gini (méthode de Brown) : 0.442
```

[ ]:
```
#L'indice de Gini calculé est de 0.884 :

# Cette valeur élevée de l'indice de Gini indique une inégalité dans la␣
 ↪distribution du chiffre d'affaires parmi les clients :
#  - une petite fraction des clients contribue à une grande part du chiffre␣
 ↪d'affaires total.

# Cela peut indiquer une dépendance vis à-vis de ces clients clés et souligner␣
 ↪l'importance de stratégies de diversification et de fidélisation de la␣
 ↪clientèle.
```

6.3 - Top 10 des clients par chiffre d'affaires total et détails pour chaques catégories

[ ]:
```
# Calcul du chiffre d'affaires par clients pour chaques catégories
df_CA_par_categ_clients = df_lapage.groupby(['client_id', 'categ'])['price'].
 ↪sum().reset_index(name='CA_par_categ')

df_CA_par_categ_clients
```

[ ]:
```
      client_id categ  CA_par_categ
0           c_1     0        360.15
1           c_1     1        214.00
2           c_1     2         54.87
3          c_10     0        263.87
4          c_10     1        809.77
…           …   …            …
19947     c_998     0        176.34
19948     c_998     1        242.47
19949     c_998     2       2403.41
19950     c_999     0        153.19
19951     c_999     1        548.21

[19952 rows x 3 columns]
```

[ ]:
```
# Fonction pivot pour transformer les catégories en colonnes séparées
df_CA_categ_pivot = df_CA_par_categ_clients.pivot(index='client_id',␣
 ↪columns='categ', values='CA_par_categ').fillna(0)

# Cmpréhension de liste pour renommer les colonne
df_CA_categ_pivot.columns = [f'C.A_catégories {c}' for c in df_CA_categ_pivot.
 ↪columns]

df_CA_categ_pivot
```

[ ]:
```
           C.A_catégories 0  C.A_catégories 1  C.A_catégories 2
client_id
```

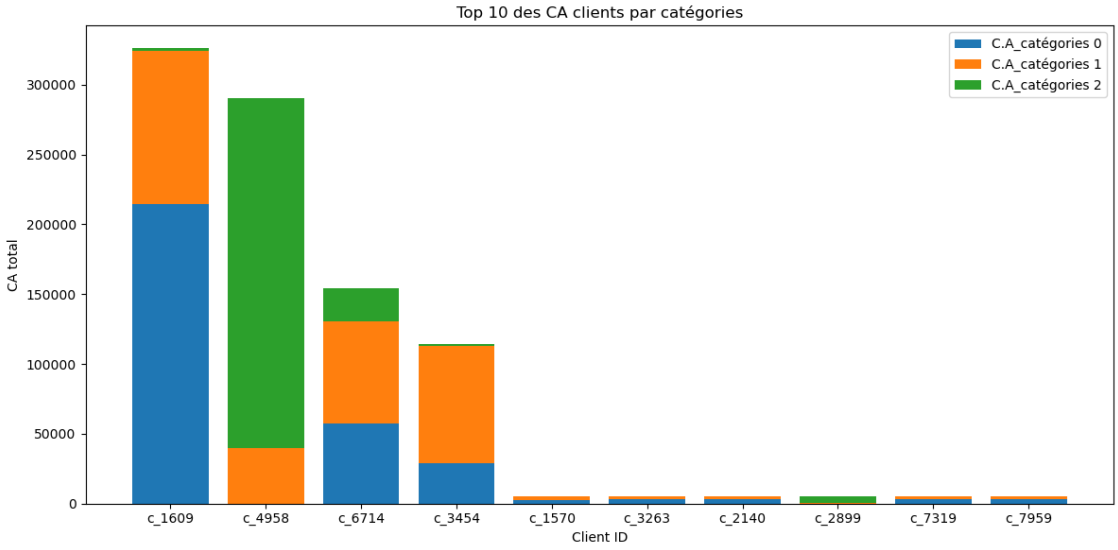
```
c_1                360.15          214.00           54.87
c_10               263.87          809.77          279.96
c_100               31.38           84.94          138.53
c_1000             386.85         1905.03            0.00
c_1001             907.73          616.12          300.00
…                   …               …               …
c_995              125.67           63.74            0.00
c_996              485.77         1151.57            0.00
c_997              207.28          471.75          810.98
c_998              176.34          242.47         2403.41
c_999              153.19          548.21            0.00

[8600 rows x 3 columns]
```

[ ]:
```
# Merge sur 'client_id' pour ajouter les colonnes au df_lapage
df_lapage = df_lapage.merge(df_CA_categ_pivot, on='client_id', how='left').
 ↪fillna(0)

df_lapage
```

[ ]:
```
       client_id    sex  age id_prod                       date session_id  \
0         c_4410  Femme   57   1_483 2021-03-13 21:35:55.949042     s_5913
1         c_4410  Femme   57  0_1111 2021-03-22 01:27:49.480137     s_9707
2         c_4410  Femme   57   1_385 2021-03-22 01:40:22.782925     s_9707
3         c_4410  Femme   57  0_1455 2021-03-22 14:29:25.189266     s_9942
4         c_4410  Femme   57  0_1420 2021-03-22 22:31:25.825764    s_10092
…          …    …  …     …                          …        …
687529      c_84  Femme   42  0_1472 2022-05-14 00:24:49.391917   s_208110
687530      c_84  Femme   42  0_1438 2022-05-29 06:11:50.316631   s_215697
687531      c_84  Femme   42   1_459 2022-12-17 00:16:56.629536   s_313173
687532      c_84  Femme   42  0_1104 2022-12-17 00:24:14.357525   s_313173
687533      c_84  Femme   42   1_688 2022-12-17 00:36:24.148027   s_313173

        price categ  CA_total  C.A_catégories 0  C.A_catégories 1  \
0       15.99     1   1376.82            501.56            729.27
1       19.99     0   1376.82            501.56            729.27
2       25.99     1   1376.82            501.56            729.27
3        8.99     0   1376.82            501.56            729.27
4       11.53     0   1376.82            501.56            729.27
…       …   …       …                 …                 …
687529  12.49     0    427.51            254.09            173.42
687530   9.31     0    427.51            254.09            173.42
687531  15.99     1    427.51            254.09            173.42
687532  13.21     0    427.51            254.09            173.42
687533  18.19     1    427.51            254.09            173.42

        C.A_catégories 2
```

```
0                  145.99
1                  145.99
2                  145.99
3                  145.99
4                  145.99
…                   …
687529               0.00
687530               0.00
687531               0.00
687532               0.00
687533               0.00

[687534 rows x 12 columns]
```

[ ]:
```python
# Recalculer le ca total pour le df df_CA_categ_pivot
df_CA_categ_pivot['CA_total'] = df_CA_categ_pivot.sum(axis=1)
df_CA_categ_pivot_tri = df_CA_categ_pivot.sort_values('CA_total',␣
 ↪ascending=False)

# Sélectionner les 10 premiers clients
top_10 = df_CA_categ_pivot_tri.head(10)

# Créer le graphique
fig, ax = plt.subplots(figsize=(12, 6))

bottom = np.zeros(10)
for category in ['C.A_catégories 0', 'C.A_catégories 1', 'C.A_catégories 2']:
    ax.bar(top_10.index, top_10[category], bottom=bottom, label=category)
    bottom += top_10[category]

ax.set_title("Top 10 des CA clients par catégories")
ax.set_xlabel("Client ID")
ax.set_ylabel("CA total")
ax.legend()
plt.tight_layout()
plt.show()
```




6.4 - Definition des clients Outlier par le CA (methode IQR)

[ ]:
```
#outlier du graphique 6.1 par le calcul (IQR)

# Définir les seuils pour les outliers graphique 6.1
Q3, Q1 = np.percentile(df_CA_par_clients_tri['CA_total'], [75, 25])
iqr = Q3 - Q1
seuil_bas = Q1 - 1.5 * iqr
seuil_haut = Q3 + 1.5 * iqr

# Identifier les outliers
outliers = df_CA_par_clients_tri[df_CA_par_clients_tri['CA_total'] > seuil_haut]

# Analyser les outliers
nombre_outliers = outliers.shape[0]
proportion_outliers = round(nombre_outliers / df_CA_par_clients_tri['CA_total'].
  ↪shape[0],2)

print(seuil_bas,seuil_haut)
print(f"Nombre d'outliers: {nombre_outliers}")
print(f"Proportion d'outliers: {proportion_outliers}")
```

```
-1289.7549999999997 3650.2049999999995
Nombre d'outliers: 251
Proportion d'outliers: 0.03
```

6.5 - Definition des clients Outlier par le CA (methode Zscore)

[ ]:
```
# Calcul de la moyenne et de l'écart-type du nombre de transactions
moyenne_transactions = df_CA_par_clients_tri['CA_total'].mean()
ecart_type_transactions = df_CA_par_clients_tri['CA_total'].std()

# Calcul du Z-score
# Z-score = (valeur - moyenne) / écart-type
df_CA_par_clients_tri['z_score'] = (df_CA_par_clients_tri['CA_total'] -␣
 ↪moyenne_transactions) / ecart_type_transactions

# Filtrer pour obtenir les client_id avec des les nombres de transactions ou le␣
 ↪Z-score est supérieur à 3
transactions_zscore_sup3 =␣
 ↪df_CA_par_clients_tri[df_CA_par_clients_tri['z_score'] > 3][['client_id',␣
 ↪'CA_total', 'z_score']]

print(transactions_zscore_sup3)
```

```
     client_id   CA_total    z_score
2724    c_3454  114110.57  21.663838
6337    c_6714  153918.60  29.315150
4388    c_4958  290227.03  55.514345
677     c_1609  326039.89  62.397765
```

[ ]:
```
# Création de la variable Clients_B2B avec les client_id ou pour le nombre de␣
 ↪transactions est supperieur a 3 z_score
Clients_B2B = transactions_zscore_sup3['client_id']
Clients_B2B
```

[ ]:
```
2724    c_3454
6337    c_6714
4388    c_4958
677     c_1609
Name: client_id, dtype: object
```

Comparativement à la methode IQR (idem sur le graphique distribution CA par clients), le Z-score être plus approprié car il se concentre sur des écarts plus significatifs afin de cibler les 4 clients B2B

6.6 - Exclusion des 4 clients B2B

[ ]:
```
df_CA_par_clients
```

[ ]:
```
     client_id  CA_total
0          c_1    629.02
1         c_10   1353.60
2        c_100    254.85
3       c_1000   2291.88
4       c_1001   1823.85
…          …        …
```

```
8595     c_995    189.41
8596     c_996   1637.34
8597     c_997   1490.01
8598     c_998   2822.22
8599     c_999    701.40

[8600 rows x 2 columns]
```

[ ]:
```
# Conserve uniquement les clients B2B du df_CA_par_clients
df_lapage_b2b = df_lapage[df_lapage['client_id'].isin(Clients_B2B)].copy()

# exclut les clients B2B du df_lapage
df_lapage = df_lapage[~df_lapage['client_id'].isin(Clients_B2B)].copy()

# Conserve uniquement les clients B2B du df_CA_par_clients
df_CA_par_clients_B2B = df_CA_par_clients[df_CA_par_clients['client_id'].
 ↪isin(Clients_B2B)].copy()

# exclut les clients B2B du df_CA_par_clients
df_CA_par_clients = df_CA_par_clients[~df_CA_par_clients['client_id'].
 ↪isin(Clients_B2B)].copy()

df_CA_par_clients_B2B
```

[ ]:
```
     client_id   CA_total
677     c_1609  326039.89
2724    c_3454  114110.57
4388    c_4958  290227.03
6337    c_6714  153918.60
```

[ ]:
```
df_CA_par_clients
```

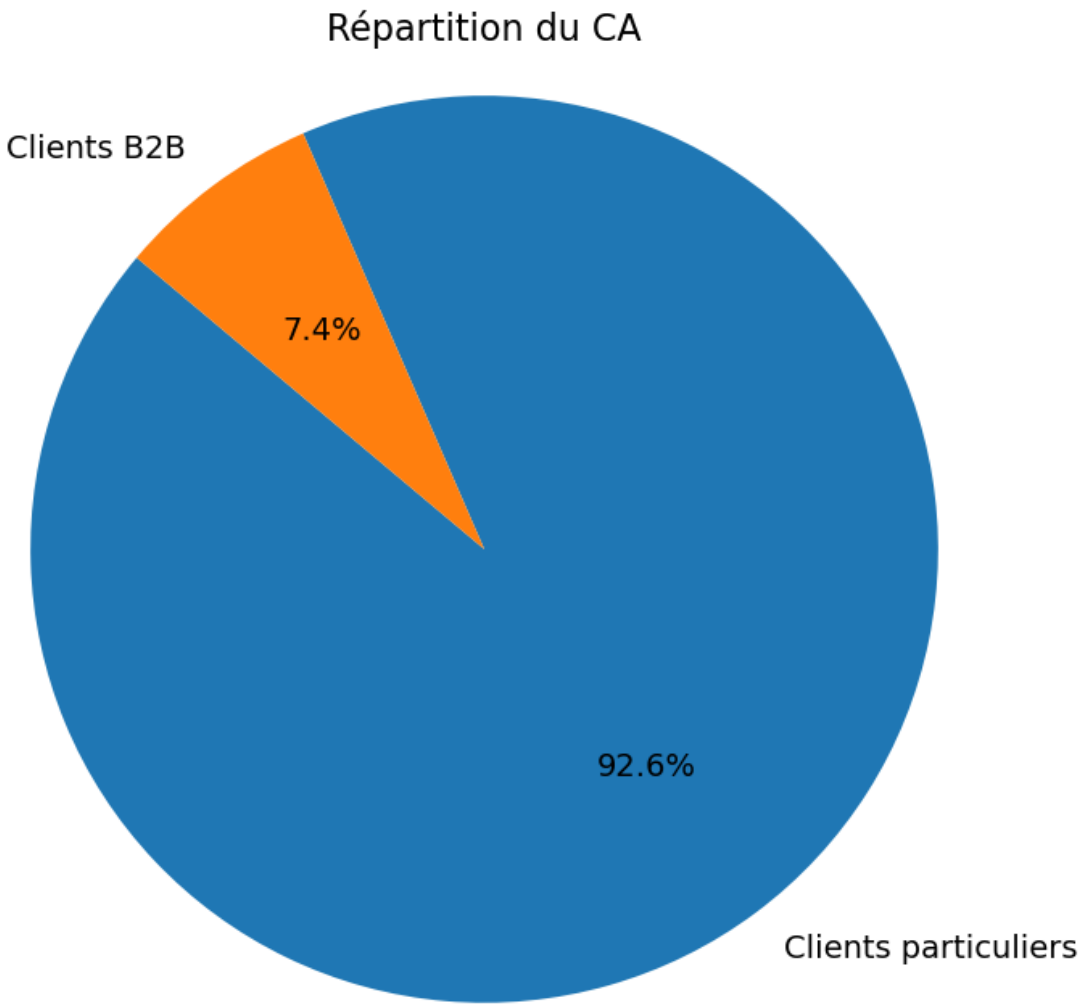
[ ]:
```
     client_id  CA_total
0          c_1    629.02
1         c_10   1353.60
2        c_100    254.85
3       c_1000   2291.88
4       c_1001   1823.85
…          …        …
8595     c_995    189.41
8596     c_996   1637.34
8597     c_997   1490.01
8598     c_998   2822.22
8599     c_999    701.40

[8596 rows x 2 columns]
```

6.7 - Répartition du chiffre d'affaires entre clients et B2B et détails des catégories pour les B2B,

[ ]:
```
df_CA_comp = pd.DataFrame({
    'Clients B2B': [df_CA_par_clients_B2B['CA_total'].sum()],
    'Clients particuliers': [df_CA_par_clients['CA_total'].sum()]
})

print(df_CA_comp)
```

```
   Clients B2B  Clients particuliers
0    884296.09           11143367.01
```

[ ]:
```
# Préparation des données pour le graphique
labels = ['Clients particuliers', 'Clients B2B']
values = [df_CA_comp['Clients particuliers'][0], df_CA_comp['Clients B2B'][0]]

# Création du pie chart
plt.figure(figsize=(8, 8))
wedges, texts, autotexts = plt.pie(values, labels=labels, autopct='%1.1f%%',␣
 ↪startangle=140, textprops={'fontsize': 14})

plt.title('Répartition du CA', fontsize = 16)
plt.axis('equal')  # Assure que le pie chart soit circulaire

# Affichage du graphique
plt.show()
```

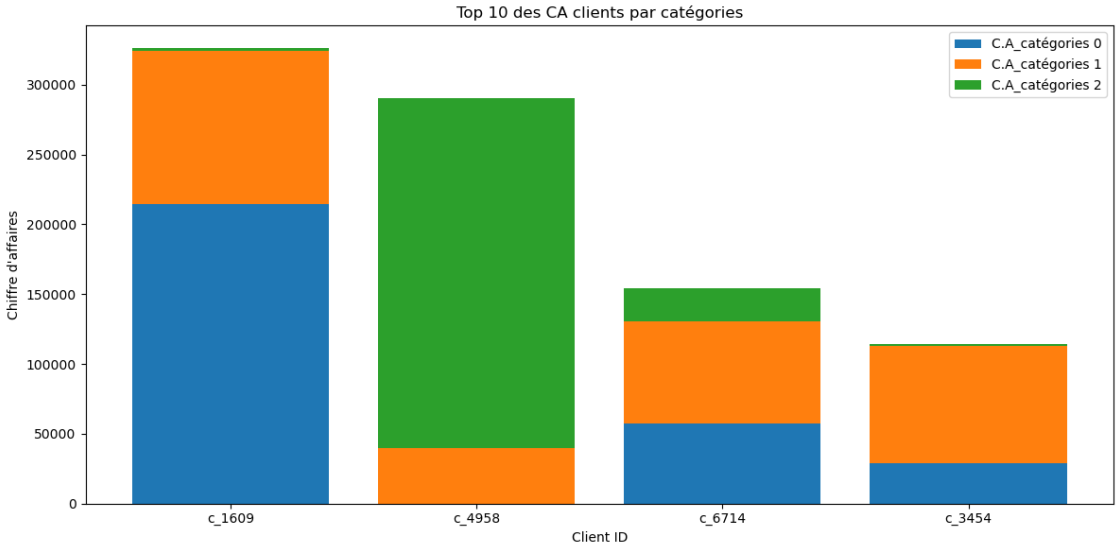



[ ]:
```
# Conserve uniquement les clients B2B dans le df_categ_pivot pour le graphique
df_CA_categ_pivot_B2B = df_CA_categ_pivot[df_CA_categ_pivot.index.
 ↪isin(Clients_B2B)]
df_CA_categ_pivot_B2B_tri = df_CA_categ_pivot_B2B.sort_values('CA_total',␣
 ↪ascending=False)
df_CA_categ_pivot_B2B_tri
```

[ ]:

| client_id | C.A_catégories 0 | C.A_catégories 1 | C.A_catégories 2 | CA_total |
|---|---|---|---|---|
| c_1609 | 214447.24 | 110091.44 | 1501.21 | 326039.89 |
| c_4958 | 48.76 | 39841.93 | 250336.34 | 290227.03 |
| c_6714 | 57254.59 | 73566.22 | 23097.79 | 153918.60 |
| c_3454 | 28779.69 | 84055.66 | 1275.22 | 114110.57 |

[ ]:
```
# Créer le graphique
fig, ax = plt.subplots(figsize=(12, 6))

bottom = np.zeros(4)
for category in ['C.A_catégories 0', 'C.A_catégories 1', 'C.A_catégories 2']:
    ax.bar(df_CA_categ_pivot_B2B_tri.index,␣
 ↪df_CA_categ_pivot_B2B_tri[category], bottom=bottom, label=category)
    bottom += df_CA_categ_pivot_B2B_tri[category]

ax.set_title("Top 10 des CA clients par catégories")
ax.set_xlabel("Client ID")
ax.set_ylabel("Chiffre d'affaires")
ax.legend()
plt.tight_layout()
plt.show()
```




6.8 - Top 10 des clients par chiffre d'affaires total et détails pour chaques catégories (clients B2B exclus)

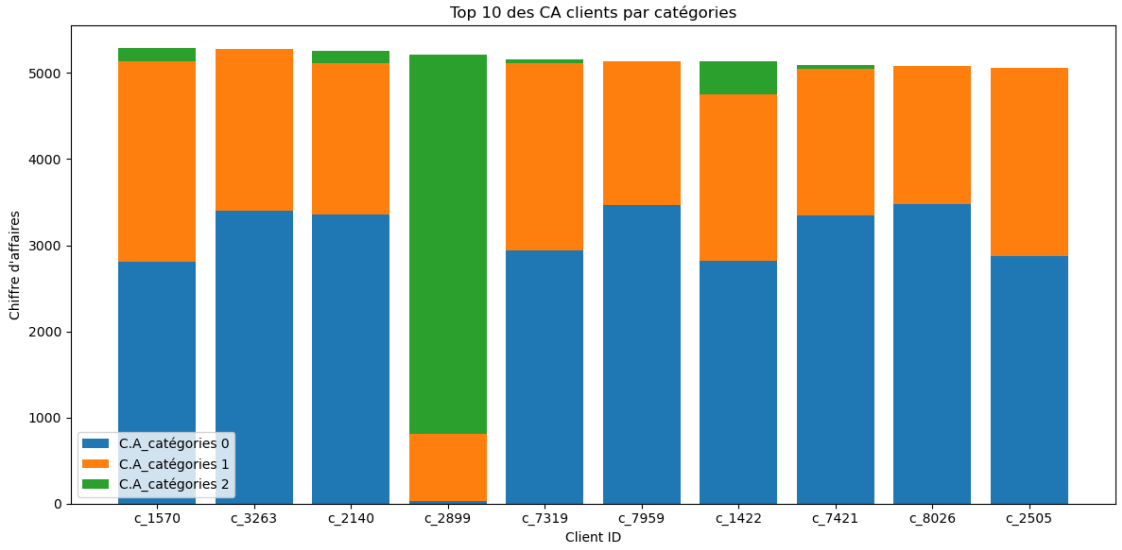
[ ]:
```
# Exclue les B2B dans le df_categ_pivot pour le graphique
df_CA_categ_pivot = df_CA_categ_pivot[~df_CA_categ_pivot.index.
 ↪isin(Clients_B2B)]
df_CA_categ_pivot_tri = df_CA_categ_pivot.sort_values('CA_total',␣
 ↪ascending=False)
df_CA_categ_pivot_tri
```

[ ]:

| client_id | C.A_catégories 0 | C.A_catégories 1 | C.A_catégories 2 | CA_total |
|---|---|---|---|---|
| c_1570 | 2812.80 | 2327.03 | 145.99 | 5285.82 |

```
c_3263              3399.29          1877.58              0.00   5276.87
c_2140              3360.28          1753.91            145.99   5260.18
c_2899                25.38           779.01           4409.66   5214.05
c_7319              2936.17          2175.61             43.99   5155.77
…                         …                …                 …        …
c_4478                13.36             0.00              0.00     13.36
c_4648                 0.00            11.20              0.00     11.20
c_8114                 9.98             0.00              0.00      9.98
c_8140                 8.30             0.00              0.00      8.30
c_8351                 6.31             0.00              0.00      6.31

[8596 rows x 4 columns]
```

[ ]:
```python
# Sélectionner les 10 premiers clients
top_10 = df_CA_categ_pivot_tri.head(10)

# Créer le graphique
fig, ax = plt.subplots(figsize=(12, 6))

bottom = np.zeros(10)
for category in ['C.A_catégories 0', 'C.A_catégories 1', 'C.A_catégories 2']:
    ax.bar(top_10.index, top_10[category], bottom=bottom, label=category)
    bottom += top_10[category]

ax.set_title("Top 10 des CA clients par catégories")
ax.set_xlabel("Client ID")
ax.set_ylabel("Chiffre d'affaires")
ax.legend()
plt.tight_layout()
plt.show()
```



6.9 - 80/20 en Chiffre d'Affaires (Analyse de Pareto)

[ ]:
```
# Calcul du chiffre d'affaires par produits
df_CA_par_Prods = df_lapage.groupby('id_prod')['price'].sum().
 ↪sort_values(ascending=False).reset_index(name='CA_par_produits')

# Calcul du cumul de CA
ca_par_Prods_cumsum = df_CA_par_Prods['CA_par_produits'].cumsum()

# Calcul du CA total et détermination du seuil 80/20
total_CA_Prods = ca_par_Prods_cumsum.iloc[-1]
pareto_threshold2 = total_CA_Prods * 0.8

# Détermination des produits qui composent les 80% du CA
top_80_Prods = df_CA_par_Prods[ca_par_Prods_cumsum <= pareto_threshold2]

# Calcul du pourcentage des produits dans le top 80%
percent_80_Prods = len(top_80_Prods) / len(df_CA_par_Prods) * 100

# Détermination des produits qui composent les 20% restants du CA
top_20_Prods = df_CA_par_Prods[ca_par_Prods_cumsum > pareto_threshold2]

# Calcul du pourcentage des produits dans le top 20%
percent_20_Prods = len(top_20_Prods) / len(df_CA_par_Prods) * 100

# Ajout d'une impression pour visualiser les résultats
print(f"Pourcentage des produits qui représentent 80% du CA: {percent_80_Prods:.
 ↪2f}%")
print(f"Pourcentage des produits qui représentent 20% du CA: {percent_20_Prods:.
 ↪2f}%")
```

```
Pourcentage des produits qui représentent 80% du CA: 21.49%
Pourcentage des produits qui représentent 20% du CA: 78.51%
```

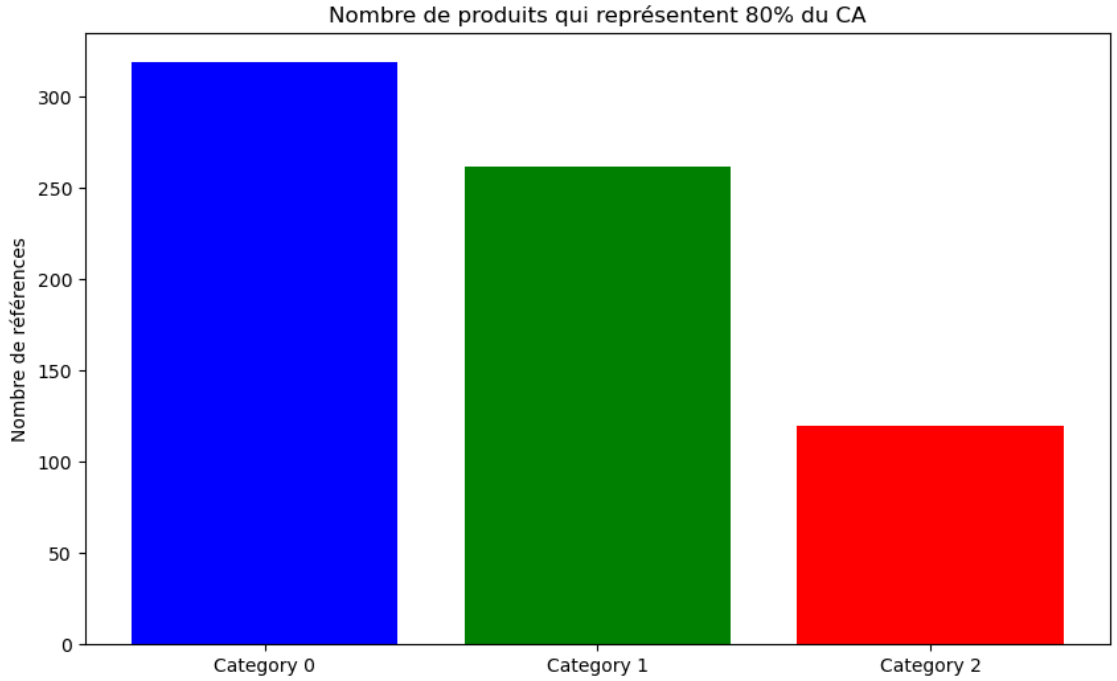
[ ]:
```
# Extraction des id_prod qui représentent 80% du CA
liste_produits_80CA = top_80_Prods['id_prod'].tolist()

# Impression des id_prod
print(f"Les id_prod qui représentent 80% du CA sont : {liste_produits_80CA}")

df_produits_CA80 = df_produits[df_produits['id_prod'].isin(liste_produits_80CA)]

# Compter le nombre de références pour chaques catégories
df_produits_CA80 = df_produits_CA80.groupby('categ')['id_prod'].count()
df_produits_CA80
```

```
Les id_prod qui représentent 80% du CA sont : ['2_159', '2_135', '2_112',
'2_102', '1_369', '1_395', '2_209', '1_414', '1_383', '2_166', '2_110', '1_498',
'2_39', '1_406', '1_366', '2_43', '1_385', '1_431', '1_378', '1_379', '1_417',
'1_389', '2_202', '1_456', '2_208', '2_167', '2_37', '2_101', '2_207', '2_108',
'2_109', '1_432', '1_396', '1_413', '2_163', '1_425', '1_403', '2_32', '1_286',
'1_397', '2_191', '2_158', '1_457', '1_437', '1_392', '1_412', '2_160', '1_426',
'1_376', '1_407', '1_279', '1_475', '2_156', '1_480', '1_400', '1_441', '1_388',
'2_62', '1_442', '1_445', '1_381', '1_468', '1_267', '1_446', '2_96', '2_155',
'1_265', '1_262', '2_104', '2_181', '2_46', '1_451', '1_256', '1_277', '1_281',
'1_287', '1_343', '1_351', '2_164', '1_282', '2_41', '2_184', '1_274', '2_147',
'1_353', '2_113', '1_462', '1_433', '0_1421', '0_1441', '1_483', '1_325',
'2_175', '1_459', '1_367', '2_150', '1_464', '1_365', '2_169', '1_466', '2_172',
'1_305', '2_162', '2_211', '1_313', '2_230', '2_79', '1_257', '1_288', '1_363',
'0_1414', '2_7', '0_1451', '2_161', '1_249', '2_185', '2_144', '1_370', '1_293',
'0_1417', '2_226', '1_337', '2_19', '1_328', '1_434', '0_1448', '2_170', '2_9',
'1_398', '0_1430', '2_151', '0_1470', '1_278', '1_320', '0_1353', '2_179',
'1_336', '1_296', '2_10', '0_1432', '1_448', '0_1342', '1_488', '0_1355',
'1_339', '1_312', '1_449', '0_1473', '1_453', '0_1358', '1_304', '1_246',
'2_136', '1_493', '2_215', '1_374', '0_1476', '1_247', '1_301', '0_1335',
'2_153', '0_1344', '1_250', '0_1479', '2_13', '1_461', '0_1366', '1_643',
'1_285', '2_157', '2_168', '1_347', '2_127', '1_350', '0_1410', '1_670',
'1_283', '2_12', '2_1', '1_260', '2_213', '0_1373', '1_271', '0_1345', '1_280',
'1_503', '2_146', '0_1390', '0_1399', '0_1425', '1_647', '1_264', '1_667',
'0_1357', '1_352', '1_435', '0_1334', '1_652', '1_436', '1_358', '0_1349',
'1_444', '0_1435', '0_1363', '1_653', '1_316', '0_1388', '2_145', '0_1590',
'1_618', '0_1426', '2_0', '2_141', '1_635', '1_644', '0_1407', '0_1469',
'2_228', '0_1350', '2_137', '0_1578', '0_1422', '2_173', '0_1485', '0_1471',
'2_148', '1_348', '1_251', '0_1609', '0_1403', '1_616', '0_1397', '0_1376',
'0_1431', '1_364', '2_227', '2_115', '1_495', '0_1394', '0_1452', '0_1621',
'1_615', '2_139', '0_1545', '0_1340', '1_318', '0_1606', '0_1475', '0_1517',
'0_1472', '0_1420', '0_1412', '1_253', '1_661', '2_140', '0_1604', '0_1521',
'0_1382', '1_362', '0_1384', '0_1457', '1_311', '2_138', '1_660', '0_1635',
'1_333', '1_710', '0_1587', '0_1567', '0_1378', '1_712', '0_1599', '1_612',
'0_1583', '0_1461', '0_1569', '0_1445', '1_649', '1_735', '2_143', '1_242',
'0_1377', '0_1556', '1_625', '0_1121', '1_701', '0_1580', '0_1362', '0_1474',
'0_1416', '0_1348', '1_634', '1_544', '2_223', '0_1383', '0_1393', '2_165',
'0_1386', '0_1600', '0_1411', '2_225', '0_1641', '2_154', '1_651', '1_259',
'1_706', '2_116', '0_1438', '2_26', '1_708', '0_1503', '0_1401', '0_1434',
'0_1505', '0_1111', '0_1419', '1_734', '0_1330', '0_1586', '1_695', '0_1562',
'0_1626', '0_989', '0_1424', '1_669', '1_622', '0_1157', '0_1542', '1_504',
'0_1555', '0_1540', '1_730', '0_1144', '1_686', '1_663', '0_1616', '1_310',
'1_676', '0_1584', '0_1367', '2_236', '0_1396', '0_1370', '0_1385', '0_1387',
'0_1630', '1_606', '2_222', '0_1493', '1_299', '0_1133', '0_1532', '1_303',
'0_1612', '0_1124', '0_1446', '0_1325', '2_217', '2_142', '1_677', '1_248',
'0_1429', '0_1326', '0_1598', '0_1455', '1_597', '0_1131', '0_1094', '1_538',
'0_1543', '1_523', '0_1338', '1_718', '1_721', '1_308', '1_522', '0_1083',
'1_507', '0_1108', '0_1464', '0_1418', '1_724', '0_1608', '0_1351', '0_1596',
'2_120', '0_1341', '0_1553', '2_119', '1_631', '0_1591', '1_621', '1_688',
```

```
'2_235', '0_1453', '0_1524', '0_1611', '1_698', '0_1113', '0_1551', '0_1617',
'0_1103', '1_521', '2_233', '1_617', '2_210', '0_1631', '0_1059', '0_1061',
'1_694', '1_727', '1_524', '1_607', '2_212', '0_1564', '0_1042', '1_516',
'0_1082', '0_1516', '0_1339', '0_991', '1_611', '1_558', '0_1501', '0_1328',
'1_582', '2_149', '1_542', '0_1106', '2_187', '0_1450', '0_1073', '2_190',
'2_221', '0_1560', '2_231', '0_1104', '2_237', '0_1546', '0_1090', '0_1640',
'0_1072', '2_220', '0_1566', '0_1160', '1_640', '0_1499', '0_1404', '0_1458',
'0_1574', '0_1454', '1_510', '0_1514', '0_1057', '0_1636', '1_702', '1_673',
'0_1443', '0_1628', '1_13', '2_171', '0_1354', '0_1603', '1_685', '2_214',
'0_1440', '2_232', '0_1632', '0_1528', '1_601', '2_189', '0_1166', '0_1130',
'1_579', '1_639', '0_1482', '2_194', '2_224', '0_1067', '0_1541', '2_203',
'1_619', '1_589', '0_1182', '0_1009', '1_593', '0_1097', '0_1156', '2_219',
'1_53', '0_1490', '1_515', '0_1558', '0_1125', '0_1468', '0_1272', '0_1391',
'1_586', '1_527', '2_216', '0_1380', '1_683', '0_1258', '0_1003', '1_592',
'0_1520', '1_245', '0_1613', '0_1559', '0_1489', '1_530', '0_1409', '0_1297',
'2_40', '0_1307', '0_1011', '0_1065', '1_547', '0_1162', '2_3', '1_508',
'0_1247', '2_176', '0_1298', '0_1173', '0_1081', '1_580', '0_1134', '1_559',
'0_1506', '0_1347', '0_1375', '1_157', '0_1442', '0_1030', '0_1309', '1_564',
'0_1264', '0_1572', '0_1096', '0_1513', '0_1049', '0_1639', '1_731', '1_10',
'0_1236', '0_1075', '1_529', '1_692', '0_1405', '1_630', '0_1356', '0_1308',
'0_1548', '0_1034', '0_1333', '1_723', '0_1623', '0_1152', '0_1392', '0_1593',
'0_1183', '0_1099', '1_556', '0_1565', '0_1', '1_87', '0_1050', '1_550',
'1_581', '0_1126', '0_1091', '1_537', '0_1494', '0_1622', '1_679', '0_1321',
'1_505', '0_1123', '1_34', '1_541', '0_1243', '1_49', '1_715', '1_568',
'0_1508', '0_1143', '1_566', '0_1319', '2_205', '0_1381', '0_1311', '0_1547',
'0_1291', '0_1306', '0_1188', '2_196', '1_736', '0_1460', '1_632', '0_996',
'0_1573', '0_1158', '0_1323', '0_1447', '0_1287', '2_182', '1_11', '1_572',
'0_1527', '0_1467', '0_1449', '0_1313', '0_2069', '0_1502', '0_1105', '0_1253',
'0_1039', '0_1437', '0_1015', '0_1605', '2_180', '1_535', '0_1060', '1_570',
'0_1249', '0_1281', '0_1132', '1_720', '0_993', '0_1368', '1_520', '0_1189',
'1_565', '1_585', '0_1398', '1_614', '1_241', '1_187', '0_1643', '1_726',
'0_1538', '1_183', '0_1481', '0_1053', '0_1199', '0_0', '0_1216', '0_1038',
'1_91', '0_1289', '0_1018', '1_162', '0_1352', '0_2002', '2_218', '0_1557',
'0_1238', '1_158', '1_684', '0_1043', '0_1256', '0_1224', '2_229', '2_206',
'0_997', '0_1192', '0_1026', '0_1428', '0_1260', '1_560', '0_1480', '1_45',
'0_1112', '1_548', '1_534', '1_165', '1_60', '1_161', '1_738', '0_1280',
'0_998', '1_43', '0_1163', '1_583', '0_1180', '1_506', '0_1400', '1_192',
'0_1320', '0_1360', '0_1210', '2_174', '1_562', '1_664', '0_1465', '0_1571',
'0_1638', '1_61', '1_54', '1_536']
```

[ ]:
```
categ
0    319
1    262
2    120
Name: id_prod, dtype: int64
```

[ ]:
```
# Couleurs pour les catégories
colors = ['blue', 'green', 'red']


plt.figure(figsize=(10, 6))
plt.bar(['Category 0', 'Category 1', 'Category 2'],
        [df_produits_CA80.iloc[0], df_produits_CA80.iloc[1], df_produits_CA80.
 ↪iloc[2]],
        color=colors)

plt.title("Nombre de produits qui représentent 80% du CA")
plt.ylabel('Nombre de références')

plt.show()
```




[ ]:
```
# Calcul du cumul de CA
ca_par_clients_cumsum = df_CA_par_clients['CA_total'].cumsum()

# Calcul du CA total et détermination du seuil 80/20
total_CA = ca_par_clients_cumsum.iloc[-1]
pareto_threshold = total_CA * 0.8

# Détermination des clients qui composent les 80% du CA
top_80_clients = df_CA_par_clients[ca_par_clients_cumsum <= pareto_threshold]
```

```
# Calcul du pourcentage des clients dans le top 80%
percent_80_clients = len(top_80_clients) / len(df_CA_par_clients) * 100

# Détermination des clients qui composent les 20% restants du CA
top_20_clients = df_CA_par_clients[ca_par_clients_cumsum > pareto_threshold]

# Calcul du pourcentage des clients dans le top 20%
percent_20_clients = len(top_20_clients) / len(df_CA_par_clients) * 100

# Ajout d'une impression pour visualiser les résultats
print(f"Pourcentage des clients qui génèrent 80% du CA: {percent_80_clients:.
 ↪2f}%")
print(f"Pourcentage des clients qui génèrent 20% du CA: {percent_20_clients:.
 ↪2f}%")
```

```
Pourcentage des clients qui génèrent 80% du CA: 79.99%
Pourcentage des clients qui génèrent 20% du CA: 20.01%
```

Etape 7 - Indicateurs des ventes (KPI's)

7.1 - Nombre de transactions par clients et par categories

[ ]:
```
# Group by client_id puis count transactions
df_transactions_groupés = df_lapage.groupby('client_id').size().
 ↪reset_index(name='Nombre de transactions')

df_transactions_groupés
```

[ ]:

| | client_id | Nombre de transactions |
|---|---|---|
| 0 | c_1 | 43 |
| 1 | c_10 | 58 |
| 2 | c_100 | 8 |
| 3 | c_1000 | 126 |
| 4 | c_1001 | 103 |
| … | … | … |
| 8591 | c_995 | 14 |
| 8592 | c_996 | 96 |
| 8593 | c_997 | 59 |
| 8594 | c_998 | 55 |
| 8595 | c_999 | 46 |

[8596 rows x 2 columns]

[ ]:
```
# création du graphique nombre de transactions par client

fig = px.box(df_transactions_groupés, y='Nombre de transactions', title='Nombre␣
 ↪de transactions par client')
```

```
fig.show()
```

[ ]:
```
# Jointure avec df_lapage pour ajouter la colonne Nombre de transactions
df_lapage = df_lapage.merge(df_transactions_groupés, on='client_id', how='left')
df_lapage
```

[ ]:

| | client_id | sex | age | id_prod | date | session_id | \ |
|---|---|---|---|---|---|---|---|
| 0 | c_4410 | Femme | 57 | 1_483 | 2021-03-13 21:35:55.949042 | s_5913 | |
| 1 | c_4410 | Femme | 57 | 0_1111 | 2021-03-22 01:27:49.480137 | s_9707 | |
| 2 | c_4410 | Femme | 57 | 1_385 | 2021-03-22 01:40:22.782925 | s_9707 | |
| 3 | c_4410 | Femme | 57 | 0_1455 | 2021-03-22 14:29:25.189266 | s_9942 | |
| 4 | c_4410 | Femme | 57 | 0_1420 | 2021-03-22 22:31:25.825764 | s_10092 | |
| … | … | … | … | … | … | … | |
| 640729 | c_84 | Femme | 42 | 0_1472 | 2022-05-14 00:24:49.391917 | s_208110 | |
| 640730 | c_84 | Femme | 42 | 0_1438 | 2022-05-29 06:11:50.316631 | s_215697 | |
| 640731 | c_84 | Femme | 42 | 1_459 | 2022-12-17 00:16:56.629536 | s_313173 | |
| 640732 | c_84 | Femme | 42 | 0_1104 | 2022-12-17 00:24:14.357525 | s_313173 | |
| 640733 | c_84 | Femme | 42 | 1_688 | 2022-12-17 00:36:24.148027 | s_313173 | |

| | price | categ | CA_total | C.A_catégories 0 | C.A_catégories 1 | \ |
|---|---|---|---|---|---|---|
| 0 | 15.99 | 1 | 1376.82 | 501.56 | 729.27 | |
| 1 | 19.99 | 0 | 1376.82 | 501.56 | 729.27 | |
| 2 | 25.99 | 1 | 1376.82 | 501.56 | 729.27 | |
| 3 | 8.99 | 0 | 1376.82 | 501.56 | 729.27 | |
| 4 | 11.53 | 0 | 1376.82 | 501.56 | 729.27 | |
| … | … | … | … | … | … | |
| 640729 | 12.49 | 0 | 427.51 | 254.09 | 173.42 | |
| 640730 | 9.31 | 0 | 427.51 | 254.09 | 173.42 | |
| 640731 | 15.99 | 1 | 427.51 | 254.09 | 173.42 | |
| 640732 | 13.21 | 0 | 427.51 | 254.09 | 173.42 | |
| 640733 | 18.19 | 1 | 427.51 | 254.09 | 173.42 | |

| | C.A_catégories 2 | Nombre de transactions |
|---|---|---|
| 0 | 145.99 | 76 |
| 1 | 145.99 | 76 |
| 2 | 145.99 | 76 |
| 3 | 145.99 | 76 |
| 4 | 145.99 | 76 |
| … | … | … |
| 640729 | 0.00 | 28 |
| 640730 | 0.00 | 28 |
| 640731 | 0.00 | 28 |
| 640732 | 0.00 | 28 |
| 640733 | 0.00 | 28 |

[640734 rows x 13 columns]

[ ]:
```
# Calcul du nb_transactions par catégories pour chaques clients
df_Transactions_par_categ_client = df_lapage.groupby(['client_id',␣
↪'categ'])['Nombre de transactions'].size().
↪reset_index(name='Nb_transactions_par_categ')

df_Transactions_par_categ_client.head()
```

[ ]:
```
  client_id categ  Nb_transactions_par_categ
0       c_1     0                         30
1       c_1     1                         12
2       c_1     2                          1
3      c_10     0                         20
4      c_10     1                         34
```

[ ]:
```
# Création d'un boxplot nombre de transactions par clients et par catégories
fig = px.box(df_lapage, x='categ', y='Nombre de transactions', color='categ',
             title='Nombre de transactions par clients et par catégories')
fig.update_layout(xaxis_title='Catégories', yaxis_title='Prix', boxgap=0.4)

fig.show()
```

[ ]:
```
# Fonction pivot pour transformer les catégories en colonnes séparées
df_transactions_categ_pivot = df_Transactions_par_categ_client.
↪pivot(index='client_id', columns='categ',␣
↪values='Nb_transactions_par_categ').fillna(0)

# Cmpréhension de liste pour renommer les colonne
df_transactions_categ_pivot.columns = [f'Nombre de transactions catégorie {c}'␣
↪for c in df_transactions_categ_pivot.columns]

# Merge les deux DataFrames sur 'client_id'
df_lapage = df_lapage.merge(df_transactions_categ_pivot, on='client_id',␣
↪how='left').fillna(0)
df_lapage
```

[ ]:
```
       client_id    sex  age id_prod                       date session_id  \
0          c_4410  Femme   57   1_483 2021-03-13 21:35:55.949042     s_5913
1          c_4410  Femme   57  0_1111 2021-03-22 01:27:49.480137     s_9707
2          c_4410  Femme   57   1_385 2021-03-22 01:40:22.782925     s_9707
3          c_4410  Femme   57  0_1455 2021-03-22 14:29:25.189266     s_9942
4          c_4410  Femme   57  0_1420 2021-03-22 22:31:25.825764    s_10092
…             …     …  …       …                          …          …
640729       c_84  Femme   42  0_1472 2022-05-14 00:24:49.391917   s_208110
640730       c_84  Femme   42  0_1438 2022-05-29 06:11:50.316631   s_215697
640731       c_84  Femme   42   1_459 2022-12-17 00:16:56.629536   s_313173
640732       c_84  Femme   42  0_1104 2022-12-17 00:24:14.357525   s_313173
640733       c_84  Femme   42   1_688 2022-12-17 00:36:24.148027   s_313173
```

```
        price categ  CA_total  C.A_catégories 0  C.A_catégories 1  \
0       15.99     1   1376.82            501.56            729.27
1       19.99     0   1376.82            501.56            729.27
2       25.99     1   1376.82            501.56            729.27
3        8.99     0   1376.82            501.56            729.27
4       11.53     0   1376.82            501.56            729.27
…       …   …       …                 …                 …
640729  12.49     0    427.51            254.09            173.42
640730   9.31     0    427.51            254.09            173.42
640731  15.99     1    427.51            254.09            173.42
640732  13.21     0    427.51            254.09            173.42
640733  18.19     1    427.51            254.09            173.42

        C.A_catégories 2  Nombre de transactions  \
0                 145.99                      76
1                 145.99                      76
2                 145.99                      76
3                 145.99                      76
4                 145.99                      76
…                  …                         …
640729              0.00                      28
640730              0.00                      28
640731              0.00                      28
640732              0.00                      28
640733              0.00                      28

        Nombre de transactions catégorie 0  \
0                                      38.0
1                                      38.0
2                                      38.0
3                                      38.0
4                                      38.0
…                                     …
640729                                 21.0
640730                                 21.0
640731                                 21.0
640732                                 21.0
640733                                 21.0

        Nombre de transactions catégorie 1  Nombre de transactions catégorie 2
0                                      37.0                                 1.0
1                                      37.0                                 1.0
2                                      37.0                                 1.0
3                                      37.0                                 1.0
4                                      37.0                                 1.0
…                                     …                                    …
```

```
640729                                7.0                                0.0
640730                                7.0                                0.0
640731                                7.0                                0.0
640732                                7.0                                0.0
640733                                7.0                                0.0

[640734 rows x 16 columns]
```

7.2 - Evolution du nombre de transactions (mensuel)

[ ]:
```
#Nombre de transactions total :

NbDeTransactionsTotal = len(df_lapage)

print("Nombre transactions total : ", NbDeTransactionsTotal)
```

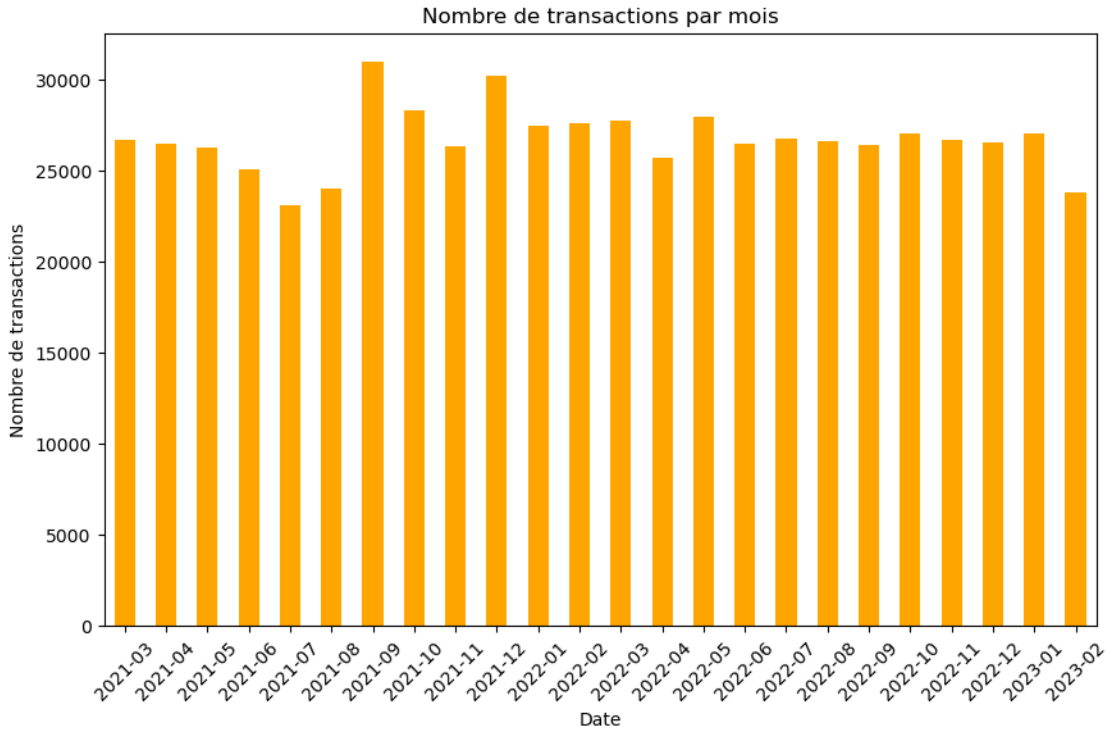
```
Nombre transactions total :  640734
```

[ ]:
```
# Extraire l'année et le mois puis grouper par cette nouvelle colonne
df_lapage['Date'] = df_lapage['date'].dt.to_period('M')
df_transactions_mensuel = df_lapage.groupby('Date').size()

print(df_transactions_mensuel)
```

```
Date
2021-03    26647
2021-04    26470
2021-05    26272
2021-06    25038
2021-07    23075
2021-08    23955
2021-09    30975
2021-10    28313
2021-11    26319
2021-12    30208
2022-01    27416
2022-02    27569
2022-03    27703
2022-04    25659
2022-05    27946
2022-06    26479
2022-07    26715
2022-08    26622
2022-09    26387
2022-10    27014
2022-11    26641
2022-12    26519
2023-01    27019
2023-02    23773
```

```
Freq: M, dtype: int64
```

[ ]:
```
plt.figure(figsize=(10, 6))
df_transactions_mensuel.plot(kind='bar', color='orange')
plt.title('Nombre de transactions par mois')
plt.ylabel('Nombre de transactions')
plt.xticks(rotation=45)
plt.show()
```




7.3 - Nombre de sessions par clients

[ ]:
```
# Calcul du nombre de sessions uniques par client
df_Nb_de_sessions = df_lapage.groupby('client_id')['session_id'].nunique().
  ↪reset_index(name='Nb_Session')

# Fusion avec le DataFrame principal
df_lapage = df_lapage.merge(df_Nb_de_sessions, on='client_id', how='left')

df_lapage
```

[ ]:
```
      client_id    sex  age id_prod                       date session_id  \
0         c_4410  Femme   57   1_483 2021-03-13 21:35:55.949042     s_5913
1         c_4410  Femme   57  0_1111 2021-03-22 01:27:49.480137     s_9707
2         c_4410  Femme   57   1_385 2021-03-22 01:40:22.782925     s_9707
```

```
3         c_4410  Femme   57  0_1455 2021-03-22 14:29:25.189266     s_9942
4         c_4410  Femme   57  0_1420 2021-03-22 22:31:25.825764    s_10092
…           …    …  …     …                          …         …
640729      c_84  Femme   42  0_1472 2022-05-14 00:24:49.391917   s_208110
640730      c_84  Femme   42  0_1438 2022-05-29 06:11:50.316631   s_215697
640731      c_84  Femme   42   1_459 2022-12-17 00:16:56.629536   s_313173
640732      c_84  Femme   42  0_1104 2022-12-17 00:24:14.357525   s_313173
640733      c_84  Femme   42   1_688 2022-12-17 00:36:24.148027   s_313173

        price categ  CA_total  C.A_catégories 0  C.A_catégories 1  \
0       15.99     1   1376.82            501.56            729.27
1       19.99     0   1376.82            501.56            729.27
2       25.99     1   1376.82            501.56            729.27
3        8.99     0   1376.82            501.56            729.27
4       11.53     0   1376.82            501.56            729.27
…       …   …       …                 …                 …
640729  12.49     0    427.51            254.09            173.42
640730   9.31     0    427.51            254.09            173.42
640731  15.99     1    427.51            254.09            173.42
640732  13.21     0    427.51            254.09            173.42
640733  18.19     1    427.51            254.09            173.42

        C.A_catégories 2  Nombre de transactions  \
0                 145.99                      76
1                 145.99                      76
2                 145.99                      76
3                 145.99                      76
4                 145.99                      76
…                  …                        …
640729              0.00                      28
640730              0.00                      28
640731              0.00                      28
640732              0.00                      28
640733              0.00                      28

        Nombre de transactions catégorie 0  \
0                                     38.0
1                                     38.0
2                                     38.0
3                                     38.0
4                                     38.0
…                                     …
640729                                21.0
640730                                21.0
640731                                21.0
640732                                21.0
640733                                21.0
```

```
        Nombre de transactions catégorie 1  \
0                                     37.0
1                                     37.0
2                                     37.0
3                                     37.0
4                                     37.0
…                                      …
640729                                 7.0
640730                                 7.0
640731                                 7.0
640732                                 7.0
640733                                 7.0

        Nombre de transactions catégorie 2     Date  Nb_Session
0                                     1.0  2021-03          52
1                                     1.0  2021-03          52
2                                     1.0  2021-03          52
3                                     1.0  2021-03          52
4                                     1.0  2021-03          52
…                                      …       …           …
640729                                0.0  2022-05          14
640730                                0.0  2022-05          14
640731                                0.0  2022-12          14
640732                                0.0  2022-12          14
640733                                0.0  2022-12          14

[640734 rows x 18 columns]
```

7.4 - Panier moyen par clients

[ ]:
```
# Calcul du panier moyen par client
df_lapage['Panier_Moyen'] = round((df_lapage['CA_total'] /␣
 ↪df_lapage['Nb_Session']),2)

df_lapage
```

[ ]:
```
       client_id    sex  age id_prod                       date session_id  \
0         c_4410  Femme   57   1_483 2021-03-13 21:35:55.949042     s_5913
1         c_4410  Femme   57  0_1111 2021-03-22 01:27:49.480137     s_9707
2         c_4410  Femme   57   1_385 2021-03-22 01:40:22.782925     s_9707
3         c_4410  Femme   57  0_1455 2021-03-22 14:29:25.189266     s_9942
4         c_4410  Femme   57  0_1420 2021-03-22 22:31:25.825764    s_10092
…           …     …  …      …                             …         …
640729      c_84  Femme   42  0_1472 2022-05-14 00:24:49.391917   s_208110
640730      c_84  Femme   42  0_1438 2022-05-29 06:11:50.316631   s_215697
640731      c_84  Femme   42   1_459 2022-12-17 00:16:56.629536   s_313173
640732      c_84  Femme   42  0_1104 2022-12-17 00:24:14.357525   s_313173
```

```
640733      c_84  Femme   42   1_688 2022-12-17 00:36:24.148027   s_313173

        price categ  CA_total  C.A_catégories 0  C.A_catégories 1  \
0       15.99     1   1376.82            501.56            729.27
1       19.99     0   1376.82            501.56            729.27
2       25.99     1   1376.82            501.56            729.27
3        8.99     0   1376.82            501.56            729.27
4       11.53     0   1376.82            501.56            729.27
…       …   …       …                 …                 …
640729  12.49     0    427.51            254.09            173.42
640730   9.31     0    427.51            254.09            173.42
640731  15.99     1    427.51            254.09            173.42
640732  13.21     0    427.51            254.09            173.42
640733  18.19     1    427.51            254.09            173.42

        C.A_catégories 2  Nombre de transactions  \
0                 145.99                      76
1                 145.99                      76
2                 145.99                      76
3                 145.99                      76
4                 145.99                      76
…                  …                         …
640729              0.00                      28
640730              0.00                      28
640731              0.00                      28
640732              0.00                      28
640733              0.00                      28

        Nombre de transactions catégorie 0  \
0                                     38.0
1                                     38.0
2                                     38.0
3                                     38.0
4                                     38.0
…                                     …
640729                                21.0
640730                                21.0
640731                                21.0
640732                                21.0
640733                                21.0

        Nombre de transactions catégorie 1  \
0                                     37.0
1                                     37.0
2                                     37.0
3                                     37.0
4                                     37.0
```

```
…                                       …
640729                                  7.0
640730                                  7.0
640731                                  7.0
640732                                  7.0
640733                                  7.0

        Nombre de transactions catégorie 2     Date  Nb_Session  Panier_Moyen
0                                      1.0  2021-03          52         26.48
1                                      1.0  2021-03          52         26.48
2                                      1.0  2021-03          52         26.48
3                                      1.0  2021-03          52         26.48
4                                      1.0  2021-03          52         26.48
…                                      …    …           …            …
640729                                 0.0  2022-05          14         30.54
640730                                 0.0  2022-05          14         30.54
640731                                 0.0  2022-12          14         30.54
640732                                 0.0  2022-12          14         30.54
640733                                 0.0  2022-12          14         30.54

[640734 rows x 19 columns]
```

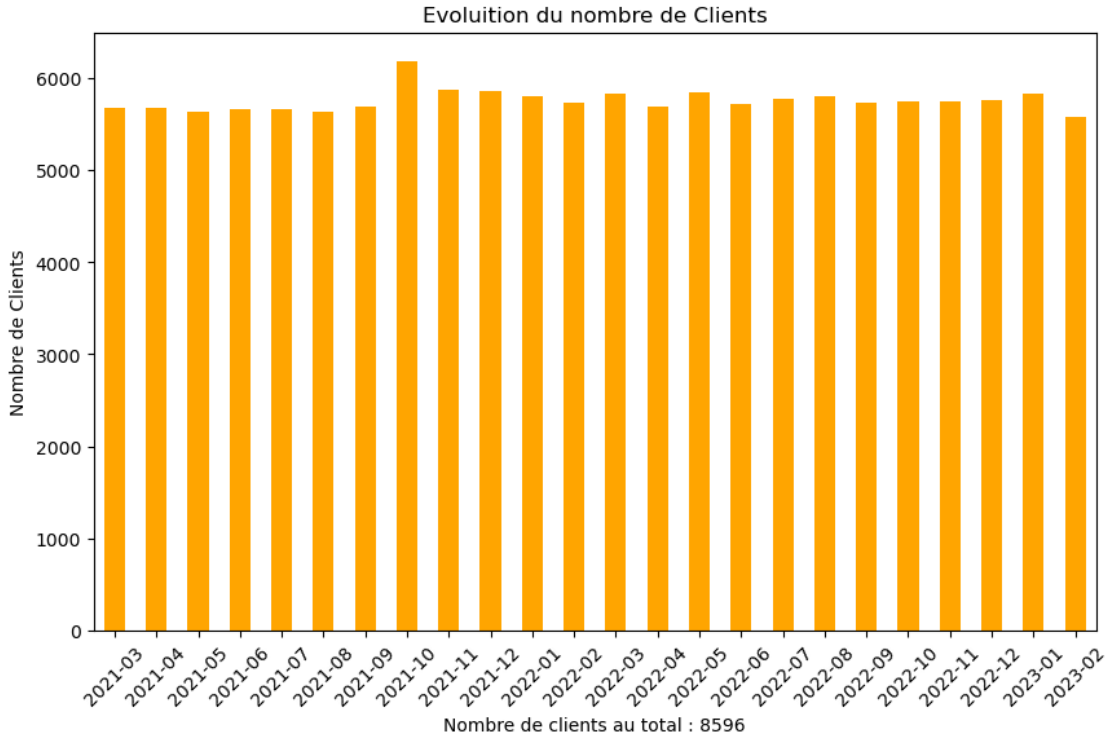
7.5 - Evolution du nombre de clients (mensuel)

[ ]:
```
df_Dateindex['Date'] = df_Dateindex.index.to_period('M')
df_clients_par_mois = df_Dateindex.groupby('Date')['client_id'].nunique()
df_clients_par_mois
```

[ ]:
```
Date
2021-03    5676
2021-04    5674
2021-05    5644
2021-06    5659
2021-07    5672
2021-08    5642
2021-09    5693
2021-10    6190
2021-11    5875
2021-12    5867
2022-01    5809
2022-02    5729
2022-03    5835
2022-04    5695
2022-05    5843
2022-06    5717
2022-07    5778
2022-08    5805
2022-09    5738
```

```
2022-10    5755
2022-11    5749
2022-12    5765
2023-01    5828
2023-02    5587
Freq: M, Name: client_id, dtype: int64
```

[ ]:
```
plt.figure(figsize=(10, 6))
df_clients_par_mois.plot(kind='bar', color='orange')
plt.title('Evoluition du nombre de Clients')
plt.xlabel('Nombre de clients au total : 8596')
plt.ylabel('Nombre de Clients')
plt.xticks(rotation=45)
plt.show()
```




7.6 - Nombre de références vendus par mois

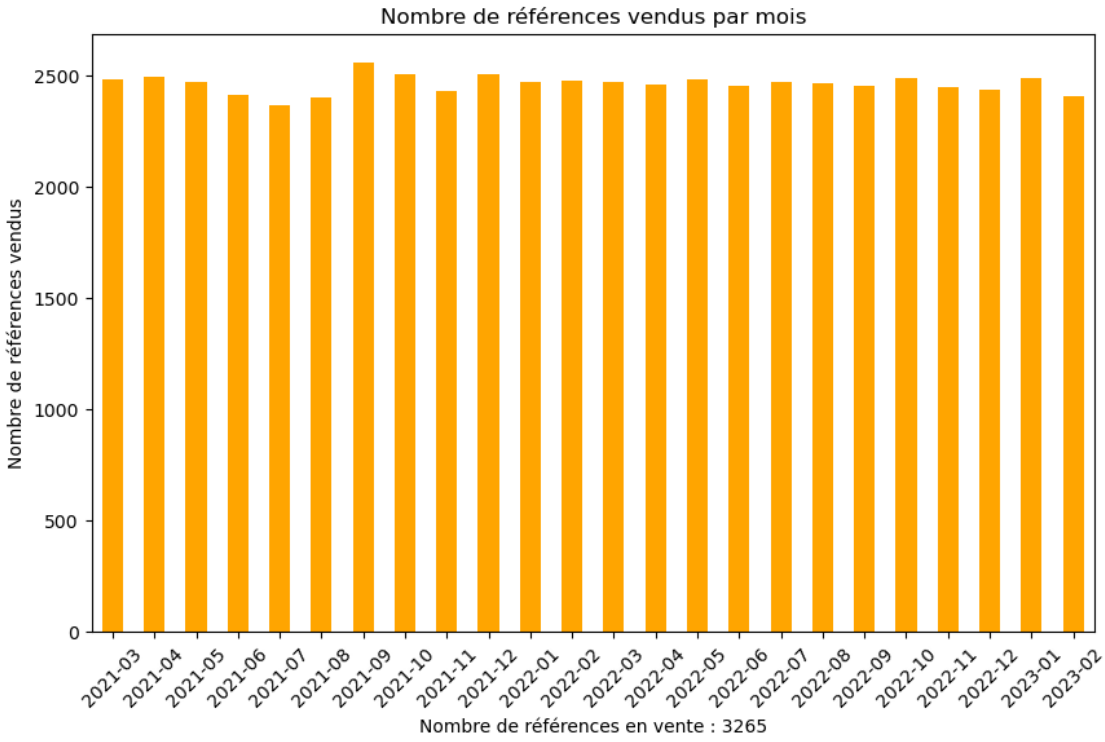
[ ]:
```
#Nombre de références au total :

NbDeReferencesTotal = df_lapage['id_prod'].nunique()

print("Nombre de références au total :", NbDeReferencesTotal)
```

```
Nombre de références au total : 3262
```

[ ]:
```
df_Dateindex['Date'] = df_Dateindex.index.to_period('M')
df_nb_de_references = df_Dateindex.groupby('Date')['id_prod'].nunique()
df_nb_de_references
```

[ ]:
```
Date
2021-03    2482
2021-04    2492
2021-05    2471
2021-06    2414
2021-07    2369
2021-08    2404
2021-09    2560
2021-10    2506
2021-11    2432
2021-12    2505
2022-01    2470
2022-02    2477
2022-03    2474
2022-04    2462
2022-05    2485
2022-06    2451
2022-07    2470
2022-08    2463
2022-09    2452
2022-10    2489
2022-11    2448
2022-12    2438
2023-01    2488
2023-02    2407
Freq: M, Name: id_prod, dtype: int64
```

[ ]:
```
plt.figure(figsize=(10, 6))
df_nb_de_references.plot(kind='bar', color='orange')
plt.title('Nombre de références vendus par mois')
plt.xlabel('Nombre de références en vente : 3265')
plt.ylabel('Nombre de références vendus')
plt.xticks(rotation=45)
plt.show()
```




7.7 - Tops et Flops Références Vendues par An

[ ]:
```
# Extraction de l'année
df_lapage['Année'] = df_lapage['date'].dt.year

# Groupe par année, categorie et id_prod, compter les transactions
vente_an = df_lapage.groupby(['Année', 'categ', 'id_prod']).size().
 ↪reset_index(name='Ventes')

# Top références vendues par année et categorie
top_refs_annee = vente_an.sort_values(by=['Année', 'categ',␣
 ↪'Ventes'],ascending=[True, True, False]).groupby(['Année', 'categ']).
 ↪head(10).reset_index(drop=True)

# Flop références vendues par année et categorie
flop_refs_annee = vente_an.sort_values(by=['Année', 'categ',␣
 ↪'Ventes'],ascending=[True, True, True]).groupby(['Année', 'categ']).head(10).
 ↪reset_index(drop=True)

# Définir Année, categ, et id_prod comme index pour éviter la répétition␣
 ↪et␣pour un affichage propre
top_refs_annee.set_index(['Année', 'categ', 'id_prod'], inplace=True)
flop_refs_annee.set_index(['Année', 'categ', 'id_prod'], inplace=True)
```

```
print(top_refs_annee)
print(flop_refs_annee)
```

```
                     Ventes
Année categ id_prod
2021  0     0_1422      544
            0_1437      497
            0_1434      495
            0_1431      494
            0_1425      493
            0_1428      493
            0_1430      492
            0_1416      491
            0_0         488
            0_1409      487
      1     1_369       877
            1_417       854
            1_414       835
            1_498       809
            1_413       801
            1_425       791
            1_396       769
            1_407       769
            1_412       765
            1_398       764
      2     2_135       365
            2_102       357
            2_112       344
            2_110       328
            2_39        317
            2_37        315
            2_207       297
            2_209       289
            2_208       288
            2_46        266
2022  0     0_1425      622
            0_1431      616
            0_1432      614
            0_1441      586
            0_1424      580
            0_1434      580
            0_1422      579
            0_1445      578
            0_1446      575
            0_1411      573
      1     1_369      1135
```

| | | | |
|---|---|---|---|
| | | 1_417 | 1114 |
| | | 1_414 | 1082 |
| | | 1_498 | 1076 |
| | | 1_425 | 1068 |
| | | 1_412 | 983 |
| | | 1_403 | 981 |
| | | 1_413 | 977 |
| | | 1_396 | 961 |
| | | 1_407 | 960 |
| | 2 | 2_102 | 491 |
| | | 2_135 | 487 |
| | | 2_112 | 437 |
| | | 2_39 | 425 |
| | | 2_37 | 392 |
| | | 2_208 | 386 |
| | | 2_110 | 376 |
| | | 2_209 | 375 |
| | | 2_207 | 363 |
| | | 2_109 | 345 |
| 2023 | 0 | 0_1440 | 110 |
| | | 0_1437 | 103 |
| | | 0_1473 | 102 |
| | | 0_1434 | 96 |
| | | 0_1435 | 96 |
| | | 0_1421 | 95 |
| | | 0_1429 | 95 |
| | | 0_1422 | 94 |
| | | 0_1428 | 94 |
| | | 0_1451 | 94 |
| | 1 | 1_414 | 197 |
| | | 1_369 | 193 |
| | | 1_406 | 167 |
| | | 1_412 | 167 |
| | | 1_425 | 167 |
| | | 1_397 | 163 |
| | | 1_381 | 162 |
| | | 1_413 | 157 |
| | | 1_498 | 157 |
| | | 1_417 | 155 |
| | 2 | 2_102 | 93 |
| | | 2_112 | 89 |
| | | 2_39 | 77 |
| | | 2_135 | 68 |
| | | 2_208 | 68 |
| | | 2_37 | 66 |
| | | 2_110 | 65 |
| | | 2_202 | 65 |
| | | 2_209 | 61 |

```
            2_207        59
                      Ventes
Année categ id_prod
2021  0     0_1071         1
            0_1101         1
            0_1107         1
            0_1116         1
            0_1120         1
            0_1122         1
            0_1150         1
            0_1151         1
            0_1165         1
            0_1191         1
      1     1_122          1
            1_151          1
            1_402          1
            1_404          1
            1_408          1
            1_409          1
            1_411          1
            1_420          1
            1_484          1
            1_494          1
      2     2_126          1
            2_23           1
            2_28           1
            2_4            1
            2_75           1
            2_77           1
            2_78           1
            2_88           1
            2_95           1
            2_98           1
2022  0     0_1012         1
            0_1032         1
            0_1037         1
            0_1114         1
            0_1116         1
            0_1141         1
            0_1171         1
            0_1213         1
            0_1233         1
            0_1237         1
      1     1_117          1
            1_146          1
            1_226          1
            1_393          1
            1_402          1
```

| | | | |
|---|---|---|---|
| | | 1_404 | 1 |
| | | 1_408 | 1 |
| | | 1_411 | 1 |
| | | 1_420 | 1 |
| | | 1_492 | 1 |
| | 2 | 2_103 | 1 |
| | | 2_111 | 1 |
| | | 2_122 | 1 |
| | | 2_81 | 1 |
| | | 2_126 | 2 |
| | | 2_16 | 2 |
| | | 2_24 | 2 |
| | | 2_25 | 2 |
| | | 2_27 | 2 |
| | | 2_28 | 2 |
| 2023 | 0 | 0_100 | 1 |
| | | 0_1005 | 1 |
| | | 0_1006 | 1 |
| | | 0_1010 | 1 |
| | | 0_1019 | 1 |
| | | 0_1032 | 1 |
| | | 0_106 | 1 |
| | | 0_1084 | 1 |
| | | 0_1088 | 1 |
| | | 0_1092 | 1 |
| | 1 | 1_112 | 1 |
| | | 1_113 | 1 |
| | | 1_115 | 1 |
| | | 1_118 | 1 |
| | | 1_119 | 1 |
| | | 1_120 | 1 |
| | | 1_121 | 1 |
| | | 1_128 | 1 |
| | | 1_138 | 1 |
| | | 1_143 | 1 |
| | 2 | 2_103 | 1 |
| | | 2_11 | 1 |
| | | 2_114 | 1 |
| | | 2_117 | 1 |
| | | 2_118 | 1 |
| | | 2_123 | 1 |
| | | 2_125 | 1 |
| | | 2_131 | 1 |
| | | 2_133 | 1 |
| | | 2_190 | 1 |

[ ]:
```
# Calcul du top 10 global
top_10_global = vente_an.nlargest(10, 'Ventes')[['id_prod', 'Ventes']]

# Calcul du flop 10 global
flop_10_global = vente_an.nsmallest(10, 'Ventes')[['id_prod', 'Ventes']]

# Création de la figure et des axes
fig, ax = plt.subplots(figsize=(12, 8))
ax.axis('tight')
ax.axis('off')

# Préparation des données pour le tableau
table_data = pd.concat([top_10_global, flop_10_global])
table_data = table_data.reset_index(drop=True)  # Supprime l'index

# Création du tableau
table = ax.table(cellText=table_data.values,
                 colLabels=['ID Produit', 'Ventes'],
                 cellLoc='center',
                 loc='center')

# Ajustement de la taille de la police
table.auto_set_font_size(False)
table.set_fontsize(9)
table.scale(1, 1.5)  # Ajustement de la hauteur des lignes

plt.title("Top 10 et Flop 10 des ventes globales")
plt.tight_layout()
plt.show()

# Affichage des données dans la console
print("Top 10 global:")
print(top_10_global.to_string(index=False))
print("\nFlop 10 global:")
print(flop_10_global.to_string(index=False))
```

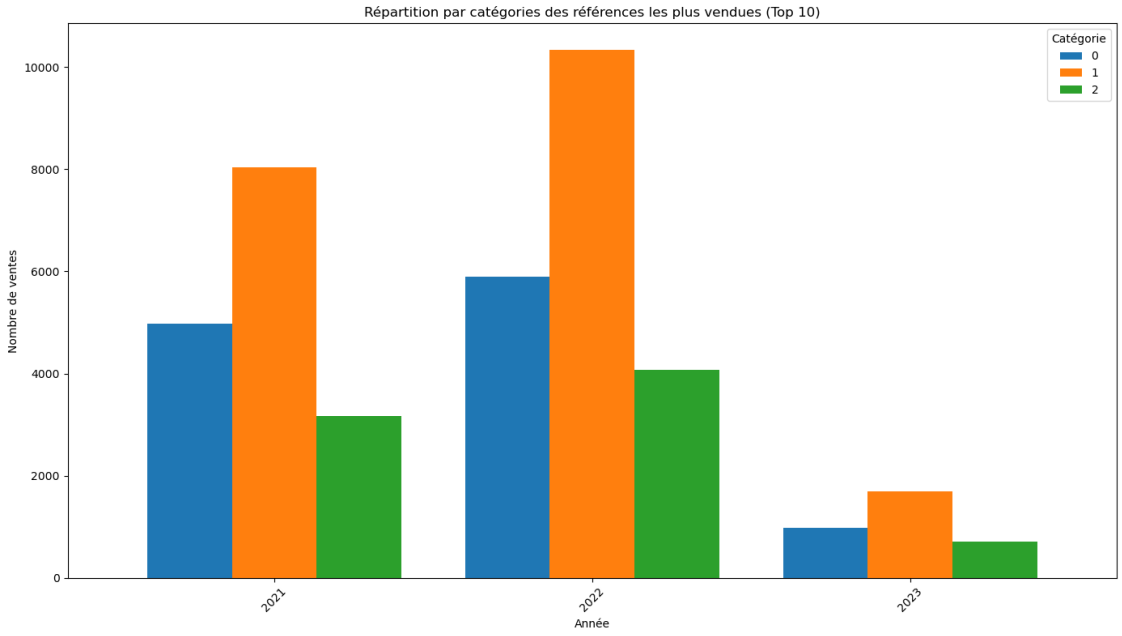
Top 10 et Flop 10 des ventes globales

| ID Produit | Ventes |
|---|---|
| 1_369 | 1135 |
| 1_417 | 1114 |
| 1_414 | 1082 |
| 1_498 | 1076 |
| 1_425 | 1068 |
| 1_412 | 983 |
| 1_403 | 981 |
| 1_413 | 977 |
| 1_396 | 961 |
| 1_407 | 960 |
| 0_1071 | 1 |
| 0_1101 | 1 |
| 0_1107 | 1 |
| 0_1116 | 1 |
| 0_1120 | 1 |
| 0_1122 | 1 |
| 0_1150 | 1 |
| 0_1151 | 1 |
| 0_1165 | 1 |
| 0_1191 | 1 |

```
Top 10 global:
id_prod  Ventes
  1_369    1135
  1_417    1114
  1_414    1082
  1_498    1076
  1_425    1068
  1_412     983
  1_403     981
  1_413     977
  1_396     961
  1_407     960

Flop 10 global:
id_prod  Ventes
 0_1071       1
 0_1101       1
 0_1107       1
 0_1116       1
 0_1120       1
 0_1122       1
 0_1150       1
 0_1151       1
 0_1165       1
```

```
0_1191        1
```

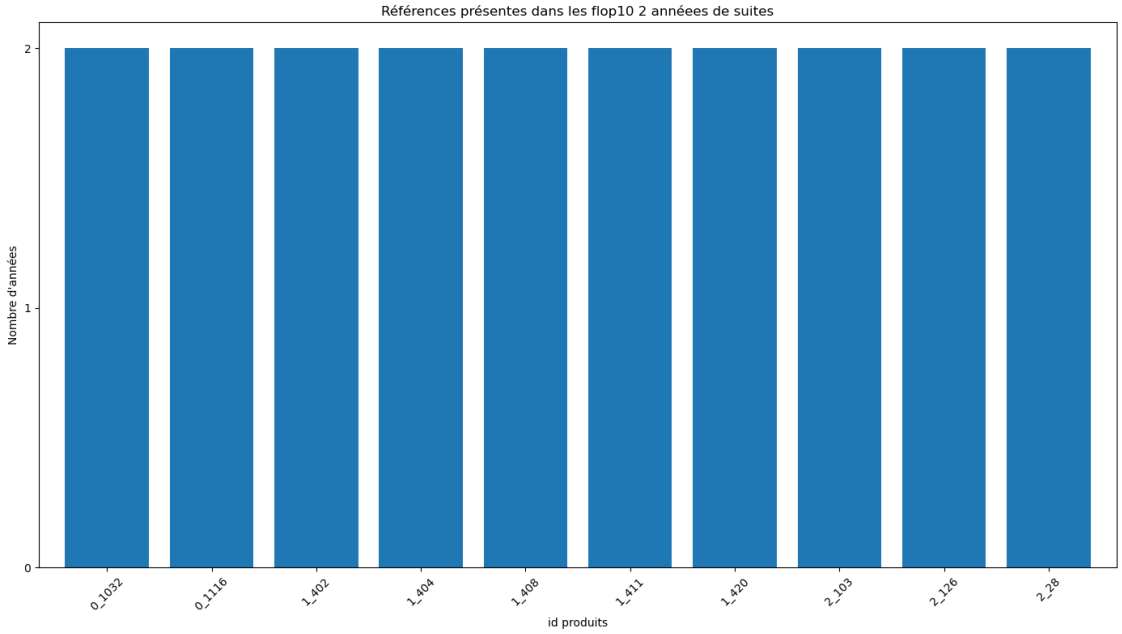
[ ]:
```python
# Grouper les données par année et catégorie, et compter le nombre d'id_prod
grouped_data = top_refs_annee.groupby(['Année', 'categ'])['Ventes'].sum().
↪unstack()

# Création du barplot
fig, ax = plt.subplots(figsize=(14, 8))

# Utiliser plot.bar() pour créer le graphique
grouped_data.plot(kind='bar', ax=ax, width=0.8)

# Personnaliser le graphique
ax.set_xlabel('Année')
ax.set_ylabel('Nombre de ventes')
ax.set_title('Répartition par catégories des références les plus vendues (Top␣
↪10)')

# Rotation des étiquettes de l'axe x pour une meilleure lisibilité
plt.xticks(rotation=45)

# Ajuster la légende
ax.legend(title='Catégorie')

# Ajuster l'espacement
plt.tight_layout()

# Afficher le graphique
plt.show()
```



[ ]:
```
# Flop des 10 références les moins vendues par années et categories
flop_refs_annee = vente_an.sort_values(by=['Année', 'categ',␣
 ↪'Ventes'],ascending=[True, True, True]).groupby(['Année', 'categ']).head(10).
 ↪reset_index(drop=True)

# Conserve les lignes ou id prod apparait 2 fois ou plus
flop_refs_annee = flop_refs_annee[flop_refs_annee.
 ↪duplicated('id_prod',keep=False)]
# Group by 'id_prod' et compte le nombre de lignes
flop_refs_annee2 = flop_refs_annee.groupby('id_prod').size()
# Convertir l'index (id_prod) en liste
list_flop_refs_annee2 = flop_refs_annee2.index.tolist()
print('Références les moins vendues 2 années de suites',list_flop_refs_annee2)
```

```
Références les moins vendues 2 années de suites ['0_1032', '0_1116', '1_402',
'1_404', '1_408', '1_411', '1_420', '2_103', '2_126', '2_28']
```

[ ]:
```
# Création du barplot
fig, ax = plt.subplots(figsize=(14, 8))
# Utiliser plot.bar() pour créer le graphique
flop_refs_annee2.plot(kind='bar', ax=ax, width=0.8)
# Personnaliser le graphique
ax.set_xlabel('id produits')
ax.set_ylabel("Nombre d'années")
ax.set_title('Références présentes dans les flop10 2 annéees de suites')
# Rotation des étiquettes de l'axe x pour une meilleure lisibilité
plt.xticks(rotation=45)
# Définir les graduations de l'axe y comme des entiers
ax.yaxis.set_major_locator(MaxNLocator(integer=True))
# Ajuster l'espacement
plt.tight_layout()
# Afficher le graphique
plt.show()
```




Etape 8 - Etudes des relations demandées

-> 5 relations à étudier :

8.2 le lien entre le genre d'un client et les catégories des livres achetés,

8.3 le lien entre l'âge des clients et le montant total des achats,

8.4 le lien entre l'âge des clients et la fréquence d'achat,

8.5 le lien entre l'âge des clients et la taille du panier moyen,

8.6 le lien entre l'âge des clients et la catégorie des livres achetés

8.1 - Préparation de la matrice pour les études des relations demandées

[ ]:
```
# Supprimer les colonnes 'Date' et 'Année'
df_lapage = df_lapage.drop(columns=['Date',␣
  ↪'Année','id_prod','date','session_id','price'])
```

[ ]:
```
#Création df_R ( df_R pour dataframe Relations)
df_R = df_lapage.copy()
df_R = df_lapage.drop(columns=['categ'])

# Conservez la première occurrence de chaques lignes totalement identiques dans␣
  ↪toutes leurs colonnes, ne conservant ainsi que les lignes totalement uniques.
df_R = df_R.drop_duplicates(keep='first')

# Vérification de l'unicité des client_id
Test_client_id_unique = df_R['client_id'].is_unique
```

```
print("Tous les client_id sont-ils uniques ?", Test_client_id_unique)

df_R
```

```
Tous les client_id sont-ils uniques ? True
```

[ ]:

| | client_id | sex | age | CA_total | C.A_catégories 0 | C.A_catégories 1 | \ |
|---|---|---|---|---|---|---|---|
| 0 | c_4410 | Femme | 57 | 1376.82 | 501.56 | 729.27 | |
| 76 | c_7839 | Femme | 49 | 568.92 | 227.15 | 341.77 | |
| 113 | c_1699 | Femme | 40 | 190.60 | 152.44 | 38.16 | |
| 128 | c_5961 | Femme | 62 | 1150.42 | 331.30 | 819.12 | |
| 204 | c_5320 | Homme | 81 | 396.53 | 163.87 | 232.66 | |
| … | … | … | … | … | … | … | |
| 640265 | c_7920 | Homme | 68 | 1925.15 | 413.81 | 1511.34 | |
| 640369 | c_7403 | Femme | 54 | 2592.73 | 549.45 | 2043.28 | |
| 640533 | c_5119 | Homme | 50 | 729.69 | 334.43 | 283.54 | |
| 640580 | c_5643 | Femme | 56 | 1995.95 | 459.64 | 1536.31 | |
| 640706 | c_84 | Femme | 42 | 427.51 | 254.09 | 173.42 | |

| | C.A_catégories 2 | Nombre de transactions | \ |
|---|---|---|---|
| 0 | 145.99 | 76 | |
| 76 | 0.00 | 37 | |
| 113 | 0.00 | 15 | |
| 128 | 0.00 | 76 | |
| 204 | 0.00 | 24 | |
| … | … | … | |
| 640265 | 0.00 | 104 | |
| 640369 | 0.00 | 164 | |
| 640533 | 111.72 | 47 | |
| 640580 | 0.00 | 126 | |
| 640706 | 0.00 | 28 | |

| | Nombre de transactions catégorie 0 | \ |
|---|---|---|
| 0 | 38.0 | |
| 76 | 22.0 | |
| 113 | 13.0 | |
| 128 | 37.0 | |
| 204 | 12.0 | |
| … | … | |
| 640265 | 34.0 | |
| 640369 | 55.0 | |
| 640533 | 40.0 | |
| 640580 | 44.0 | |
| 640706 | 21.0 | |

| | Nombre de transactions catégorie 1 | \ |
|---|---|---|

```
0                                    37.0
76                                   15.0
113                                   2.0
128                                  39.0
204                                  12.0
…                                     …
640265                               70.0
640369                              109.0
640533                               6.0
640580                               82.0
640706                               7.0

        Nombre de transactions catégorie 2  Nb_Session  Panier_Moyen
0                                      1.0          52         26.48
76                                     0.0          20         28.45
113                                    0.0           6         31.77
128                                    0.0          58         19.83
204                                    0.0          18         22.03
…                                       …           …             …
640265                                 0.0          48         40.11
640369                                 0.0         123         21.08
640533                                 1.0          25         29.19
640580                                 0.0          95         21.01
640706                                 0.0          14         30.54

[8596 rows x 13 columns]
```

8.2 - lien entre le genre d'un client et les catégories des livres achetés

### 0.0.2 Variables

- Le genre d'un clients (sexe) : Variable **qualitative**
- Catégories de livres achetées : Variable **qualitative**.

**Deux variables Qualitatives**

Pour étudier le lien entre des variables **Qualitatives** je commence par crée un **tableau de contingence** puis je réalise un **test de Chi_2**

[ ]:
```
# choix des colonnes pour l'analyse :
df_R_sex_categ = df_R[['sex','Nombre de transactions catégorie 0","Nombre de␣
 ↪transactions catégorie 1","Nombre de transactions catégorie 2"]]

# Créer un tableau de contingence : Sexe du client et le nombre de transactions␣
 ↪par categories de livre
tableau_contingence = df_R_sex_categ.groupby('sex').sum()

# Convertir les colonnes float en type int
```

```
tableau_contingence = tableau_contingence.astype({
    'Nombre de transactions catégorie 0': 'int',
    'Nombre de transactions catégorie 1': 'int',
    'Nombre de transactions catégorie 2': 'int'
})

tableau_contingence.columns = ['catégorie 0', 'catégorie 1', 'catégorie 2']#␣
 ↪Changer les noms des colonnes pour les annotations
tableau_contingence
```

[ ]:

| sex | catégorie 0 | catégorie 1 | catégorie 2 |
|---|---|---|---|
| Femme | 200793 | 115721 | 16980 |
| Homme | 186488 | 104884 | 15868 |

### 0.0.3 Test du chi-carré (chi2)

### 0.0.4 Hypothèses

**Hypothèse Nulle (H0)** : Il n'y a pas de lien entre le genre d'un client et les catégories des livres achetés.

**Hypothèse Alternative (H1)** : Il existe un lien entre le genre d'un client et les catégories des livres achetés.

[ ]:
```
# Test Chi-2

c, p, dof, expected = chi2_contingency(tableau_contingence)

# Calculer la valeur critique du chi-carré pour 95% de confiance
alpha = 0.05
chi2_critique = round(chi2.ppf(1 - alpha, dof),2)

# Afficher les résultats
print("Valeur critique du chi-carré pour un seuil de signification␣
 ↪de",alpha,"%","et",dof,"degrés de liberté :",chi2_critique)
print("Statistique du chi-carré :", c)
print("P-valeur :", p,"\n")

# Interprétation de la valeur p
if p < 0.05:
    conclusion = "Nous rejetons l'hypothèse nulle (H0) : Il n'y a pas␣
 ↪d'association entre le genre d'un client et les catégories des livres␣
 ↪achetés."
else:
    conclusion = "Nous ne rejetons pas l'hypothèse nulle (H0): Il n'y a pas␣
 ↪d'association entre le genre d'un client et les catégories des livres␣
 ↪achetés."
```

```
print(conclusion)
```

```
Valeur critique du chi-carré pour un seuil de signification de 0.05 % et 2
degrés de liberté : 5.99
Statistique du chi-carré : 22.66856665178056
P-valeur : 1.1955928116587024e-05

Nous rejetons l'hypothèse nulle (H0) : Il n'y a pas d'association entre le genre
d'un client et les catégories des livres achetés.
```

### 0.0.5 Interprétation des résultats du test de chi2

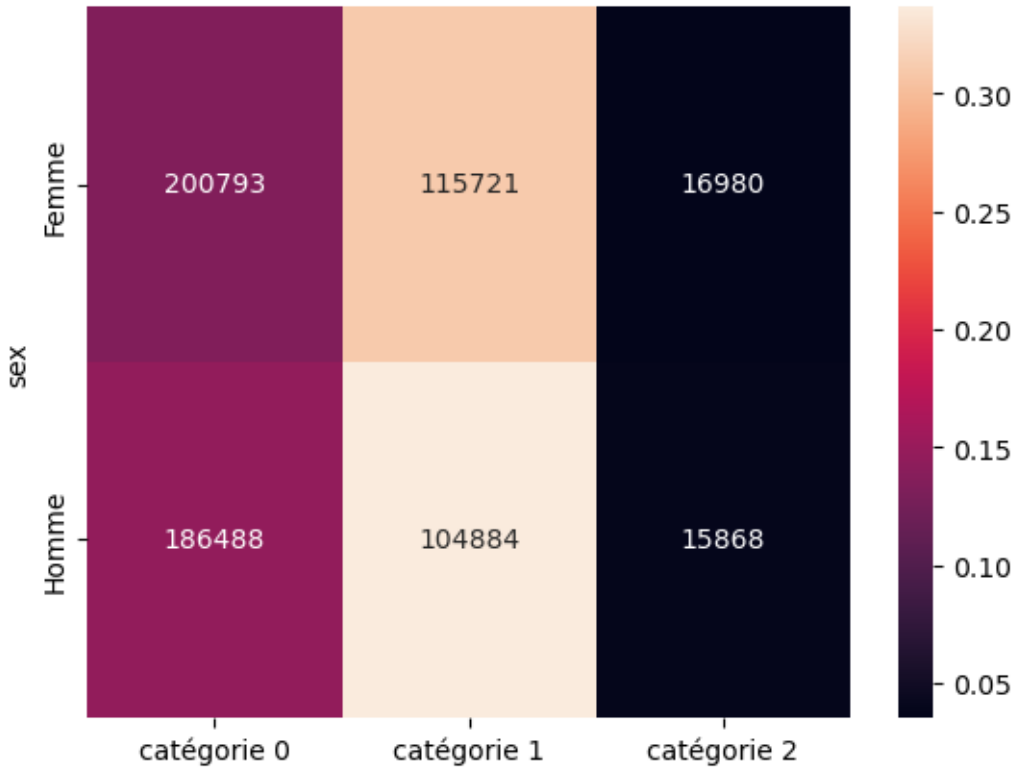
La p-valeur obtenue 1.195928116587024e-05 est bien inférieure au seuil de significativité (0,05%).

La statistique du chi-carré (22.66...) est superieur a la valeur critique du chi-carré.(5.99)

**Nous rejettons l'hypothèse H0 d'indépendance entre le genre et la catégorie, avec un risuqe très minime de se tromper.**

**Par conséquent : Il existe un lien entre le genre d'un client et la categories de livres achetés.**

[ ]:
```
# Création de la Heatmap

# dfmesure ou chaque cellule représente la contribution au chi-carré de la␣
 ↪différence entre les valeurs observées et attendues.
mesure = (tableau_contingence - expected) ** 2 / expected # Utilisation de␣
 ↪expected (valeurs attendues du test Chi-2) pour calculer mesure.

table = mesure / c # Normalisations de la mesure en divisant chaque cellule par␣
 ↪la statistique du chi-carré.

sns.heatmap(table, annot=tableau_contingence, fmt="d")  #Le paramètre fmt="d"␣
 ↪formatte les annotations comme des entiers.
# table représente les couleurs dans la heatmap, montrant les contributions␣
 ↪normalisées au chi-carré.
# annot=tableau_contingence représente les chiffres annotés dans chaque cellule␣
 ↪de la heatmap, montrant les valeurs observées du tableau de contingence.

plt.show()
```




### 0.0.6 Analyse de la heatmap

La heatmap montre les contributions normalisées au test du chi-carré pour les transactions par sexe et catégories de livres achetées. On peut considérer cette valeur comme une contribution à la non-indépendance entre le genre d'un client et les catégories des livres achetés

Plus cette contribution sera proche de 1, plus la case en question sera source de non-indépendance

**-La variation des couleurs indique un lien plus important entre le sexe et la catégorie 1.**

### 0.0.7 Calculer le coefficient T de Tschuprow

Le coefficient T de Tschuprow est une mesure de liaison entre deux variables qualitatives. Il est basé sur le chi-carré ( $^2$) et prend en compte la taille de l'échantillon et le nombre de catégories dans chaque variable.

[ ]:
```
n = tableau_contingence.to_numpy().sum()  # Somme totale des fréquences␣
↪observées
r = tableau_contingence.shape[0]  # Nombre de lignes dans le tableau de␣
↪contingence
k = tableau_contingence.shape[1]  # Nombre de colonnes dans le tableau de␣
↪contingence
```

```
# c = Valeurs obtenues du test du chi-carré

# Calcul du coeff T de Tschuprow
T = np.sqrt((c / n) / np.sqrt((r - 1) * (k - 1)))

print(f"Le coefficient T de Tschuprow est {T:.3f}")
```

```
Le coefficient T de Tschuprow est 0.005
```

Le coefficient T de Tschuprow varie entre 0 et 1. Le coefficient T de Tschuprow calculé est de 0.005, proche de 0 indique une liaison très faible.

### 0.0.8 Conclusion

**-Les résultats du test de chi2 ont permits de rejeter l'hypothèse nulle H_0 et montré qu'il existe un lien entre le sexe et les catégories de lives achetées**

**-Le coefficient T de Tschuprow obtenu à partir des valeurs du test chi2 montre que ce lien est très faible**

8.3 - Lien entre l'âge des clients et le montant total des achats

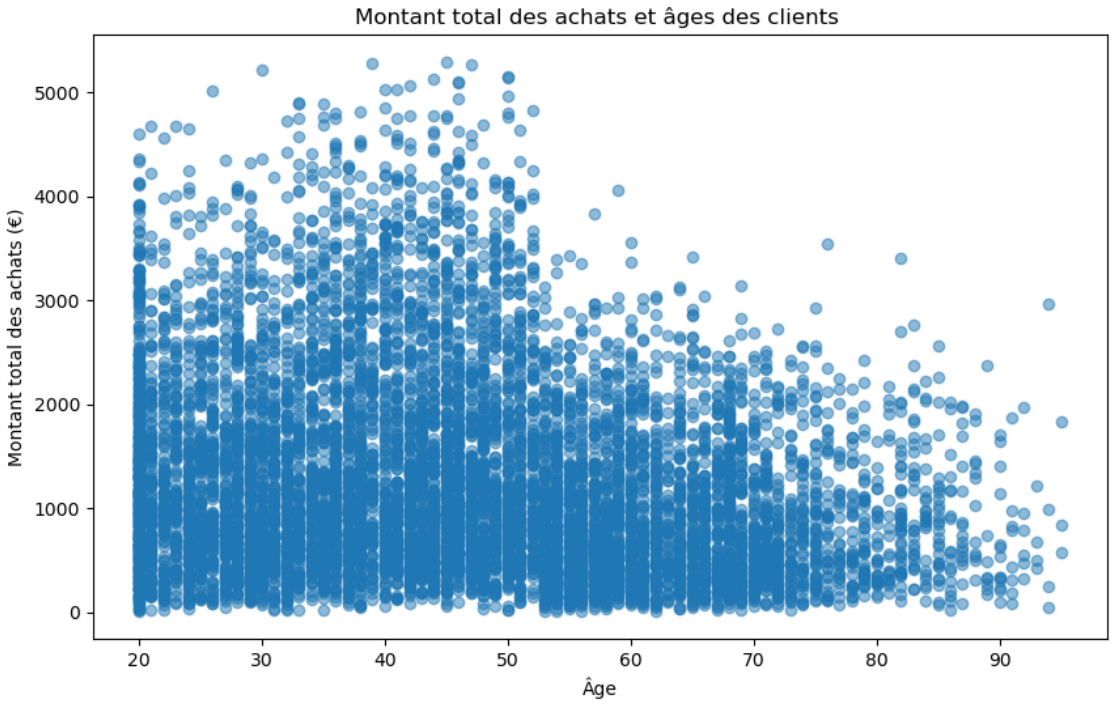
### 0.0.9 Variables

- Âge : Variable **quantitative**
- Montant des achats : Variable **quantitative**

**Deux variables Quantitatives**

Pour étudier le lien entre des variables **Quantitatives** je commence par visualiser les données à l'aide d'un **graphique en nuage de points**

[ ]:
```
# Scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df_R['age'], df_R['CA_total'], alpha=0.5)
plt.title('Montant total des achats et âges des clients ')
plt.xlabel('Âge')
plt.ylabel('Montant total des achats (€)')
plt.show()
```




### 0.0.10 Analyse graphique du nuage de points

**La tendance générale semble être baissière** Cela pourrait montrer que plus l'âge augmente plus le montant des achats diminue.

**La variaance des montants des achats parait plus grande chez les clients plus jeunes**

[ ]:
```
# calcul du coéff de corrélation de Pearson
pearson_corr, p_value = pearsonr(df_R["age"], df_R["CA_total"])

print("Corrélation de Pearson entre l'âge et le montant total des achats :",␣
 ↪round(pearson_corr,2))

corr=abs(pearson_corr) #Conversion du coeff en valeur absolue

if corr > 0.5 :
    conclusion = " - Il existe une corrélation linéaire entre le montant total␣
 ↪des achats et l'âge des clients."
else:
    conclusion = " - Il n'existe pas de corrélation linéaire entre le montant␣
 ↪total des achats et l'âge des clients."

print("\n",conclusion)
```

```
Corrélation de Pearson entre l'âge et le montant total des achats : -0.19
```

```
  - Il n'existe pas de corrélation linéaire entre le montant total des achats et
l'âge des clients.
```

### 0.0.11 Conclusion

Il n'existe pas de corrélation linéaire entre l'age des client et le mopntant total des achats, par conséquent :

**Cela indique que l'âge des clients et le montant total des achats ne sont pas liées par une corrélation linaire.**

8.4 - Lien entre l'âge des clients et la fréquence d'achat

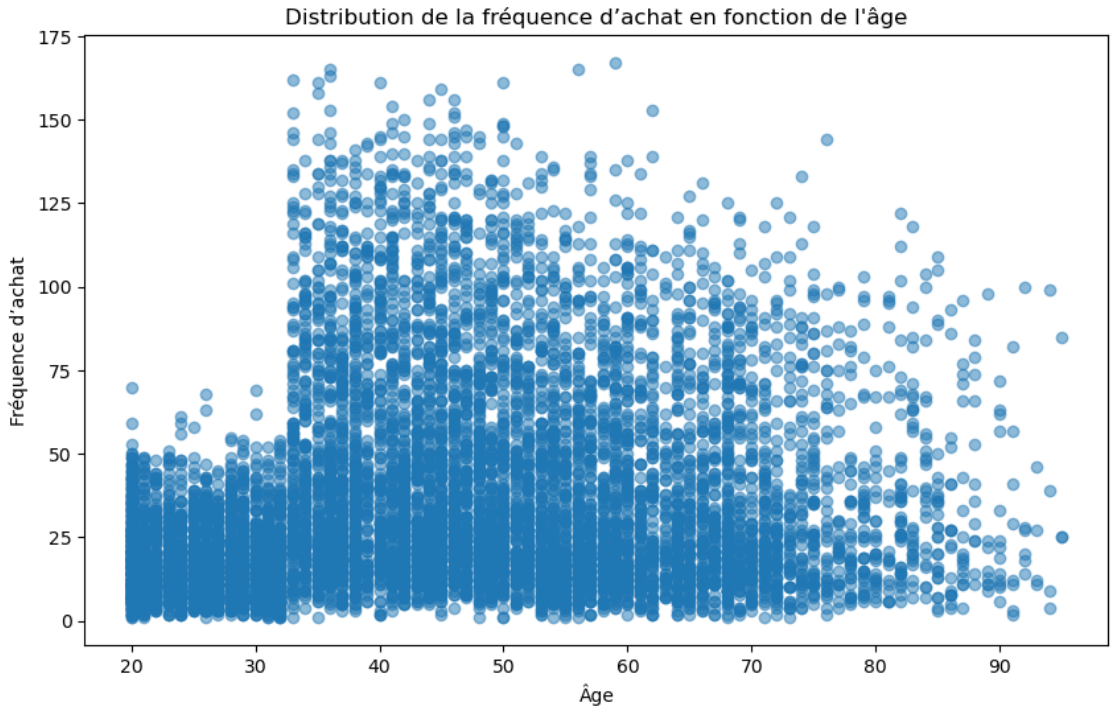
### 0.0.12 Variables

- Âge : Variable **quantitative**
- Fréquence d'achats : Variable **quantitative**

**Deux variables Quantitatives**

Pour étudier le lien entre des variables **Quantitatives** je commence par visualiser les données à l'aide d'un **graphique en nuage de points**

[ ]:
```
# Scatter plot pour la fréquence d'achat
plt.figure(figsize=(10, 6))
plt.scatter(df_R['age'], df_R['Nb_Session'], alpha=0.5)
plt.title("Distribution de la fréquence d'achat en fonction de l'âge")
plt.xlabel('Âge')
plt.ylabel("Fréquence d'achat")
plt.show()
```




### 0.0.13 Analyse visuel du nuage de points

**Tendance générale :** La tendance de la distribution de la fréquence en fonction de l'age parrait baissière.

**Variabilité :** La variabilité des montants des achats est plus importante apres 33ans

[ ]:
```
# Calculer la corrélation de Pearson entre l'âge et la fréquence d'achat
pearson_corr, p_value = pearsonr(df_R['age'], df_R['Nb_Session'])

print("Corrélation de Pearson entre l'âge et la fréquence d'achat :",␣
 ↪round(pearson_corr,2),"\n")

corr=abs(pearson_corr) #Conversion du coeff en valeur absolue

if corr > 0.5 :
    conclusion = " - Il existe une corrélation linéaire entre la fréquence␣
 ↪d'achat et l'age des clients."
else:
    conclusion = " - Il n'existe pas de corrélation linéaire entre la fréquence␣
 ↪d'achat et l'age des clients."

print(conclusion)
```

```
Corrélation de Pearson entre l'âge et la fréquence d'achat : 0.16
```

```
 - Il n'existe pas de corrélation linéaire entre la fréquence d'achat et l'age
des clients.
```

### 0.0.14 Conclusion

Il n'existe pas de corrélation linéaire entre l'age d'un client et la fréquence d'achat, par conséquent :

**Cela indique que l'âge des clients et la fréquence d'achats ne sont pas liées par une corrélation linaire.**

### 0.0.15 Discrétiser l'âge en deux groupes : avant et après 33 ans

[ ]:
```
# Discrétiser l'âge en deux groupes : avant et après 33 ans
df_R['age_groupeses'] = pd.cut(df_R['age'], bins=[0, 33, 100], labels=['<= 33␣
 ↪ans', '> 33 ans'])

# Convertir les catégories en type "category" pour éviter les avertissements
df_R['age_groupeses'] = df_R['age_groupeses'].astype('category')

# Afficher les données agrégées pour vérifier
print(df_R.groupby('age_groupeses', observed=True)['Nb_Session'].describe())
```

```
                count       mean        std  min   25%   50%   75%    max
age_groupeses
<= 33 ans      2411.0  20.877229  16.114807  1.0  10.0  17.0  27.0  162.0
> 33 ans       6185.0  43.998545  32.799326  1.0  18.0  34.0  63.0  167.0
```

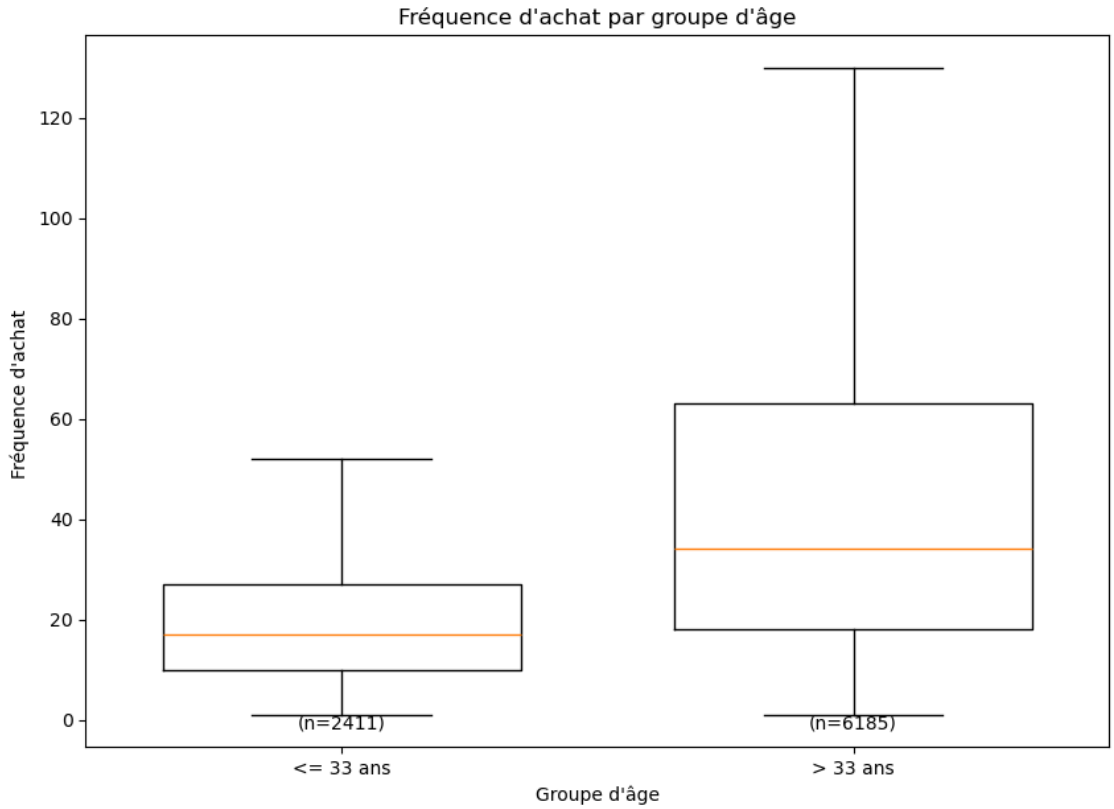
### 0.0.16 Variables

- Âge (discrétisé) : Après discrétisation, l'âge devient une variable qualitative avec deux niveaux ($<= 52$ ans et $> 52$ ans).
- Montant total des achats : Variable quantitative.

**Variables Quantitative et Qualitative -> Visualisation à l'aide d'un graphique en boîte à moustache :**

[ ]:
```
# Visualisation avec un boxplot
groupes = []
for label, group in df_R.groupby('age_groupeses', observed=True):
    frequences = group['Nb_Session']
    if len(frequences) > 0:
        g = {
            'valeurs': frequences,
            'label': label,
            'taille': len(frequences),
            'quartiles': [np.percentile(frequences, p) for p in [25, 50, 75]]
        }
```

```
        groupes.append(g)

# Création du boxplot
plt.figure(figsize=(10, 7))
plt.boxplot([g["valeurs"] for g in groupes],
            labels=[g["label"] for g in groupes],
            showfliers=False,  # on ne prend pas en compte les outliers
            widths=0.7)  # largeur graphique des boxplots

# Affichage des effectifs de chaque classe
for i, g in enumerate(groupes):
    plt.text(i + 1, min(g["valeurs"]), "(n={})".format(g["taille"]),
             horizontalalignment='center', verticalalignment='top')

plt.xlabel('Groupe d\'âge')
plt.ylabel('Fréquence d\'achat ')
plt.title('Fréquence d\'achat par groupe d\'âge')
plt.show()
```



### 0.0.17 Analyse visuel des boites à moustaches

Le boxplot montre la distribution de la fréquence d'achat pour les deux groupes d'âge.

- Les clients de plus de 33 ans ont une fréquence d'achat médiane plus élevée que ceux de 33 ans ou moins.
- Le groupe d'âge > 33 ans présente une plus grande variabilité dans la fréquence d'achat.
- Le nombre d'observations est plus élevé pour le groupe > 33 ans (n=6185) par rapport au groupe <= 33 ans (n=2411).

**Ces observations indiquent que les clients plus âgés ont tendance à acheter plus fréquemment que les clients plus jeunes.**

Cela pourrait indiquer un lien entre les deux groupes d'age et la fréquence d'achat.

### 0.0.18 Hypothèses

**Hypothèse Nulle (H0)** : Il n'y a pas de lien entre les deux groupes d'ages et la fréquence d'achat.

**Hypothèse Alternative (H1)** : Il existe un Lien entre les deux groupes d'ages et la fréquence d'achat.

[ ]:
```
# Création d'un tableau ANOVA à un facteur renvoyés pour la somme des carrés de␣
 ↪type II

model = ols('Nb_Session ~ age_groupeses', data = df_R).fit()
aov_table = sm.stats.anova_lm(model, typ=2)

def anova_table(aov):

#Calcul somme des carrés moyen (mean_sq)
    aov['mean_sq'] = aov[:]['sum_sq']/aov[:]['df']

## Calcul du rapport de corrélation eta carré (eta_squared)
    aov['eta_sq'] = aov[:-1]['sum_sq']/sum(aov['sum_sq'])

#Mesurer l'intensité de l'effet avec omega carré (omega squared)
    aov['omega_sq'] = (aov[:-1]['sum_sq']-(aov[:-1]['df']*aov['mean_sq'][-1]))/
 ↪(sum(aov['sum_sq'])+aov['mean_sq'][-1])

    cols = ['sum_sq', 'df', 'mean_sq', 'F', 'PR(>F)', 'eta_sq', 'omega_sq']
    aov = aov[cols]
    return aov

anova_table(aov_table)
```

```
C:\Users\WadJa\AppData\Local\Temp\ipykernel_65840\2210745588.py:15:
FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
```

```
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
```

[ ]:
```
                     sum_sq      df         mean_sq            F  \
age_groupeses  9.273968e+05     1.0  927396.787825  1095.002407
Residual       7.278567e+06  8594.0     846.935844          NaN

                      PR(>F)    eta_sq  omega_sq
age_groupeses  4.019252e-226  0.113015    0.1129
Residual                 NaN       NaN       NaN
```

### 0.0.19 Conclusion

Le résultat montre que le rapport de corrélation eta-squared (omega carré) 0.11 :

Ce niveau de corrélation eta carré, confirmé par oméga carré montre **qu'il n'existe pas de lien entre la fréquence d'achat et les deux groupes d'âges**

8.5 - Lien entre l'âge des clients et la taille du panier moyen

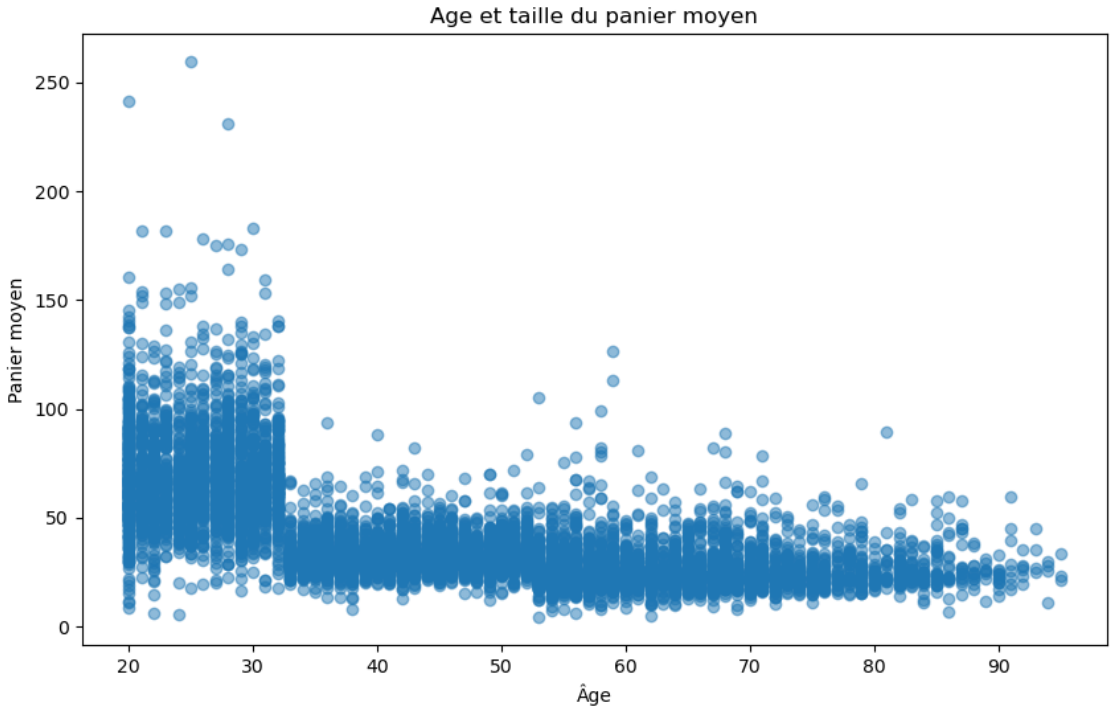
### 0.0.20 Variables

- Âge : Variable **quantitative**
- Taille du panier moyen : Variable **quantitative**

**Deux variables Quantitatives**

Pour étudier le lien entre des variables **Quantitatives** je commence par visualiser les données à l'aide d'un **graphique en nuage de points**

[ ]:
```
# Variable Quantitative et Quantitative -> Représentation graphique en nuage de point
plt.figure(figsize=(10, 6))
plt.scatter(df_R['age'], df_R['Panier_Moyen'], alpha=0.5)
plt.title('Age et taille du panier moyen')
plt.xlabel('Âge')
plt.ylabel('Panier moyen')
plt.show()
```




### 0.0.21 Analyse visuel du nuage de points

**Tendance générale :** Il y a une tendance générale parrait baissière mais si ont considère deux deux groupe avant et après 33ans elle parrait stable

**Variabilité :** La variabilité du montant de panier moyen diminue fortement apres à 33ans

**Population de clients :** Le montant du manier moyen parrait supperieur avant 33 ans.

[ ]:
```
# Calculer la corrélation de Pearson et la covariance entre l'âge et la taille␣
 ↪du panier moyen
pearson_corr, p_value = pearsonr(df_R['age'], df_R['Panier_Moyen'])

print("Corrélation de Pearson entre l'âge et la taille du panier moyen :",␣
 ↪round(pearson_corr,2),"\n")

corr=abs(pearson_corr) #Conversion du coeff en valeur absolue

print("P-value pour la corrélation de Pearson :", p_value,"\n")

# Vérification de l'hypothèse
alpha = 0.05
if p_value < alpha and corr > 0.5 :
    print("Il existe une corrélation linéaire entre la taille du panier moyenet␣
 ↪l'age des clients.")
```

```
else:
    print("Il n'existe pas de corrélation linéaire entre la taille du panier␣
 ↪moyen et l'age des clients.")
```

```
Corrélation de Pearson entre l'âge et la taille du panier moyen : -0.62

P-value pour la corrélation de Pearson : 0.0

Il existe une corrélation linéaire entre la taille du panier moyenet l'age des
clients.
```

### 0.0.22 Conclusion

Nous rejetons l'hypothèse nulle H_0 car il existe un lien de corrélation linéaire entre l'age d'un client et la taille du panier moyen, par conséquent :

**Cela indique que l'âge des clients et la taille du panier moyen sont liées par une corrélation linéaire.**

### 0.0.23 Discrétiser l'âge en deux groupes :

[ ]:
```
# Discrétiser l'âge avant et après 33 ans

df_R.loc[:, 'groupes_ages'] = pd.cut(df_R['age'], bins=[0, 33, 100],␣
 ↪labels=['Inf ou égale a 33 ans', 'Suppérieur a 33 ans'])
```

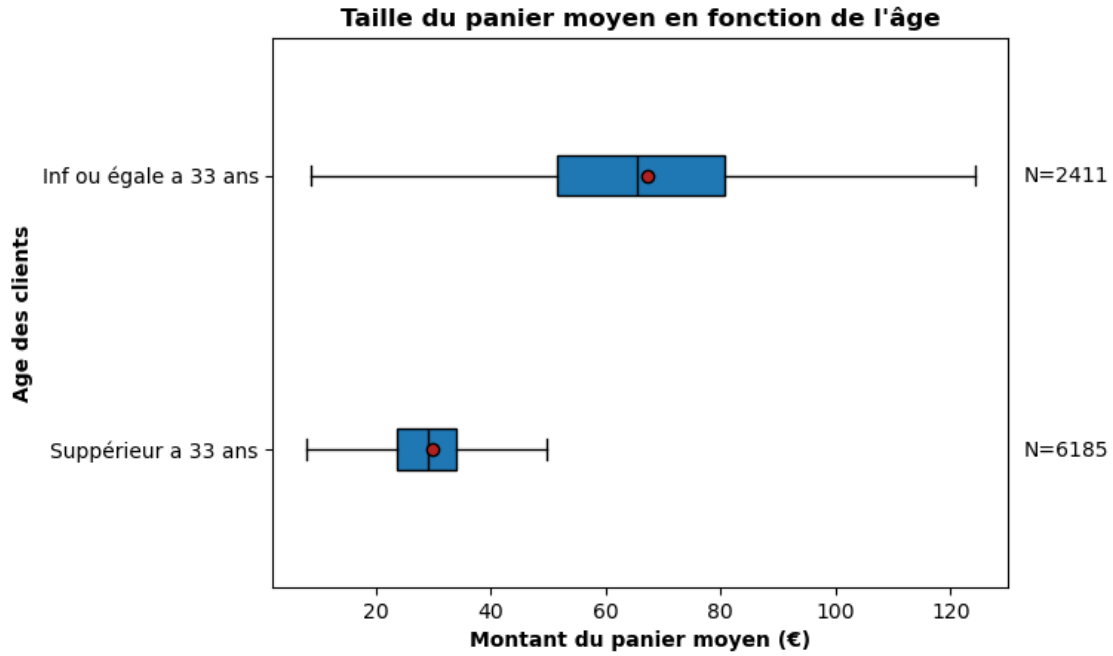
### 0.0.24 Variables

- Âge (discrétisé) : Après discrétisation, l'âge devient une variable qualitative avec deux niveaux ( <=33 ans et >33 ans).
- Taille du panier moyen : Variable quantitative.

**Variables Quantitative et Qualitative -> Visualisation à l'aide d'un graphique en boîte à moustache :**

[ ]:
```
# Visualisation à l'aide d'un graphique en boîte à moustache

X = 'groupes_ages'
Y = 'Panier_Moyen'

modalites = sorted(df_R[X].unique(), reverse=True) # sorted pour trie les␣
 ↪catégories en ordre inverse

groupes = []
for m in modalites:
    groupes.append(df_R[df_R[X]==m][Y])

# Calculer les effectifs pour chaque groupe
effectifs = [len(g) for g in groupes]
```

```
medianprops = {'color':"black"}
meanprops = {'marker':'o', 'markeredgecolor':'black',
            'markerfacecolor':'firebrick'}

# Ajouter les effectifs pour chaque groupe
for i, eff in enumerate(effectifs, start=1):
    plt.text(1.02, i, f'N={eff}', va='center', ha='left', transform=plt.gca().
 ↪get_yaxis_transform())


plt.boxplot(groupes, labels=modalites, showfliers=False,␣
 ↪medianprops=medianprops,
            vert=False, patch_artist=True, showmeans=True, meanprops=meanprops)
plt.title("Taille du panier moyen en fonction de l'âge", fontweight='bold',␣
 ↪fontsize = 12)
plt.ylabel("Age des clients", fontweight='bold')
plt.xlabel('Montant du panier moyen (€)', fontweight='bold')
plt.show()
```




### 0.0.25 Analyse visuel des boites à moustaches

Les clients de plus de 33 ans ont une taille de panier moyen globalement plus faible que ceux de 33 ans ou moins. Le groupe d'âge inf ou égale a 33 ans présente une variabilité plus importante au

niveau de la taille du panier moyen. Le nombre d'observations N est plus élevé pour le groupe > 33 ans (n=6185) par rapport au groupe <= 33 ans (n=2411).

Cela pourrait indiquer un lien entre les deux groupes d'age et la taille du panier moyen.

### 0.0.26 Hypothèses

**Hypothèse Nulle (H0)** : Il n'y a pas de lien entre les deux groupes d'ages et la taille du panier moyen.

**Hypothèse Alternative (H1)** : Il existe un Lien entre les deux groupes d'ages et la taille du panier moyen.

[ ]:
```
# Création d'un tableau ANOVA à un facteur renvoyés pour la somme des carrés de␣
 ↪type II

model = ols('Panier_Moyen ~ groupes_ages', data = df_R).fit()
aov_table = sm.stats.anova_lm(model, typ=2)

def anova_table(aov):

#Calcul somme des carrés moyen (mean_sq)
    aov['mean_sq'] = aov[:]['sum_sq']/aov[:]['df']

## Calcul du rapport de corrélation eta carré (eta_squared)
    aov['eta_sq'] = aov[:-1]['sum_sq']/sum(aov['sum_sq'])

#Mesurer l'intensité de l'effet avec omega carré (omega squared)
    aov['omega_sq'] = (aov[:-1]['sum_sq']-(aov[:-1]['df']*aov['mean_sq'][-1]))/
 ↪(sum(aov['sum_sq'])+aov['mean_sq'][-1])

    cols = ['sum_sq', 'df', 'mean_sq', 'F', 'PR(>F)', 'eta_sq', 'omega_sq']
    aov = aov[cols]
    return aov

anova_table(aov_table)
```

```
C:\Users\WadJa\AppData\Local\Temp\ipykernel_65840\1507631496.py:15:
FutureWarning:

Series.__getitem__ treating keys as positions is deprecated. In a future
version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
```

[ ]:

| | sum_sq | df | mean_sq | F | PR(>F) | \ |
|---|---|---|---|---|---|---|
| groupes_ages | 2.423443e+06 | 1.0 | 2.423443e+06 | 10257.16631 | 0.0 | |
| Residual | 2.030490e+06 | 8594.0 | 2.362683e+02 | NaN | NaN | |

```
                 eta_sq  omega_sq
groupes_ages   0.544113  0.544031
Residual          NaN       NaN
```

[ ]:
```
#tester la normalité des résidus :

shapiro(model.resid)
```

```
c:\Users\WadJa\anaconda3\envs\Anaconda_3_12\Lib\site-
packages\scipy\stats\_axis_nan_policy.py:531: UserWarning:

scipy.stats.shapiro: For N > 5000, computed p-value may not be accurate. Current
N is 8596.
```

[ ]:
```
ShapiroResult(statistic=0.8787907240619794, pvalue=4.6606272107621094e-63)
```

**Analyse de la normalité des résidus (test de Shapiro)**

Le test est non significatif, W = 0.8787907240619794, p = 4.6606272107621094e-63, ce qui indique que **les résidus sont distribués normalement.**

[ ]:
```
#Non-homogéneité de la variance (hétéroscédasticité) avec le test de LEVENE) :

levene(df_R['groupes_ages'] == 'Inf ou égale a 33 ans', df_R['groupes_ages'] ==␣
 ↪'Suppérieur a 33 ans')
```

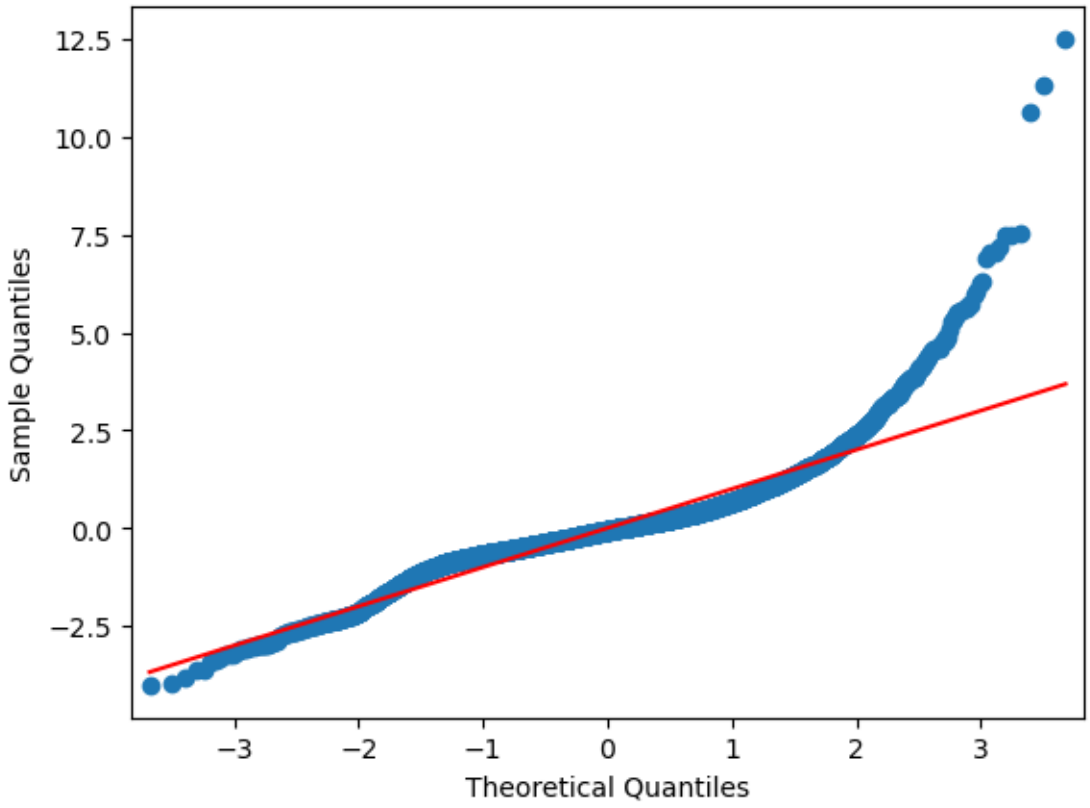
[ ]:
```
LeveneResult(statistic=0.0, pvalue=1.0)
```

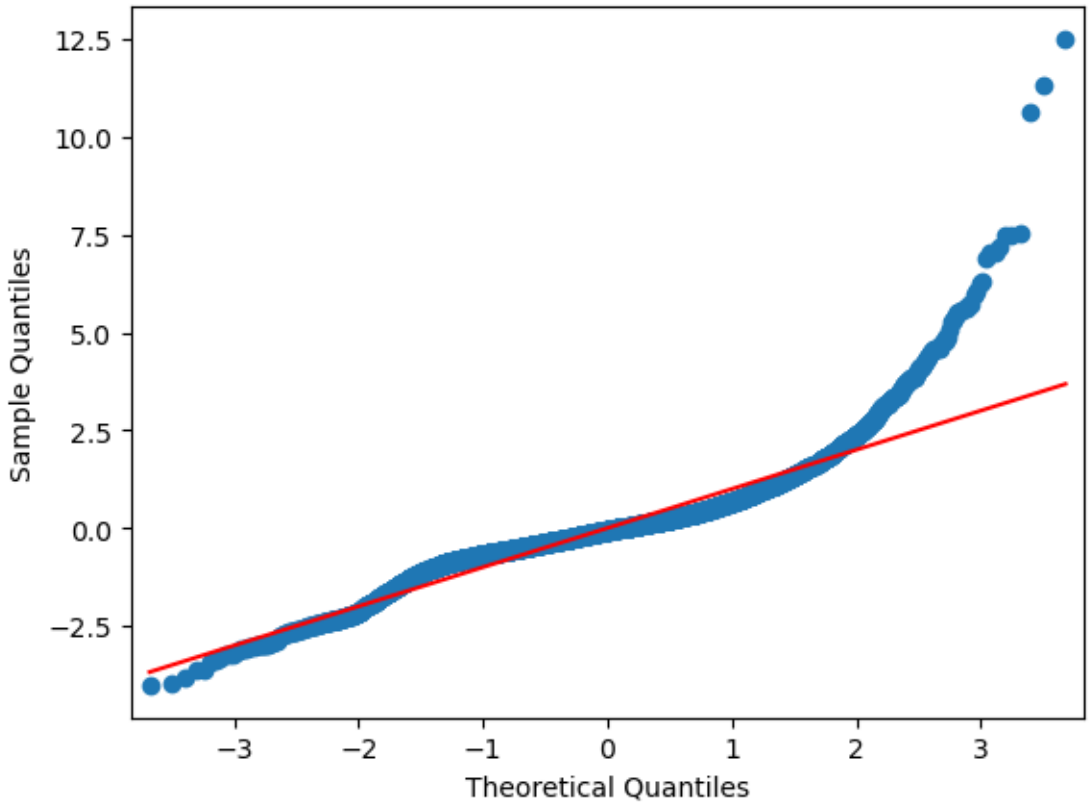
LeveneResult(statistic=0.0, pvalue=1.0)

**Le test d'homogénéité des variances de Levene n'est pas significatif.**

[ ]:
```
#Crée le diagramme Quantile-Quantile ou diagramme Q-Q plot permettant d'évaluer␣
 ↪la pertinence de l'ajustement d'une distribution donnée à un modèle␣
 ↪théorique.
pplot = gofplots.ProbPlot(model.resid, fit = True)
pplot.qqplot(line = 's')
```

[ ]:


12.5
10.0
7.5
5.0
2.5
0.0
−2.5
Sample Quantiles
−3
−2
−1
0
1
2
3
Theoretical Quantiles




### 0.0.27 Conclusion

Le résultat montre que le rapport de corrélation eta-squared (omega carré) 0.54 :

-Cela qui indique que 54% de la taille du panier moyen est expliqué par les différences entre les deux catégories d'ages

Ce niveau de corrélation, confirmé par oméga carré montre qu'il existe **un lien entre la taille du panier moyen et les deux groupes d'âges**

La p-value PR(>F) des groupes d'ages (0) indique que nous pouvons rejetez l'hypothèse nulle (H0), confirmant que'il **existe un lien entre l'age des clients et la taille du panier moyen.**

8.6 - Lien entre l'âge des clients et la catégorie des livres achetés

### 0.0.28 Variables

- Âge : Variable **quantitative**
- Catégories de livres achetées : Variable **qualitative**.

**Variables Quantitative et Qualitative**

[ ]:
```
df_lapage
```

[ ]:

| | client_id | sex | age | categ | CA_total | C.A_catégories 0 | \ |
|---|---|---|---|---|---|---|---|
| 0 | c_4410 | Femme | 57 | 1 | 1376.82 | 501.56 | |
| 1 | c_4410 | Femme | 57 | 0 | 1376.82 | 501.56 | |
| 2 | c_4410 | Femme | 57 | 1 | 1376.82 | 501.56 | |
| 3 | c_4410 | Femme | 57 | 0 | 1376.82 | 501.56 | |
| 4 | c_4410 | Femme | 57 | 0 | 1376.82 | 501.56 | |
| … | … | … | … | … | … | … | |
| 640729 | c_84 | Femme | 42 | 0 | 427.51 | 254.09 | |
| 640730 | c_84 | Femme | 42 | 0 | 427.51 | 254.09 | |
| 640731 | c_84 | Femme | 42 | 1 | 427.51 | 254.09 | |
| 640732 | c_84 | Femme | 42 | 0 | 427.51 | 254.09 | |
| 640733 | c_84 | Femme | 42 | 1 | 427.51 | 254.09 | |

| | C.A_catégories 1 | C.A_catégories 2 | Nombre de transactions | \ |
|---|---|---|---|---|
| 0 | 729.27 | 145.99 | 76 | |
| 1 | 729.27 | 145.99 | 76 | |
| 2 | 729.27 | 145.99 | 76 | |
| 3 | 729.27 | 145.99 | 76 | |
| 4 | 729.27 | 145.99 | 76 | |
| … | … | … | … | |
| 640729 | 173.42 | 0.00 | 28 | |
| 640730 | 173.42 | 0.00 | 28 | |
| 640731 | 173.42 | 0.00 | 28 | |
| 640732 | 173.42 | 0.00 | 28 | |
| 640733 | 173.42 | 0.00 | 28 | |

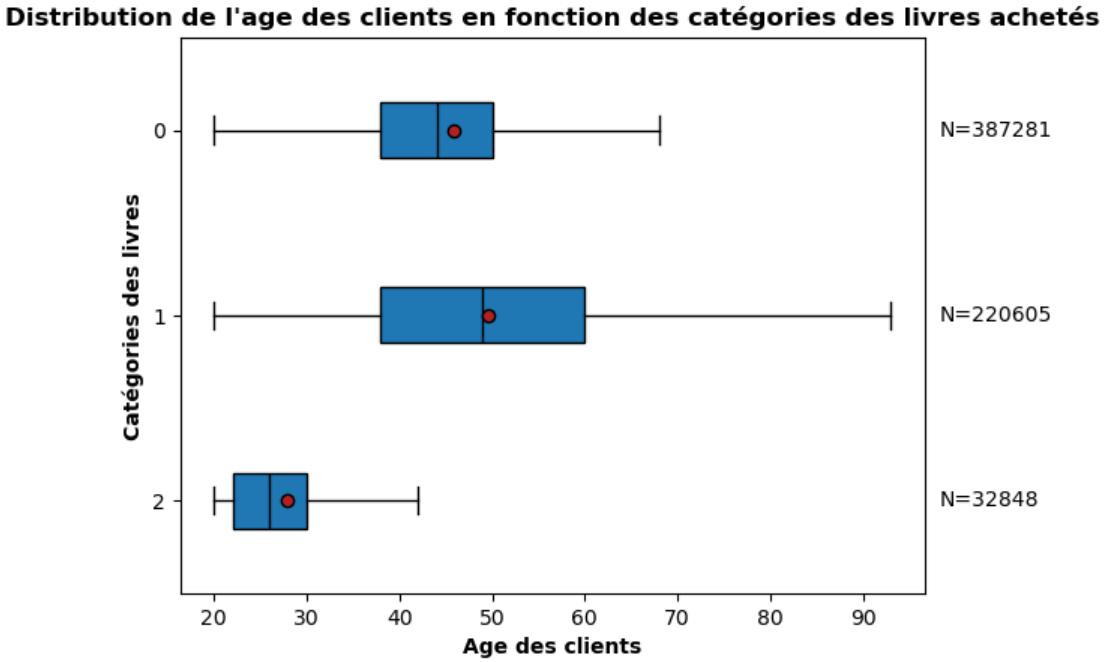
| | Nombre de transactions catégorie 0 | \ |
|---|---|---|
| 0 | 38.0 | |
| 1 | 38.0 | |
| 2 | 38.0 | |
| 3 | 38.0 | |
| 4 | 38.0 | |
| … | … | |
| 640729 | 21.0 | |
| 640730 | 21.0 | |
| 640731 | 21.0 | |
| 640732 | 21.0 | |
| 640733 | 21.0 | |

| | Nombre de transactions catégorie 1 | \ |
|---|---|---|
| 0 | 37.0 | |
| 1 | 37.0 | |
| 2 | 37.0 | |
| 3 | 37.0 | |
| 4 | 37.0 | |
| … | … | |
| 640729 | 7.0 | |

```
640730                                    7.0
640731                                    7.0
640732                                    7.0
640733                                    7.0

        Nombre de transactions catégorie 2  Nb_Session  Panier_Moyen
0                                      1.0          52         26.48
1                                      1.0          52         26.48
2                                      1.0          52         26.48
3                                      1.0          52         26.48
4                                      1.0          52         26.48
…                                      …            …          …
640729                                 0.0          14         30.54
640730                                 0.0          14         30.54
640731                                 0.0          14         30.54
640732                                 0.0          14         30.54
640733                                 0.0          14         30.54

[640734 rows x 14 columns]
```

[ ]:
```python
# Visualisation à l'aide d'un graphique en boîte à moustache

X = 'categ'
Y = 'age'

modalites = sorted(df_lapage[X].unique(), reverse=True) # sorted pour trie les␣
 ↪catégories en ordre inverse

groupes = []
for m in modalites:
    groupes.append(df_lapage[df_lapage[X]==m][Y])

# Calculer les effectifs pour chaque groupe
effectifs = [len(g) for g in groupes]


medianprops = {'color':"black"}
meanprops = {'marker':'o', 'markeredgecolor':'black',
            'markerfacecolor':'firebrick'}

# Ajouter les effectifs pour chaque groupe
for i, eff in enumerate(effectifs, start=1):
    plt.text(1.02, i, f'N={eff}', va='center', ha='left', transform=plt.gca().
 ↪get_yaxis_transform())
```

```
plt.boxplot(groupes, labels=modalites, showfliers=False,␣
 ↪medianprops=medianprops,
            vert=False, patch_artist=True, showmeans=True, meanprops=meanprops)
plt.title("Distribution de l'age des clients en fonction des catégories des␣
 ↪livres achetés", fontweight='bold', fontsize = 12)
plt.ylabel("Catégories des livres", fontweight='bold')
plt.xlabel('Age des clients', fontweight='bold')
plt.show()
```




### 0.0.29 Analyse visuel des boites à moustaches

La distribution de l'âge des acheteurs varie selon la catégorie, cela pourrait laisser suggérer que différentes catégories de livres sont préférées par des groupes d'âge distincts.

### 0.0.30 Hypothèses

**Hypothèse Nulle (H0)** : Il n'y a pas de lien entre l'âge des clients et la catégorie des livres achetés.

**Hypothèse Alternative (H1)** : Il existe un lien entre l'âge des clients et la catégorie des livres achetés.

[ ]:
```
# ANOVA

X = "categ"
Y = "age"
```

```
def eta_squared(x,y):
    moyenne_y = y.mean()
    classes = []
    for classe in x.unique():
        yi_classe = y[x==classe]
        classes.append({'ni': len(yi_classe),
                        'moyenne_classe': yi_classe.mean()})
    SCT = sum([(yj-moyenne_y)**2 for yj in y])
    SCE = sum([c['ni']*(c['moyenne_classe']-moyenne_y)**2 for c in classes])
    return SCE/SCT

print("Le rapport de corrélation vaut",␣
 ↪round(eta_squared(df_lapage[X],df_lapage[Y]), 2))
```

```
Le rapport de corrélation vaut 0.11
```

#### 0.0.31 Conclusion

Le résultat du test ANOVA montre que le rapport de corrélation (eta-squared) 0.11 :

-Cela qui indique que 11% de la variabilité de l'âge est expliquée par les différences entre les catégories de livres.

Ce niveau de corrélation suggère **qu'il n'y a pas d'association entre l'âge des clients et les catégories de livres qu'ils achètent.**

#### 0.0.32 Visualisation du nombre de transactions par categories en fonction de de l'age

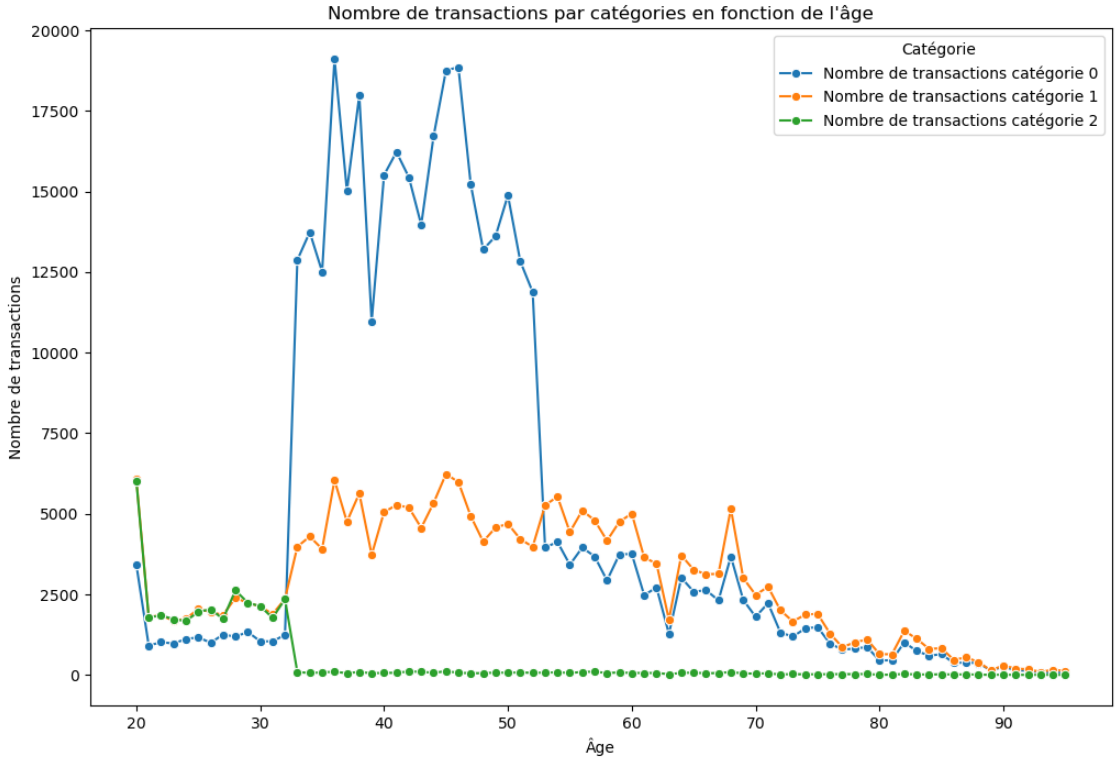
#### 0.0.33 Variables

- Âge : Variable quantitavive
- Nombre de transactions : Variable quantitative.

**Variables Quantitatives -> Visualisation à l'aide d'un nuage de point ou graphique en ligne :**

[ ]:
```
# Préparation de la matrice pour crée un tableau de contingence :
df_R_age_categ = df_R[['age','Nombre de transactions catégorie 0","Nombre de␣
 ↪transactions catégorie 1","Nombre de transactions catégorie 2"]]

# Créer un tableau de contingence : Age du client et le nombre de transactions␣
 ↪par categories de livres achetées
tableau_contingence = df_R_age_categ.groupby('age').sum()

# Convertir les colonnes float en type int
tableau_contingence = tableau_contingence.astype({
    'Nombre de transactions catégorie 0': 'int',
    'Nombre de transactions catégorie 1': 'int',
```

```
    'Nombre de transactions catégorie 2': 'int'
})

# Transposer le tableau pour avoir les catégories à la verticale
tableau_transpose = tableau_contingence.T

# Convertir le tableau transposé en DataFrame
df_transpose = tableau_transpose.reset_index().melt(id_vars='index',␣
 ↪var_name='age', value_name='Nb_transactions')
df_transpose.rename(columns={'index': 'category'}, inplace=True)

# Afficher la représentation graphique du nombre de transaction des trois␣
 ↪catégories en fonction de l'âge
plt.figure(figsize=(12, 8))
sns.lineplot(data=df_transpose, x='age', y='Nb_transactions', hue='category',␣
 ↪marker='o')
plt.title('Nombre de transactions par catégories en fonction de l\'âge')
plt.xlabel('Âge')
plt.ylabel('Nombre de transactions')
plt.legend(title='Catégorie')
plt.show()
```



#### 0.0.34 Discrétiser l'âge en deux groupes : Avant et apres 33ans :

[ ]:
```
# Discrétiser l'âge en deux groupes : avant et après 33 ans
df_R['age_groupes'] = pd.cut(df_R['age'], bins=[0, 33, 100], labels=['<= 33␣
↪ans', '> 33 ans'])
```

#### 0.0.35 effectuer une analyse de Variance (ANOVA) avec les deux groupes d'age identifié pour effectuer une analyse de Variance (ANOVA)

[ ]:
```
# Créer un tableau de contingence avec les groupes d'âge et les catégories de␣
↪transactions
tableau_contingence_age_categ = df_R.groupby('age_groupes',␣
↪observed=True)[['Nombre de transactions catégorie 0', 'Nombre de␣
↪transactions catégorie 1', 'Nombre de transactions catégorie 2']].sum().
↪reset_index()

tableau_contingence_age_categ
```

[ ]:
```
  age_groupes  Nombre de transactions catégorie 0  \
0   <= 33 ans                             29565.0
1    > 33 ans                            357716.0

   Nombre de transactions catégorie 1  Nombre de transactions catégorie 2
0                             33963.0                             29962.0
1                            186642.0                              2886.0
```

[ ]:
```
# Groupes d'âge pour l'analyse ANOVA
groupe1 =␣
↪tableau_contingence_age_categ[tableau_contingence_age_categ['age_groupes']␣
↪== '<= 33 ans'][['Nombre de transactions catégorie 0', 'Nombre de␣
↪transactions catégorie 1', 'Nombre de transactions catégorie 2']].values.
↪flatten()
groupe2 =␣
↪tableau_contingence_age_categ[tableau_contingence_age_categ['age_groupes']␣
↪== '> 33 ans'][['Nombre de transactions catégorie 0', 'Nombre de␣
↪transactions catégorie 1', 'Nombre de transactions catégorie 2']].values.
↪flatten()

# Test ANOVA
f_stat, p_value_anova = scipy.stats.f_oneway(groupe1, groupe2)

print("F-statistic:", f_stat)
print("P-value:", p_value_anova)

# Conclusion
alpha = 0.05
if p_value_anova < alpha:
```

```
    print("Nous rejetons l'hypothèse nulle (H0). Il n'éxiste pas de lien entre␣
 ↪l'âge des clients et les categories de livres acheters.")
else:
    print("Nous ne pouvons pas rejeter l'hypothèse nulle (H0). Il éxiste un␣
 ↪lien entre l'âge des clients et les categories de livres acheters.")
```

```
F-statistic: 2.1790766586542367
P-value: 0.21394097515852772
Nous ne pouvons pas rejeter l'hypothèse nulle (H0). Il éxiste un lien entre
l'âge des clients et les categories de livres acheters.
```

Analyse des résultats de l'ANOVA Tableau de contingence Le tableau de contingence présente le nombre de transactions par catégorie pour les deux groupes d'âge (<= 33 ans et > 33 ans). Les résultats montrent que :

Les clients de plus de 33 ans ont un nombre total de transactions (pour toutes les catégories combinées) plus élevé que ceux de 33 ans ou moins. Résultats de l'ANOVA F-statistic : 2.179876858542367 P-value : 0.21394997515852772 Interprétation des résultats de l'ANOVA La p-value obtenue (0.21394997515852772) est supérieure au seuil de signification de 0.05. Par conséquent, nous ne pouvons pas rejeter l'hypothèse nulle (H0).

Conclusion : Il n'y a pas de différence significative dans les moyennes des transactions entre les groupes d'âge (<= 33 ans et > 33 ans) pour les différentes catégories de produits. Cela signifie que l'âge des clients n'a pas un effet significatif sur le nombre moyen de transactions par catégorie de produit.