Analyse des ventes d'une librairie

avec Python

Importation des données :

In [150…
```
# Importation des librairies :
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

In [2]:
```
import scipy as sp
from scipy.stats import pearsonr
from scipy.stats import bartlett
from scipy.stats import shapiro
from scipy.stats import chi2_contingency
from scipy.stats import kendalltau, spearmanr
import statsmodels.api as sm
```

In [3]:
```
# Importation des données :
cust = pd.read_csv('customers.csv')
prod = pd.read_csv('products.csv')
trans = pd.read_csv('transactions.csv')
```

Partie 1 : Nettoyage et préparation des données

1.1. Nettoyage des données :

1.1.1. Données "Customers" :

In [4]:
```
cust.head()
```

Out[4]:

	client_id	sex	birth
0	c_4410	f	1967
1	c_7839	f	1975
2	c_1699	f	1984
3	c_5961	f	1962
4	c_5320	m	1943

In [5]:
```
cust.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8623 entries, 0 to 8622
```

```
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   client_id  8623 non-null   object
 1   sex        8623 non-null   object
 2   birth      8623 non-null   int64
dtypes: int64(1), object(2)
memory usage: 202.2+ KB
```

In [6]:
```
cust.nunique()
```

Out[6]:
```
client_id    8623
sex             2
birth          76
dtype: int64
```

1.1.2. Données "Products" :

In [7]:
```
prod.head()
```

Out[7]:

	id_prod	price	categ
0	0_1421	19.99	0
1	0_1368	5.13	0
2	0_731	17.99	0
3	1_587	4.99	1
4	0_1507	3.99	0

In [8]:
```
prod.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3287 entries, 0 to 3286
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id_prod  3287 non-null   object
 1   price    3287 non-null   float64
 2   categ    3287 non-null   int64
dtypes: float64(1), int64(1), object(1)
memory usage: 77.2+ KB
```

In [9]:
```
prod.nunique()
```

Out[9]:
```
id_prod    3287
price      1455
categ         3
dtype: int64
```

In [10]:
```
prod.categ.value_counts()
```

Out[10]:
```
0    2309
1     739
2     239
Name: categ, dtype: int64
```

1.1.3. Données "Transactions" :

In [11]:
```
trans.head()
```

Out[11]:

	id_prod	date	session_id	client_id

0	0_1518	2022-05-20 13:21:29.043970	s_211425	c_103
1	1_251	2022-02-02 07:55:19.149409	s_158752	c_8534
2	0_1277	2022-06-18 15:44:33.155329	s_225667	c_6714
3	2_209	2021-06-24 04:19:29.835891	s_52962	c_6941
4	0_1509	2023-01-11 08:22:08.194479	s_325227	c_4232

In [12]:
```
trans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 679532 entries, 0 to 679531
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   id_prod     679532 non-null  object
 1   date        679532 non-null  object
 2   session_id  679532 non-null  object
 3   client_id   679532 non-null  object
dtypes: object(4)
memory usage: 20.7+ MB
```

In [13]:
```
trans.nunique()
```

Out[13]:
```
id_prod         3267
date          679371
session_id    342316
client_id       8602
dtype: int64
```

Traitement des doublons :

In [14]:
```
# doublons "transaction"
trans_dup = trans[trans.duplicated()]
trans_dup
```

Out[14]:

	id_prod	date	session_id	client_id
27778	T_0	test_2021-03-01 02:30:02.237437	s_0	ct_1
52424	T_0	test_2021-03-01 02:30:02.237419	s_0	ct_0
96687	T_0	test_2021-03-01 02:30:02.237412	s_0	ct_1
130188	T_0	test_2021-03-01 02:30:02.237419	s_0	ct_0
139339	T_0	test_2021-03-01 02:30:02.237443	s_0	ct_1
...	...	...	...	...
653098	T_0	test_2021-03-01 02:30:02.237432	s_0	ct_0
657830	T_0	test_2021-03-01 02:30:02.237417	s_0	ct_0
662081	T_0	test_2021-03-01 02:30:02.237427	s_0	ct_1
671647	T_0	test_2021-03-01 02:30:02.237424	s_0	ct_1
679180	T_0	test_2021-03-01 02:30:02.237425	s_0	ct_1

126 rows × 4 columns

Suppression des doublons :

In [15]:
```
trans = trans.drop_duplicates()
trans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 679406 entries, 0 to 679531
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   id_prod     679406 non-null  object
 1   date        679406 non-null  object
 2   session_id  679406 non-null  object
 3   client_id   679406 non-null  object
dtypes: object(4)
memory usage: 25.9+ MB
```

126 doublons supprimés

1.1.4. Rapprochement des données :

In [16]:
```
set(trans.id_prod) - set(prod.id_prod)
```

Out[16]:
```
{'0_2245'}
```

In [17]:
```
trans[trans.id_prod == '0_2245']
```

Out[17]:

	id_prod	date	session_id	client_id
2633	0_2245	2022-09-23 07:22:38.636773	s_272266	c_4746
10106	0_2245	2022-07-23 09:24:14.133889	s_242482	c_6713
11727	0_2245	2022-12-03 03:26:35.696673	s_306338	c_5108
15675	0_2245	2021-08-16 11:33:25.481411	s_76493	c_1391
16377	0_2245	2022-07-16 05:53:01.627491	s_239078	c_7954
...	...	...	...	...
669730	0_2245	2021-08-25 09:06:03.504061	s_80395	c_131
670682	0_2245	2022-03-06 19:59:19.462288	s_175311	c_4167
671286	0_2245	2022-05-16 11:35:20.319501	s_209381	c_4453
675679	0_2245	2022-02-11 09:05:43.952857	s_163405	c_1098
677996	0_2245	2021-12-14 22:34:54.589921	s_134446	c_4854

221 rows × 4 columns

Le produit référencé "0_2245" et de catégorie 0 est vendu 221 fois sans posséder de données dans la table "produit".

In [149...
```
set(prod.id_prod) - set(trans.id_prod)
```

Out[149]:
```
{'0_1014',
 '0_1016',
 '0_1025',
 '0_1062',
 '0_1119',
 '0_1318',
 '0_1620',
 '0_1624',
```

```
 '0_1645',
 '0_1780',
 '0_1800',
 '0_2308',
 '0_299',
 '0_310',
 '0_322',
 '0_510',
 '1_0',
 '1_394',
 '2_72',
 '2_86',
 '2_87'}
```

In [18]:
```
len(set(prod.id_prod) - set(trans.id_prod))
```

Out[18]:
```
21
```

21 produits référencés ne font pas partie d'une transaction

Rapprochement des tables "transactions" et "products" :

In [19]:
```
trans_prod = pd.merge(trans, prod, on ='id_prod', how = 'left')
```

In [20]:
```
trans_prod.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 679406 entries, 0 to 679405
Data columns (total 6 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   id_prod     679406 non-null  object 
 1   date        679406 non-null  object 
 2   session_id  679406 non-null  object 
 3   client_id   679406 non-null  object 
 4   price       679185 non-null  float64
 5   categ       679185 non-null  float64
dtypes: float64(2), object(4)
memory usage: 36.3+ MB
```

Rapprochement des trois tables :

In [21]:
```
data = pd.merge(trans_prod, cust, on ='client_id', how = 'left')
```

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 679406 entries, 0 to 679405
Data columns (total 8 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   id_prod     679406 non-null  object 
 1   date        679406 non-null  object 
 2   session_id  679406 non-null  object 
 3   client_id   679406 non-null  object 
 4   price       679185 non-null  float64
 5   categ       679185 non-null  float64
 6   sex         679406 non-null  object 
 7   birth       679406 non-null  int64  
dtypes: float64(2), int64(1), object(5)
memory usage: 46.7+ MB
```

1.1.5. Traitement des valeurs nulles est des incohérences :

In [23]:
```
data.isna().sum()
```

Out[23]:
```
id_prod        0
date           0
session_id     0
client_id      0
price        221
categ        221
sex            0
birth          0
dtype: int64
```

In [24]:
```
data_nan = data[(data.categ.isna()) | (data.price.isna())]
data_nan
```

Out[24]:

	id_prod	date	session_id	client_id	price	categ	sex	birth
2633	0_2245	2022-09-23 07:22:38.636773	s_272266	c_4746	NaN	NaN	m	1940
10106	0_2245	2022-07-23 09:24:14.133889	s_242482	c_6713	NaN	NaN	f	1963
11727	0_2245	2022-12-03 03:26:35.696673	s_306338	c_5108	NaN	NaN	m	1978
15675	0_2245	2021-08-16 11:33:25.481411	s_76493	c_1391	NaN	NaN	m	1991
16377	0_2245	2022-07-16 05:53:01.627491	s_239078	c_7954	NaN	NaN	m	1973
...	...	...	...	...	...	...	...	...
669606	0_2245	2021-08-25 09:06:03.504061	s_80395	c_131	NaN	NaN	m	1981
670558	0_2245	2022-03-06 19:59:19.462288	s_175311	c_4167	NaN	NaN	f	1979
671162	0_2245	2022-05-16 11:35:20.319501	s_209381	c_4453	NaN	NaN	m	1981
675554	0_2245	2022-02-11 09:05:43.952857	s_163405	c_1098	NaN	NaN	m	1986
677871	0_2245	2021-12-14 22:34:54.589921	s_134446	c_4854	NaN	NaN	m	1968

221 rows × 8 columns

Imputation des prix nuls par la moyenne des prix de même catégorie :

Les valeur nulles concernent qu'un seul produit référencé "0_2245" de catégorie "0"

In [25]:
```
data_nan.id_prod.value_counts()
```

Out[25]:
```
0_2245    221
Name: id_prod, dtype: int64
```

In [26]:
```
data.categ.value_counts()
```

Out[26]:
```
0.0    415533
1.0    227169
2.0     36483
Name: categ, dtype: int64
```

In [27]:
```
# moyenne des prix de la catégorie "0"
mean_price_categ_0 = round(data[data.categ == 0].price.mean(), 2)
# Remplacement des prixs manquants par la moyenne des prix de la catégorie "0"
data.price = data.price.fillna(mean_price_categ_0)
```

Imputation des catégories nulles par celle du produit :

In [28]:
```
# Remplacement des catégories manquantes par "0"
data.categ = data.categ.fillna(0)
```

In [29]:
```
data[data.id_prod == '0_2245']
```

Out[29]:

	id_prod	date	session_id	client_id	price	categ	sex	birth
2633	0_2245	2022-09-23 07:22:38.636773	s_272266	c_4746	10.64	0.0	m	1940
10106	0_2245	2022-07-23 09:24:14.133889	s_242482	c_6713	10.64	0.0	f	1963
11727	0_2245	2022-12-03 03:26:35.696673	s_306338	c_5108	10.64	0.0	m	1978
15675	0_2245	2021-08-16 11:33:25.481411	s_76493	c_1391	10.64	0.0	m	1991
16377	0_2245	2022-07-16 05:53:01.627491	s_239078	c_7954	10.64	0.0	m	1973
...	...	...	...	...	...	...	...	...
669606	0_2245	2021-08-25 09:06:03.504061	s_80395	c_131	10.64	0.0	m	1981
670558	0_2245	2022-03-06 19:59:19.462288	s_175311	c_4167	10.64	0.0	f	1979
671162	0_2245	2022-05-16 11:35:20.319501	s_209381	c_4453	10.64	0.0	m	1981
675554	0_2245	2022-02-11 09:05:43.952857	s_163405	c_1098	10.64	0.0	m	1986
677871	0_2245	2021-12-14 22:34:54.589921	s_134446	c_4854	10.64	0.0	m	1968

221 rows × 8 columns

Traitement des valeurs incohérantes :

In [30]:
```
data.describe(include = 'all')
```

Out[30]:

	id_prod	date	session_id	client_id	price	categ	sex	birth
count	679406	679406	679406	679406	679406.000000	679406.000000	679406	679406.000000
unique	3267	679371	342316	8602	NaN	NaN	2	NaN
top	1_369	test 2021-03-01 02:30:02.237449	s_0	c_1609	NaN	NaN	m	NaN
freq	2252	2	74	25488	NaN	NaN	340967	NaN
mean	NaN	NaN	NaN	NaN	17.450547	0.441761	NaN	1977.813665
std	NaN	NaN	NaN	NaN	18.326442	0.594984	NaN	13.575971
min	NaN	NaN	NaN	NaN	-1.000000	0.000000	NaN	1929.000000
25%	NaN	NaN	NaN	NaN	8.870000	0.000000	NaN	1970.000000
50%	NaN	NaN	NaN	NaN	13.990000	0.000000	NaN	1980.000000
75%	NaN	NaN	NaN	NaN	18.990000	1.000000	NaN	1987.000000
max	NaN	NaN	NaN	NaN	300.000000	2.000000	NaN	2004.000000

In [31]:
```
# Visualisation des données incohérantes avec prix négatif et date commençant par "test"
data_inco = data[(data.price < 0) | (data.date.str.startswith('test'))]
data_inco
```

Out[31]:

	id_prod	date	session_id	client_id	price	categ	sex	birth

3019	T_0	test 2021-03-01 02:30:02.237419	s_0	ct_0	-1.0	0.0	f	2001
5138	T_0	test 2021-03-01 02:30:02.237425	s_0	ct_0	-1.0	0.0	f	2001
9668	T_0	test 2021-03-01 02:30:02.237437	s_0	ct_1	-1.0	0.0	m	2001
10728	T_0	test 2021-03-01 02:30:02.237436	s_0	ct_0	-1.0	0.0	f	2001
15292	T_0	test 2021-03-01 02:30:02.237430	s_0	ct_0	-1.0	0.0	f	2001
...	...	...	...	...	...	...	...	...
577222	T_0	test 2021-03-01 02:30:02.237424	s_0	ct_0	-1.0	0.0	f	2001
592959	T_0	test 2021-03-01 02:30:02.237422	s_0	ct_1	-1.0	0.0	m	2001
607783	T_0	test 2021-03-01 02:30:02.237412	s_0	ct_0	-1.0	0.0	f	2001
625936	T_0	test 2021-03-01 02:30:02.237422	s_0	ct_0	-1.0	0.0	f	2001
670556	T_0	test 2021-03-01 02:30:02.237449	s_0	ct_1	-1.0	0.0	m	2001

74 rows × 8 columns

Tout les livres avec des données incohérantes sont de catégorie "0"

In [32]:
```
data_inco.categ.value_counts()
```

Out[32]:
```
0.0    74
Name: categ, dtype: int64
```

In [33]:
```
data_inco.id_prod.value_counts()
```

Out[33]:
```
T_0    74
Name: id_prod, dtype: int64
```

In [34]:
```
# Remplacement des dates qui commencent par "test"
data.loc[data.date.str.startswith("test"), 'date'] = data.loc[data.date.str.startswith("
```

In [35]:
```
# Remplacement des prix négatifs par la moyennes des prix de la catégorie "0"
data.loc[data.price < 0, 'price'] = mean_price_categ_0
```

In [36]:
```
# Vérification des résultats
data[data.id_prod == 'T_0']
```

Out[36]:

	id_prod	date	session_id	client_id	price	categ	sex	birth
3019	T_0	2021-03-01 02:30:02.237419	s_0	ct_0	10.64	0.0	f	2001
5138	T_0	2021-03-01 02:30:02.237425	s_0	ct_0	10.64	0.0	f	2001
9668	T_0	2021-03-01 02:30:02.237437	s_0	ct_1	10.64	0.0	m	2001
10728	T_0	2021-03-01 02:30:02.237436	s_0	ct_0	10.64	0.0	f	2001
15292	T_0	2021-03-01 02:30:02.237430	s_0	ct_0	10.64	0.0	f	2001
...	...	...	...	...	...	...	...	...
577222	T_0	2021-03-01 02:30:02.237424	s_0	ct_0	10.64	0.0	f	2001
592959	T_0	2021-03-01 02:30:02.237422	s_0	ct_1	10.64	0.0	m	2001
607783	T_0	2021-03-01 02:30:02.237412	s_0	ct_0	10.64	0.0	f	2001
625936	T_0	2021-03-01 02:30:02.237422	s_0	ct_0	10.64	0.0	f	2001
670556	T_0	2021-03-01 02:30:02.237449	s_0	ct_1	10.64	0.0	m	2001

74 rows × 8 columns

1.2. Préparation des données :

1.2.1. Découpage par jours , mois, trimestres, années :

Formatage des dates :

In [37]:
```
#Formatage des dates:
data['date'] =  pd.to_datetime(data['date'], format='%Y/%m/%d %H:%M:%S.%f', errors = 'co
```

In [38]:
```
# Vérification des éventuelles erreurs de formatage
data.date.isna().sum()
```

Out[38]:
```
0
```

Découpage des données par périodes :

In [39]:
```
# découpage par jours
data['jour'] = data['date'].dt.strftime('%Y-%m-%d')
data['jour'] = pd.to_datetime(data['jour'])
# découpage par mois
data['mois'] = data.date.dt.to_period('M')
data['mois'] = data['mois'].astype(str)
```

In [40]:
```
# découpage en trimestre
# date de début
start_date = data.date.min()
# date de fin
end_date = data.date.max()
# nombre de mois
nbr_mois = pd.date_range(start_date, end_date, freq='M')
# nombre de trimestres
n = int(len(nbr_mois)/3)
```

In [41]:
```
# les labels de trimestres
labels = []
for i in range(1,n+1):
    labels.append(i)
```

In [42]:
```
data['trimestre'] = pd.cut(data.date, n, labels = labels)
```

In [43]:
```
# Découpage par années
import datetime
def year(date):
    if date <  datetime.datetime(2022, 3, 1):
        return "Année 1"
    elif date >=  datetime.datetime(2022, 3, 1):
        return "Année 2"
data['annee_exo'] = data['jour'].apply(year)
```

In [44]:
```
# Découpage par jours ouvrés et en weekend
data['jour_semaine']= data.date.dt.day_name()
def sem(jour):
    if jour in ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']:
        return 'semaine'
    elif jour in ['Saturday', 'Sunday']:
```

```
        return 'weekend'
data['week'] = data['jour_semaine'].apply(sem)
```

Transformation des dates de naissances en ages :

In [45]:
```
# age
data['age'] = data.date.dt.year - data.birth
```

In [46]:
```
labels_age = [ '< 30 ans', '30 - 50 ans', '> 50 ans']
data['cat_age'] = pd.cut(data.age, bins = [16, 29, 50, 94], labels = labels_age )
data['cat_age'] = data['cat_age'].astype(str)
```

In [47]:
```
# Remplacement de 'f' par 'femme' et 'm' par 'homme'
data.sex = data.sex.replace('f', 'femme')
data.sex = data.sex.replace('m', 'homme')
```

In [151…
```
# Remplacement des catégories
data.categ = data.categ.replace(0.0, 'Cat 0')
data.categ = data.categ.replace(1.0, 'Cat 1')
data.categ = data.categ.replace(2.0, 'Cat 2')
```

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

Out[49]:

	id_prod	date	session_id	client_id	price	categ	sex	birth	jour	mois	trimestre	annee_exo
0	0_1518	2022-05-20 13:21:29.043970	s_211425	c_103	4.18	Cat 0	femme	1986	2022-05-20	2022-05	5	Année 2
1	1_251	2022-02-02 07:55:19.149409	s_158752	c_8534	15.99	Cat 1	homme	1988	2022-02-02	2022-02	4	Année 1
2	0_1277	2022-06-18 15:44:33.155329	s_225667	c_6714	7.99	Cat 0	femme	1968	2022-06-18	2022-06	6	Année 2
3	2_209	2021-06-24 04:19:29.835891	s_52962	c_6941	69.99	Cat 2	homme	2000	2021-06-24	2021-06	2	Année 1
4	0_1509	2023-01-11 08:22:08.194479	s_325227	c_4232	4.99	Cat 0	homme	1980	2023-01-11	2023-01	8	Année 2

Partie 2 : Analyse des données

In [50]:
```
# fonction de groupage
def grby_sum(df, x, y):
    return df.groupby(x)[[y]].sum().reset_index()
def grby_count(df, x, y):
    return df.groupby(x)[[y]].count().reset_index()
```

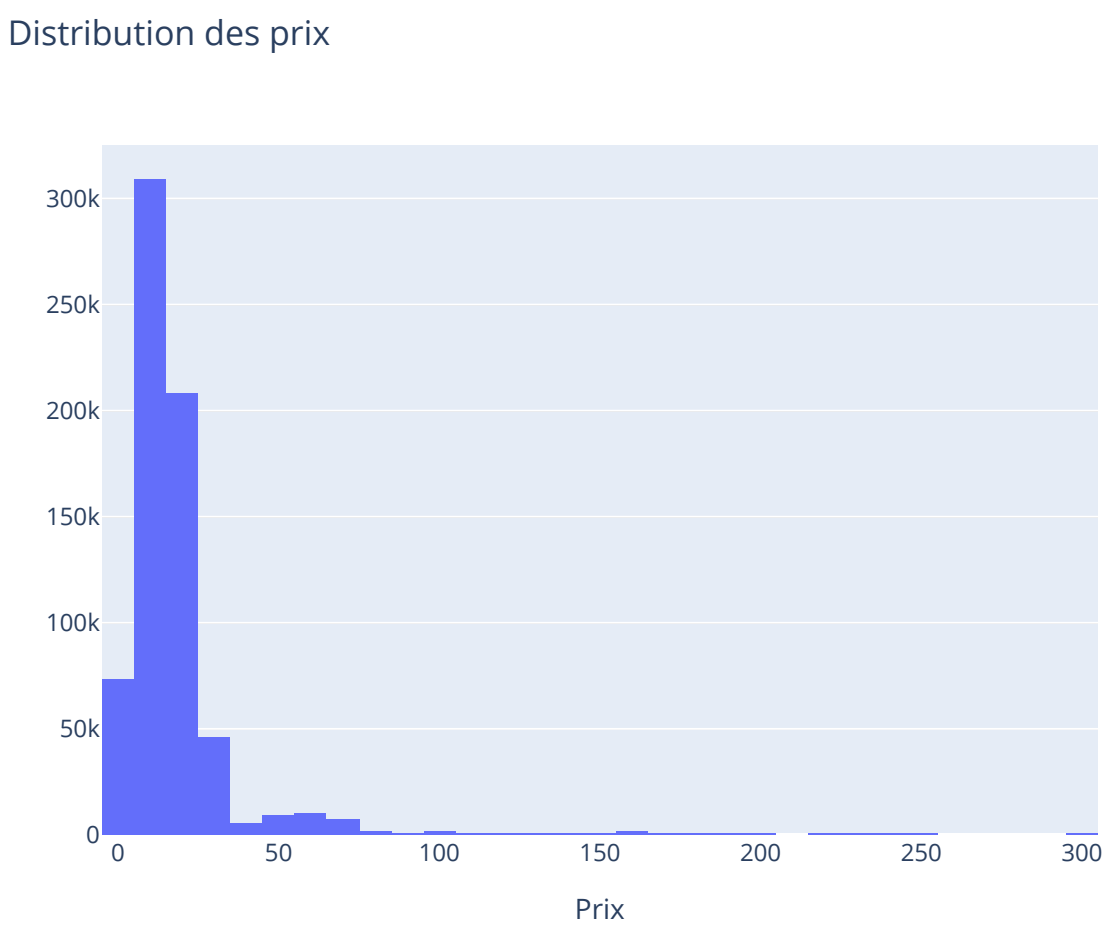
2.1. Analyse des prix :

Les mesures de tendance centrale :

In [51]:
```
print('le prix modale est égal à :', data.price.mode()[0], '€')
print('le prix moyen est égal à :', round(data.price.mean(), 2),'€')
print('le prix médian est égal à :', data.price.median(), '€')
# Histogramme
fig = px.histogram(data, x="price", nbins = 30, title = "Distribution des prix")
fig.update_layout(
```

```
    xaxis_title = "Prix",
    yaxis_title = ""
)
fig.show()
```

```
le prix modale est égal à : 15.99 €
le prix moyen est égal à : 17.45 €
le prix médian est égal à : 13.99 €
```



La majorité des prix se concentrent à gauche et sont inférieurs à 50 €

Les mesures de dispersion :

In [153…

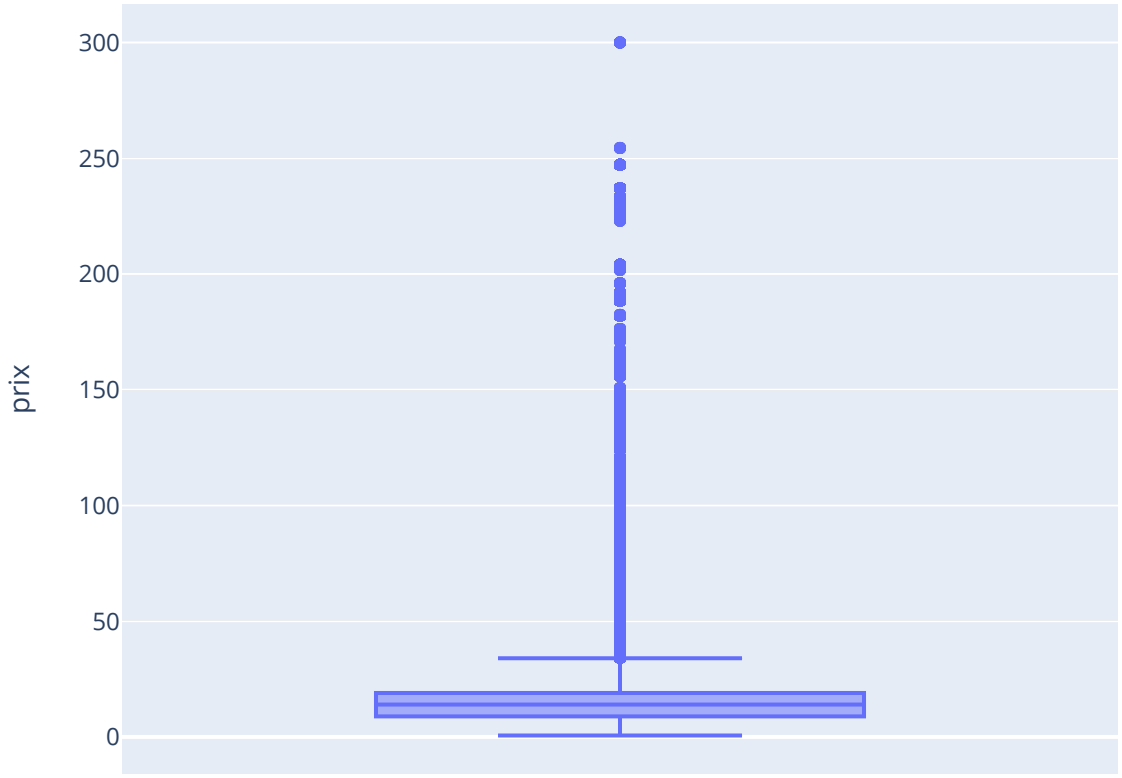
```
var_price = round(data.price.var(),3)
print('\nLa variance empirique du prix est égale à :',var_price)# Variance emperique
var_sb_price = round(data.price.var(ddof=0),3)
print('La variance empirique sans biais du prix est égale à :',var_sb_price)# variance e
std_price = round(data.price.std(), 3)
print("L'écart-type empirique du prix est égal à",std_price,' €\n') # écart-type empiriq
```

```
La variance empirique du prix est égale à : 335.826
La variance empirique sans biais du prix est égale à : 335.826
L'écart-type empirique du prix est égal à 18.326  €
```

In [157…

```
# Représentation de la répartition des prix:
fig = px.box(data, y="price")
fig.update_layout(
    title = "Répartition des prix",
    yaxis_title = "prix"
```

```
)
fig.show()
```



2.2. Analyse du chiffre d'affaires :

2.2.1. Chiffre d'affaires :

In [54]:
```
data1 = data[data.annee_exo == 'Année 1'] # données Année 1
data2 = data[data.annee_exo == 'Année 2'] # données Année 2

ca_total_annee1 = round(data1.price.sum(), 2)
ca_total_annee2 = round(data2.price.sum(), 2)
ca_total = ca_total_annee1 + ca_total_annee2

print("Le chiffre d'affaires total pour la première année est de ", ca_total_annee1, "€")
print("Le chiffre d'affaires total pour la seconde année est de ", ca_total_annee2, "€")
print("Le chiffre d'affaires total pour les deux années est de ", ca_total, "€")
```

```
Le chiffre d'affaires total pour la première année est de  5833620.33 €
Le chiffre d'affaires total pour la seconde année est de  6023247.15 €
Le chiffre d'affaires total pour les deux années est de  11856867.48 €
```

2.2.2. Tendance et évolution du chiffre d'affaires :

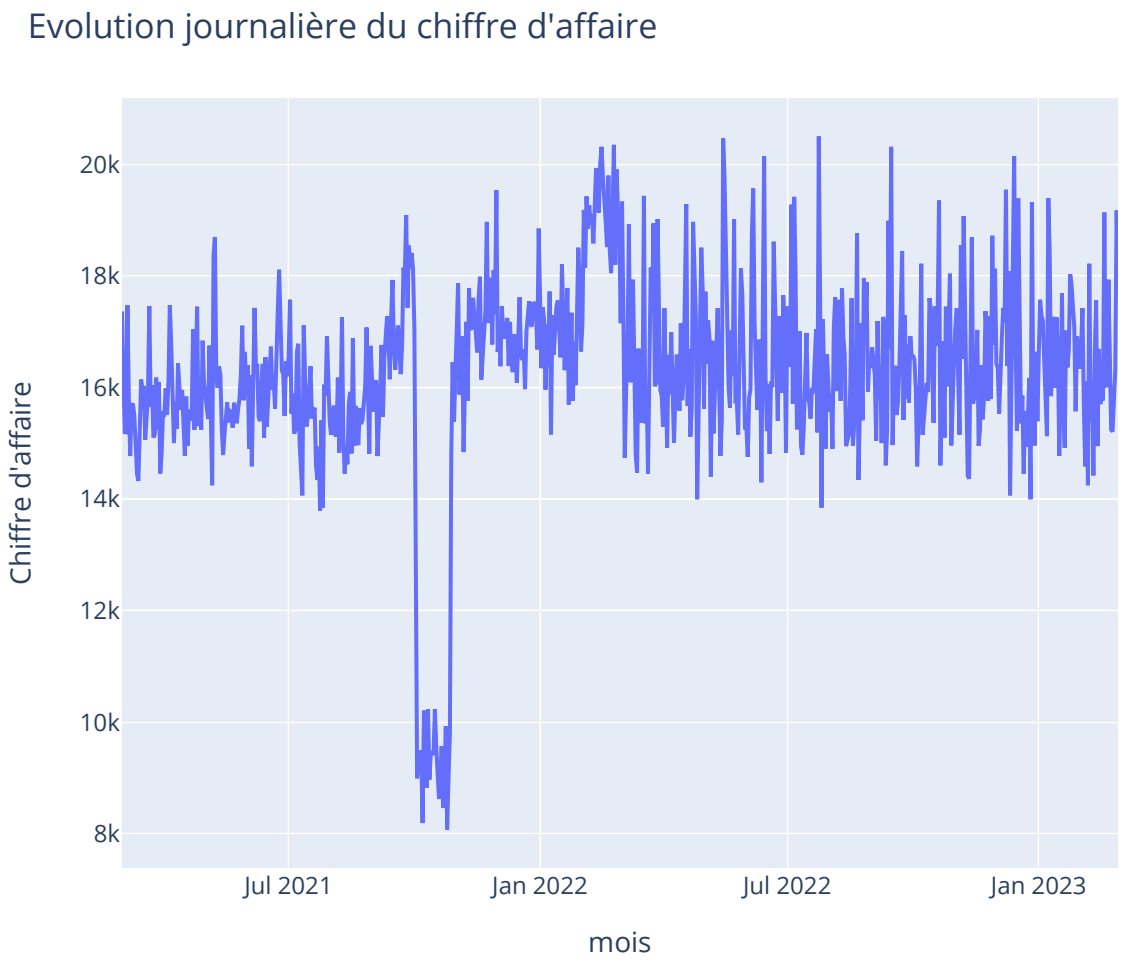
Evolution du chiffre d'affaires par jour/mois/trimestre :

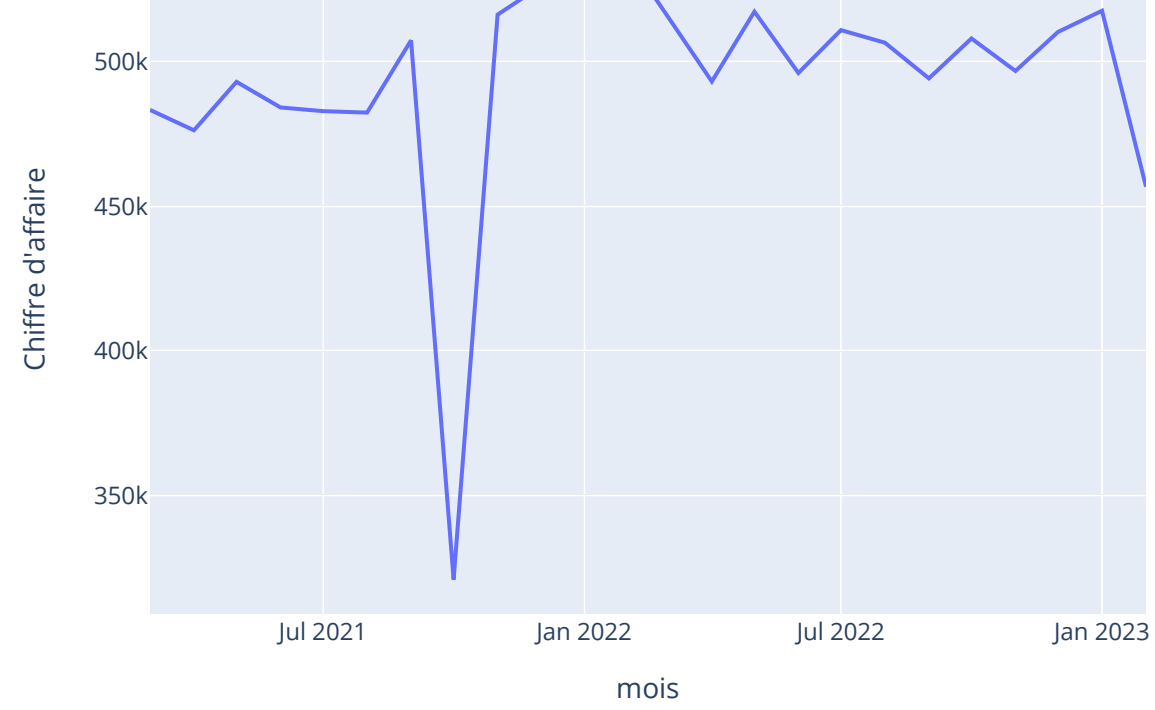
In [158…
```
# CA par jour/mois/trimestre
```

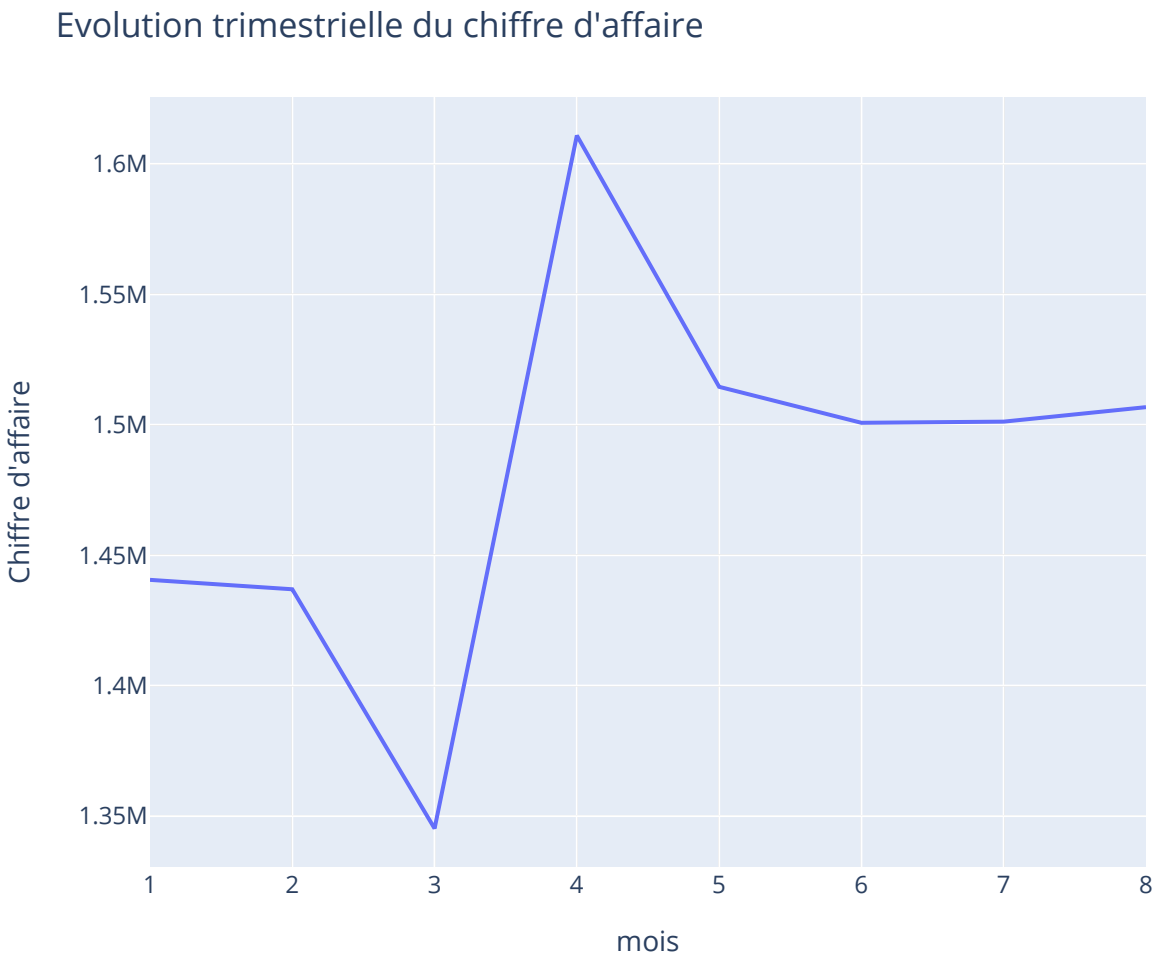
```
data.mois = pd.to_datetime(data.mois)
ca_trimestre = grby_sum(data, 'trimestre', 'price')
ca_mois = grby_sum(data, 'mois', 'price')
ca_jours = grby_sum(data, 'jour', 'price')
```

In [159...

```
fig1 = px.line(ca_mois, x="mois", y="price", title = "Evolution mensuelle du chiffre d'a
fig2 = px.line(ca_jours, x="jour", y="price")
fig3 = px.line(ca_trimestre, x="trimestre", y="price")
fig1.update_layout(
    xaxis_title = 'mois',
    yaxis_title = "Chiffre d'affaire")
fig2.update_layout(
    title = "Evolution journalière du chiffre d'affaire",
    xaxis_title = 'mois',
    yaxis_title = "Chiffre d'affaire")
fig3.update_layout(
    title = "Evolution trimestrielle du chiffre d'affaire",
    xaxis_title = 'mois',
    yaxis_title = "Chiffre d'affaire")

fig2.show()
fig1.show()
fig3.show()
```







On constate une baisse signficative du chiffre d'affaires au mois d'octobre 2021.

Tendance du chiffre d'affaires (Méthode de la moyenne mobile) :

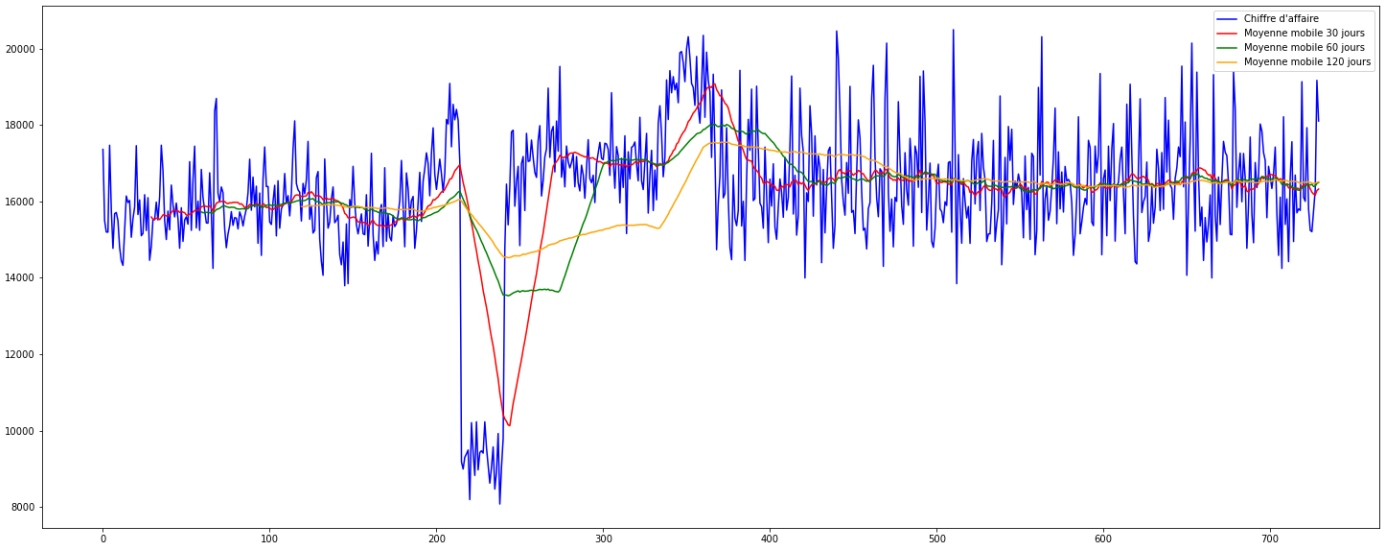
In [57]:
```
# CA par jour
ca_jours.head()
```

Out[57]:

	jour	price
0	2021-03-01	17363.22
1	2021-03-02	15497.09
2	2021-03-03	15198.69
3	2021-03-04	15196.07
4	2021-03-05	17471.37

In [161…

```
# Calcul et représentation des moyennes mobiles :
plt.figure(figsize = (25,10))
# Moyennes mobiles sur 30/60/120 jours
rolling_mean1 = ca_jours.rolling(window = 30).mean()
rolling_mean2 = ca_jours.rolling(window = 60).mean()
rolling_mean3 = ca_jours.rolling(window = 120).mean()
# Représentation graphique des moyennes mobiles
plt.plot(ca_jours.price, color = 'blue', label = "Chiffre d'affaire")
plt.plot(rolling_mean1.price, color = 'red', label = "Moyenne mobile 30 jours")
plt.plot(rolling_mean2.price, color = 'green', label = "Moyenne mobile 60 jours")
plt.plot(rolling_mean3.price, color = 'orange', label = "Moyenne mobile 120 jours")
plt.legend(loc = 'best')
plt.show()
```



On constate que l'évolution du chiffre d'affaire est stationnaire. Nous allons utiliser un test de Dickey-Fuller augmenté (ADF) pour confirmer cette hypothèse.

On définie les hypothèses: </br> H0 : La série est non stationnaire si p-value > 5% </br> H1 : La série est stationnaire si p-value < 5%

In [163…

```
#Test de Dickey-Fuller augmenté (ADF)
from statsmodels.tsa.stattools import adfuller
result = adfuller(ca_jours.price)

print('Statistiques ADF : ',format(result[0]))
print('p-value : ',format(result[1]))
print('Valeurs Critiques :')
for key, value in result[4].items():
    print('\t{}: {}'.format(key, value))
```

```
Statistiques ADF :  -4.778714886185212
p-value :  5.9721006639759034e-05
Valeurs Critiques :
```

```
        1%: -3.4394018678958544
        5%: -2.865534780932388
        10%: -2.5688972751486325
```

La P-value est inférieure au seuil de 0.05 et la statistique ADF est inférieur aux valeurs critiques. On rejète donc l'hypothèse nulle H0.</br> ~H0 : La série est non stationnaire si p-value > 5%~ </br> H1 : La série est stationnaire si p-value < 5%

La série qui représente l'évolution du chiffre d'affaire dans le temps est bien stationnaire </br>

2.2.3. Analyse de la saisonnalité :

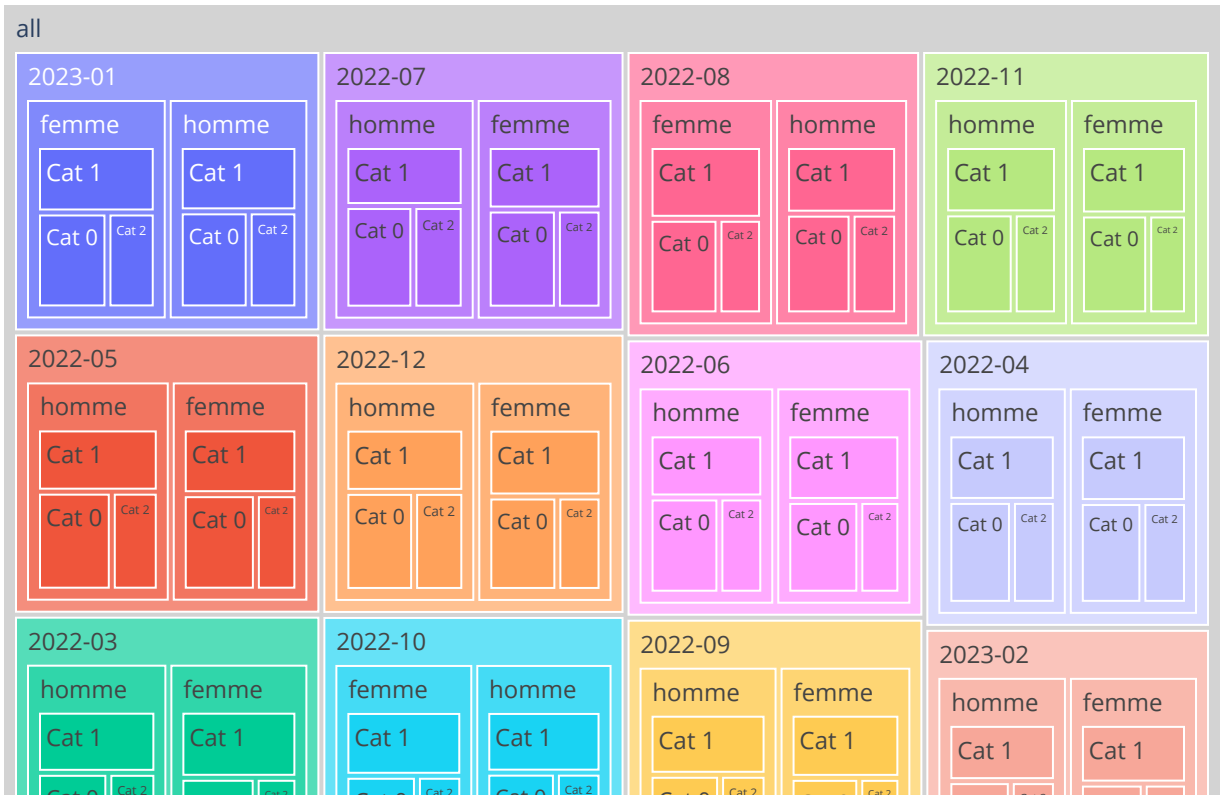
Saisonnalité annuelle :

In [165…
```
# Chiffre d'affaire mensuel par sexes et catégories pour une année entière (Année 2)
data_saison = grby_sum(data2, ['mois', 'sex', 'categ'],'price')
data_saison.head()
```

Out[165]:

	mois	sex	categ	price
0	2022-03	femme	Cat 0	96080.67
1	2022-03	femme	Cat 1	106351.99
2	2022-03	femme	Cat 2	53218.39
3	2022-03	homme	Cat 0	95501.31
4	2022-03	homme	Cat 1	100133.27

In [166…
```
fig = px.treemap(data_saison, path=[px.Constant("all"), 'mois','sex', 'categ'], values='
fig.update_traces(root_color="lightgrey")
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```



Sur une année entière, on remarque une similitude des surfaces représentants les chiffres d'affaires mensuels en fonction du sex et des catégories. On peut conclure qu'il n'y a pas de saisonnalité.

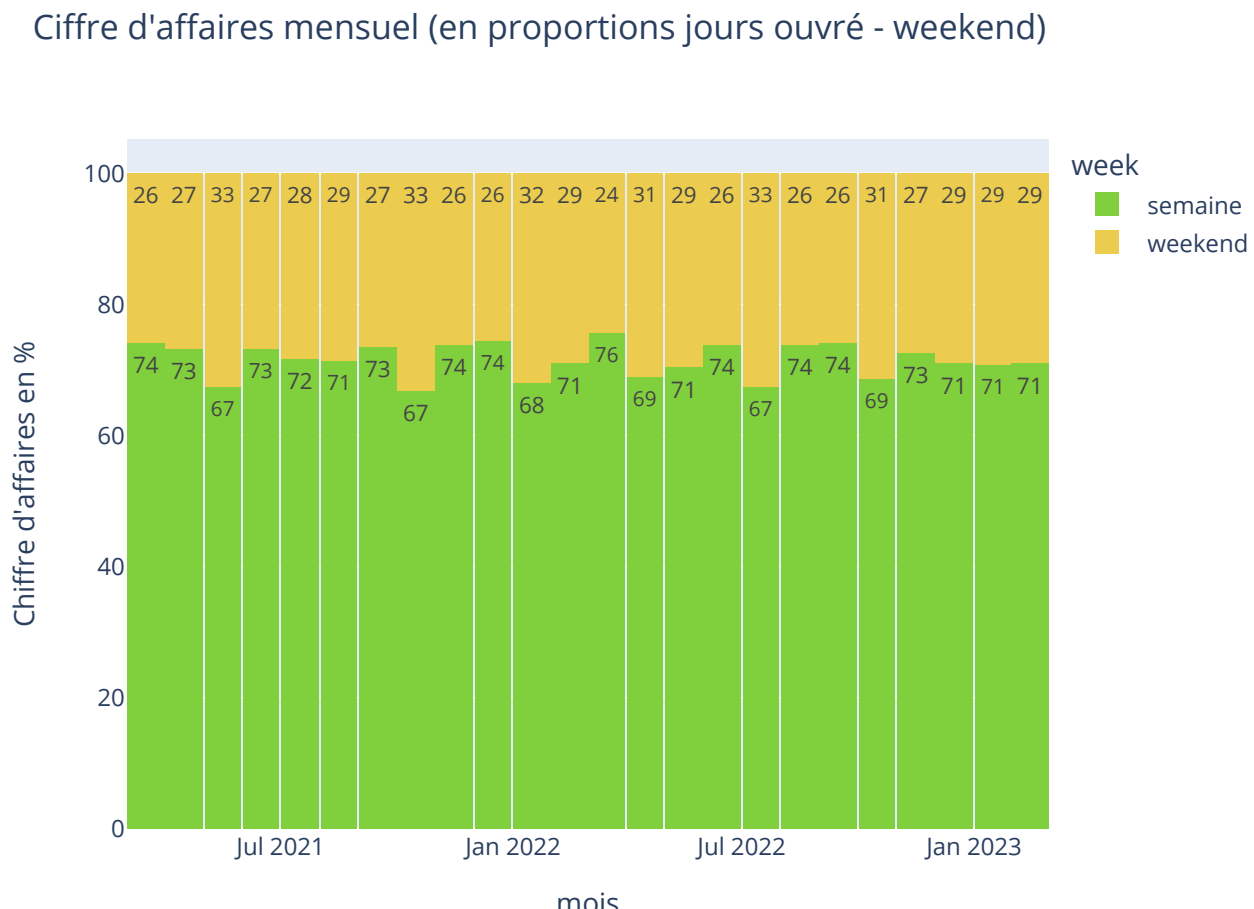
Saisonnalité weekend / jours ouvrés :

In [62]:
```
D =grby_sum(data, ['jour', 'week'], 'price')
D.head()
```

Out[62]:

	jour	week	price
0	2021-03-01	semaine	17363.22
1	2021-03-02	semaine	15497.09
2	2021-03-03	semaine	15198.69
3	2021-03-04	semaine	15196.07
4	2021-03-05	semaine	17471.37

In [146…
```
fig = px.histogram(D, x="jour", y = 'price', color="week",text_auto= '.2s', barnorm = "p
                   title="Ciffre d'affaires mensuel (en proportions jours ouvré - weeken
                   color_discrete_sequence=['#80D03D', '#EBCC4F'])
fig.update_layout(
    xaxis_title = 'mois',
    yaxis_title = "Chiffre d'affaires en %")
fig.show()
```



mois

On constate qu'il n'y a pas de saisonnalité en terme de vente entre weekend et jours ouvrés.

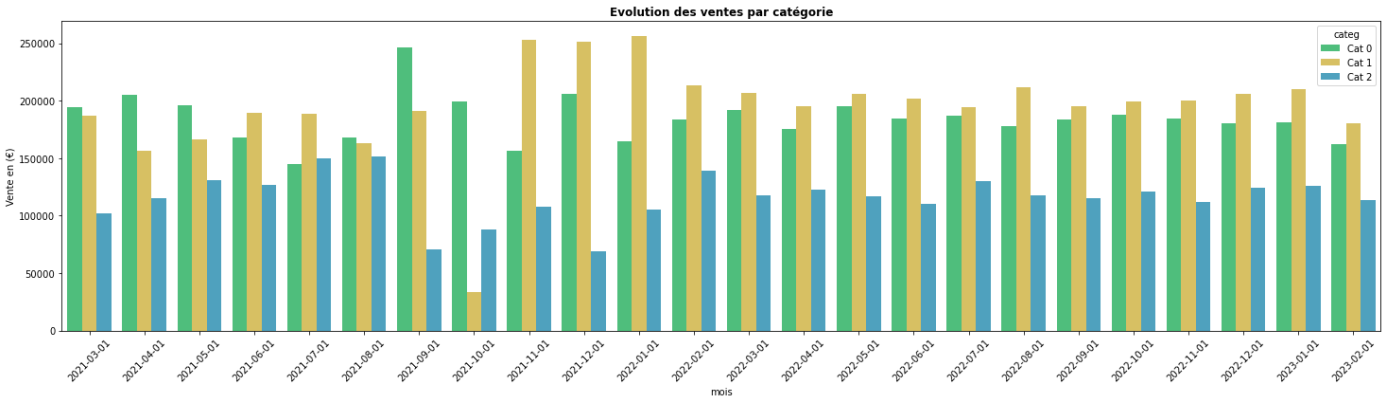
2.3. Analyse des ventes du mois d'octobre 2021 :

In [64]:
```
# Chiffre d'affaires par mois et par catégorie
ventes_mois_categ = grby_sum(data, ['mois', 'categ'], 'price')
ventes_mois_categ['mois'] = ventes_mois_categ['mois'].astype('str')
ventes_mois_categ.head()
```

Out[64]:

	mois	categ	price
0	2021-03-01	Cat 0	194512.29
1	2021-03-01	Cat 1	186974.17
2	2021-03-01	Cat 2	101837.27
3	2021-04-01	Cat 0	205371.42
4	2021-04-01	Cat 1	156138.35

In [168…
```
# Représentation des ventes mensuelles par catégories sur les deux années
plt.figure(figsize = (25,6))
cols = ['#3DD079', '#EBCC4F', '#3DAAD0']
sns.barplot(data = ventes_mois_categ, x = 'mois', y = 'price', hue = 'categ', palette=co
plt.title("Evolution des ventes par catégorie", fontweight ="bold")
plt.xticks(rotation=45)
plt.ylabel("Vente en (€)")
plt.show()
```



On constate une baisse significative des ventes "de catégorie 1" pour le mois d'octobre 2021. Nous allons analyser en détail cette baisse.

In [132…
```
# Vente du mois d'octobre
data_oct2021 = data[data.mois == '2021-10']
ventes_oct2021_categ = grby_sum(data_oct2021, ['jour', 'categ'], 'price')
ventes_oct2021_categ.head()
```

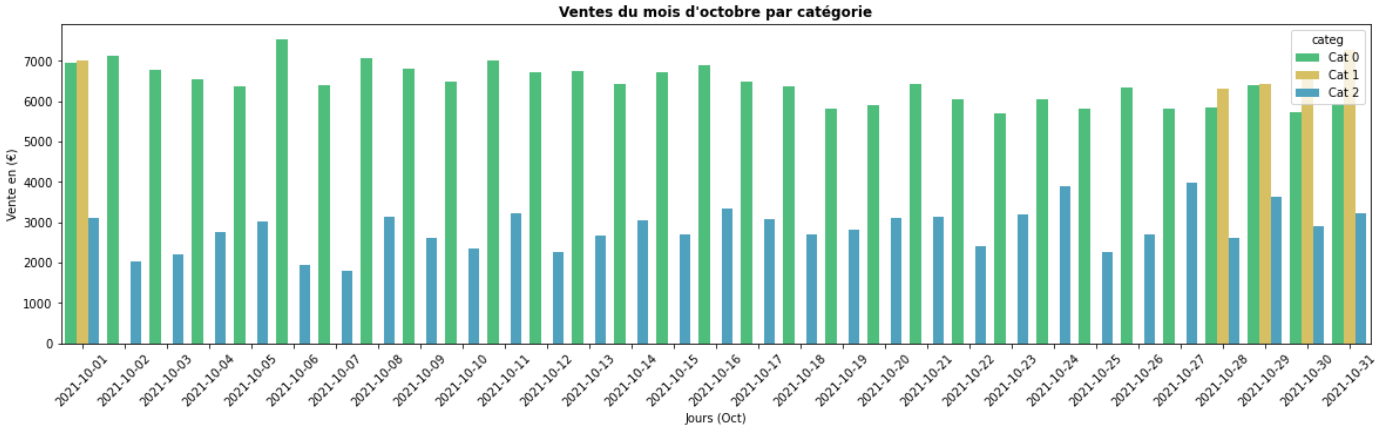
Out[132]:

	jour	categ	price
0	2021-10-01	Cat 0	6950.50
1	2021-10-01	Cat 1	7003.79
2	2021-10-01	Cat 2	3104.05

3	2021-10-02	Cat 0	7141.01
4	2021-10-02	Cat 2	2041.12

In [133…
```
ventes_oct2021_categ['jour'] = ventes_oct2021_categ['jour'].astype('str')
```

In [169…
```
# Représentation des ventes par catégories du mois d'octobre
plt.figure(figsize = (20,5))
sns.barplot(data = ventes_oct2021_categ, x = 'jour', y = 'price', hue ='categ', palette=
plt.title("Ventes du mois d'octobre par catégorie", fontweight ="bold")
plt.xticks(rotation=45)
plt.xlabel("Jours (Oct)")
plt.ylabel("Vente en (€)")
plt.show()
```



On constate une abscence des ventes de la "catégorie 1" du 02 au 27 octobre 2021. On peut supposer qu'il y a eu une rupture de stocks ou un problème d'approvisionnement ...etc

2.4. Chiffre d'affaires et références produits :

In [135…
```
# Les 10 premières références en terme de CA
grby_sum(data,'id_prod', 'price').nlargest(10, "price")
```

Out[135]:

	id_prod	price
3097	2_159	94893.50
3071	2_135	69334.95
3046	2_112	65407.76
3035	2_102	60736.78
3153	2_209	56971.86
2620	1_395	54356.25
2592	1_369	54025.48
3044	2_110	53846.25
3202	2_39	53060.85
3105	2_166	52449.12

In [136…
```
# Les 10 dérnières références en terme de CA
grby_sum(data,'id_prod', 'price').nsmallest(10, "price")
```

Out[136]:

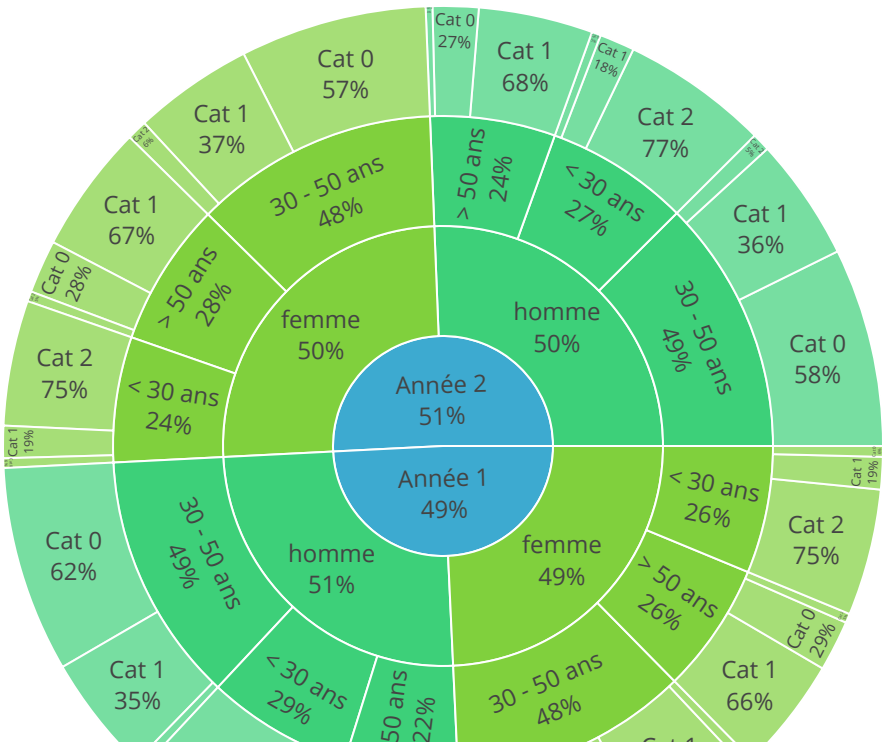
	id_prod	price

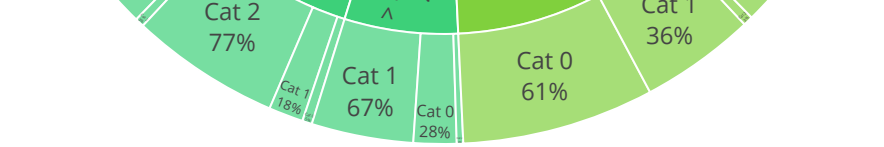
595	0 1539	0.99
313	0 1284	1.38
719	0 1653	1.98
665	0 1601	1.99
1785	0 541	1.99
2080	0 807	1.99
802	0 1728	2.27
549	0 1498	2.48
2180	0 898	2.54
925	0 1840	2.56

2.5.Chiffre d'affaires et l'ensemble des variables :

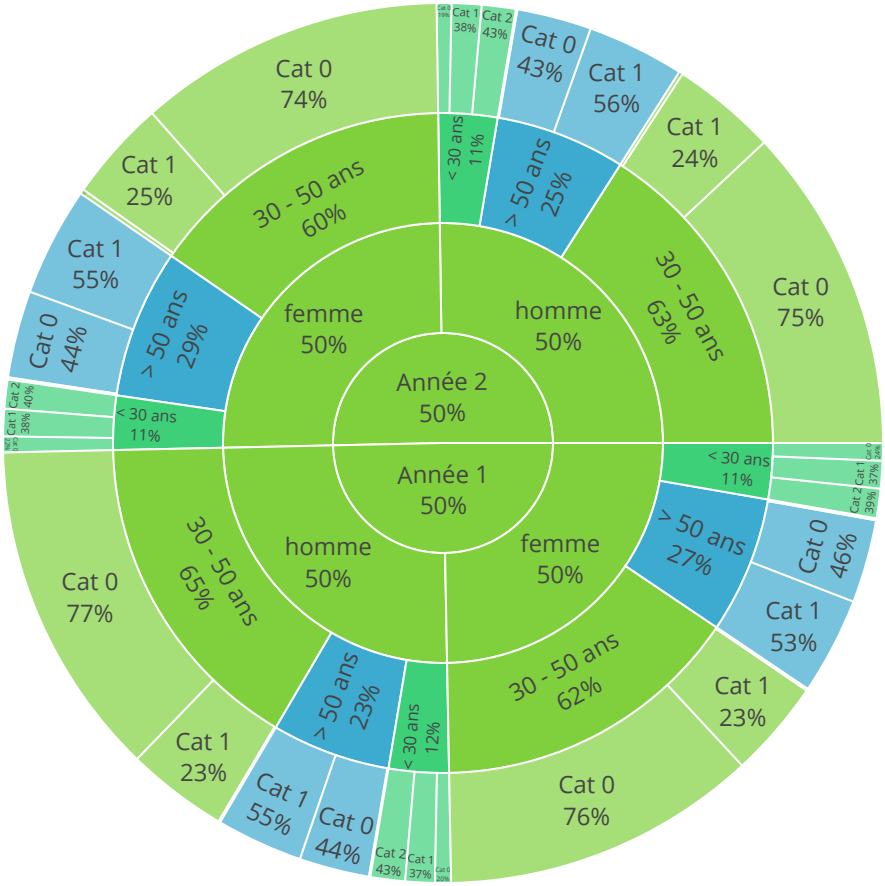
In [137...
```
df1 = grby_count(data, ['annee_exo','cat_age','sex', 'categ'], 'client_id')
df2 = grby_sum(data, ['annee_exo','cat_age','sex', 'categ'], 'price')
```

In [140...
```
fig2 = px.sunburst(df2, path=['annee_exo', 'sex', 'cat_age', 'categ'], values='price', c
                   width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07

fig2.update_traces(textinfo = 'label+percent parent')
fig2.update_layout(title_text="Chiffre d'affaires", title_x = 0.5, font_size=12)
fig2.show()

fig1 = px.sunburst(df1, path=['annee_exo','sex', 'cat_age', 'categ'], values='client_id'
                   width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07
fig1.update_traces(textinfo = 'label+percent parent')
fig1.update_layout(title_text="Volume de vente", title_x = 0.5, font_size=12)
fig1.show()
```





Volume de vente



2.5.1. Chiffre d'affaires et catégories produits :

Répartition du chiffre d'affaires par catégorie :

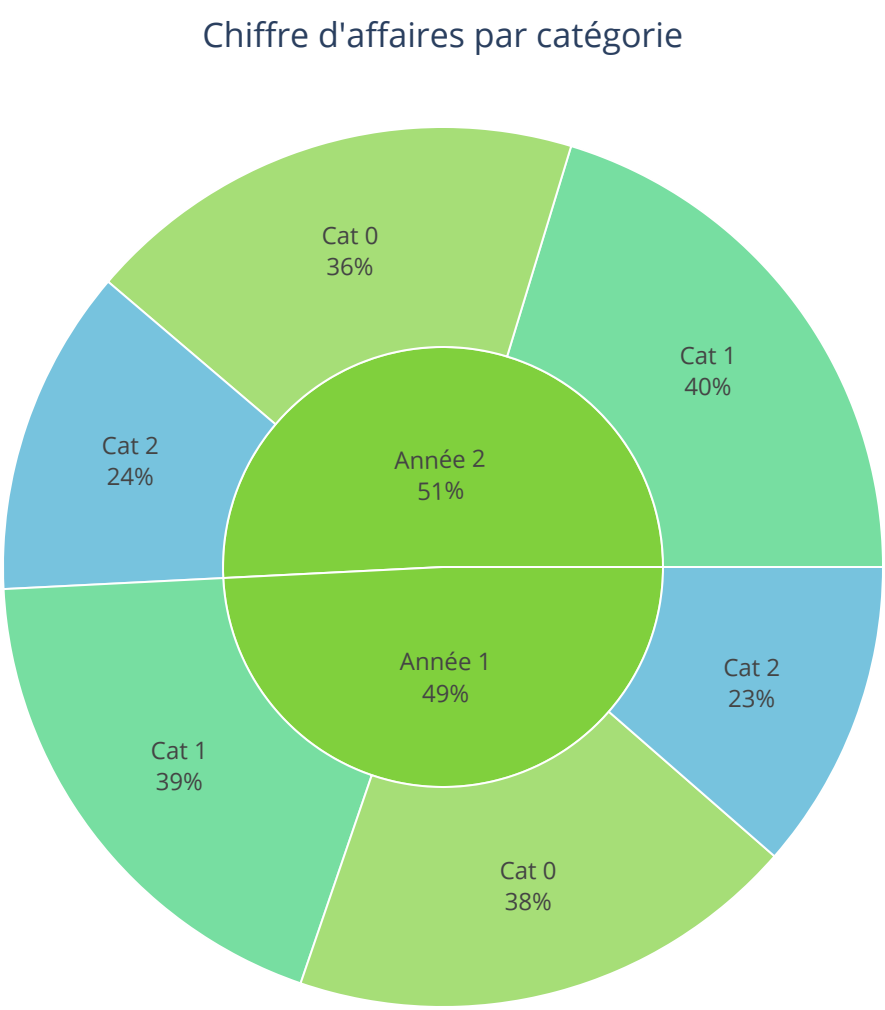
In [73]:
```
ca_categ = grby_sum(data, ['categ', 'annee_exo'], 'price')
ca_categ
```

Out[73]:

	categ	annee_exo	price
0	Cat 0	Année 1	2232311.57
1	Cat 0	Année 2	2190558.20
2	Cat 1	Année 1	2247384.41

3	Cat 1	Année 2	2406338.28
4	Cat 2	Année 1	1353924.35
5	Cat 2	Année 2	1426350.67

In [74]:
```
fig = px.sunburst(ca_categ, path=['annee_exo', 'categ'], values='price', color ='categ',
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Chiffre d'affaires par catégorie", title_x = 0.5, font_siz
fig.show()
```



Volume de ventes par catégorie :

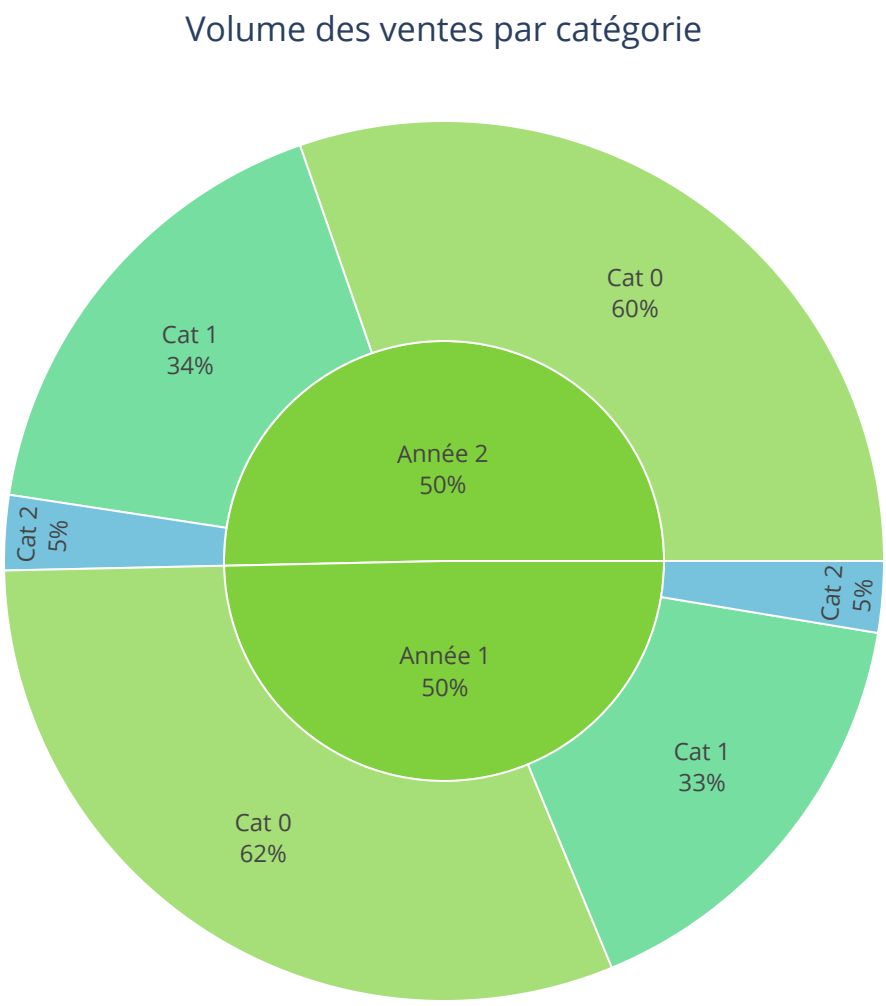
In [75]:
```
volume_categ = grby_count(data, ['categ', 'annee_exo'], 'client_id')
volume_categ
```

Out[75]:

	categ	annee_exo	client_id
0	Cat 0	Année 1	209839
1	Cat 0	Année 2	205915
2	Cat 1	Année 1	109735

3	Cat 1	Année 2	117434
4	Cat 2	Année 1	17788
5	Cat 2	Année 2	18695

In [76]:
```
fig = px.sunburst(volume_categ, path=['annee_exo', 'categ'], values='client_id', color =
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Volume des ventes par catégorie", title_x = 0.5, font_size
fig.show()
```



2.5.2. Chiffre d'affaires et catégories d'age :

Répartition du chiffre d'affaires par catégorie d'age :

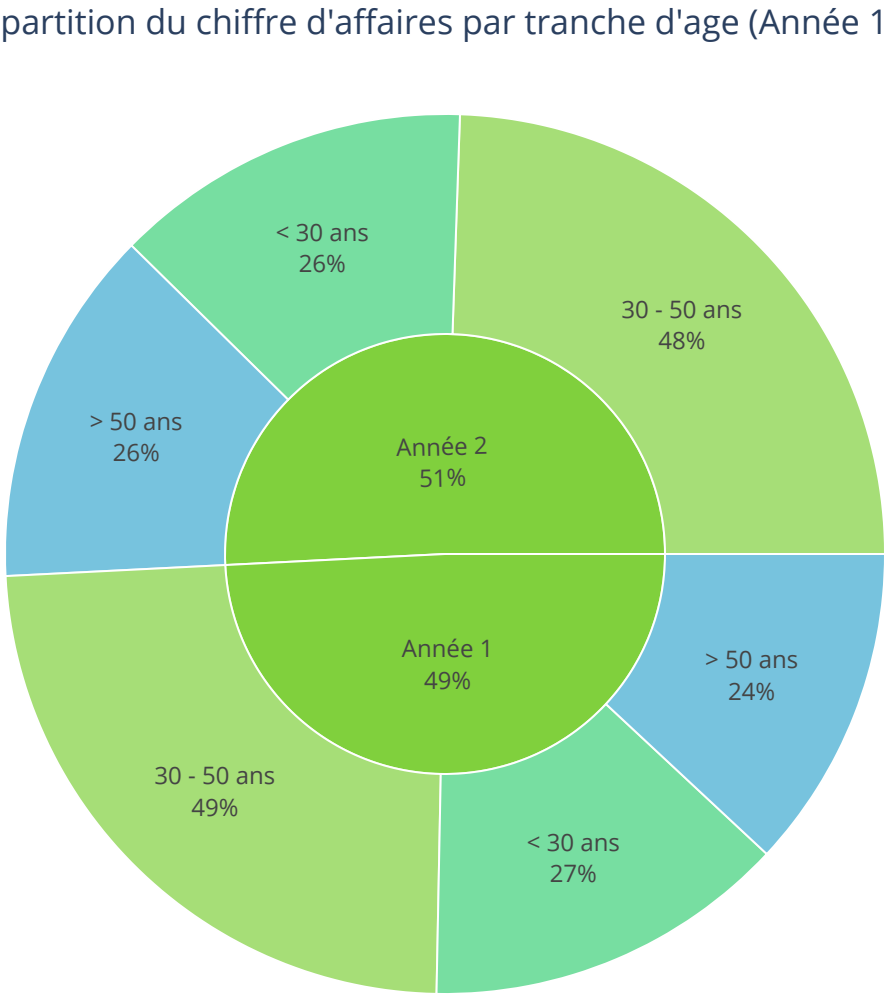
In [77]:
```
ca_cat_age = grby_sum(data, ['cat_age', 'annee_exo'], 'price')
ca_cat_age
```

Out[77]:

	cat_age	annee_exo	price
0	30 - 50 ans	Année 1	2832989.80
1	30 - 50 ans	Année 2	2899885.70

2	< 30 ans	Année 1	1584533.62
3	< 30 ans	Année 2	1561800.94
4	> 50 ans	Année 1	1416096.91
5	> 50 ans	Année 2	1561560.51

In [78]:
```
fig = px.sunburst(ca_cat_age, path=['annee_exo', 'cat_age'], values='price', color ='cat
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Répartition du chiffre d'affaires par tranche d'age (Année
fig.show()
```



Répartition par catégorie d'age

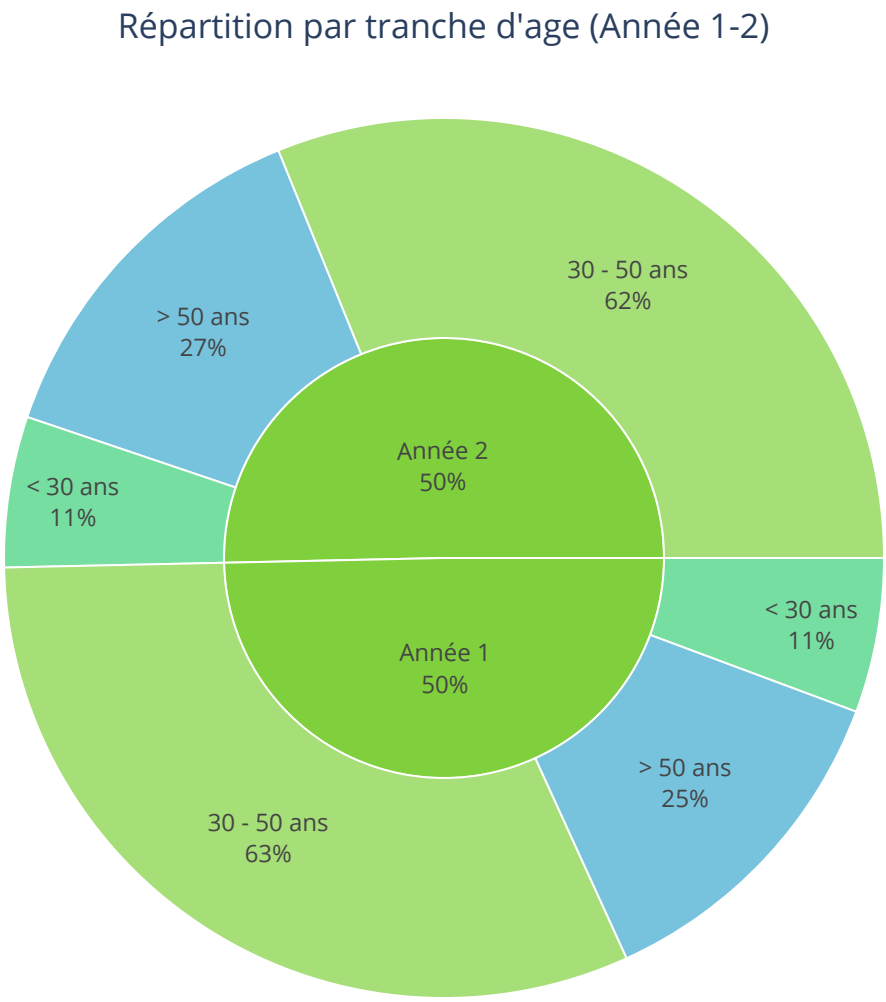
In [79]:
```
age_repartition = grby_count(data, ['cat_age', 'annee_exo'], 'client_id')
age_repartition
```

Out[79]:

	cat age	annee exo	client id
0	30 - 50 ans	Année 1	213801
1	30 - 50 ans	Année 2	211604
2	< 30 ans	Année 1	38560

3	< 30 ans	Année 2	37618
4	> 50 ans	Année 1	85001
5	> 50 ans	Année 2	92822

In [80]:
```
fig = px.sunburst(age_repartition, path=['annee_exo', 'cat_age'], values='client_id', co
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD079
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Répartition par tranche d'age (Année 1-2)", title_x = 0.5,
fig.show()
```



2.5.3. Chiffre d'affaires et genre :

Répartition du chiffre d'affaires par genre :

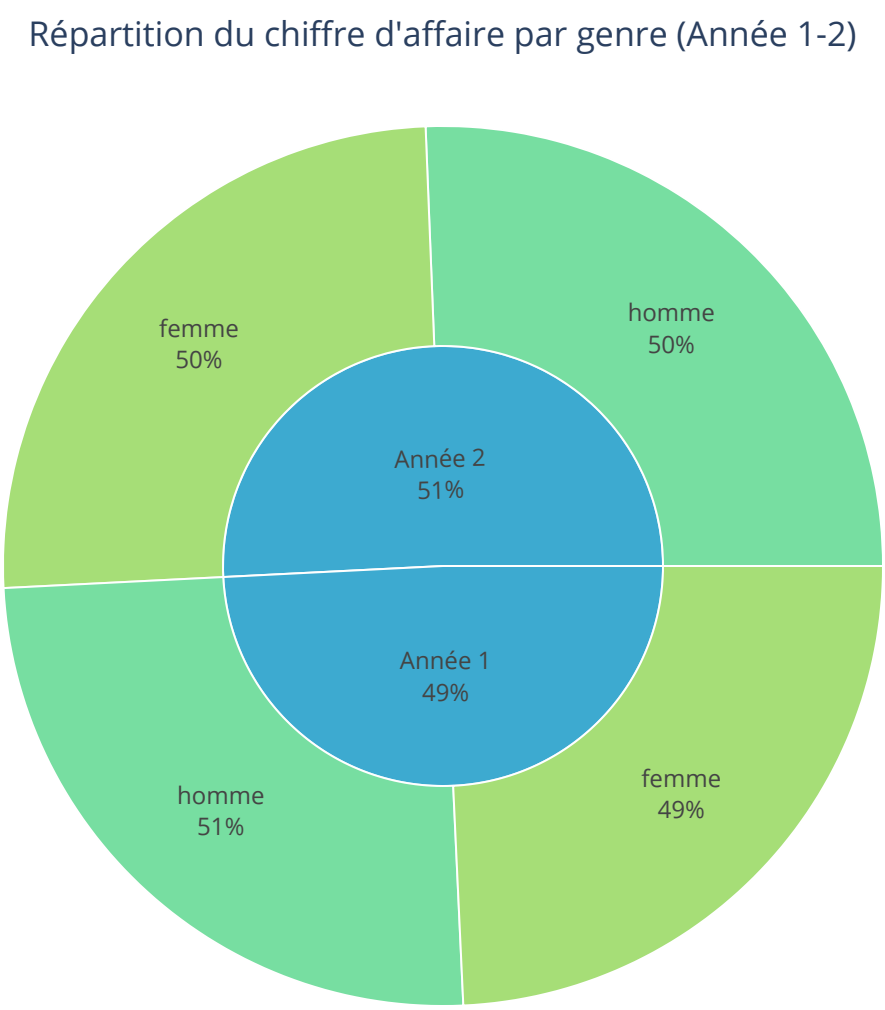
In [81]:
```
ca_sex = grby_sum(data, ['sex', 'annee_exo'], 'price')
ca_sex
```

Out[81]:

	sex	annee_exo	price
0	femme	Année 1	2876988.59
1	femme	Année 2	2984333.10

2	homme	Année 1	2956631.74
3	homme	Année 2	3038914.05

In [82]:
```
fig = px.sunburst(ca_sex, path=['annee_exo', 'sex'], values='price', color ='sex',
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Répartition du chiffre d'affaire par genre (Année 1-2)", t
fig.show()
```



Répartition des clients par genre :

In [83]:
```
sex_repartition = grby_count(data, ['sex', 'annee_exo'], 'client_id')
sex_repartition
```

Out[83]:

	sex	annee_exo	client_id
0	femme	Année 1	167838
1	femme	Année 2	170601
2	homme	Année 1	169524
3	homme	Année 2	171443

In [84]:
```
fig = px.sunburst(sex_repartition, path=['annee_exo', 'sex'], values='client_id', color
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD07
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Répartition par genre (Année 1-2)", title_x = 0.5, font_si
fig.show()
```

2.6. Profils de clients:

Typologie clients:

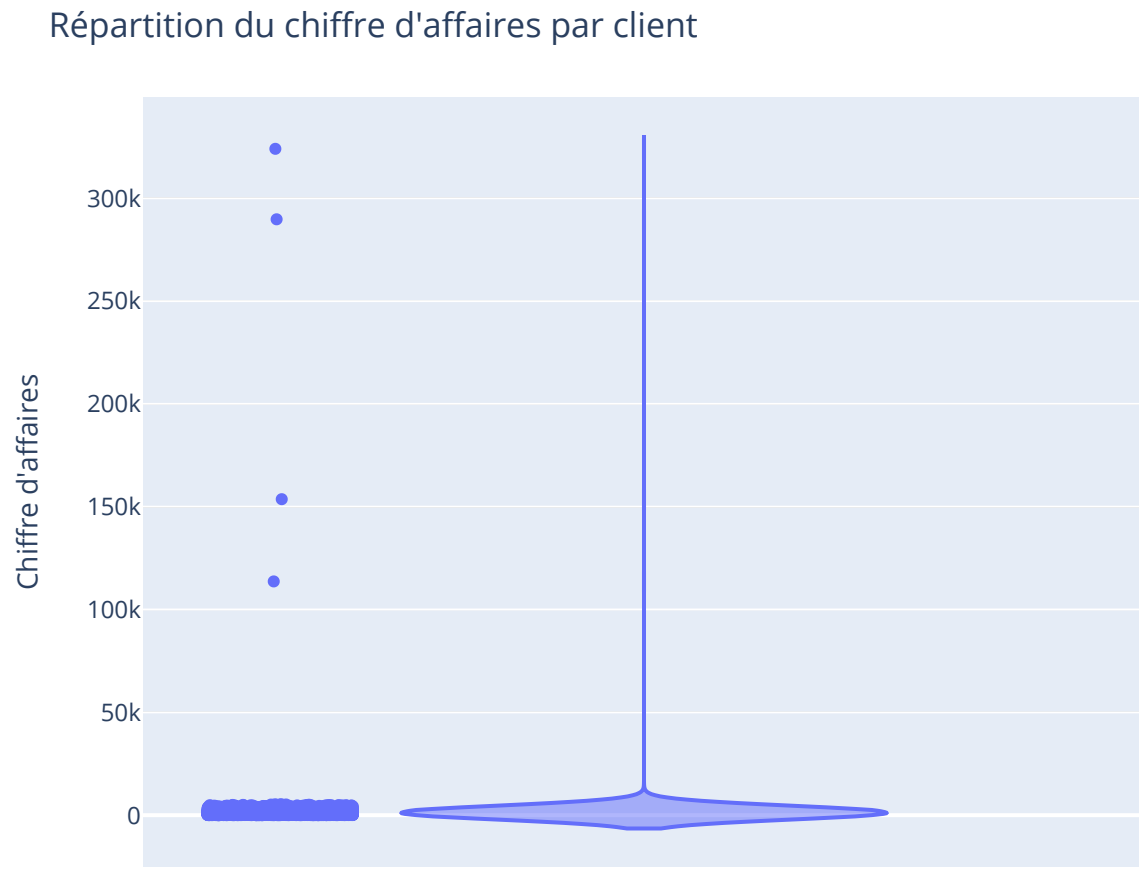
In [85]:
```
# CA par client
ca_client = grby_sum(data,'client_id', 'price')
ca_client.nlargest(10, 'price')
```

Out[85]:

	client_id	price
677	c_1609	324033.35
4388	c_4958	289760.34
6337	c_6714	153662.76
2724	c_3454	113669.85
2513	c_3263	5276.87

634	c_1570	5271.62
2108	c_2899	5214.05
1268	c_2140	5208.82
7006	c_7319	5155.77
7791	c_8026	5093.22

In [86]:
```
fig = px.violin(ca_client, y="price", points="all")
fig.update_layout(
    title = "Répartition du chiffre d'affaires par client",
    #xaxis_title = 'date',
    yaxis_title = "Chiffre d'affaires",
    #showlegend = True
)
fig.show()
```



Quatre clients identifiés (c_1609, c_4958, c_6714, c_3454) se distinguent des autres clients par leur grande contribution au chiffre d'affaires généré. On peut supposer qu'ils corespondent à des revendeurs.

Etudions de plus près ces clients qu'on va nommer par la suite "Clients BtoB" et les autres clients "Clients BtoC".

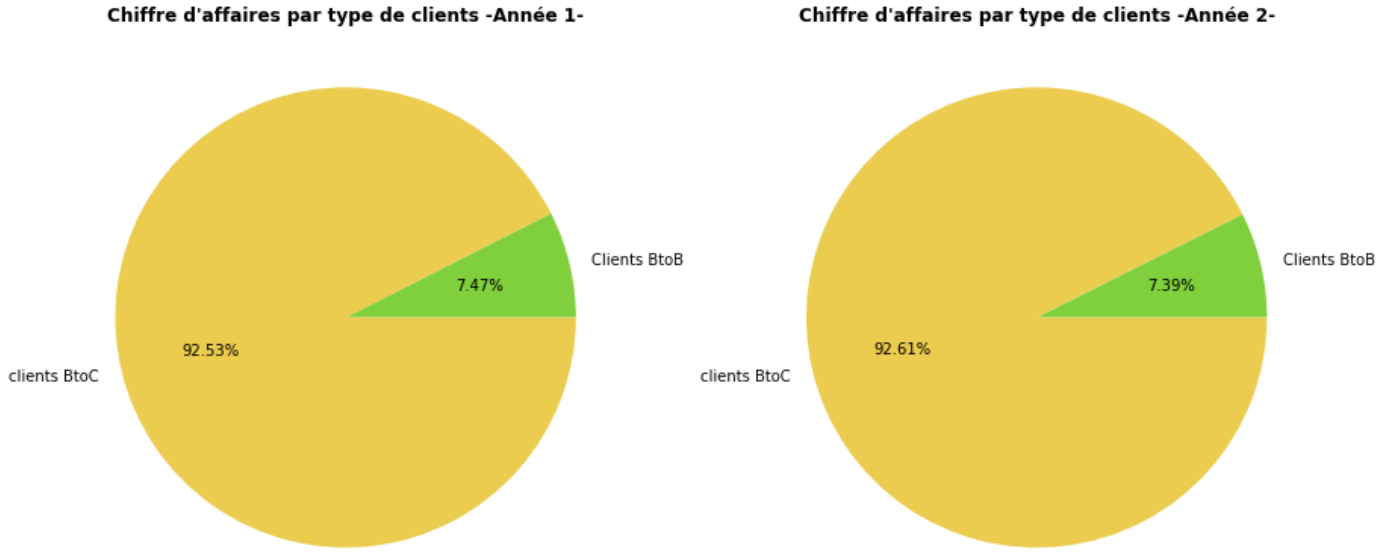
In [87]:
```
client_BtoB = ['c_1609', 'c_4958', 'c_6714', 'c_3454']
```

In [88]:
```
# Données sur les clients BtoB pour chaque année
data1_client_BtoB = data1[data1.client_id.isin(client_BtoB)]
```

```
data2_client_BtoB = data2[data2.client_id.isin(client_BtoB)]
```

In [89]:
```
# CA des clients BtoB
ca_annee1_BtoB = data1_client_BtoB.price.sum()
ca_annee2_BtoB = data2_client_BtoB.price.sum()
```

In [130…
```
plt.figure(figsize = (15,10))

colors = sns.color_palette(['#80D03D', '#EBCC4F'])
labels = ['Clients BtoB', 'clients BtoC']

plt.subplot(1,2,1)
plt.pie((ca_annee1_BtoB, ca_total_annee1 - ca_annee1_BtoB), labels = labels, colors = co
plt.title("Chiffre d'affaires par type de clients -Année 1-", fontweight ="bold")

plt.subplot(1,2,2)
plt.pie((ca_annee2_BtoB, ca_total_annee2 - ca_annee2_BtoB), labels = labels, colors = co
plt.title("Chiffre d'affaires par type de clients -Année 2-", fontweight ="bold")

plt.show()
```



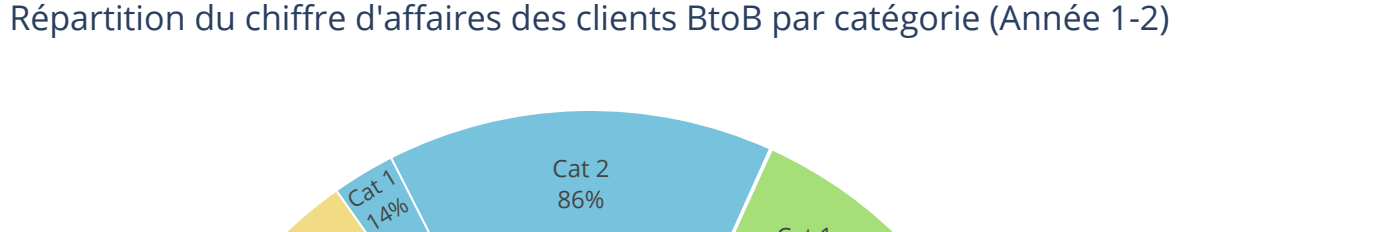
Les clients BtoB contribuent à peu près à 7% du chiffre d'affaires total.

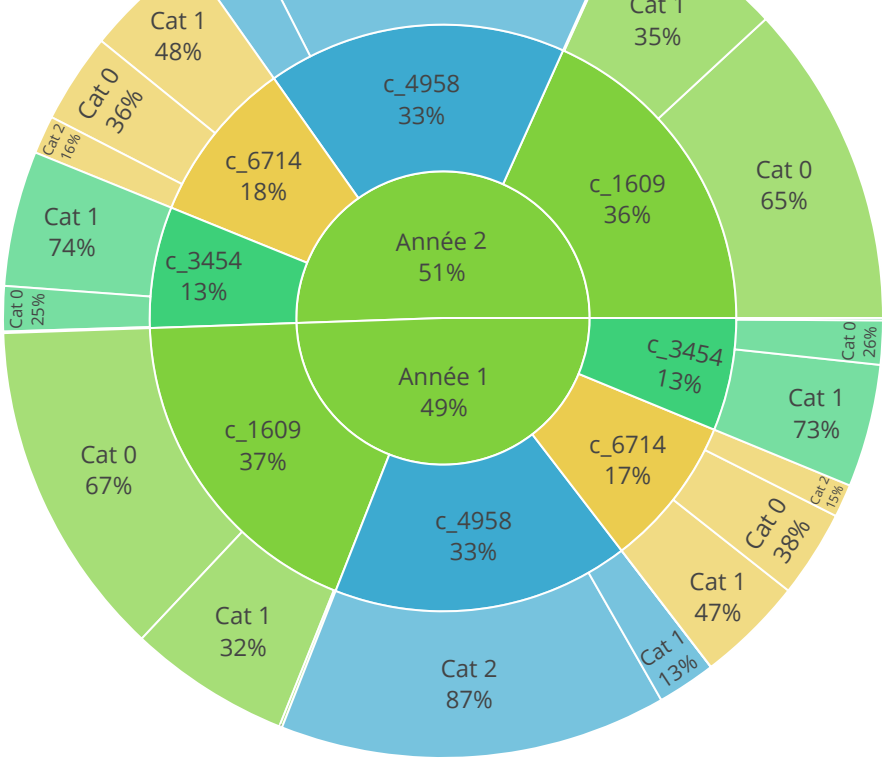
Répartition du chiffre d'affaires par type de clients :

In [91]:
```
data_client_BtoB = data[data.client_id.isin(client_BtoB)]
```

In [92]:
```
ff = grby_sum(data_client_BtoB, ['client_id', 'annee_exo', 'categ'], 'price')
```

In [93]:
```
fig = px.sunburst(ff, path=['annee_exo', 'client_id', 'categ'], values='price', color ='
                  width =600, height = 600, color_discrete_sequence=['#80D03D', '#3DD0
fig.update_traces(textinfo = 'label+percent parent')
fig.update_layout(title_text="Répartition du chiffre d'affaires des clients BtoB par cat
fig.show()
```





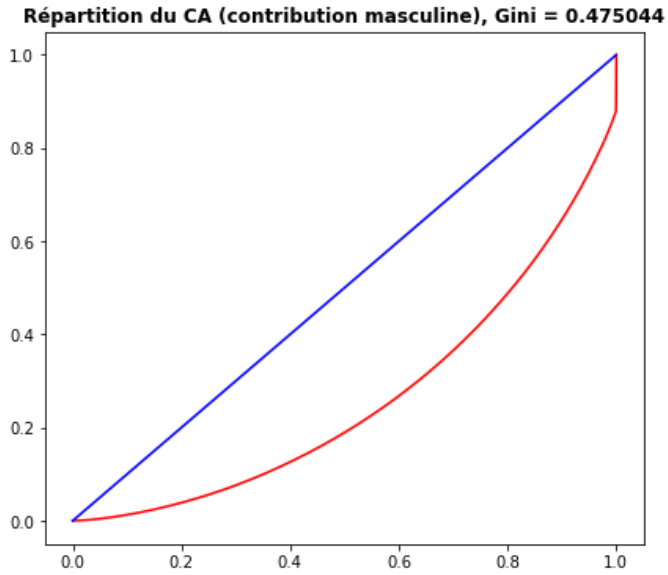
2.7. Contribution du genre dans la création du chiffre d'affaires :

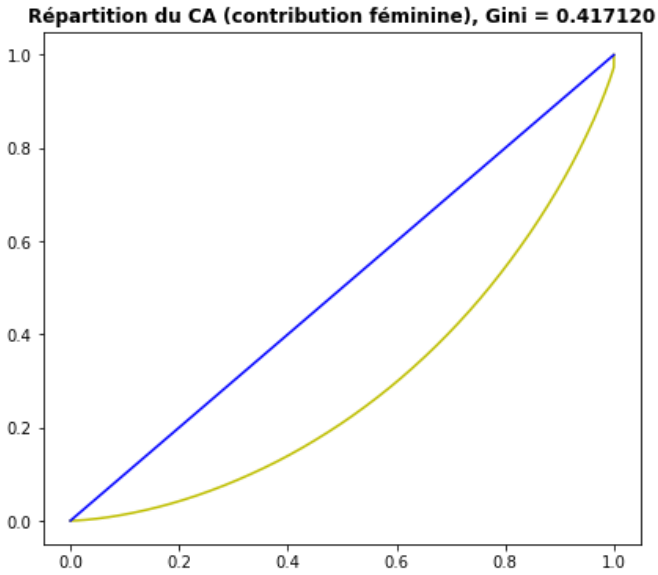
In [94]:
```
ca_client_m = data.query('sex == "homme"').groupby('client_id')[['price']].sum().reset_i
ca_client_f = data.query('sex == "femme"').groupby('client_id')[['price']].sum().reset_i
```

In [95]:
```
# Calcul Lorenz
data_lorenz_m = ca_client_m['price']
data_lorenz_f = ca_client_f['price']
nm = len(data_lorenz_m)
nf = len(data_lorenz_f)
Ym = np.cumsum(np.sort(data_lorenz_m))/ data_lorenz_m.sum()
Yf = np.cumsum(np.sort(data_lorenz_f))/ data_lorenz_f.sum()
Ym = np.append([0],Ym)
Yf = np.append([0],Yf)
Xm = np.linspace(0-1/nm,1+1/nm,nm+1)
Xf = np.linspace(0-1/nf,1+1/nf,nf+1)
# Calcul de l'indice de Gini
AUCm = (Ym.sum() -Ym[-1]/2 -Ym[0]/2)/nm
AUCf = (Yf.sum() -Yf[-1]/2 -Yf[0]/2)/nf
Sm = 0.5 - AUCm
Sf = 0.5 - AUCf
gini_m = 2*Sm
gini_f = 2*Sf
```

In [96]:
```
# Courbes de Lorenz
plt.figure(figsize = (15,6))
plt.subplot(1,2,1)
sns.lineplot(Xm, Ym, color = 'r')
sns.lineplot([0,1], [0,1], color = 'b')
plt.title("Répartition du CA (contribution masculine), Gini = %f" % gini_m, fontweight =
plt.subplot(1,2,2)
sns.lineplot(Xf, Yf, color = 'y')
sns.lineplot([0,1], [0,1], color = 'b')
```

```
plt.title("Répartition du CA (contribution féminine), Gini = %f" % gini_f, fontweight ="
plt.show()
```





Les femmes et les hommes contribuent d'une manière moyennement équitable dans la création du chiffre d'affaires total. On peut dire aussi qu'il y a une similitude dans le comportement d'achat des clients hommes et femmes.

Partie 3: Relations entre variables

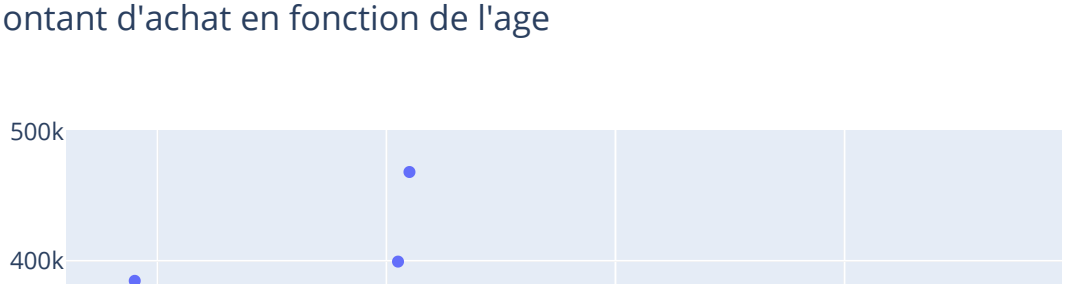
3.1. Age et montant d'achats :

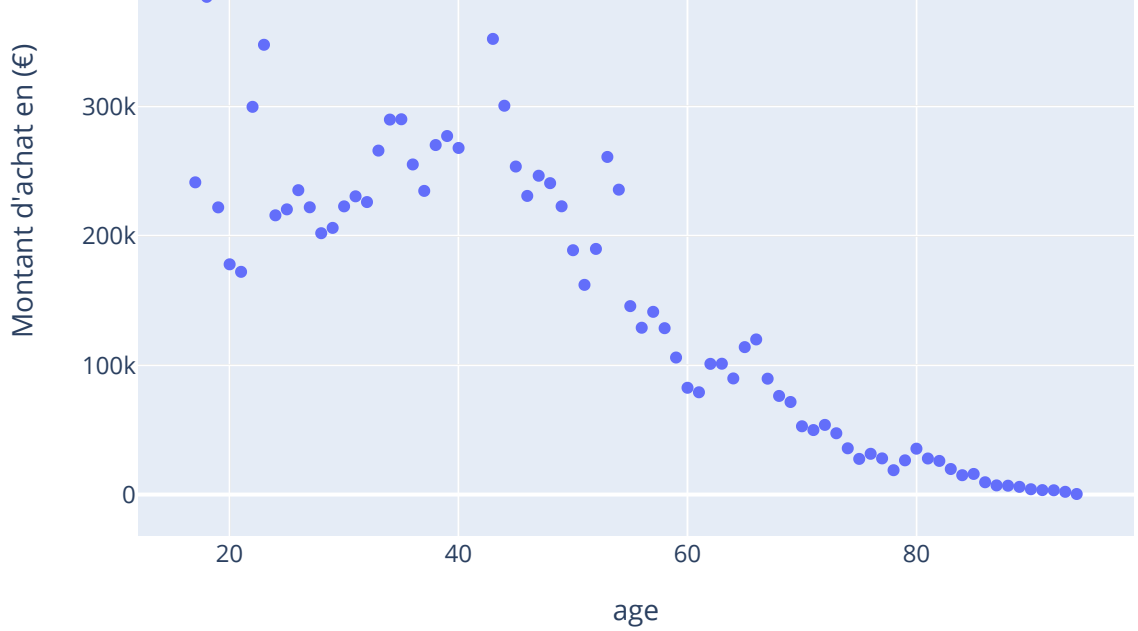
In [97]:
```
age_ca = grby_sum(data,'age', 'price')
age_ca.head()
```

Out[97]:

	age	price
0	17	241265.83
1	18	384592.43
2	19	221910.74
3	20	177913.92
4	21	172102.39

In [98]:
```
fig = px.scatter(age_ca, x="age", y="price", title = "Montant d'achat en fonction de l'a
fig.update_layout(
    xaxis_title = 'age',
    yaxis_title = "Montant d'achat en (€)")
fig.show()
```





Le lien ici entre le chiffre d'affaires et l'age (deux variables quantitatives) est partiellement linéaire. Nous allons utiliser ici la corrélation de Pearson et de Spearman.

Posons les hypothèses de départ :</br> H0 : Variables indépendantes si p-value > 5% </br> H1 : Variables non indépendantes si p-value < 5%

In [99]:
```
pearsonr(age_ca.age, age_ca.price)
```

Out[99]:
```
(-0.856799660216099, 1.459407662056002e-23)
```

In [100…
```
spearmanr(age_ca.age, age_ca.price)
```

Out[100]:
```
SpearmanrResult(correlation=-0.8720267074697454, pvalue=2.730425001740127e-25)
```

Dans les deux cas le résultat de la p-value est inférieur à 0.05, on rejète donc l'hypothèse H0:</br> </br> ~H0 : Variables indépendantes si p-value > 5%~ </br> H1 : Variables non indépendantes si p-value < 5% </br></br> On peut dire que l'age et le montant d'achat des clients sont corrélés

3.2. Age et nombre d'achats

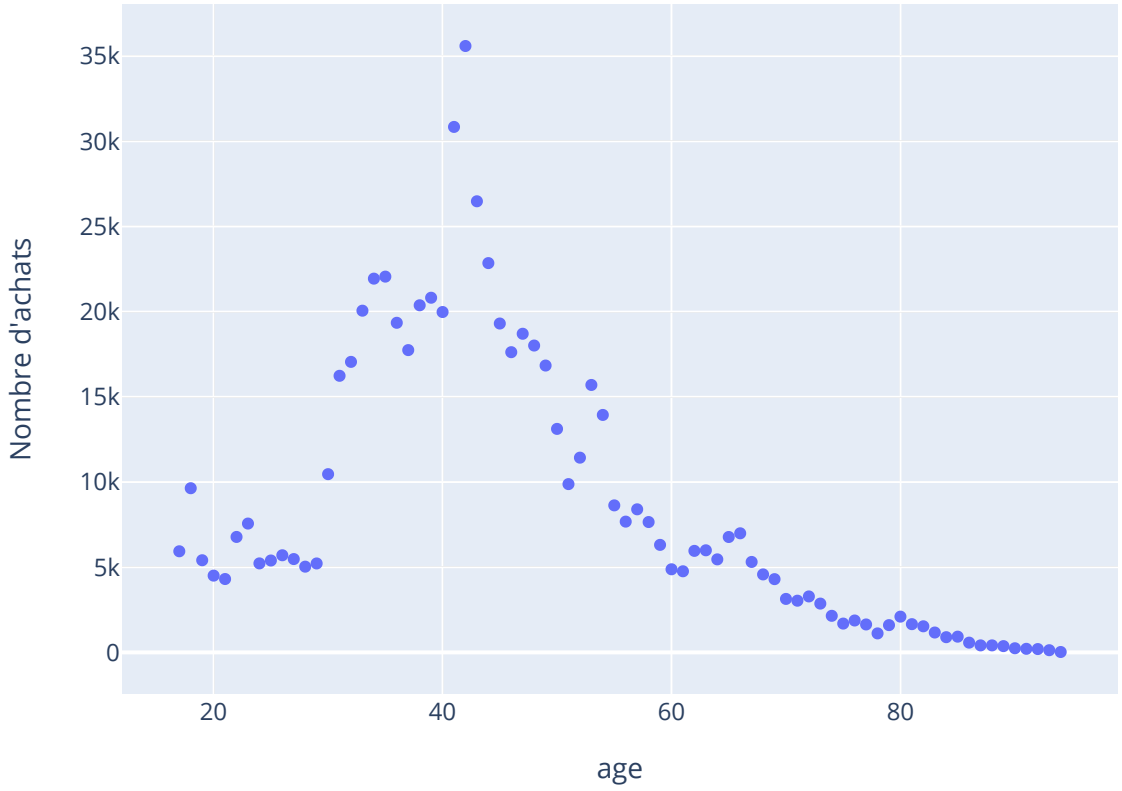
In [101…
```
age_achat = grby_count(data, 'age', 'client_id')
age_achat.head()
```

Out[101]:

	age	client_id
0	17	5935
1	18	9636
2	19	5409
3	20	4506
4	21	4306

In [102…
```
fig = px.scatter(age_achat, x="age", y="client_id", title = "Nombre d'achats en fonction
fig.update_layout(
```

```
    xaxis_title = 'age',
    yaxis_title = "Nombre d'achats")
fig.show()
```



Le lien ici entre le nombre d'achats et l'age (deux variables quantitatives) est partiellement linéaire. Nous allons utiliser ici la corrélation Pearson et de Spearman.

Posons les hypothèses de départ :</br> H0 : Variables indépendantes si p-value > 5% </br> H1 : Variables non indépendantes si p-value < 5%

In [103…
```
pearsonr(age_achat.age, age_achat.client_id)
```

Out[103]:
```
(-0.5555773708371885, 1.2935855004108499e-07)
```

In [104…
```
spearmanr(age_achat.age, age_achat.client_id)
```

Out[104]:
```
SpearmanrResult(correlation=-0.7059143388257313, pvalue=5.275863065515789e-13)
```

Dans les deux cas le résultat de la p-value est inférieur à 0.05, on rejète donc l'hypothèse H0:</br> </br> ~H0 : Variables indépendantes si p-value > 5%~ </br> H1 : Variables non indépendantes si p-value < 5% </br></br> On peut dire que l'age et le montant d'achat des clients sont corrélés

3.3. Age et panier moyen

In [105…
```
age_panier = data.groupby('age').agg({'price': ['sum'], 'client_id':['count']}).reset_in
age_panier['panier_moyen'] = round(age_panier.iloc[:,1]/age_panier.iloc[:,2])
```

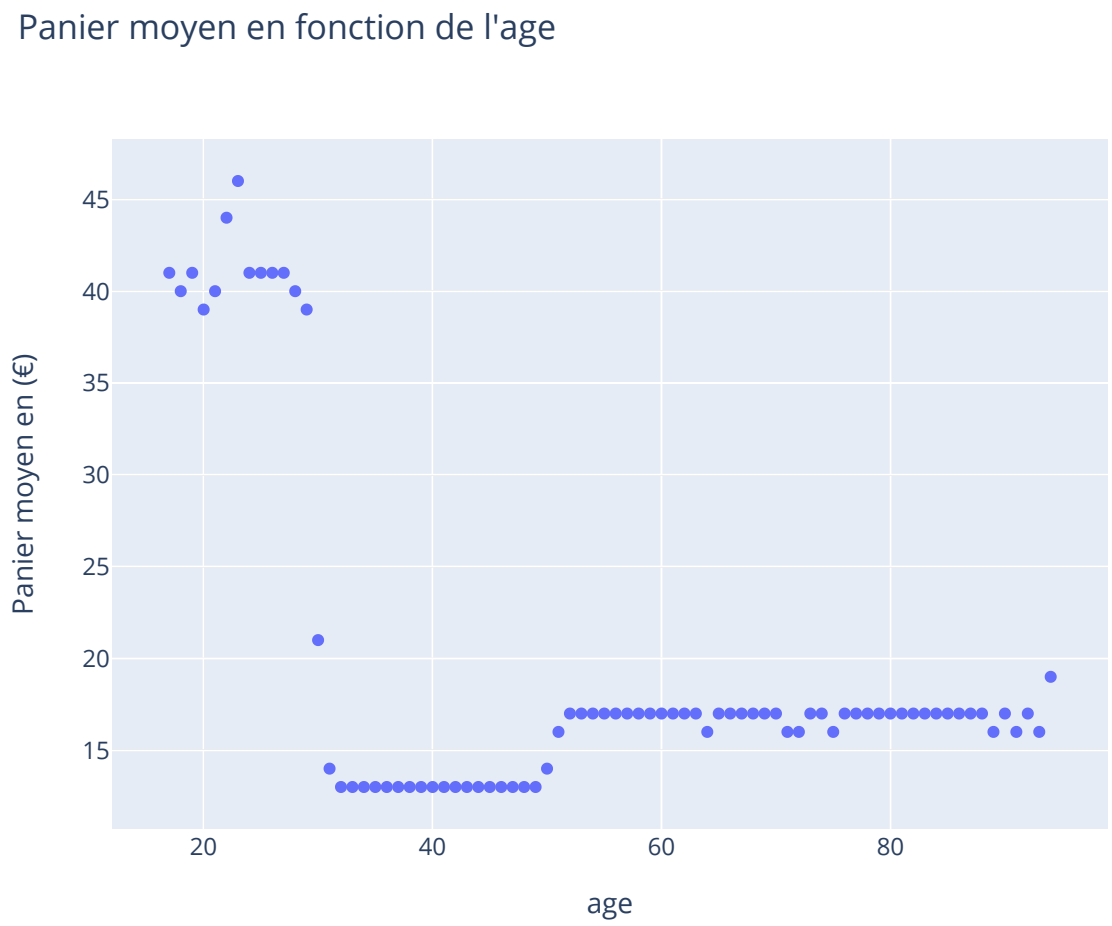
```
age_panier = age_panier[['age', 'panier_moyen']]
age_panier.head()
```

Out[105]:

	age	panier_moyen
0	17	41.0
1	18	40.0
2	19	41.0
3	20	39.0
4	21	40.0

In [106…

```
fig = px.scatter(age_panier, x= 'age', y = 'panier_moyen', title = "Panier moyen en fonc
fig.update_layout(
    xaxis_title = 'age',
    yaxis_title = "Panier moyen en (€)")
fig.show()
```



Il est évident que le panier moyen est en lien avec des tranches d'age bien définies. A savoir, les :</br>

- moins de 30 ans : autour de 40 € </br>
- entre 30 et 50 ans : autour de 13 € </br>
- plus de 50 ans : autour de 17 € </br>

3.4. Lien entre l'age des clients et la catégorie de livres achetés :

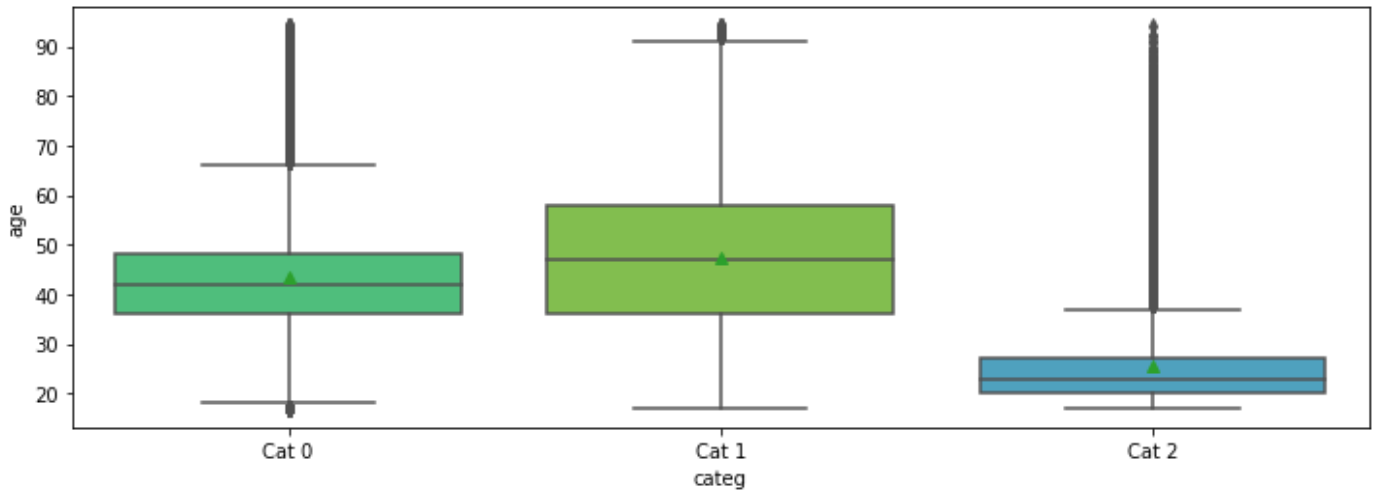
Ici on va utiliser le test d'analyse de variance type ANOVA entre une variable qualitative "catégorie de livres" et quantitative "age des clients".

In [107…
```
age_categ = data[['age', 'categ']]
age_categ.head()
```

Out[107]:

	age	categ
0	36	Cat 0
1	34	Cat 1
2	54	Cat 0
3	21	Cat 2
4	43	Cat 0

In [128…
```
plt.subplots(figsize=(12,4))
ax = sns.boxplot(y = 'age', x = 'categ', data = age_categ, showmeans= True, palette = co
plt.show()
```



Vérification des hypothèse paramétriques :

Test d'homoscédasticité :

In [109…
```
# Calcul des variances
age_categ.groupby("categ")['age'].agg('var')
```

Out[109]:
```
categ
Cat 0    126.094214
Cat 1    240.529843
Cat 2     96.325097
Name: age, dtype: float64
```

Le test de Bartlett permet de tester si les variances sont significativement différentes ou non. Posons les hypothèses de départ :

H0 : Les variances de chaque groupe sont égales si p-value > 5% </Br> H1 : Les variances de chaque groupe ne sont pas toutes égales < 5%

In [110…
```
# test de Bartlett
from scipy.stats import bartlett
bartlett = bartlett(age_categ.age[age_categ.categ == 'Cat 0'],
                    age_categ.age[age_categ.categ == 'Cat 1'],
```

```
                    age_categ.age[age_categ.categ == 'Cat 2'])
print(" La statistique de bartlett est égale à ", bartlett[0])
print(" La p-value de bartlett est égale à ", bartlett[1])
```

```
 La statistique de bartlett est égale à  36527.06257330086
 La p-value de bartlett est égale à  0.0
```

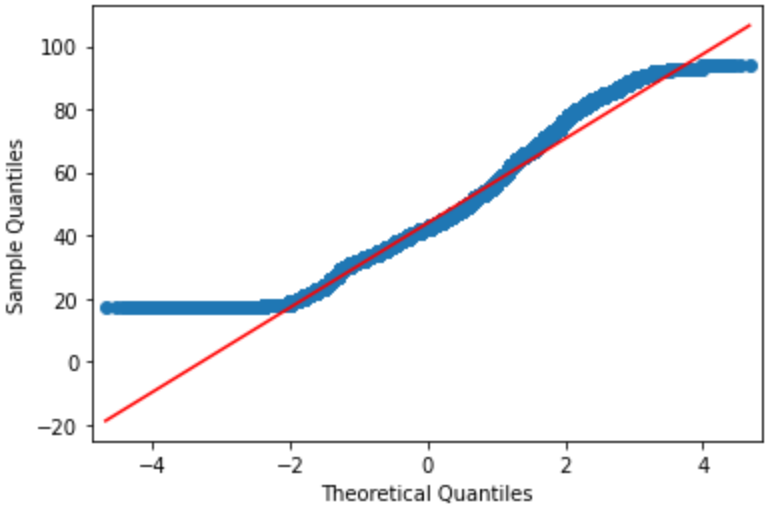
On rejète donc l'hypotèse nulle car la p-value est inférieure à 0.05. On peut dire que les variances de chaque groupe ne sont pas toutes égales.

Test de normalité:

Test de normalité par le graphique q-q plot :

In [111…

```
import statsmodels.api as sm
sm.qqplot(age_categ['age'], line = "r")
plt.show()
```



Le test de Smirnov-Kermogolv permet de tester si les variances sont significativement différentes ou non. Posons les hypothèses de départ :</br></br> H0 : La distribution suit une loi normale si p-value > 5% </br> H1 : La distribution ne suit pas une loi normale si p-value < 5%

In [113…

```
# Test de Smirnov-Kermogolov
from scipy import stats
from scipy.stats import kstest
kstest(age_categ['age'], stats.norm.cdf)
```

Out[113]:

```
KstestResult(statistic=1.0, pvalue=0.0)
```

On rejète l'hypotèse nulle car la p-value est inférieure à 0.05. On peut dire que la distibution ne suit pas une loi normale.</br></br> ~H0 : La distribution suit une loi normale si p-value > 5%~ </br> H1 : La distribution ne suit pas une loi normale si p-value < 5%

Les hypothèses de test ANOVA ne sont pas réunies. Nous allons donc utiliser le test non paramétrique de Kruskal-Wallis.</br></br> Posons les hypothèses de départ : </br> H0 : Les trois catégories ne différent pas pour l'age si p-value > 5% </br> H1 : Au moins une catégorie diffère des autres pour l'age si p-value < 5%

In [114…

```
# test non paramétrique de Kruskal
kstat, pval = sp.stats.kruskal(*[group["age"].values for name, group in age_categ.groupb
```

In [115…
```
kstat
```

Out[115]:
```
79491.97844148098
```

In [116…
```
pval
```

Out[116]:
```
0.0
```

Le test est significatif, Kstat = 79491.9784 et p-value = 0.0, on peut donc supposer qu'au moins une des 3 catégories diffère des autres pour l'age des clients.

3.5. Lien entre le genre d'un client et les catégories des livres achetés :

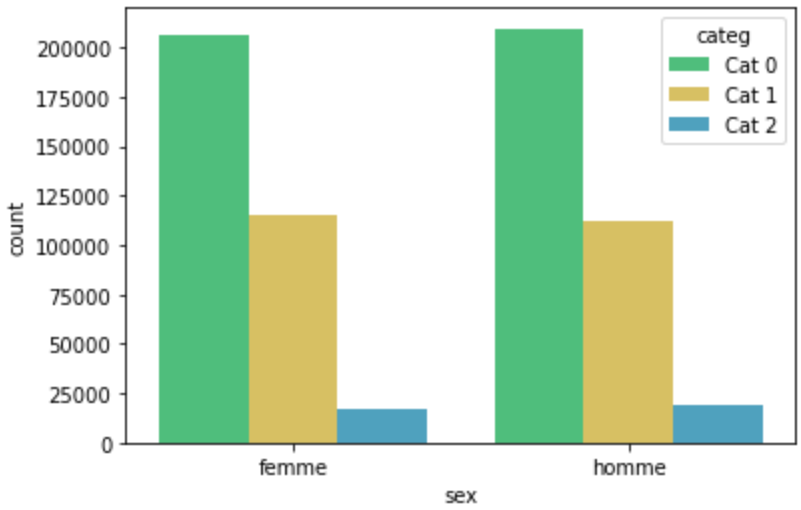
Il s'agit ici de deux variables qualitatives "sexe du client" et "catégorie de livre". On utilisera le test d'indépendance du khi-deux.

In [117…
```
# Séléction des données "sexe" et "catégorie":
categ_sex = data[['categ', 'sex']]
categ_sex.head()
```

Out[117]:

	categ	sex
0	Cat 0	femme
1	Cat 1	homme
2	Cat 0	femme
3	Cat 2	homme
4	Cat 0	homme

In [170…
```
sns.countplot(data = categ_sex, x = 'sex', hue = 'categ', palette = cols)
plt.show()
```



Hypothèse du test Khi-2

Posons les hypothèses de départ </br> H0 : Le sexe des clients et la catégorie de livre sont indépendants si p-value > 5%
H1 : Le sexe des clients et la catégorie de livre sont non indépendants si p-value < 5%

In [119…
```
# Matrice de contingence :
```

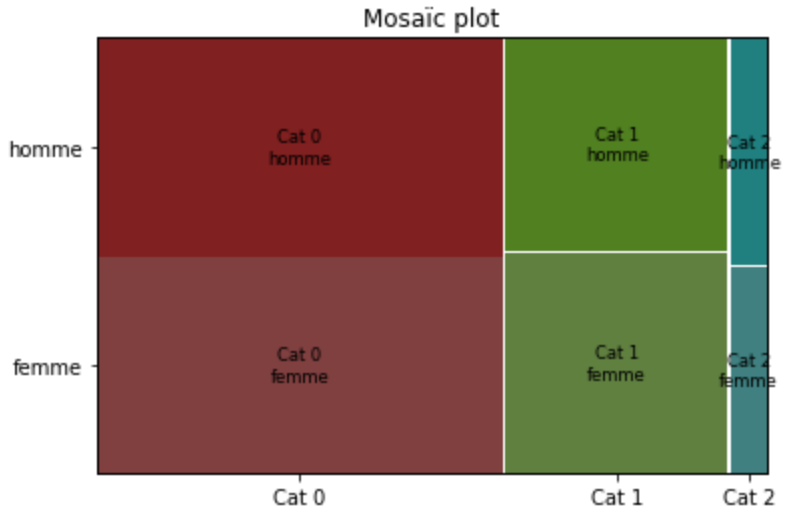
```
categ_sex_cross = pd.crosstab(categ_sex.categ, categ_sex.sex)
categ_sex_cross
```

Out[119]:

categ	sex	femme	homme
Cat 0		206257	209497
Cat 1		114899	112270
Cat 2		17283	19200

Visualisons grace à un graphe mosaïque s'il'y a un lien ou une influence entre les deux variables.

In [120…
```
from statsmodels.graphics.mosaicplot import mosaic
mosaic(categ_sex, ['categ', 'sex'], title='Mosaïc plot')
plt.show()
```



Les aires de chaque bloc du graphique sont très différentes, cela indique qu'il y a un lien ou influence entre nos deux variables.

In [121…
```
from scipy.stats import chi2_contingency
Khi2_obs, p_value, ddl, effectif_theorique = chi2_contingency(categ_sex_cross)
```

In [122…
```
print( "le résultat du test Khi-2 nous fourni une statistique de", Khi2_obs)
print( "le résultat du test Khi-2 nous fourni une p-value de", p_value)
```

```
le résultat du test Khi-2 nous fourni une statistique de 146.99906487909777
le résultat du test Khi-2 nous fourni une p-value de 1.2010432285664067e-32
```

On rejète donc l'hypotèse nulle car p-value est inférieure au seuil fixé de 5% . On peut dire que le sexe des clients et la catégorie de livre sont non indépendants.</br></br> H0 : Le sexe des clients et la catégorie de livre sont indépendants si p-value > 5%
H1 : Le sexe des clients et la catégorie de livre sont non indépendants si p-value < 5%

Pour mesurer l'intensité de la liaison entre nos deux variables qualitatives, nous allons calculer le coefficient de "V de Cramer" et le "T Tschuprov" (basés sur les résultats du test Khi-2).

In [123…
```
# Test Cramer et de Tschuprov
l = len(categ_sex.categ.unique())
c = len(categ_sex.sex.unique())
N = len(categ_sex)
mini = min((l - 1), (c - 1))
```

```
V = np.sqrt(Khi2_obs/(N * mini))
T = np.sqrt(Khi2_obs/(N * ddl))
print("Le résultat de V de Cramer est égal à", V)
print("Le résultat du coefficient de Tschuprov est égal à", T)
```

```
Le résultat de V de Cramer est égal à 0.014709320032271354
Le résultat du coefficient de Tschuprov est égal à 0.0104010599414622
```

Les deux coefficient sont proche de zéro, cela veut dire que la liaison entre le sexe et la catégorie de livre est faible.