

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
from scipy import stats as st
```

```
In [2]: #ctps = pd.read_csv('ctcp.csv')
```

```
In [3]: ctp = pd.read_csv('ctp_partie2.csv')
ctc.head()
```

```
Out[3]:
```

	id_produ	date	session_id	client_id	price	categ	sex	birth
0	0_1483	2021-04-10 18:37:28.723910	s_18746	c_4450	4.99	0.0	f	1977
1	2_226	2022-02-03 01:55:53.276402	s_159142	c_277	65.75	2.0	f	2000
2	1_374	2021-09-23 15:13:46.938559	s_94290	c_4270	10.71	1.0	f	1979
3	0_2186	2021-10-17 03:27:18.783634	s_105936	c_4597	4.20	0.0	m	1963
4	0_1351	2021-07-17 20:34:25.800563	s_63642	c_1242	8.99	0.0	f	1980

```
In [4]: #Je change le format de date pour separer date et heure
ctc['date'] = ctc['date'].astype('datetime64[ns]')#transfo format date methode datetime
ctc['price'] = pd.to_numeric(ctc['price'])#transformation de la variable prix en numerique
ctc['time'] = ctc['date'].apply(lambda x: x.time()), ctc['date'] = ctc['date'].apply(lambda x: x.date())
#methode lambda pour ajouter pour ajouter une colonne 'time' pour pouvoir separer date et time
ctc.head()
```

```
Out[4]:
```

	id_produ	date	session_id	client_id	price	categ	sex	birth	time
0	0_1483	2021-04-10	s_18746	c_4450	4.99	0.0	f	1977	18:37:28.723910
1	2_226	2022-02-03	s_159142	c_277	65.75	2.0	f	2000	01:55:53.276402
2	1_374	2021-09-23	s_94290	c_4270	10.71	1.0	f	1979	15:13:46.938559
3	0_2186	2021-10-17	s_105936	c_4597	4.20	0.0	m	1963	03:27:18.783634
4	0_1351	2021-07-17	s_63642	c_1242	8.99	0.0	f	1980	20:34:25.800563

```
In [5]: #help(pd.to_numeric)
```

```
In [6]: #pour simplifier l'analyse je calcule l'age et ajoute une colonne 'age'
ctc['age'] = 2021 - ctc['birth']
ctc['age'] = pd.to_numeric(ctc['age'], errors='coerce')
ctc.head()
```

```
Out[6]:
```

	id_produ	date	session_id	client_id	price	categ	sex	birth	time	age
0	0_1483	2021-04-10	s_18746	c_4450	4.99	0.0	f	1977	18:37:28.723910	44
1	2_226	2022-02-03	s_159142	c_277	65.75	2.0	f	2000	01:55:53.276402	21
2	1_374	2021-09-23	s_94290	c_4270	10.71	1.0	f	1979	15:13:46.938559	42
3	0_2186	2021-10-17	s_105936	c_4597	4.20	0.0	m	1963	03:27:18.783634	58
4	0_1351	2021-07-17	s_63642	c_1242	8.99	0.0	f	1980	20:34:25.800563	41

Je procede a l'analyse

```
In [7]: #Agrégation des données transactionnelles par fréquence mensuelle (methode .groupby())
ctc_evol = ctc.groupby('date').sum().reset_index() #reset_index permet d'indexer correctement dans la nouvelle table
ctc_evol['ventes_keuros'] = ctc_evol['price'] / 1000 #transformation des prix kiloeuros
ctc_evol = ctc_evol[['date', 'ventes_keuros']]
ctc_evol.tail()
```

```
Out[7]:
```

	date	ventes_keuros
360	2022-02-24	20.20037
361	2022-02-25	18.20326
362	2022-02-26	19.75958
363	2022-02-27	19.02183
364	2022-02-28	18.71994

```
In [8]: ctc_evol.shape
```

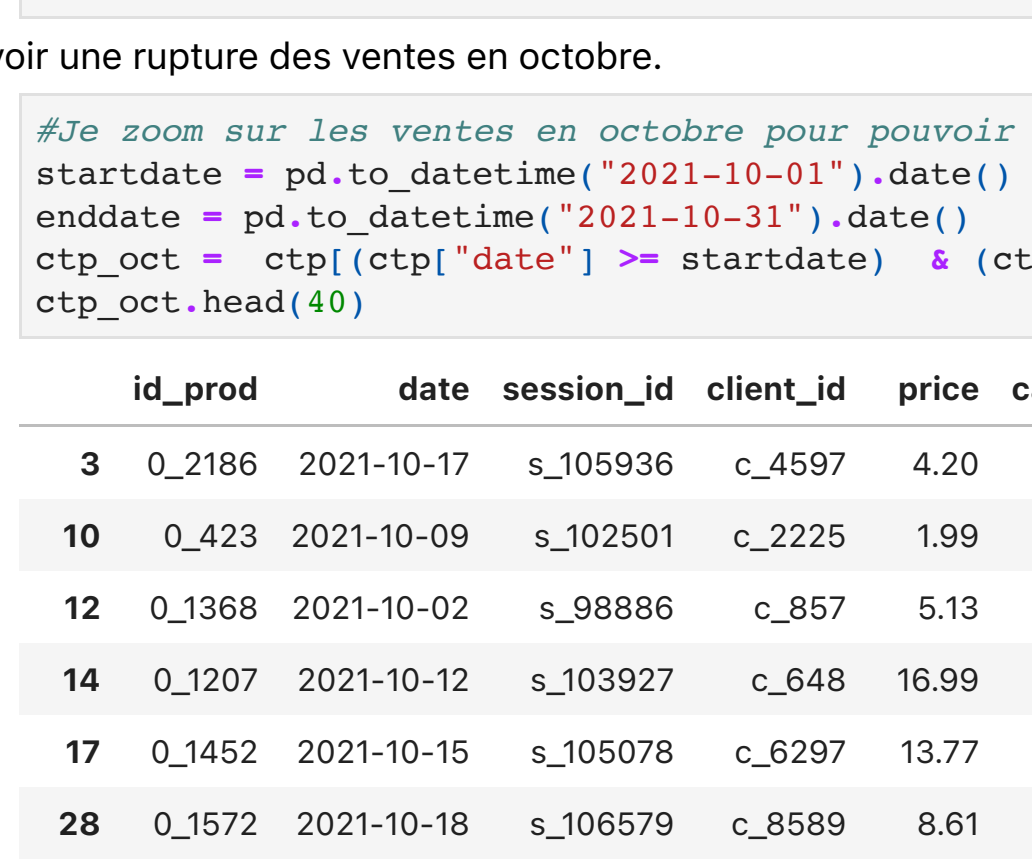
```
Out[8]:
```

```
(365, 2)
```

365 ligne pour 365 jours de l'année

```
In [9]: #ctc_evol.plot(x='date', y='ventes_keuros')
ventes = ctc_evol['ventes_keuros']
temps = ctc_evol['date']
fig, ax = plt.subplots()
plt.plot(temps, ventes)

ax.set(xlabel='temps (mois)', ylabel='Ventes (K€)',
       title='Evolution des ventes') #permet de legender les axes et la graphique
ax.grid() #ajoute une grille, facilite l'analyse
fig.savefig('test.png') #exporter et enregistrer le graphique
plt.show()
```



```
In [10]: #help(plt.subplots)
```

il semble avoir une rupture des ventes en octobre.

```
In [11]: #Je zoom sur les ventes en octobre pour pouvoir identifier des raisons pour cette baisse
startdate = pd.to_datetime('2021-10-01').date()
enddate = pd.to_datetime('2021-10-31').date()
ctc_oct = ctc[ctc['date'] >= startdate & (ctc['date'] <= enddate)]
ctc_oct.head(40)
```

```
Out[11]:
```

	id_produ	date	session_id	client_id	price	categ	sex	birth	time	age
3	0_2186	2021-10-17	s_105936	c_4597	4.20	0.0	m	1963	03:27:18.783634	58
10	0_423	2021-10-09	s_102501	c_2225	1.99	0.0	f	1949	23:50:33.907269	72
12	0_1368	2021-10-02	s_38886	c_857	5.13	0.0	m	1965	10:56:43.188179	56
14	0_1207	2021-10-12	s_103927	c_648	16.99	0.0	m	1976	22:50:12.290635	45
17	0_1452	2021-10-15	s_105078	c_6297	13.77	0.0	f	1969	10:07:12.401758	52
28	0_1572	2021-10-18	s_106579	c_8589	8.61	0.0	m	1958	10:44:56.742021	63
31	0_1127	2021-10-02	s_38883	c_2041	5.99	0.0	f	1975	10:41:36.135881	46
55	0_1034	2021-10-11	s_103080	c_4870	14.38	0.0	f	1981	04:23:19.599571	40
142	0_1420	2021-10-17	s_106247	c_390	11.53	0.0	f	1982	19:28:49.619771	39
160	0_1348	2021-10-17	s_106138	c_1656	12.03	0.0	f	1982	13:29:31.536194	39
163	2_227	2021-10-22	s_108583	c_5841	50.99	2.0	m	1996	17:41:17.347164	25
175	0_1435	2021-10-13	s_104041	c_3000	13.99	0.0	f	1985	05:50:12.700268	36
242	0_1247	2021-10-05	s_100274	c_7850	15.99	0.0	f	1981	08:28:58.911451	40
260	2_233	2021-10-03	s_99180	c_4958	172.99	2.0	m	1999	01:57:22.634598	22
331	1_348	2021-10-01	s_98327	c_2543	16.15	1.0	m	1952	08:16:02.908615	69
332	0_1253	2021-10-26	s_110214	c_412	12.99	0.0	m	1976	03:13:10.751184	45
344	1_435	2021-10-28	s_111122	c_7146	11.99	1.0	m	1971	01:36:30.625115	50
350	0_515	2021-10-20	s_107496	c_6999	11.99	0.0	f	1981	10:05:59.517060	40
351	0_1623	2021-10-20	s_107730	c_1561	7.99	0.0	f	1982	22:02:37.538613	39
368	0_1562	2021-10-21	s_107940	c_3970	14.73	0.0	f	1991	08:26:19.739630	30
389	0_1521	2021-10-13	s_104093	c_722	17.99	0.0	f	1987	08:13:27.087469	34
426	0_1434	2021-10-18	s_106482	c_1609	8.58	0.0	m	1980	05:54:03.183680	41
443	0_1209	2021-10-11	s_103004	c_4780	8.99	0.0	f	1954	04:04:05.992073	67
458	0_1287	2021-10-11	s_103082	c_1855	11.99	0.0	f	1987	04:37:14.556655	34
459	0_410	2021-10-23	s_109088	c_1080	25.23	0.0	f	1969	19:04:21.253947	52
466	0_1644	2021-10-18	s_106420	c_6406	4.77	0.0	m	1988	03:30:01.073138	33
478	0_1123	2021-10-31	s_112551	c_2239	12.99	0.0	m	1989	04:08:41.904426	32
487	0_2171	2021-10-25	s_110069	c_4824	4.99	0.0	m	1974	20:42:44.831312	47
513	0_944	2021-10-17	s_106067	c_8326	13.38	0.0	m	1983	10:19:17.616939	38
577	0_1474	2021-10-30	s_112100	c_89	9.88	0.0	m	1971	03:47:06.880002	50
583	0_1301	2021-10-16	s_105489	c_528	5.99	0.0	f	1989	07:56:19.682714	32
624	0_1564	2021-10-21	s_107880	c_8537	11.12	0.0	f	1991	05:16:04.282100	30
625	0_1038	2021-10-29	s_111864	c_1200	9.74	0.0	m	1979	16:37:30.563775	42
627	0_1632	2021-10-06	s_100936	c_887	9.21	0.0	m	1976	16:27:50.702968	45
644	0_1447	2021-10-09	s_102025	c_7856	4.99	0.0	f	1983	00:18:52.57433	38
655	0_1804	2021-10-31	s_112873	c_5797	16.95	0.0	m	1983	20:17:36.803427	38
656	1_392	2021-10-28	s_111152	c_8516	18.11	1.0	m	1967	03:12:19.943022	54
659	0_1586	2021-10-26	s_110553	c_4015	12.71	0.0	m	1986	20:18:16.744314	35
672	0_1412	2021-10-13	s_104314	c_7245	11.73	0.0	f	1983	19:36:20.548005	38
694	0_1877	2021-10-08	s_101944	c_2037	9.99	0.0	m	1971	18:57:08.279713	50

la categorie 1 semble presenter uniquement a certains jours les ventes de la categorie 1 semble sous-representees car il s'apparait que au debut du mois d'octobre (1. octobre) et en fin de mois (28/29/30/31) octobre le supprime donc le mois d'octobre car des données sont manquantes, affiche le graphique pour le mois d'octobre uniquement pour la cat 1 pour pouvoir augmenter

```
In [12]: #Je zoom sur les ventes en octobre pour pouvoir identifier des raisons pour cette baisse
startdate2 = pd.to_datetime('2021-10-01').date()
enddate2 = pd.to_datetime('2021-10-31').date()
ctc_sOct = ctc[ctc['date'] == startdate2 | (ctc['date'] > enddate2)]
ctc_sOct.describe(include = "all")
```

```
# != exclusion des dates qui ne contient pas 2021-10 str.contains 2021-10
```

```
Out[12]:
```

	id_produ	date	session_id	client_id	price	categ	sex	birth	time	age
count	315232	315232	315232	315232	315232.000000	315232	315232.000000	315232	315232.000000	
unique	3263	334	157648	8598	NaN	NaN	2	NaN	315232	NaN
top	1_369	2021-09-30	s_118668	c_1609	NaN	NaN	m	NaN	23:05:52.903346	NaN
freq	1066	1311	14	11839	NaN	NaN	NaN	158144	NaN	1
mean	NaN	NaN	NaN	NaN	17.379074	0.446823	NaN	1977.751983	NaN	43.248017
std	NaN	NaN	NaN	NaN	17.862571	0.592598	NaN	13.604129	NaN	13.604129
min	NaN	NaN	NaN	NaN	0.620000	0.000000	NaN	1929.000000	NaN	17.000000
25%	NaN	NaN	NaN	NaN	8.990000	0.000000	NaN	1970.000000	NaN	34.000000
50%	NaN	NaN	NaN	NaN	13.990000	0.000000	NaN	1980.000000	NaN	41.000000
75%	NaN	NaN	NaN	NaN	19.040000	1.000000	NaN	1987.000000	NaN	51.000000
max	NaN	NaN	NaN	NaN	300.000000	2.000000	NaN	2004.000000	NaN	92.000000

347322 (ctp) - 22298 (ctp_sOct) = 325024 (ctp_ssOct)

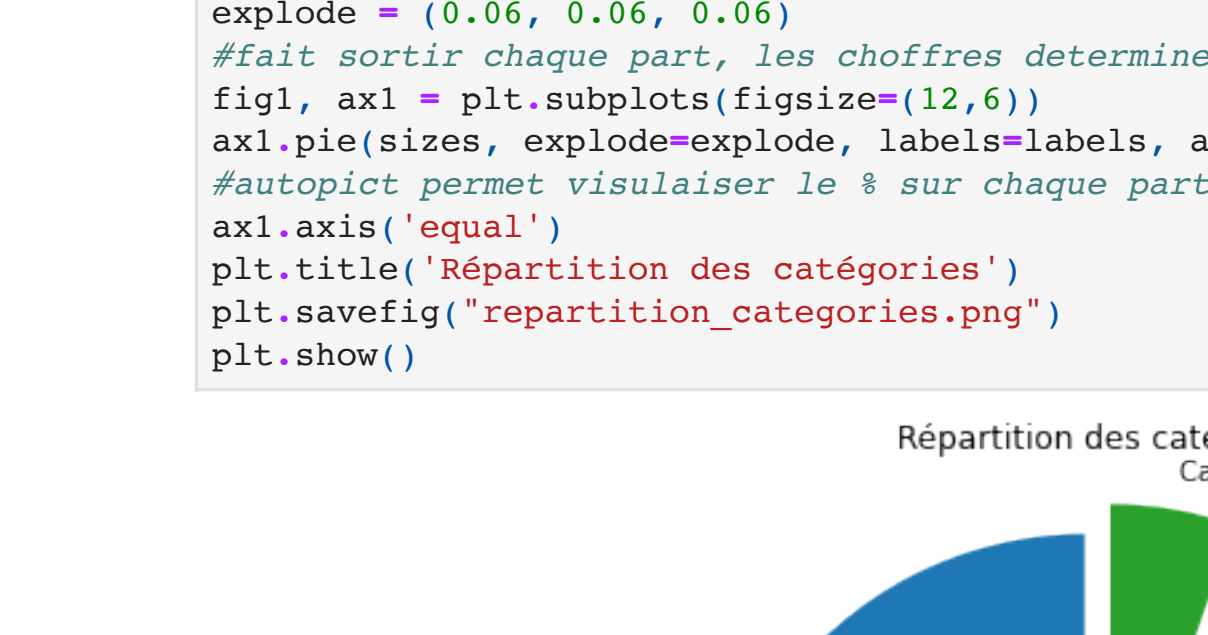
```
In [13]: #Agrégation des données transactionnelles par fréquence mensuelle (methode .groupby())
ctc_ssOct = ctc.groupby('date').sum().reset_index()
ctc_ssOct['ventes_keuros'] = ctc_ssOct['price'] / 1000
ctc_ssOct = ctc_ssOct[['date', 'ventes_keuros']]
ctc_ssOct.tail()
```

```
Out[13]:
```

	date	ventes_keuros
329	2022-02-24	20.20037
330	2022-02-25	18.20326
331	2022-02-26	19.75958
332	2022-02-27	19.02183
333	2022-02-28	18.71994

```
In [14]: #ctc_ssOct.plot(x='date', y='ventes_keuros')
ventes = ctc_ssOct['ventes_keuros']
temps = ctc_ssOct['date']
fig, ax = plt.subplots()
plt.plot(temps, ventes)
```

```
ax.set(xlabel='temps (mois)', ylabel='Ventes (K€)',
       title='Evolution des ventes')
ax.grid()
fig.savefig('test.png')
plt.show()
```



```
In [15]: #Représentation des effectifs par catégories de vente(methode .value_counts())
effectif = ctp_ssOct['categ'].value_counts() #value_counts compte le nombre de chaque categorie
tab = pd.DataFrame(effectif, columns = ['categ']) #crédation du tableau a partir des effectifs
tab['n'] = effectif.values
tab['% des ventes'] = round(tab['n'] / len(ctp),2) * 100 #determine le % de chaque categorie
```

```
Out[15]:
```

	categ	n	% des ventes
0.0	190771	190771	57.0
1.0	108069	108069	32.0
2.0	16392	16392	5.0

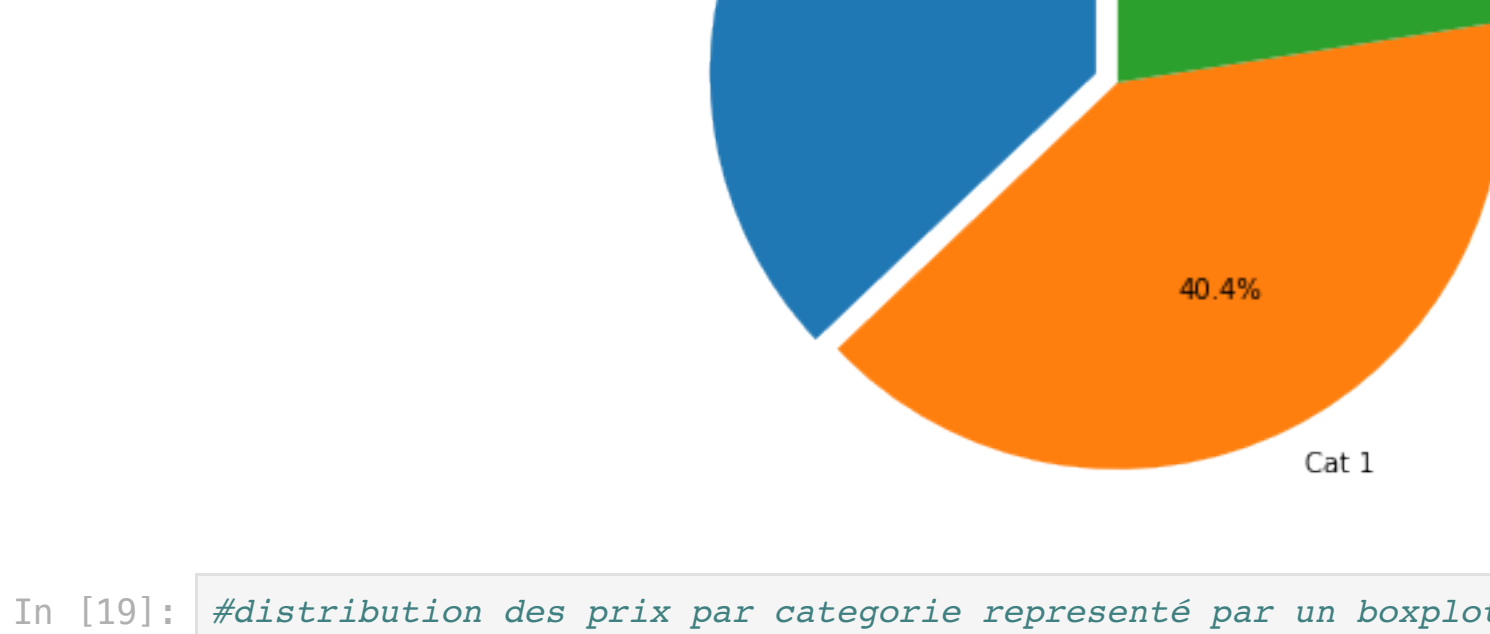
```
In [34]: #Pie Chart pour représenter la part de chacune des catégories de vente
```

```
labels = 'Cat 0', 'Cat 1', 'Cat 2'
sizes = tab['% des ventes']
explode = (0.05, 0.05, 0.05)
#fait sortir chaque part, les chiffres determine la distance pour chaque part, trois ici
ax1 = plt.subplot(figsize=(12,6))
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)
#autopct permet visualiser le % sur chaque part, startangle turn le graphique
ax1.axis('equal')
plt.title('Répartition catégories')
fig.savefig('repartition_categories.png')
plt.show()
```

```
In [17]: #help(plt.pie)
```

```
In [33]: #Je determine la contribution de chaque categorie au chiffre d'affaire par un pie chart
ctp_price = ctp_ssOct.groupby('date').sum()
```

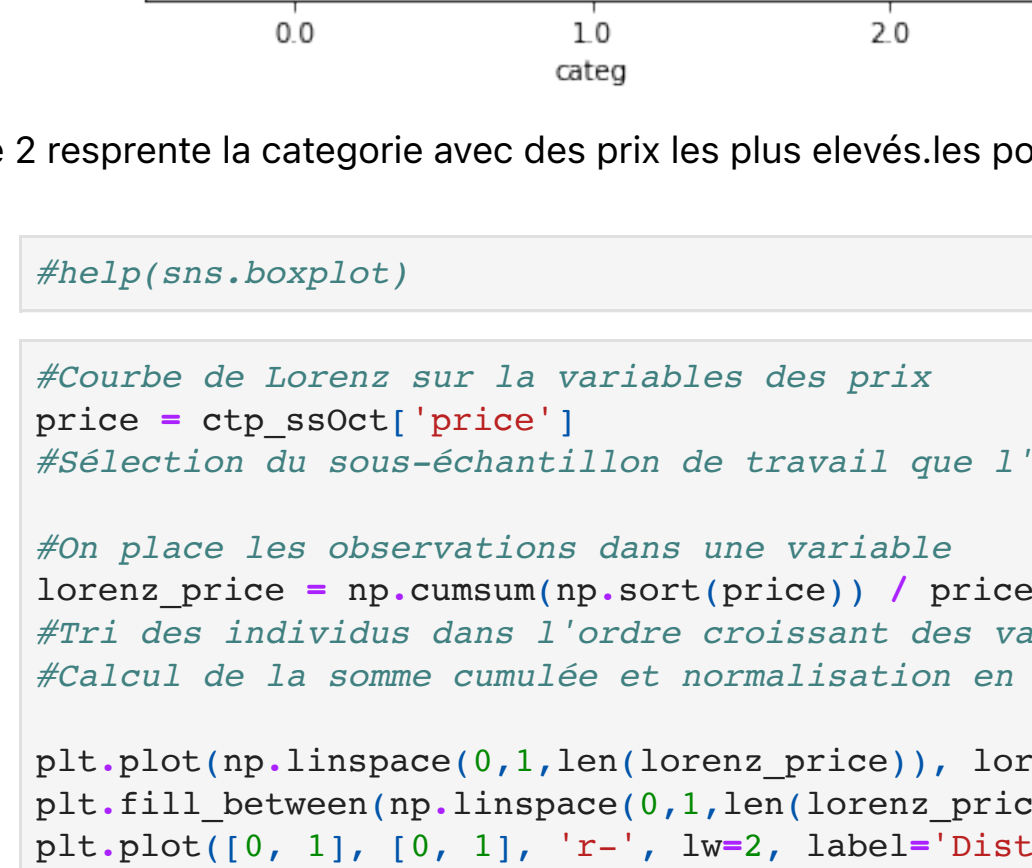
```
labels = 'Cat 0', 'Cat 1', 'Cat 2'
sizes = ctp_price['price']
explode = (0.05, 0, 0)
fig1, ax1 = plt.subplots(figsize=(12,6))
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)
ax1.axis('equal')
plt.title('Contribution des categories au CA')
fig.savefig('repartition_categories2.png')
plt.show()
```



```
In [19]: #Distribution des prix par categorie represente par un boxplot
sns.boxplot(x = ctp_ssOct['categ'], y = ctp_ssOct['price'])
```

```
Out[19]:
```

<AxesSubplot: xlabel='categ', ylabel='price'>



la categorie 2 represente la categorie avec des prix les plus élevés les points en dehors du boxplot sont les outliers. Chaque part de la boîte a moustache represente 25%. Elle indique également la mediane.

```
In [26]: #help(sns.boxplot)
```

```
In [21]: #Courbe de Lorenz sur la variables des prix
price = ctp_ssOct['price']
#Selection de sous-echantillon de travail que l'on appelle price
```

```
#On place les observations dans une variable
min_yilm, max_yilm = np.cumsum(np.sort(price)).sum()
#Tri des individus dans l'ordre croissant des valeurs de la variable,
#Calcul de la somme cumulée et normalisation en divisant par la somme des observations
plt.plot(np.linspace(0,1,len(price)), price, drawstyle='steps-post', color='rosybrown', label='Lorenz')
plt.fill_between(np.linspace(0,1,len(price)), Lorenz_price, Lorenz_price, color='rosybrown')
plt.plot(np.linspace(0,1,10), 10, 'r', lw=2, label='Distribution egalitaire')
plt.vlines(x=.76, ymin=0, ymax=5, color='blue', linestyle='--', linewidth=1, label='Medial')
plt.hlines(xmin=.76, xmax=0, y=5, color='blue', linestyle='--', linewidth=1)
plt.axis('equal')
```

```
plt.title('Courbe de Lorenz des prix de vente')
plt.xlabel('Distribution des ventes (x)')
plt.ylabel('Cumul des prix de ventes (y)')
plt.legend(loc='best')
```