In [1]:
```
#j'importe les libraries necessaires
import pandas as pd
import numpy as np
import matplotlib.pylab as plt  #pour les analyses des données et representation graphic
from datetime import datetime   #gestion des dates et des temps
import seaborn as sns           #pour les analyses des données et representation graphic
from scipy import stats as st   #pour les analyses statistics
```

Je telecharge les données pour faire les verifications habituelles: recherches des NAN ou valeurs manquants ou null Il y a 3 fichiers: customers, products et transactions

In [2]:
```
cust = pd.read_csv("customers.csv")
cust.head()
```

Out[2]:

	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 [3]:
```
cust.describe(include = "all")
```

Out[3]:

	client_id	sex	birth
count	8623	8623	8623.000000
unique	8623	2	NaN
top	c_325	f	NaN
freq	1	4491	NaN
mean	NaN	NaN	1978.280877
std	NaN	NaN	16.919535
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

In [4]:
```
cust.isna().any()
```

Out[4]:
```
client_id    False
sex          False
birth        False
dtype: bool
```

In [5]:
```
cust.isnull().any()
```

Out[5]:
```
client_id    False
sex          False
birth        False
dtype: bool
```

In [6]:
```
prod = pd.read_csv("products.csv")
prod.head()
```

Out[6]:

	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 [7]:
```
prod.describe(include = "all")
#describe permet une analyse de statitstique basic. include "all" permet de inclure toutes les colonnes,
#des variables quantitatifs comme  qualitatifs en meme temps. Sinon on peut voir que les colonnes quantitatifs.
```

Out[7]:

	id_prod	price	categ
count	3287	3287.000000	3287.000000
unique	3287	NaN	NaN
top	2_125	NaN	NaN
freq	1	NaN	NaN
mean	NaN	21.856641	0.370246
std	NaN	29.847908	0.615387
min	NaN	-1.000000	0.000000
25%	NaN	6.990000	0.000000
50%	NaN	13.060000	0.000000
75%	NaN	22.990000	1.000000
max	NaN	300.000000	2.000000

il ya des prix negatifs

In [8]:
```
# je visualize les valeurs < 0
prod_neg = prod[prod["price"] < 0]
prod_neg.head()
```

Out[8]:

	id_prod	price	categ
731	T_0	-1.0	0

In [9]:
```
prod_neg.shape #shape permet de voir la taille du df
```

Out[9]:
```
(1, 3)
```

In [10]:
```
prod.isna().any() #detection des Nan
```

Out[10]:
```
id_prod    False
price      False
categ      False
dtype: bool
```

In [11]:
```
prod.isnull().any() #detection des valeurs 0
```

Out[11]:
```
id_prod    False
price      False
categ      False
dtype: bool
```

In [12]:
```
transac = pd.read_csv("transactions.csv")
transac.head()
```

Out[12]:

	id_prod	date	session_id	client_id
0	0_1483	2021-04-10 18:37:28.723910	s_18746	c_4450
1	2_226	2022-02-03 01:55:53.276402	s_159142	c_277
2	1_374	2021-09-23 15:13:46.938559	s_94290	c_4270
3	0_2186	2021-10-17 03:27:18.783634	s_105936	c_4597
4	0_1351	2021-07-17 20:34:25.800563	s_63642	c_1242

In [13]:
```
transac.describe(include = "all")
```

Out[13]:

	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

il y a un format de date test, donc inhabituel

In [14]:
```
transac.isna().any()
```

Out[14]:
```
id_prod       False
date          False
session_id    False
client_id     False
dtype: bool
```

In [15]:
```
transac.isnull().any()
```

Out[15]:
```
id_prod       False
date          False
session_id    False
client_id     False
dtype: bool
```

In [16]:
```
#je visualise les dates "test"
transac_test = transac[transac['date'].str.contains("test")]
transac_test.head(40)
```

Out[16]:

	id_prod	date	session_id	client_id
1431	T_0	test_2021-03-01 02:30:02.237420	s_0	ct_1
2365	T_0	test_2021-03-01 02:30:02.237446	s_0	ct_1
2895	T_0	test_2021-03-01 02:30:02.237414	s_0	ct_1
5955	T_0	test_2021-03-01 02:30:02.237441	s_0	ct_0
7283	T_0	test_2021-03-01 02:30:02.237434	s_0	ct_1
13745	T_0	test_2021-03-01 02:30:02.237443	s_0	ct_0
20470	T_0	test_2021-03-01 02:30:02.237442	s_0	ct_0
22347	T_0	test_2021-03-01 02:30:02.237412	s_0	ct_1
26359	T_0	test_2021-03-01 02:30:02.237439	s_0	ct_1
26407	T_0	test_2021-03-01 02:30:02.237426	s_0	ct_0
26814	T_0	test_2021-03-01 02:30:02.237447	s_0	ct_1
27161	T_0	test_2021-03-01 02:30:02.237434	s_0	ct_0
32675	T_0	test_2021-03-01 02:30:02.237438	s_0	ct_0
34387	T_0	test_2021-03-01 02:30:02.237443	s_0	ct_0
36700	T_0	test_2021-03-01 02:30:02.237437	s_0	ct_0
38779	T_0	test_2021-03-01 02:30:02.237450	s_0	ct_0
39224	T_0	test_2021-03-01 02:30:02.237429	s_0	ct_0
40476	T_0	test_2021-03-01 02:30:02.237435	s_0	ct_0
47170	T_0	test_2021-03-01 02:30:02.237430	s_0	ct_1
48425	T_0	test_2021-03-01 02:30:02.237443	s_0	ct_1
54813	T_0	test_2021-03-01 02:30:02.237412	s_0	ct_1
56373	T_0	test_2021-03-01 02:30:02.237446	s_0	ct_0
57261	T_0	test_2021-03-01 02:30:02.237439	s_0	ct_1
57682	T_0	test_2021-03-01 02:30:02.237419	s_0	ct_0
57755	T_0	test_2021-03-01 02:30:02.237413	s_0	ct_1
58802	T_0	test_2021-03-01 02:30:02.237429	s_0	ct_0
59043	T_0	test_2021-03-01 02:30:02.237413	s_0	ct_0
59422	T_0	test_2021-03-01 02:30:02.237445	s_0	ct_0
60170	T_0	test_2021-03-01 02:30:02.237446	s_0	ct_0
60485	T_0	test_2021-03-01 02:30:02.237438	s_0	ct_1
62365	T_0	test_2021-03-01 02:30:02.237412	s_0	ct_1
70341	T_0	test_2021-03-01 02:30:02.237424	s_0	ct_1
71097	T_0	test_2021-03-01 02:30:02.237446	s_0	ct_0
72068	T_0	test_2021-03-01 02:30:02.237427	s_0	ct_0
72102	T_0	test_2021-03-01 02:30:02.237428	s_0	ct_0
72405	T_0	test_2021-03-01 02:30:02.237424	s_0	ct_1
72850	T_0	test_2021-03-01 02:30:02.237433	s_0	ct_1
75176	T_0	test_2021-03-01 02:30:02.237443	s_0	ct_1
76735	T_0	test_2021-03-01 02:30:02.237442	s_0	ct_0
77758	T_0	test_2021-03-01 02:30:02.237429	s_0	ct_1

In [17]:
```
transac_test.id_prod.unique()#unique permet de visualiser les differents objets dans une colonne
```

Out[17]:
```
array(['T_0'], dtype=object)
```

In [18]:
```
transac_test.session_id.unique()
```

Out[18]:
```
array(['s_0'], dtype=object)
```

In [19]:
```
transac_test.client_id.unique()
```

Out[19]:
```
array(['ct_1', 'ct_0'], dtype=object)
```

Resume: - Dans les 3 df pas de valeurs manquants, ou NaN, ni 0. - Dans la df products il y une ligne de prix negatif qui correspond au id_prod T_0 et categ = 0. il faut exclure les prix < 0 - Dans le df transactions il y a un format de date inhabituel contenant le prefix 'test'. Il y a 200 lignes. Il est associé avec id_prod = T_0 et session_id = s_0 et client_id = ct_O/ct_1. Le id_prod T_ 0 est associé uvec un prix = -1 dans la table products. J'exclus donc.

In [20]:
```
#j'exclus le prix = -1 dans products
prod1 = prod[prod["price"] != -1]
prod1.describe(include = "all")
```

Out[20]:

	id_prod	price	categ
count	3286	3286.000000	3286.000000
unique	3286	NaN	NaN
top	2_125	NaN	NaN
freq	1	NaN	NaN
mean	NaN	21.863597	0.370359
std	NaN	29.849786	0.615446
min	NaN	0.620000	0.000000
25%	NaN	6.990000	0.000000
50%	NaN	13.075000	0.000000
75%	NaN	22.990000	1.000000
max	NaN	300.000000	2.000000

In [21]:
```
#j'exclus les lignes avec les dates test...
transT = transac[transac["id_prod"] != "T_0"]
transT.describe(include = "all")
```

Out[21]:

	id_prod	date	session_id	client_id
count	336816	336816	336816	336816
unique	3265	336816	169194	8600
top	1_369	2021-03-01 04:48:32.880371	s_118668	c_1609
freq	1081	1	14	12855

In [22]:
```
#je verfie si il reste des lignes avec une date = 'test..'
transT[transT['date'].str.contains("test")]
```

Out[22]:

id_prod	date	session_id	client_id

Exactement 200 lignes correspandant aux lignes date = 'test...' sont exclus dans la table transT

In [23]:
```
#Je procede au merge des tables products et transactions sur id_prod
tp = pd.merge(transT, prod1, on = 'id_prod', how = 'left')
tp.head()
```

Out[23]:

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

In [24]:
```
tp.describe(include = "all")
```

Out[24]:

	id_prod	date	session_id	client_id	price	categ
count	336816	336816	336816	336816	336713.000000	336713.000000
unique	3265	336816	169194	8600	NaN	NaN
top	1_369	2021-03-01 04:48:32.880371	s_118668	c_1609	NaN	NaN
freq	1081	1	14	12855	NaN	NaN
mean	NaN	NaN	NaN	NaN	17.215189	0.430156
std	NaN	NaN	NaN	NaN	17.855445	0.591082
min	NaN	NaN	NaN	NaN	0.620000	0.000000
25%	NaN	NaN	NaN	NaN	8.610000	0.000000
50%	NaN	NaN	NaN	NaN	13.900000	0.000000
75%	NaN	NaN	NaN	NaN	18.990000	1.000000
max	NaN	NaN	NaN	NaN	300.000000	2.000000

Absence de lignes test ou de prix = -1

In [25]:
```
tp.isnull().any()
```

Out[25]:
```
id_prod       False
date          False
session_id    False
client_id     False
price          True
categ          True
dtype: bool
```

In [26]:
```
tp.isna().any()
```

Out[26]:
```
id_prod       False
date          False
session_id    False
client_id     False
price          True
categ          True
dtype: bool
```

In [27]:
```
#Il y a des NaN dans price et categ. Je les visualize.
tp_isna = tp[tp.price.isna() == True]
tp_isna.head()
```

Out[27]:

	id_prod	date	session_id	client_id	price	categ
6231	0_2245	2021-06-17 03:03:12.668129	s_49705	c_1533	NaN	NaN
10797	0_2245	2021-06-16 05:53:01.627491	s_49323	c_7954	NaN	NaN
14045	0_2245	2021-11-24 17:35:59.911427	s_124474	c_5120	NaN	NaN
17480	0_2245	2022-02-28 18:08:49.875709	s_172304	c_4964	NaN	NaN
21071	0_2245	2021-03-01 00:09:29.301897	s_3	c_580	NaN	NaN

In [28]:
```
tp_isna.describe(include = "all")
```

Out[28]:

	id_prod	date	session_id	client_id	price	categ
count	103	103	103	103	0.0	0.0
unique	1	103	103	100	NaN	NaN
top	0_2245	2022-01-15 06:19:44.035000	s_167890	c_1025	NaN	NaN
freq	103	1	1	2	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

In [29]:
```
tp_isna.id_prod.unique()
```

Out[29]:
```
array(['0_2245'], dtype=object)
```

In [30]:
```
tp_isnull = tp[tp.price.isnull() == True]
tp_isnull.head()
```

Out[30]:

	id_prod	date	session_id	client_id	price	categ
6231	0_2245	2021-06-17 03:03:12.668129	s_49705	c_1533	NaN	NaN
10797	0_2245	2021-06-16 05:53:01.627491	s_49323	c_7954	NaN	NaN
14045	0_2245	2021-11-24 17:35:59.911427	s_124474	c_5120	NaN	NaN
17480	0_2245	2022-02-28 18:08:49.875709	s_172304	c_4964	NaN	NaN
21071	0_2245	2021-03-01 00:09:29.301897	s_3	c_580	NaN	NaN

In [31]:
```
tp_isnull.describe(include = "all")
```

Out[31]:

	id_prod	date	session_id	client_id	price	categ
count	103	103	103	103	0.0	0.0
unique	1	103	103	100	NaN	NaN
top	0_2245	2022-01-15 06:19:44.035000	s_167890	c_1025	NaN	NaN
freq	103	1	1	2	NaN	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

In [32]:
```
tp_isnull.id_prod.unique()
```

Out[32]:
```
array(['0_2245'], dtype=object)
```

Resumé: - les prix et categ manquants sont associés exclusivement avec le id_prod = 0_2245 - le id_prod = 0_2245 appartient a la categorie 0 a cause du prefix 0_, j'attribue donc 0 pour la categ - j'attribue le prix moyenne de la categorie 0 comme prix pour le id_prod = 0_2245

In [ ]:
```

```

In [33]:
```
#Je determine le prix moyen pour la categorie 0 afin de l'attribuer au id_prod = 0_2245
tp_cat0 = tp[tp["categ"] == 0.0]
tp_cat0.head()
```

Out[33]:

	id_prod	date	session_id	client_id	price	categ
0	0_1483	2021-04-10 18:37:28.723910	s_18746	c_4450	4.99	0.0
3	0_2186	2021-10-17 03:27:18.783634	s_105936	c_4597	4.20	0.0
4	0_1351	2021-07-17 20:34:25.800563	s_63642	c_1242	8.99	0.0
5	0_1085	2021-09-15 05:47:48.215162	s_90139	c_2526	3.99	0.0
6	0_1508	2021-08-29 05:39:01.055455	s_82100	c_5799	8.03	0.0

In [34]:
```
price_mean_cat0 = tp_cat0["price"].mean()
price_mean_cat0
```

Out[34]:
```
10.646828235274288
```

In [35]:
```
#J'ajoute le id_prod = 0_2245 a la table initiale products' en utilisant la method 'append'.
prod_02245 = prod1.append(pd.DataFrame([["0_2245", 10.65, 0.0]],columns=['id_prod', 'price', 'categ']))
prod_02245.describe(include = "all")
```

Out[35]:

	id_prod	price	categ
count	3287	3287.000000	3287.000000
unique	3287	NaN	NaN
top	2_125	NaN	NaN
freq	1	NaN	NaN
mean	NaN	21.860186	0.370246
std	NaN	29.845884	0.615387
min	NaN	0.620000	0.000000
25%	NaN	6.990000	0.000000
50%	NaN	13.060000	0.000000
75%	NaN	22.990000	1.000000
max	NaN	300.000000	2.000000

In [36]:
```
#Je verifie si la ligne etait bien ajouté
prod_02245_T = prod_02245[prod_02245["id_prod"] == "0_2245"]
prod_02245_T
```

Out[36]:

	id_prod	price	categ
0	0_2245	10.65	0.0

Comme indiqué dans la ligne top id_prod etait bien ajouté au df products et la table products est passée de 3286 ligne a 3287

In [37]:
```
#Je procede au merge des tables products ou prodT(contenant le id_prod 0_2245) et transactions sur id_prod
tp2 = pd.merge(transT, prod_02245, on = 'id_prod', how = 'left')
tp2.head()
```

Out[37]:

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

In [38]:
```
tp2.describe(include = 'all')
```

Out[38]:

	id_prod	date	session_id	client_id	price	categ
count	336816	336816	336816	336816	336816.000000	336816.000000
unique	3265	336816	169194	8600	NaN	NaN
top	1_369	2021-03-01 04:48:32.880371	s_118668	c_1609	NaN	NaN
freq	1081	1	14	12855	NaN	NaN
mean	NaN	NaN	NaN	NaN	17.213181	0.430024
std	NaN	NaN	NaN	NaN	17.853084	0.591039
min	NaN	NaN	NaN	NaN	0.620000	0.000000
25%	NaN	NaN	NaN	NaN	8.610000	0.000000
50%	NaN	NaN	NaN	NaN	13.900000	0.000000
75%	NaN	NaN	NaN	NaN	18.990000	1.000000
max	NaN	NaN	NaN	NaN	300.000000	2.000000

In [39]:
```
tp2.isnull().any()
```

Out[39]:
```
id_prod       False
date          False
session_id    False
client_id     False
price         False
categ         False
dtype: bool
```

In [40]:
```
tp2.isna().any()
```

Out[40]:
```
id_prod       False
date          False
session_id    False
client_id     False
price         False
categ         False
dtype: bool
```

pas de données manquantes dans tp2. Je procede au merge avec le df customers Pour rapelle le df transactions contenait une date test associé avec le client_id ct_0/ct_1 Avant le merge je verifie si customers contient ces client_id. si oui je vais les supprimers avant le merge.

In [41]:
```
#ctp_evol10 =  ctp[(ctp.date > startdate)  & (ctp.date < enddate)]
cust_ct01 = cust[(cust.client_id == "ct_0") | (cust.client_id == "ct_1")]
cust_ct01.head()
```

Out[41]:

	client_id	sex	birth
2735	ct_0	f	2001
8494	ct_1	m	2001

In [42]:
```
cust_ct01.shape
```

Out[42]:
```
(2, 3)
```

la table customers contient bien deux lignes de client_id ct0 et c1. il faut les eclure car ces client_id sont exclus dans les autres tables

In [43]:
```
custT = cust[(cust["client_id"] != "ct_0") & (cust["client_id"] != "ct_1") ]
custT.shape
```

Out[43]:
```
(8621, 3)
```

In [44]:
```
cust.shape
```

Out[44]:
```
(8623, 3)
```

In [45]:
```
#je verifie si ct0/1 etaient bien exclus
custT_ct01 = custT[(custT.client_id == "ct_0") | (custT.client_id == "ct_1")]
custT_ct01
```

Out[45]:

client_id	sex	birth

Les client_id ct0/1 n'apparaissent plus dans la table customers

In [46]:
```
#je procede au merge de la table tp avec la table custT(sans ct0/1)

ctp = pd.merge(tp2, custT, on='client_id', how='left')
ctp.head()
```

Out[46]:

	id_prod	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 [47]:
```
ctp.describe(include = "all")
```

Out[47]:

	id_prod	date	session_id	client_id	price	categ	sex	birth
count	336816	336816	336816	336816	336816.000000	336816.000000	336816	336816.000000
unique	3265	336816	169194	8600	NaN	NaN	2	NaN
top	1_369	2021-03-01 04:48:32.880371	s_118668	c_1609	NaN	NaN	m	NaN
freq	1081	1	14	12855	NaN	NaN	169241	NaN
mean	NaN	NaN	NaN	NaN	17.213181	0.430024	NaN	1977.823396
std	NaN	NaN	NaN	NaN	17.853084	0.591039	NaN	13.523923
min	NaN	NaN	NaN	NaN	0.620000	0.000000	NaN	1929.000000
25%	NaN	NaN	NaN	NaN	8.610000	0.000000	NaN	1971.000000
50%	NaN	NaN	NaN	NaN	13.900000	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 [48]:
```
ctp.isnull().any()
```

Out[48]:
```
id_prod       False
date          False
session_id    False
client_id     False
price         False
categ         False
sex           False
birth         False
dtype: bool
```

In [49]:
```
ctp.isna().any()
```

Out[49]:
```
id_prod       False
date          False
session_id    False
client_id     False
price         False
categ         False
sex           False
birth         False
dtype: bool
```

la nouvelle table ne contient plus des valeurs manquantes.

In [50]:
```
#exportation des données du bilans alimentaires hors sousnutrition
ctp.to_csv('ctp_partie2.csv', index=False)
```

In [ ]:
```

```