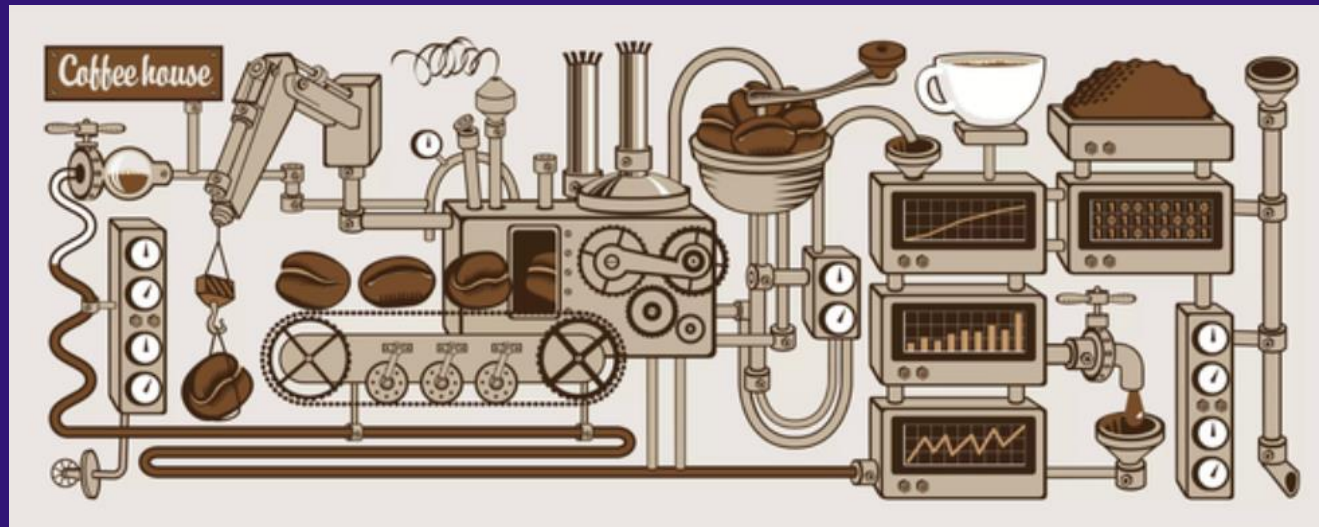



# Projet 2

# Rester livres

Nacim BOUGHANMI





Je suis Data Analyst dans **une grande chaine de librairie**, nommée **Rester livres**, qui propose depuis peu la vente de livre en ligne.

Mon manager et le service informatique de l'entreprise m'ont mis à disposition **la base de données des ventes de l'entreprise** et m'a donné pour mission:

- **Nettoyage du jeu de données**
- **Analyse des données**
- **Quelques questions supplémentaires**

# Nettoyage du jeu de données

# I. Analyse pré-exploratoire de ce jeu de données

```
Entrée [52]: import pandas
import matplotlib.pyplot as plt
import numpy as np
import datetime
import seaborn as sns
```

```
Entrée [2]: df_customers = pandas.read_csv("customers.csv")
df_products = pandas.read_csv("products.csv")
df_transactions = pandas.read_csv("transactions.csv")
```

J'importe l'intégralité de mon Data Set afin de vérifier la qualité de mon jeu de données.

# I. Analyse pré-exploratoire de ce jeu de données

```
Entrée [57]: df_customers['client_id'].isnull().value_counts()
```

```
Out[57]: False      8623  
        Name: client_id, dtype: int64
```

```
Entrée [56]: df_customers['birth'].isnull().value_counts()
```

```
Out[56]: False      8623  
        Name: birth, dtype: int64
```

```
Entrée [7]: df_products['price'].isnull().value_counts()
```

```
Out[7]: False      3287  
        Name: price, dtype: int64
```

```
Entrée [8]: df_products['categ'].isnull().value_counts()
```

```
Out[8]: False      3287  
        Name: categ, dtype: int64
```

```
Entrée [9]: df_transactions['client_id'].isnull().value_counts()
```

```
Out[9]: False     337016  
        Name: client_id, dtype: int64
```

```
Entrée [10]: df_transactions['session_id'].isnull().value_counts()
```

```
Out[10]: False     337016  
         Name: session_id, dtype: int64
```

```
Entrée [11]: df_transactions['date'].isnull().value_counts()
```

```
Out[11]: False     337016  
         Name: date, dtype: int64
```

À l'aide des **fonctions de bases de la librairie de pandas** je vérifie le nombre de manquant par indicateur.

## II. Élimination des valeurs négatives

Entrée [12]: `df_products.describe()`

Out[12]:

	price	categ
count	3287.000000	3287.000000
mean	21.856641	0.370246
std	29.847908	0.615387
min	-1.000000	0.000000
25%	6.990000	0.000000
50%	13.060000	0.000000
75%	22.990000	1.000000
max	300.000000	2.000000

Entrée [58]:

```
df_products = df_products.drop(df_products[df_products.price < 0].index)
df_products
df_products.describe()
```

Out[58]:

	price	categ
count	3286.000000	3286.000000
mean	21.863597	0.370359
std	29.849786	0.615446
min	0.620000	0.000000
25%	6.990000	0.000000
50%	13.075000	0.000000
75%	22.990000	1.000000
max	300.000000	2.000000

En utilisant la fonction **describe** de la librairie pandas nous pouvons remarquer que nous avons des valeurs **négatives**, ici je vais **drop** tous les prix des produit **inférieur à 0**.

### III. Élimination des valeurs “test”

Entrée [16]: `df_transactions.describe(include='all')`

Out[16]:

	id_prod		date	session_id	client_id
count	337016		337016	337016	337016
unique	3266		336855	169195	8602
top	1_369	test_2021-03-01 02:30:02.237413		s_0	c_1609
freq	1081		13	200	12855

Entrée [17]: `df_transactions = df_transactions[df_transactions['date'].str.contains('test') == False]`  
`df_transactions.describe(include='all')`

Out[17]:

	id_prod		date	session_id	client_id
count	336816		336816	336816	336816
unique	3265		336816	169194	8600
top	1_369	2021-10-31 02:21:47.099872		s_118668	c_1609
freq	1081		1	14	12855

En utilisant la fonction **describe(include = all)** nous pouvons remarquer que nous avons des valeurs “test”, ici je vais **drop** tous les dates de transaction incluant le caractère “test”.

# IV. Jointure des Data Frames

```
Entrée [19]: df_j= pandas.merge(df_customers, df_transactions, how = 'outer')
df_j
```

Out[19]:

	client_id	sex	birth	id_prod	date	session_id
--	-----------	-----	-------	---------	------	------------

0	c_4410	f	1967	0_1455	2021-03-22 14:29:25.189266	s_9942
---	--------	---	------	--------	----------------------------	--------

1	c_4410	f	1967	0_1376	2021-09-24 22:58:27.418343	s_94984
---	--------	---	------	--------	----------------------------	---------

2	c_4410	f	1967	1_312	20
---	--------	---	------	-------	----

3	c_4410	f	1967	1_653	20
---	--------	---	------	-------	----

4	c_4410	f	1967	0_1110	20
---	--------	---	------	--------	----

...	...	...	...	...	...
-----	-----	-----	-----	-----	-----

336834	c_84	f	1982	1_459	20
--------	------	---	------	-------	----

336835	c_84	f	1982	0_1050	20
--------	------	---	------	--------	----

336836	c_84	f	1982	0_1417	20
--------	------	---	------	--------	----

336837	c_84	f	1982	1_343	20
--------	------	---	------	-------	----

336838	c_84	f	1982	0_1571	20
--------	------	---	------	--------	----

```
Entrée [20]: df_j = pandas.merge(df_j, df_products,how = 'outer' )
df_j
```

Out[20]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
--	-----------	-----	-------	---------	------	------------	-------	-------

0	c_4410	f	1967.0	0_1455	2021-03-22 14:29:25.189266	s_9942	8.99	0.0
---	--------	---	--------	--------	----------------------------	--------	------	-----

1	c_4389	m	1984.0	0_1455	2021-07-09 11:16:18.579726	s_59967	8.99	0.0
---	--------	---	--------	--------	----------------------------	---------	------	-----

2	c_5019	f	1977.0	0_1455	2022-01-15 00:01:53.456196	s_149928	8.99	0.0
---	--------	---	--------	--------	----------------------------	----------	------	-----

3	c_7049	f	1987.0	0_1455	2021-03-04 14:01:38.698752	s_1637	8.99	0.0
---	--------	---	--------	--------	----------------------------	--------	------	-----

4	c_5110	f	1982.0	0_1455	2021-09-05 11:48:41.065009	s_85364	8.99	0.0
---	--------	---	--------	--------	----------------------------	---------	------	-----

...	...	...	...	...	...	...	...	...
-----	-----	-----	-----	-----	-----	-----	-----	-----

336856	NaN	NaN	NaN	0_525	NaN	NaN	2.99	0.0
--------	-----	-----	-----	-------	-----	-----	------	-----

336857	NaN	NaN	NaN	2_86	NaN	NaN	132.36	2.0
--------	-----	-----	-----	------	-----	-----	--------	-----

336858	NaN	NaN	NaN	0_299	NaN	NaN	22.99	0.0
--------	-----	-----	-----	-------	-----	-----	-------	-----

336859	NaN	NaN	NaN	0_510	NaN	NaN	23.66	0.0
--------	-----	-----	-----	-------	-----	-----	-------	-----

336860	NaN	NaN	NaN	0_2308	NaN	NaN	20.28	0.0
--------	-----	-----	-----	--------	-----	-----	-------	-----

Jointure externe des Data Frames à l'aide de la fonction **merge**.



# V. Imputation des valeurs manquantes

```
Entrée [21]: df_j.isnull().sum()
```

```
Out[21]: client_id    22  
sex              22  
birth            22  
id_prod          23  
date             45  
session_id       45  
price            126  
categ            126  
dtype: int64
```

```
Entrée [22]: df_j = df_j[df_j['session_id'].isnull() == False]  
df_j
```

```
Out[22]:
```

	client_id	sex	birth	id_prod	date	session_id	price	categ
0	c_4410	f	1967.0	0_1455	2021-03-22 14:29:25.189266	s_9942	8.99	0.0
1	c_4389	m	1984.0	0_1455	2021-07-09 11:16:18.579726	s_59967	8.99	0.0
2	c_5019	f	1977.0	0_1455	2022-01-15 00:01:53.456196	s_149928	8.99	0.0
3	c_7049	f	1987.0	0_1455	2021-03-04 14:01:38.698752	s_1637	8.99	0.0
4	c_5110	f	1982.0	0_1455	2021-09-05 11:48:41.065009	s_85364	8.99	0.0
...	...	...	...	...	...	...	...	...
336834	c_7135	m	1996.0	2_99	2021-07-11 20:56:49.820935	s_61009	84.99	2.0
336835	c_5828	f	1998.0	2_99	2021-11-21 01:53:46.967570	s_122697	84.99	2.0
336836	c_8260	m	1991.0	0_833	2021-09-27 23:22:40.394509	s_96558	2.99	0.0
336837	c_8138	f	1984.0	0_394	2021-11-09 09:02:38.299240	s_116986	2.14	0.0
336838	c_8327	m	1972.0	0_394	2021-12-28 22:44:11.200205	s_141516	2.14	0.0

La fonction `isnull(). sum()` me permet de compter le nombre de manquant par indicateur. Ici je supprime **tous les produits non commandés**.

# V. Imputation des valeurs

Entrée [23]: `df_j.isnull().sum()`

Out[23]:

client_id	0
sex	0
birth	0
id_prod	0
date	0
session_id	0
price	103
categ	103
dtype:	int64

Entrée [24]: `df_j[df_j.isna().any(axis=1)]`

Out[24]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
266960	c_4505	m	1976.0	0_2245	2022-01-09 09:23:31.000720	s_147220	NaN	NaN
266961	c_3468	f	1981.0	0_2245	2021-09-11 10:52:05.205583	s_88251	NaN	NaN
266962	c_1403	f	1978.0	0_2245	2022-02-15 14:26:50.187952	s_165575	NaN	NaN
266963	c_3065	f	1977.0	0_2245	2022-01-26 13:34:33.440366	s_155484	NaN	NaN
266964	c_7102	m	1983.0	0_2245	2021-04-25 19:58:42.716401	s_25704	NaN	NaN

Entrée [25]: `df_j['price'].fillna(df_products['price'].mean(), inplace = True)`  
`df_j['categ'].fillna(df_products['categ'].mode()[0], inplace = True)`  
`df_j`

Out[25]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
0	c_4410	f	1967.0	0_1455	2021-03-22 14:29:25.189266	s_9942	8.99	0.0
1	c_4389	m	1984.0	0_1455	2021-07-09 11:16:18.579726	s_59967	8.99	0.0
2	c_5019	f	1977.0	0_1455	2022-01-15 00:01:53.456196	s_149928	8.99	0.0
3	c_7049	f	1987.0	0_1455	2021-03-04 14:01:38.698752	s_1637	8.99	0.0
4	c_5110	f	1982.0	0_1455	2021-09-05 11:48:41.065009	s_85364	8.99	0.0

Ici j'impute les valeurs manquantes par des valeurs moyennes ou par une valeur prédéfinie.

# **Analyse des données**

# I. Création d'une Data Frame chiffre d'affaire

```
Entrée [28]: df_CA['date'] = pandas.to_datetime(df_CA['date'])
CA = df_CA.groupby(df_CA['date'].dt.strftime('%B %Y'))['price'].sum().reset_index()
CA
```

Out[28]:

	date	price
0	April 2021	473286.810359
1	August 2021	479501.962374
2	December 2021	523090.725179
3	February 2022	532981.379568
4	January 2022	523064.627985
5	July 2021	480986.114388
6	June 2021	481410.985179
7	March 2021	479508.802374
8	May 2021	489542.858777
9	November 2021	513108.805971
10	October 2021	319313.005180
11	September 2021	503033.643165

```
Entrée [27]: df_CA = df_j[['date', 'price']]
df_CA
```

Out[27]:

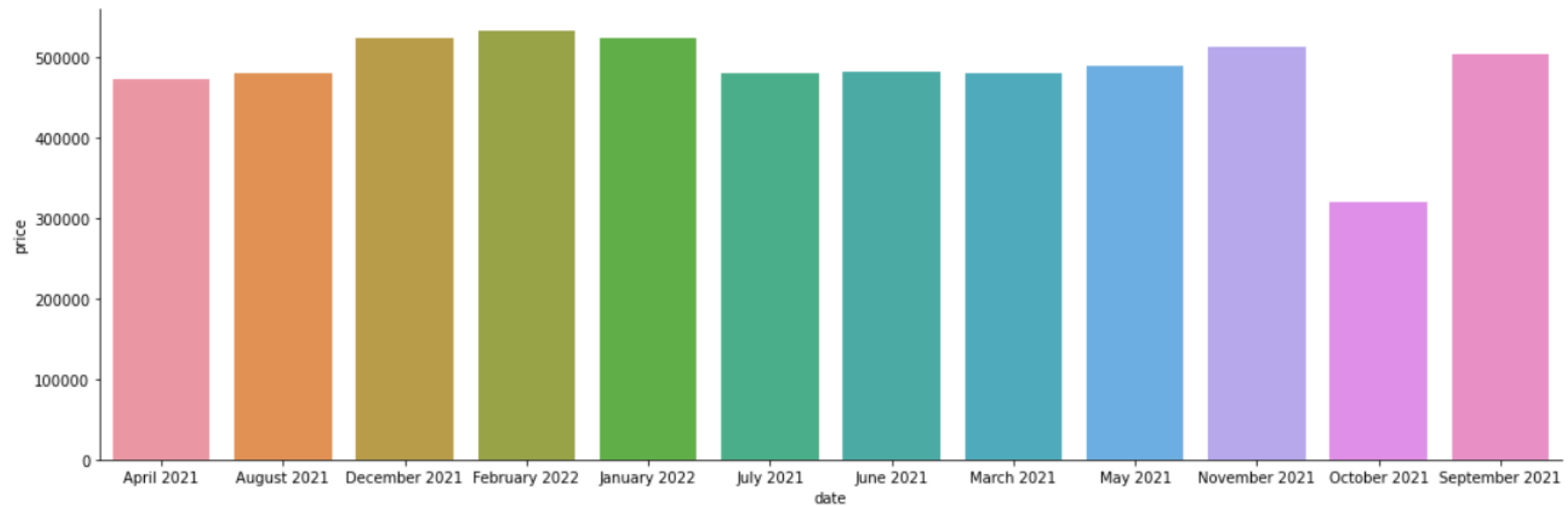
	date	price
0	2021-03-22 14:29:25.189266	8.99
1	2021-07-09 11:16:18.579726	8.99
2	2022-01-15 00:01:53.456196	8.99
3	2021-03-04 14:01:38.698752	8.99
4	2021-09-05 11:48:41.065009	8.99

Je crée une Data Frames **CA** qui me permet de visualiser par la suite leur **chiffre d'affaires par mois**.

## II. Visualisation du chiffre d'affaire par mois

```
Entrée [29]: sns.catplot(x= 'date', y= "price", kind="bar", data= CA, height=5, aspect=15/5)
```

```
Out[29]: <seaborn.axisgrid.FacetGrid at 0x1ca394c5160>
```



J'utilise la fonction `catplot` pour visualiser les chiffres d'affaires par mois.

# III. Explication du chiffre d'affaire d'octobre

```
Entrée [29]: Otc= df_j[df_j['date'].str.contains('2021-10') == True]
Otc['categ'].value_counts()
```

```
Out[29]: 0.0    18758
1.0     1666
2.0     1160
Name: categ, dtype: int64
```

```
Entrée [30]: fev= df_j[df_j['date'].str.contains('2022-')]
fev['categ'].value_counts()
```

```
Out[30]: 0.0    17273
1.0    10459
2.0     1835
Name: categ, dtype: int64
```

```
Entrée [31]: co= df_j[df_j['categ']== 2]
co
```

Out[31]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
35547	c_415	m	1993.0	2_19	2021-11-21 06:18:41.695577	s_122787	69.99	2.0
35548	c_6884	m	1994.0	2_19	2021-05-07 13:32:04.703604	s_31172	69.99	2.0
35549	c_5139	f	2000.0	2_19	2021-04-03 07:31:36.674313	s_15300	69.99	2.0
35550	c_109	f	2004.0	2_19	2022-01-06 22:08:02.218755	s_146030	69.99	2.0
35551	c_5167	f	1997.0	2_19	2022-02-15 07:55:07.443719	s_165432	69.99	2.0

```
Entrée [32]: cp= df_j[df_j['categ']== 1]
cp
```

Out[32]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
957	c_4410	f	1967.0	1_312	2022-01-29 14:07:47.482092	s_156960	24.56	1.0
958	c_2253	m	1967.0	1_312	2021-04-14 05:42:49.629624	s_20318	24.56	1.0
959	c_4767	m	1979.0	1_312	2022-02-24 20:44:46.657939	s_170349	24.56	1.0
960	c_1044	m	1956.0	1_312	2021-09-15 21:15:24.352934	s_90464	24.56	1.0
961	c_3167	f	1965.0	1_312	2021-03-16 15:52:36.065937	s_7185	24.56	1.0

Je crée différentes Data Frames qui me permettront par la suite de compter le nombre de produits commandés pour chaque catégorie par mois.

# III. Explication du chiffre d'affaire d'octobre

```
Entrée [33]: co.date = pandas.to_datetime(co.date)
df_co = co.groupby(pandas.Grouper(key='date', freq='M')).sum()
df_co[['price', 'categ']].sort_values(['price'], ascending=False)
```

Out[33]:

	price	categ
date		
2021-08-31	148635.99	3896.0
2021-07-31	147663.47	3956.0
2022-02-28	136479.72	3670.0
2021-05-31	127359.59	3306.0
2021-06-30	124209.56	3338.0
2021-04-30	111682.70	3002.0
2021-11-30	104136.00	2746.0
2022-01-31	102524.72	2740.0
2021-03-31	98771.48	2630.0
2021-10-31	86179.70	2320.0
2021-12-31	65934.49	1752.0
2021-09-30	65893.29	1748.0

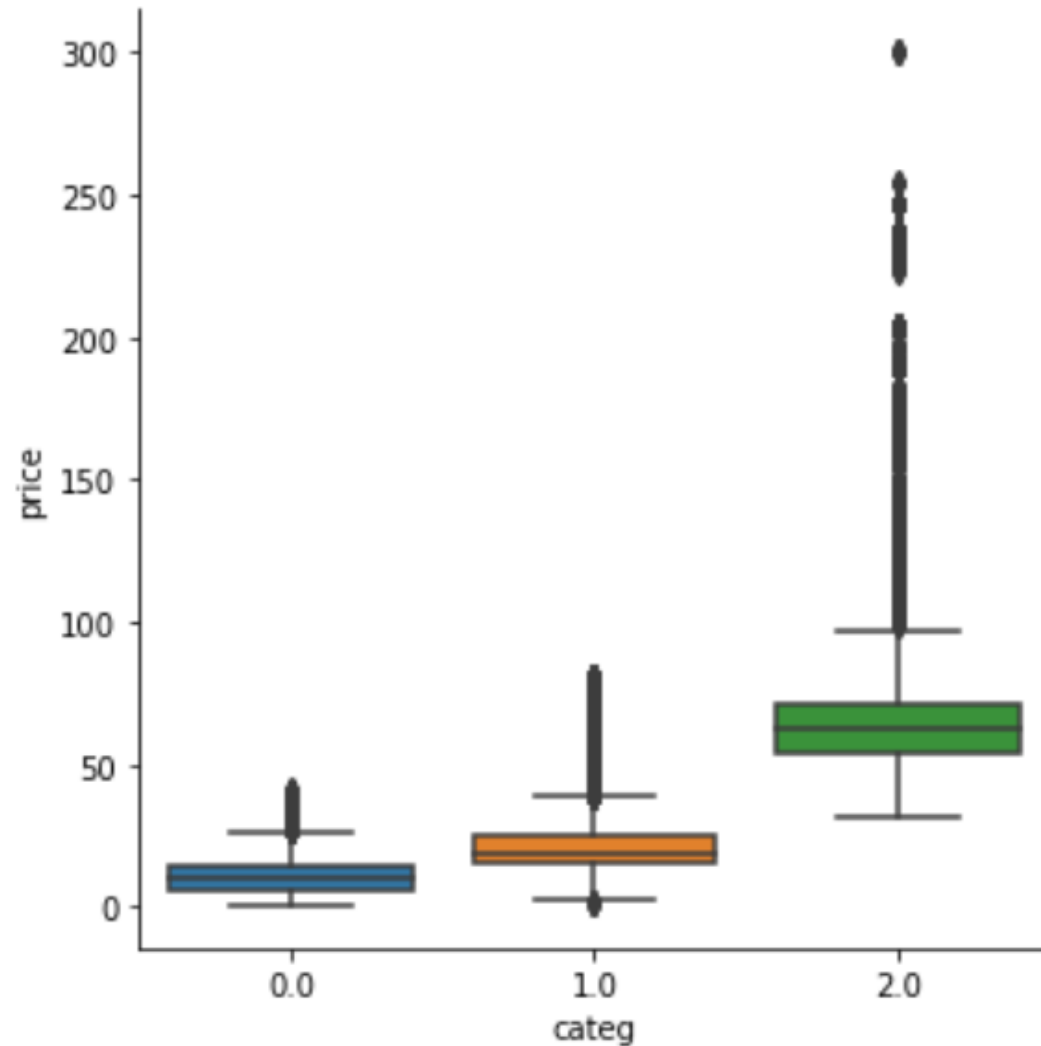
```
Entrée [34]: cp.date = pandas.to_datetime(cp.date)
df_cp = co.groupby(pandas.Grouper(key='date', freq='M')).sum()
df_cp[['price', 'categ']].sort_values(['price'], ascending=False)
```

Out[49]:

	price	categ
date		
2022-01-31	256267.92	12560.0
2021-11-30	252910.39	12316.0
2021-12-31	251026.75	12259.0
2022-02-28	213120.64	10459.0
2021-09-30	190613.78	9268.0
2021-06-30	189162.04	9264.0
2021-07-31	188523.27	9169.0
2021-03-31	186974.17	9134.0
2021-05-31	165893.40	8107.0
2021-08-31	162991.38	7954.0
2021-04-30	156138.35	7579.0
2021-10-31	33762.32	1666.0

la **df\_co** correspond au produit de la **catégorie 2**, la **df cp** correspond quant à elle au produit de la **catégorie 1**.

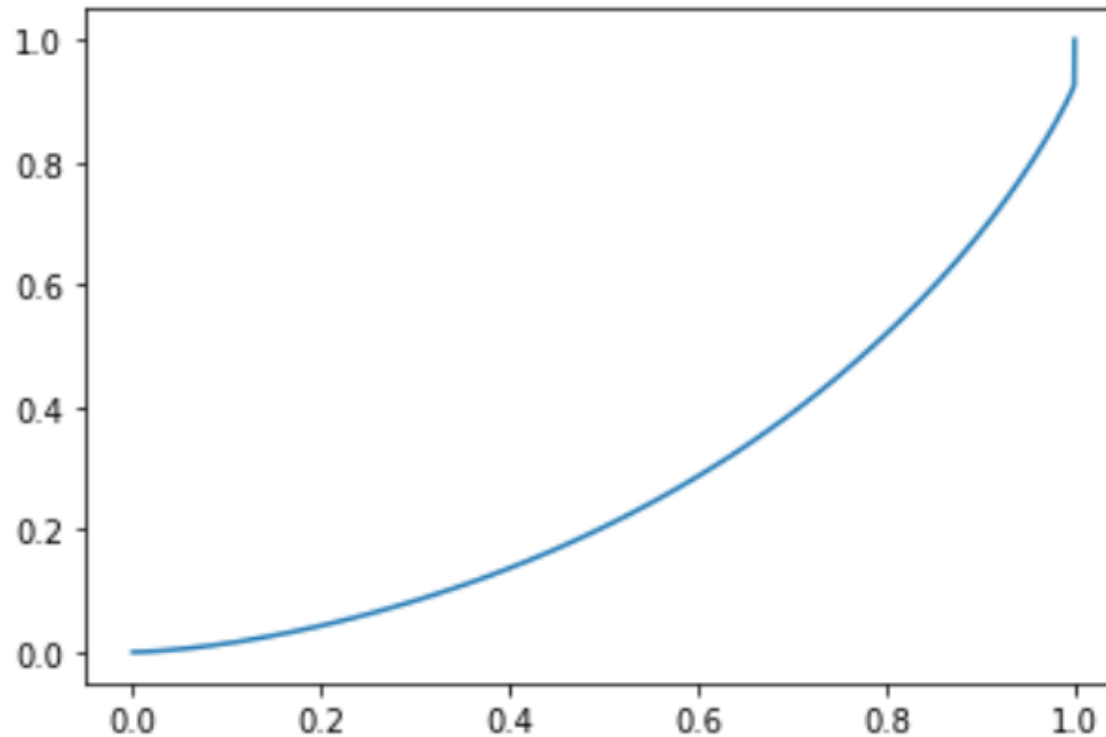
## IV. Boîte à moustaches



J'ai réalisé un diagramme dit **boîte à moustache** afin de visualiser pour chaque catégories **les différences prix**.



## V. Courbe de Lorenz et Indice de Gini



Out[38]: 0.4397259518529868

J'ai réalisé une **courbe de Lorenz** qui m'a permis de calculer l'**indice de Gini**, celui-ci nous démontre que la répartition **des dépenses de chaque client** est moyennement égalitaire.

# **Quelques questions supplémentaires**

I. Y a-t-il une corrélation entre le sexe des clients et les catégories de produits achetés ?

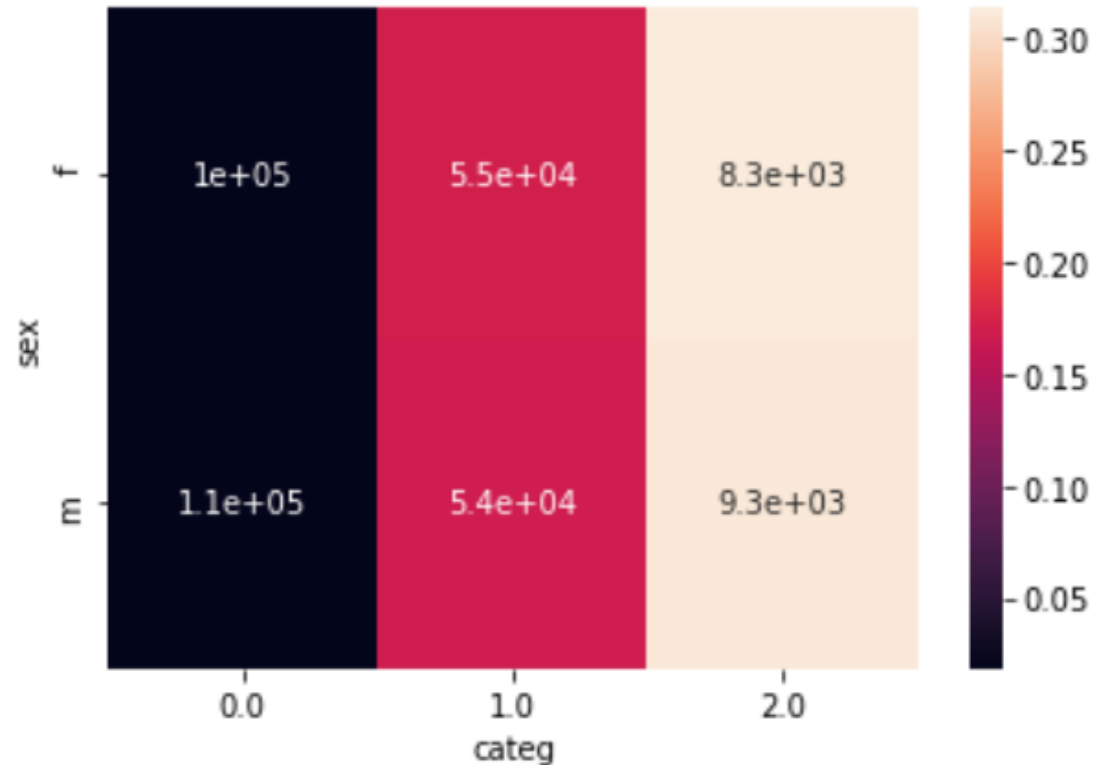
Entrée [39]:

```
X = "sex"
Y = "categ"

cont = df_j[[X,Y]].pivot_table(index=X,columns=Y,aggfunc=len,margins=True,margins_name="Total")
cont
```

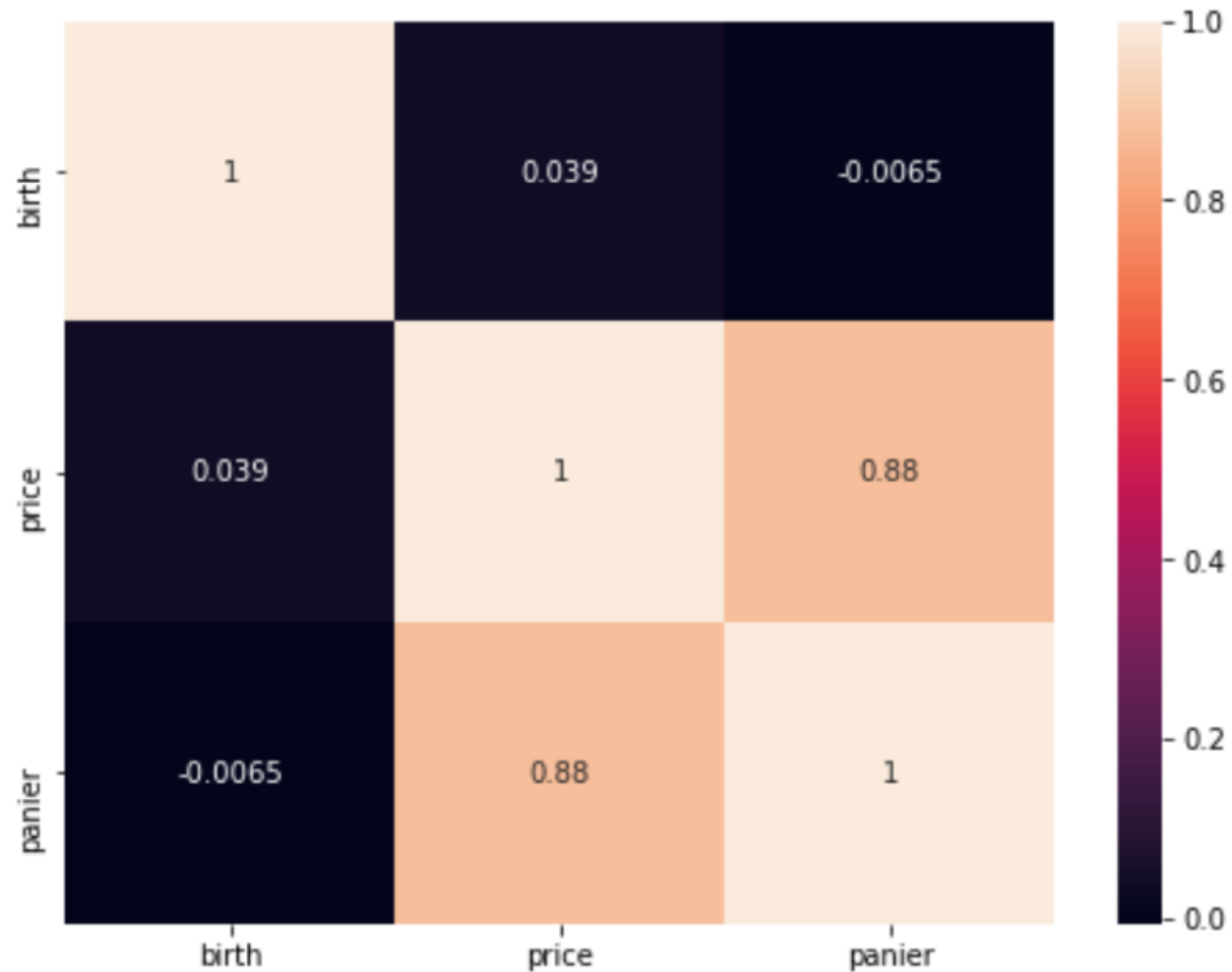
Out[39]:

categ	0.0	1.0	2.0	Total
sex				
f	103846	55469	8260	167575
m	105683	54266	9292	169241
Total	209529	109735	17552	336816



J'utilise la méthode **Chi-2** qui consiste à créer d'abord un premier **tableau dit de contingence**, qui me permet de par la suite de faire apparaître un tableau de **contingence coloré**.

II. Y a-t-il une corrélation entre l'âge des clients et le montant total des achats ?



J'utilise la fonction **corr** qui me permet de représenter un **tableau de contingence coloré**.

III. Y a-t-il une corrélation entre l'âge des clients et la fréquence d'achat ?

```

Entrée [42]: df_j.date = pandas.to_datetime(df_j.date)
df_j = df_j.sort_values(['client_id', 'date'])
df_j['previous_order'] = df_j.groupby(['client_id'])['date'].shift()

df_j['days_bw_orders'] = df_j['date'] - df_j['previous_order']

df_j['days_bw_orders'] = df_j['days_bw_orders'].apply(lambda x: x.days)

df_j.groupby('client_id')['days_bw_orders'].agg('mean')

df_j['days_bw_orders'].mean()
df_j

```

Out[42]:

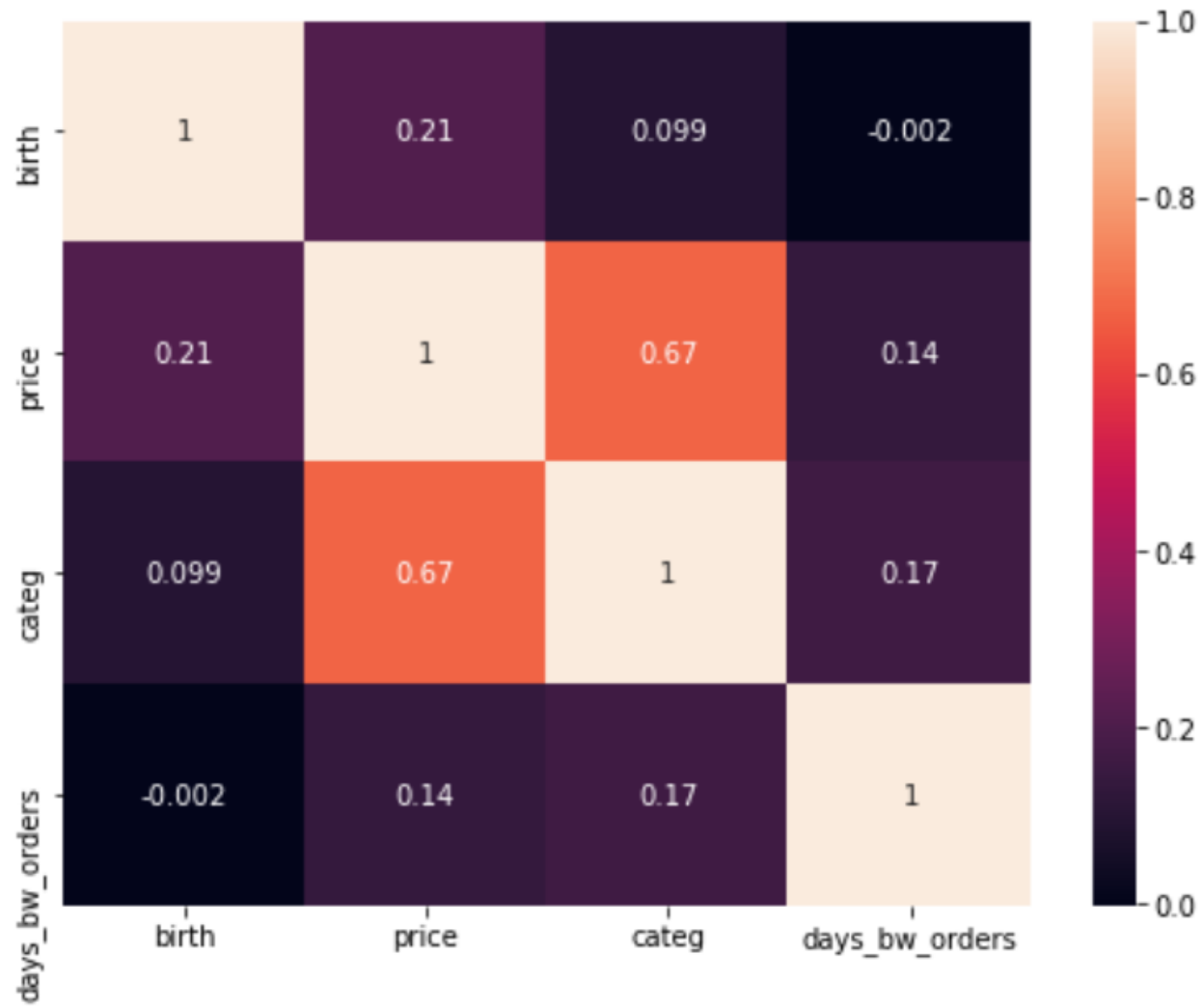
	client_id	sex	birth	id_prod	date	session_id	price	categ	previous_order	days_bw_orders
86144	c_1	m	1955.0	0_1470	2021-06-11 21:02:39.382765	s_47346	19.53	0.0	NaT	NaN
247492	c_1	m	1955.0	0_513	2021-07-21 22:41:38.769525	s_65433	11.99	0.0	2021-06-11 21:02:39.382765	40.0
148281	c_1	m	1955.0	0_1186	2021-07-25 12:17:34.446678	s_66947	12.30	0.0	2021-07-21 22:41:38.769525	3.0
68614	c_1	m	1955.0	0_1448	2021-07-26 17:37:29.438136	s_67467	18.94	0.0	2021-07-25 12:17:34.446678	1.0
74939	c_1	m	1955.0	0_1475	2021-07-27 10:30:00.293075	s_67769	11.99	0.0	2021-07-26 17:37:29.438136	0.0

Out[43]:

	client_id	birth	days_bw_orders
0	c_1	1955.0	12.105263
1	c_10	1956.0	9.592593
2	c_100	1992.0	22.400000
3	c_1000	1966.0	6.036364
4	c_1001	1982.0	6.052632
...	...	...	...
8595	c_995	1955.0	29.714286
8596	c_996	1970.0	7.951220
8597	c_997	1994.0	11.136364
8598	c_998	2001.0	12.407407
8599	c_999	1964.0	10.333333

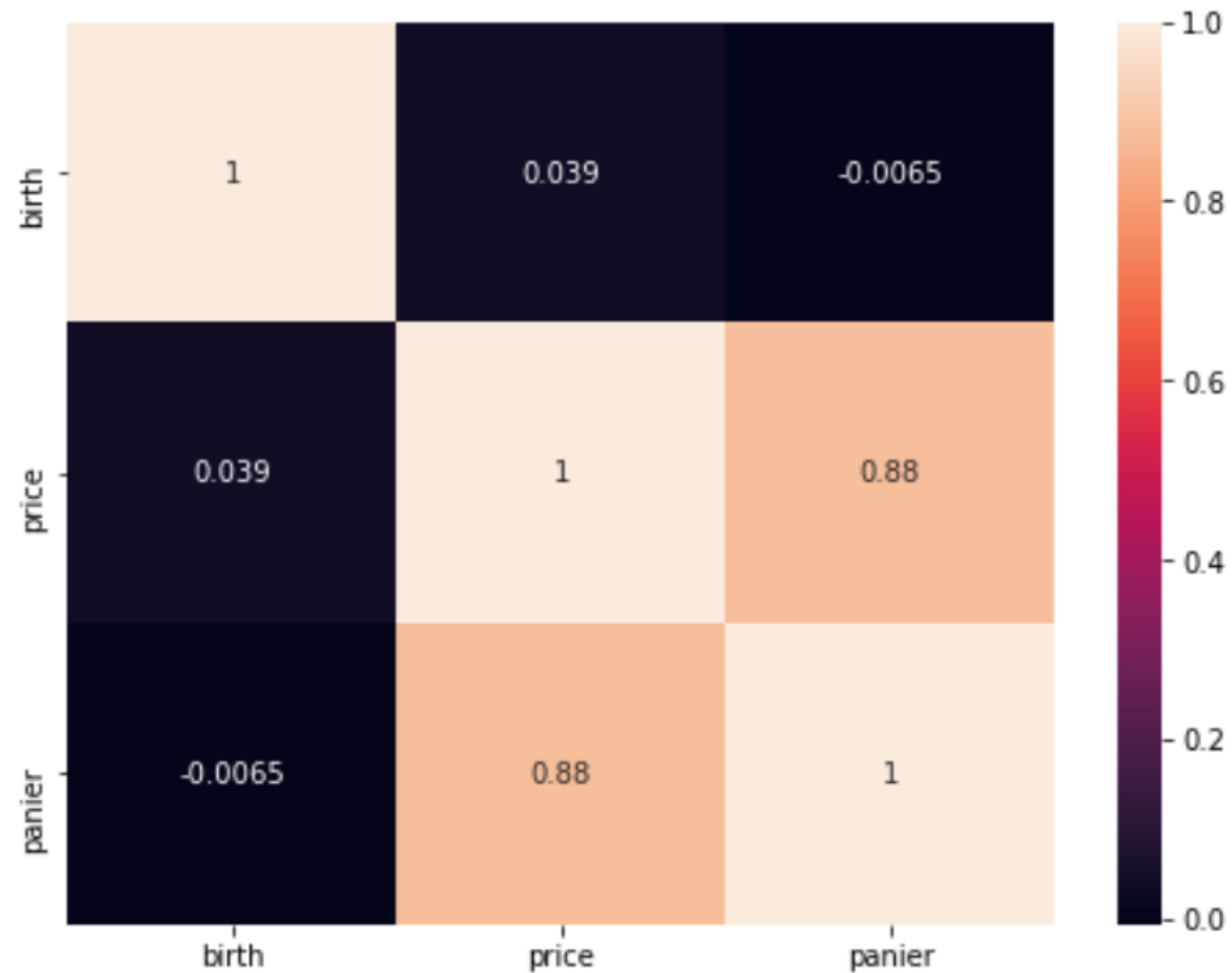
Ici je vais d'abord calculer la différence de jour entre les commandes pour chaque client, ensuite je fais la moyenne de jours pour chaque client.





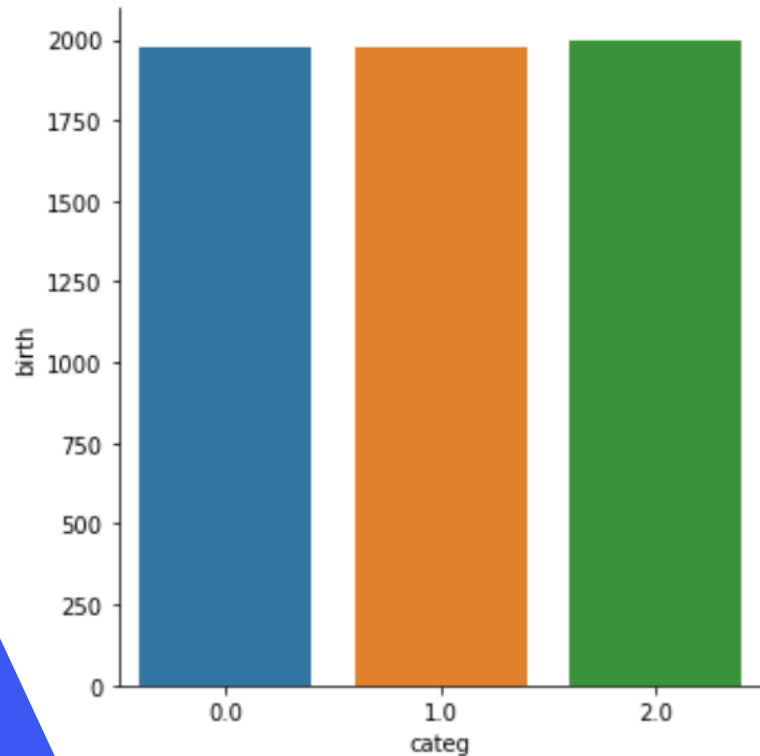
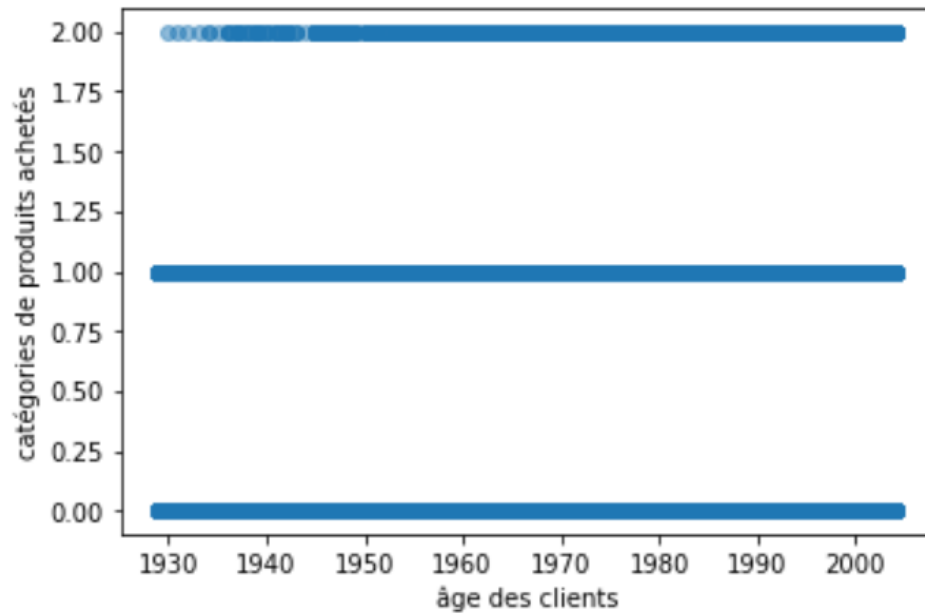
Par la suite j'utilise comme précédemment la fonction **corr**.

IV. Y a-t-il une corrélation entre l'âge des clients et la taille du panier moyen ?



Après avoir calculé le **panier pour chaque client**, je peux directement utiliser la fonction **corr** sur la Data Frame shop.

V. Y a-t-il une corrélation entre  
l'âge des clients et les catégories  
de produits achetés ?



Pour cette corrélation j'ai choisi de réaliser **3 graphiques**.