ANALYSE DU STOCK ET DES VENTES DU SITE BOTTLENECK

OBJECTIF DE CE NOTEBOOK

Bienvenue dans l'outil plébiscité par les analystes de données Jupyter.

Il s'agit d'un outil permettant de mixer et d'alterner codes, textes et graphique.

Cet outil est formidable pour plusieurs raisons:

- il permet de tester des lignes de codes au fur et à mesure de votre rédaction, de constater immédiatement le résultat d'un instruction, de la corriger si nécessaire.
- De rédiger du texte pour expliquer l'approche suivie ou les résultats d'une analyse et de le mettre en forme grâce à du code html ou plus simple avec **Markdown**
- d'agrémenter de graphiques

Pour vous aider dans vos premiers pas à l'usage de Jupyter et de Python, nous avons rédigé ce notebook en vous indiquant les instructions à suivre.

Il vous suffit pour cela de saisir le code Python répondant à l'instruction donnée.

Vous verrez de temps à autre le code Python répondant à une instruction donnée mais cela est fait pour vous aider à comprendre la nature du travail qui vous est demandée.

Et garder à l'esprit, qu'il n'y a pas de solution unique pour résoudre un problème et qu'il y a autant de résolutions de problèmes que de développeurs ;)...

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

Entrée [1]:

```
#Importation de la librairie Pandas
import pandas as pd
```

Entrée [2]:

```
import numpy as np
```

Entrée [3]:

```
import matplotlib.pyplot as plt
```

Entrée [4]:

```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\consultant\anaconda3\lib\s
ite-packages (5.15.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\consultant\anacon
da3\lib\site-packages (from plotly) (8.2.2)
Requirement already satisfied: packaging in c:\users\consultant\anaconda3\li
b\site-packages (from plotly) (19.2)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\consultant\anaco
nda3\lib\site-packages (from packaging->plotly) (2.4.2)
Requirement already satisfied: six in c:\users\consultant\anaconda3\lib\site
-packages (from packaging->plotly) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

Entrée [5]:

```
#Importation de la librairie plotly express
import plotly.express as px
```

1.2 - Chargements des fichiers

Entrée [6]:

```
#Importation du fichier web.xlsx
df_web = pd.read_excel("web.xlsx")
#Importation du fichier erp.xlsx

#importation du fichier liaison.xlsx
```

Entrée [7]:

```
df_erp=pd.read_excel("erp.xlsx")
df_erp.head()
```

Out[7]:

	product_id	onsale_web	price	stock_quantity	stock_status
0	3847	1	24.2	0	outofstock
1	3849	1	34.3	0	outofstock
2	3850	1	20.8	0	outofstock
3	4032	1	14.1	0	outofstock
4	4039	1	46.0	0	outofstock

Entrée [8]:

```
df_web=pd.read_excel("web.xlsx")
df_web.head()
```

Out[8]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	tax_clas
0	bon-cadeau-25-euros	0	0	0	0.0	10.0	taxable	Na
1	15298	0	0	0	0.0	6.0	taxable	Na
2	15296	0	0	0	0.0	0.0	taxable	Na
3	15300	0	0	0	0.0	0.0	taxable	Na
4	19814	0	0	0	0.0	3.0	taxable	Na

5 rows × 28 columns

Entrée [9]:

```
df_liaison=pd.read_excel("liaison.xlsx")
df_liaison.head()
```

Out[9]:

	product_id	id_web
0	3847	15298
1	3849	15296
2	3850	15300
3	4032	19814
4	4039	19815

1.2 - Chargements du fichiers csv

Information Mentor

L'étudiant va être très certainement confronté à des difficultés pour ouvrir ce fichier. Il faudra donc qu'il se documente sur les points suivants:
- notion d'encodage en informatique. ("UTF-8", "ISO-8859-1", etc)
- manipulation de fichier csv, tsv, etc

Pour lui faire réussir cette étape, l'étudiant devrait suivre ces directives:
- Explorer le fichier à l'aide d'un outil comme Notepad++ afin de visualiser la structure du fichier
- ... ou d'utiliser Excel pour les mêmes raisons. Mais dans ce cas l'étudiant pourrait être tenté de convertir le fichier csv en xlsx, ce que je ne recommande pas: si le fichier est trop gros pour Excel cela deviendra une souffrance de procéder ainsi
- apprendre à ourvrir un fichier avec python à l'aide de l'instruction suivante: `with open(nom_fichier, "rb") as f:`
- utiliser une bibliothèque comme `chardet` pour tenter d'identifier l'encodage du fichier

```
#Avant d'utiliser pandas pour le chargement du fichier, explorons la structure du fichier
#Pour cela utilisons l'instruction ci-dessous qui permet d'ouvrir un fichier texte
with open("caracteristiques_vins.csv", "rb") as f:
    file = f.read()

#Et affichons le:
print(file)
```

Entrée [10]:

```
import chardet as chr
```

Entrée [11]:

```
with open("caracteristiques_vins.csv", "rb") as f:
    file = f.read()
print(file)
```

```
b"post_name;poids;R\xe9gion;Domaine;Appellation;Couleur;C\xe9page;Mill\xe9
sime;Garde;Contenance;Degr\xe9 d'alcool;Temp\xe9rature d\xe9gustation;Alli
ance mets\r\npierre-jean-villa-saint-joseph-preface-2018;1.5 kg;Rh\xf4ne;P
ierre Jean Villa;Saint Joseph;Rouge;100% Syrah;2020;4-7 ans;75cl;13%;15\xb
0C;Charcuterie, Lapin, Viande rouge, Volaille\r\npierre-jean-villa-saint-j
oseph-tilde-2017;1.5 kg;Rh\xf4ne;Pierre Jean Villa;Saint Joseph;Rouge;100%
Syrah;2019;6-8 ans;75cl;13%;15\xb0C;Charcuterie, Viande rouge, Volaille\r
\npierre-jean-villa-croze-hermitage-accroche-coeur-2018;1.5 kg;Rh\xf4ne;Pi
erre Jean Villa;Crozes-Hermitage;Rouge;100% Syrah;2020;3-5 ans;75cl;13%;15
\xb0C;Viande rouge, Volaille\r\npierre-jean-villa-igp-gamine-2018;1.5 kg;R
h\xf4ne;Pierre Jean Villa;Collines Rhodaniennes;Rouge;100% Syrah;2020;3-5
ans;75cl;13%;14\xb0C;Charcuterie, Viande rouge, Volaille\r\npierre-jean-vi
lla-cote-rotie-carmina-2017;1.5 kg;Rh\xf4ne;Pierre Jean Villa;C\xf4te R\xf
4tie;Rouge;100% Syrah;2019;10-20 ans;75cl;13%;17\xb0C;Gibier, Viande rouge
\r\npierre-jean-villa-saint-joseph-saut-ange-2018;1.5 kg;Rh\xf4ne;Pierre J
ean Villa;Saint Joseph;Blanc;100% Roussanne;2020;6-8 ans;75cl;14%;12\xb0C;
Ap\xe9ritif, Fromages, Poisson en sauce, Volaille\r\npierre-gaillard-condr
ieu-2018;1.5 kg;Rh\xf4ne;Pierre Gaillard;Condrieu;Blanc;100% Viognier;202
0;3-5 ans;75cl;13%;12\xb0C;Ap\xe9ritif, Crustac\xe9s, Cuisine Asiatique, F
```

Entrée [12]:

```
df_car_vins=pd.read_csv("caracteristiques_vins.csv",sep=";",encoding="windows-1252")
df_car_vins.head()
```

Out[12]:

	post_name	poids	Région	Domaine	Appellation	Couleur	Cépage	Millésime	Garde	Cou
0	pierre-jean-villa-saint-joseph-preface-2018	1.5 kg	Rhône	Pierre Jean Villa	Saint Joseph	Rouge	100% Syrah	2020.0	4-7 ans	
1	pierre-jean-villa-saint-joseph-tilde-2017	1.5 kg	Rhône	Pierre Jean Villa	Saint Joseph	Rouge	100% Syrah	2019.0	6-8 ans	
2	pierre-jean-villa-croze-hermitage-accroche-coe...	1.5 kg	Rhône	Pierre Jean Villa	Crozes-Hermitage	Rouge	100% Syrah	2020.0	3-5 ans	
3	pierre-jean-villa-igp-gamine-2018	1.5 kg	Rhône	Pierre Jean Villa	Collines Rhodaniennes	Rouge	100% Syrah	2020.0	3-5 ans	
4	pierre-jean-villa-cote-rotie-carmina-2017	1.5 kg	Rhône	Pierre Jean Villa	Côte Rôtie	Rouge	100% Syrah	2019.0	10-20 ans	

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier erp.xlsx

Entrée [13]:

```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_erp.shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_erp.shape[1]))

print(df_erp.shape)
```

```
Le tableau comporte 825 observation(s) ou article(s)
Le tableau comporte 5 colonne(s)
(825, 5)
```

Entrée [14]:

```
#Consulter le nombre de colonnes
print(len(df_erp.columns))
```

```
5
```

Entrée [15]:

```
#La nature des données dans chacune des colonnes
print(df_erp.dtypes)
```

```
product_id          int64
onsale_web          int64
price             float64
stock_quantity      int64
stock_status       object
dtype: object
```

Entrée [16]:

```
#Le nombre de valeurs présentes dans chacune des colonnes
print(df_erp.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 5 columns):
product_id        825 non-null int64
onsale_web        825 non-null int64
price             825 non-null float64
stock_quantity    825 non-null int64
stock_status      825 non-null object
dtypes: float64(1), int64(3), object(1)
memory usage: 29.1+ KB
None
```

Entrée [17]:

```
#Afficher les 5 premières lignes de la table
df_erp.head()
```

Out[17]:

	product_id	onsale_web	price	stock_quantity	stock_status
0	3847	1	24.2	0	outofstock
1	3849	1	34.3	0	outofstock
2	3850	1	20.8	0	outofstock
3	4032	1	14.1	0	outofstock
4	4039	1	46.0	0	outofstock

Entrée [18]:

```
#Vérifier si il y a les lignes en doublons dans la colonne product_id
df_erp.duplicated('product_id').sum()
```

Out[18]:

```
0
```

Entrée [19]:

```
#Afficher les valeurs distinctes de la colonne stock_status
print(df_erp.stock_status.unique())
```

```
['outofstock' 'instock']
```

Entrée [20]:

```
#À quelle(s) autre(s) colonne(s) sont-elles liées ?
#les valeurs de la colonne stock_status liées avec le colonne stock_quantity, quand on a st
```

Entrée [21]:

```
#Création d'une colonne "stock_status_2
#La valeur de cette deuxième colonne sera fonction de la valeur dans la colonne "stock_quan
#si la valeur de la colonne "stock_quantity" est nulle renseigner "outofstock" sinon mettre
df_erp['stock_status_2']=np.where(df_erp['stock_quantity']==0,'outofstock','instock')
```

Entrée [22]:

```
#vérification
df_erp.head()
```

Out[22]:

	product_id	onsale_web	price	stock_quantity	stock_status	stock_status_2
0	3847	1	24.2	0	outofstock	outofstock
1	3849	1	34.3	0	outofstock	outofstock
2	3850	1	20.8	0	outofstock	outofstock
3	4032	1	14.1	0	outofstock	outofstock
4	4039	1	46.0	0	outofstock	outofstock

Entrée [23]:

```
#Vérifions que les 2 colonnes sont identiques:
#Les 2 colonnes sont strictement identiques si les valeurs de chaque ligne sont strictement
#La comparaison de 2 colonnes peut se réaliser simplement avec l'instruction ci-dessous:
df_erp["stock_status"] == df_erp["stock_status_2"]

#Le résultat est l'affichage de True ou False pour chacune des lignes du dataset
#C'est un bon début, mais difficile à exploiter
```

Out[23]:

```
0      True
1      True
2      True
3      True
4      True
       ...
820    True
821    True
822    True
823    True
824    True
Length: 825, dtype: bool
```

Entrée [24]:

```
#Mais il est possible de synthétiser ce résultat en effectuant la somme de cette colonne:
#True vaut 1 et False 0
#Nous devrions obtenir la somme de 824 qui correspond au nombre de lignes dans ce dataset

(df_erp["stock_status"] == df_erp["stock_status_2"]).sum()
```

Out[24]:

```
824
```

Entrée [25]:

```
#Si les colonnes ne sont absolument pas identiques ligne à ligne alors identifier la ligne
##Dans ce cas je vous ce lien pour apprendre à réaliser des filtres dans Pandas:
##https://bitbucket.org/hrojas/learn-pandas/src/master/
##Lesson 3

df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
```

Out[25]:

	product_id	onsale_web	price	stock_quantity	stock_status	stock_status_2
443	4954	1	25.0	0	instock	outofstock

Entrée [26]:

```
#Corriger la ou les données incohérentes

df_erp.loc[443,"stock_status"]="outofstock"
```

Entrée [27]:

```
df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
```

Out[27]:

	product_id	onsale_web	price	stock_quantity	stock_status	stock_status_2

2.1.1 - Analyse exploratoire de chaque variable du fichier erp.xlsx

2.1.1.1 - Analyse de la variable PRIX

Entrée [28]:

```
################
## LES PRIX  ##
################

#Vérification des prix: Y a t-il des prix non renseignés, négatif ou nul?
#Afficher le ou les prix non renseignés dans la colonne "price"
#Afficher le prix minimum de la colonne "price"
#Afficher le prix maximum de la colonne "price"
```

Entrée [29]:

```
(df_erp['price'].isnull()).sum()
```

Out[29]:

```
0
```

Entrée [30]:

```
df_erp['price'].min()
```

Out[30]:

```
5.2
```

Entrée [31]:

```
df_erp['price'].max()
```

Out[31]:

```
225.0
```

2.1.1.2 - Analyse de la variable STOCK

Entrée [32]:

```
#######################
### stock_quantity  ###
#######################

#Vérification de la colonne stock quantity
#Afficher la quantité minimum de la colonne "stock_quantity"

#Afficher la quantité maximum de la colonne "stock_quantity"
```

Entrée [33]:

```
(df_erp['stock_quantity'].isnull()).sum()
```

Out[33]:

```
0
```

Entrée [34]:

```
df_erp['stock_quantity'].min()
```

Out[34]:

```
0
```

Entrée [35]:

```
df_erp['stock_quantity'].max()
```

Out[35]:

```
578
```

2.1.1.3 - Analyse de la variable ONSALE_WEB

Entrée [36]:

```
#Vérification de la colonne onsale_web et des valeurs qu'elle contient? Que signifient-elle
(df_erp['onsale_web'].isnull()).sum()
```

Out[36]:

```
0
```

Entrée [37]:

```
df_erp['onsale_web'].unique()
```

Out[37]:

```
array([1, 0], dtype=int64)
```

Entrée [38]:

```
#Quelles sont les colonnes à conserver selon vous?
#les colonnes à conserves selon moi:product_id,onsale_web,price,stock_quantity
```

Entrée [39]:

```
df_erp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 6 columns):
product_id        825 non-null int64
onsale_web        825 non-null int64
price             825 non-null float64
stock_quantity    825 non-null int64
stock_status      825 non-null object
stock_status_2    825 non-null object
dtypes: float64(1), int64(3), object(2)
memory usage: 32.3+ KB
```

Entrée [40]:

```
#Supprimer les colonnes comportant le libellé "stock_status"
#Cette colonne est redondante avec la colonne "stock_quantity". Dans notre projet cette inf
df_erp=df_erp.drop(['stock_status','stock_status_2'],axis=1)
```

Entrée [41]:

```
#Vérification
df_erp.head()
```

Out[41]:

	product_id	onsale_web	price	stock_quantity
0	3847	1	24.2	0
1	3849	1	34.3	0
2	3850	1	20.8	0
3	4032	1	14.1	0
4	4039	1	46.0	0

2.2 - Analyse exploratoire du fichier web.xlsx

Entrée [42]:

```
#Dimension du dataset
#Nombre d'observations
#Nombre de caractéristiques
print(df_web.shape)
```

```
(1513, 28)
```

Entrée [43]:

```
#Consulter le nombre de colonnes
print(len(df_web.columns))
```

```
28
```

Entrée [44]:

```
#La nature des données dans chacune des colonne
print(df_web.dtypes)
```

```
sku                             object
virtual                          int64
downloadable                     int64
rating_count                     int64
average_rating                 float64
total_sales                    float64
tax_status                      object
tax_class                      float64
post_author                    float64
post_date                datetime64[ns]
post_date_gmt            datetime64[ns]
post_content                   float64
post_title                      object
post_excerpt                    object
post_status                     object
comment_status                  object
ping_status                     object
post_password                  float64
post_name                       object
post_modified            datetime64[ns]
post_modified_gmt        datetime64[ns]
post_content_filtered          float64
post_parent                    float64
guid                            object
menu_order                     float64
post_type                       object
post_mime_type                  object
comment_count                  float64
dtype: object
```

Entrée [45]:

```
#Le nombre de valeurs présentes dans chacune des colonnes
print(df_web.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1513 entries, 0 to 1512
Data columns (total 28 columns):
sku                     1428 non-null object
virtual                 1513 non-null int64
downloadable            1513 non-null int64
rating_count            1513 non-null int64
average_rating          1430 non-null float64
total_sales             1430 non-null float64
tax_status              716 non-null object
tax_class               0 non-null float64
post_author             1430 non-null float64
post_date               1430 non-null datetime64[ns]
post_date_gmt           1430 non-null datetime64[ns]
post_content            0 non-null float64
post_title              1430 non-null object
post_excerpt            716 non-null object
post_status             1430 non-null object
comment_status          1430 non-null object
ping_status             1430 non-null object
post_password           0 non-null float64
post_name               1430 non-null object
post_modified           1430 non-null datetime64[ns]
post_modified_gmt       1430 non-null datetime64[ns]
post_content_filtered   0 non-null float64
post_parent             1430 non-null float64
guid                    1430 non-null object
menu_order              1430 non-null float64
post_type               1430 non-null object
post_mime_type          714 non-null object
comment_count           1430 non-null float64
dtypes: datetime64[ns](4), float64(10), int64(3), object(11)
memory usage: 266.0+ KB
None
```

Entrée [46]:

```
#Selon vous, quelles sont les colonnes à conserver ?
#selon moi j'ai conservé toutes les colonnes sauf tax_class,post_content,post_password,post
```

Entrée [47]:

```
#Si vous avez défini des colonnes à supprimer, effectuer l'opération
df_web=df_web.drop(['tax_class','post_content_filtered','post_content','post_password'],axi
```

Entrée [48]:

```
#Visualisation des valeurs de la colonne sku
df_web['sku'].unique()
```

Out[48]:

```
array(['bon-cadeau-25-euros', 15298, 15296, 15300, 19814, 19815, 15303,
       14975, 16042, 14980, 16041, 15269, 14977, 16044, 16043, 16449,
       16045, 16030, 13127, 19816, 16029, 16039, 16318, 16275, 16498,
       16320, 16319, 15966, 15022, 15967, 15490, 16416, 11862, 15444,
       15953, 12045, 13074, 15941, 16069, 13072, 15440, 13435, 13078,
       13117, 16296, 16014, 16462, 16013, 16180, 15676, 16120, 15564,
       15675, 15378, 15813, 13416, 14905, 15767, 16505, 15683, 16504,
       15787, 14800, 15353, 15382, 15339, 11668, 13209, 15341, 13217, 304,
       11641, 1662, 1360, 15648, 1364, 7086, 1366, 15140, 16238, 16237,
       15141, 14944, 14941, 14751, 16093, 15668, 15373, 15375, 14474,
       15482, 13453, 15075, 16124, 15785, 15784, 15786, 14332, 16210,
       16211, 16209, 15629, 15583, 16160, 16166, 15783, 16560, 15747,
       15746, 16190, 16189, 16265, 16191, 16263, 15605, 16529, 15441,
       13032, 16256, 16322, 16295, 15656, 15655, 15415, 15414, 15413,
       16023, 16024, 15720, 15714, 15717, 15718, 15480, 15213, 15672,
       12599, 15758, 15829, 15759, 16585, 15306, 16497, 15261, 12657,
       15403, 15461, 16269, 13905, 16567, 15436, 14725, 15310, 15770,
       16097, 15428, 15033, 16317, 15032, 6616, 12203, 14253, 12476,
       14485, 14945, 15662, 15663, 15664, 15665, 15136, 16537, 16307,
       16244, 15839, 13460, 13089, 12942, 14864, 14527, nan, 14865, 15690,
       16330, 16154, 16153, 16066, 16065, 15292, 16246, 16501, 16578,
       15567, 16553, 13172, 15120, 15949, 15946, 7818, 13599, 4679, 12586,
       12588, 15940, 12587, 12589, 12585, 9562, 13854, 13853, 11585,
       11467, 11586, 13765, 13766, 11587, 9636, 12639, 12641, 12640,
       14768, 3506, 3510, 3507, 13230, 7819, 3509, 15426, 15621, 15457,
       13604, 12857, 15476, 14000, 15478, 15475, 16151, 15659, 15147,
       15660, 15148, 15149, 15146, 15145, 15801, 15452, 15038, 15030,
       15875, 16186, 14371, 10459, 14372, 11049, 15850, 15849, 812, 807,
       805, 802, 2534, 793, 791, 2179, 804, 41, 798, 2361, 15848, 16525,
       16262, 16261, 15206, 11849, 13515, 13514, 13516, 10814, 11847,
       13517, 16081, 15402, 15404, 13647, 14657, 16053, 15525, 15527,
       14676, 16057, 16056, 13762, 15280, 15282, 15281, 15283, 15934,
       15933, 15575, 16239, 14451, 16324, 15582, 13736, 13659, 15465,
       15004, 14699, 15349, 15466, 14700, 10775, 16119, 15667, 14746,
       15361, 15196, 15657, 15658, 15670, 16527, 16513, 15880, 15879,
       16010, 14950, 16540, 15729, 38, 5646, 8344, 15576, 16138, 14366,
       13412, 14632, 15315, 13627, 14184, 15429, 16132, 14680, 15859,
       16229, 14302, 16072, 14300, 13096, 16564, 13754, 15734, 15448,
       15881, 15731, 15316, 15732, 14599, 15733, 15730, 12771, 3568,
       14506, 15811, 16342, 16292, 15307, 16047, 16255, 16274, 16148,
       16149, 16289, 14981, 15773, 15776, 16037, 16038, 15807, 15952,
       15808, 16062, 16063, 14802, 13052, 14805, 14220, 14374, 14395,
       15614, 13809, 15612, 13814, 15613, 15615, 15533, 15531, 15530,
       15928, 16276, 16277, 15456, 15425, 15047, 15927, 16155, 16280,
       9937, 16281, 15554, 15106, 16283, 13379, 15338, 15337, 15737,
       15958, 16515, 16586, 11225, 16004, 14756, 16005, 14930, 13313,
       15229, 14507, 14509, 14508, 15868, 14581, 14580, 15869, 15871,
       15870, 12791, 11602, 15073, 14839, 14696, 11996, 13914, 13913,
       11997, 531, 13531, 15711, 15713, 15715, 15346, 15345, 15344, 15755,
       15677, 14561, 16022, 16011, 3383, 14149, 13904, 14141, 12494,
       15462, 15095, 14626, 12496, 12315, 15649, 14809, 15155, 12194,
       16328, 14469, 16034, 14679, 15526, 16305, 16306, 15138, 15753,
       15756, 16131, 16130, 16129, 14712, 15481, 16146, 14192, 15860,
       15863, 15861, 15862, 15864, 14819, 14828, 14827, 15202, 13959,
```

```
       13965, 13958, 13957, 13520, 13969, 19820, 19821, 15748, 19822,
       16192, 14729, 8463, 13982, 15944, 15930, 14912, 15945, 14915,
       14855, 14856, 15923, 14845, 14844, 15921, 15922, 12366, 8365,
       12365, 14647, 15812, 14661, 16304, 15797, 16094, 14736, 11736,
       15036, 15360, 15674, 13557, 15035, 16121, 14241, 14982, 15026,
       15116, 15369, 15566, 16003, 15127, 15125, 14323, 15631, 16147,
       7033, 11258, 13849, 15818, 15179, 15185, 15183, 15254, 15178,
       15184, 15180, 15264, 14338, 15561, 16213, 14692, 13291, 13895,
       15688, 14461, 11277, 15399, 13572, 14955, 13567, 15471, 15080,
       14429, 15238, 15237, 14600, 15241, 11933, 15240, 15325, 15328,
       15329, 15775, 15774, 14983, 13910, 16539, 15910, 12339, 12869,
       14095, 14099, 15856, 12881, 15857, 12882, 15227, 10014, 14265,
       14774, 14775, 14773, 15343, 15351, 16323, 523, 15432, 16472, 15895,
       15577, 15766, 15892, 16326, 15574, 13662, 11669, 13215, 13211,
       15342, 15318, 13073, 16159, 16264, 14899, 15134, 16133, 16028,
       15951, 15487, 15486, 15489, 14089, 14100, 14092, 14090, 14106,
       14101, 14797, 15201, 14923, 14573, 14569, 14570, 15834, 14596,
       15126, 14604, 16565, 16580, 16077, 13996, 15072, 11601, 12790,
       15070, 16096, 7032, 15324, 15162, 15161, 15163, 16273, 16247,
       15654, 15710, 15745, 15678, 15810, 15779, 15707, 15705, 15706,
       15704, 15473, 15479, 15647, 15769, 15434, 15764, 16071, 15781,
       16031, 15539, 16046, 15204, 15205, 15790, 15791, 15792, 15793,
       15795, 15794, 15763, 16152, 15661, 16068, 16067, 8193, 16144,
       15256, 15735, 14897, 15736, 15740, 15845, 15741, 16135, 15891,
       15887, '13127-1', 16230], dtype=object)
```

Entrée [49]:

```
#Quelles sont les valeurs qui ne semblent pas respecter la régle de codification?
#les valeurs qui ne semblent pas respecter la régle de codification sont:'13127-1','Nan','b
```

Entrée [50]:

```
#Si vous avez identifié des codes articles ne respectant pas la règle de codification, cons
```

Entrée [51]:

```
df_web[df_web['sku']=='13127-1']
```

Out[51]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_
797	13127-1	0	0	0	0.0	0.0	taxable	
1511	13127-1	0	0	0	0.0	0.0	NaN	

2 rows × 24 columns

Entrée [52]:

```
df_web[df_web['sku'].isnull()]
```

Out[52]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_aut
178	NaN	0	0	0	NaN	NaN	NaN	N
179	NaN	0	0	0	NaN	NaN	NaN	N
227	NaN	0	0	0	NaN	NaN	NaN	N
230	NaN	0	0	0	NaN	NaN	NaN	N
231	NaN	0	0	0	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	
792	NaN	0	0	0	NaN	NaN	NaN	N
793	NaN	0	0	0	NaN	NaN	NaN	N
794	NaN	0	0	0	NaN	NaN	NaN	N
795	NaN	0	0	0	NaN	NaN	NaN	N
796	NaN	0	0	0	NaN	NaN	NaN	N

85 rows × 24 columns

Entrée [53]:

```
df_web[df_web['sku']=='bon-cadeau-25-euros']
```

Out[53]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post
0	bon-cadeau-25-euros	0	0	0	0.0	10.0	taxable	
1209	bon-cadeau-25-euros	0	0	0	0.0	10.0	NaN	

2 rows × 24 columns

Entrée [54]:

```
#Pour les codes articles identifiés, réalisé une analyse et définissez l'action à entrepren
```

Entrée [55]:

```
df_web[df_web['sku'].isnull()].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 85 entries, 178 to 796
Data columns (total 24 columns):
sku                  0 non-null object
virtual              85 non-null int64
downloadable         85 non-null int64
rating_count         85 non-null int64
average_rating       2 non-null float64
total_sales          2 non-null float64
tax_status           2 non-null object
post_author          2 non-null float64
post_date            2 non-null datetime64[ns]
post_date_gmt        2 non-null datetime64[ns]
post_title           2 non-null object
post_excerpt         2 non-null object
post_status          2 non-null object
comment_status       2 non-null object
ping_status          2 non-null object
post_name            2 non-null object
post_modified        2 non-null datetime64[ns]
post_modified_gmt    2 non-null datetime64[ns]
post_parent          2 non-null float64
guid                 2 non-null object
menu_order           2 non-null float64
post_type            2 non-null object
post_mime_type       0 non-null object
comment_count        2 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(11)
memory usage: 12.9+ KB
```

Entrée [56]:

```
df_web[(df_web['sku'].isnull()) & (df_web['total_sales'].notnull())]
```

Out[56]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_aut
470	NaN	0	0	0	0.0	0.0	taxable	
471	NaN	0	0	0	0.0	0.0	taxable	

2 rows × 24 columns

Entrée [57]:

```
df_web.loc[df_web['post_name']=='pierre-jean-villa-cote-rotie-fongeant-2017','sku']='inconr
```

Entrée [58]:

```
df_web.loc[df_web['post_name']=='pierre-jean-villa-condrieu-suspendu-2018','sku']='inconnu_
```

Entrée [59]:

```
df_web[(df_web['sku'].isnull()) & (df_web['total_sales'].notnull())]
```

Out[59]:

sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_author

0 rows × 24 columns

Entrée [60]:

```
df_web.loc[df_web['post_name']=='clos-du-mont-olivet-chateauneuf-du-pape-2007-2','sku']=131
```

Entrée [61]:

```
df_web[df_web['sku']==131271]
```

Out[61]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_
797	131271	0	0	0	0.0	0.0	taxable	
1511	131271	0	0	0	0.0	0.0	NaN	

2 rows × 24 columns

Entrée [62]:

```
df_web.loc[df_web['post_name']=='bon-cadeau-de-25-euros','sku']='inconu_3'
```

Entrée [63]:

```
df_web[df_web['sku']=='inconu_3']
```

Out[63]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	pos
0	inconu_3	0	0	0	0.0	10.0	taxable	
1209	inconu_3	0	0	0	0.0	10.0	NaN	

2 rows × 24 columns

Entrée [64]:

```
df_web[df_web['sku'].isnull()].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83 entries, 178 to 796
Data columns (total 24 columns):
sku                  0 non-null object
virtual              83 non-null int64
downloadable         83 non-null int64
rating_count         83 non-null int64
average_rating       0 non-null float64
total_sales          0 non-null float64
tax_status           0 non-null object
post_author          0 non-null float64
post_date            0 non-null datetime64[ns]
post_date_gmt        0 non-null datetime64[ns]
post_title           0 non-null object
post_excerpt         0 non-null object
post_status          0 non-null object
comment_status       0 non-null object
ping_status          0 non-null object
post_name            0 non-null object
post_modified        0 non-null datetime64[ns]
post_modified_gmt    0 non-null datetime64[ns]
post_parent          0 non-null float64
guid                 0 non-null object
menu_order           0 non-null float64
post_type            0 non-null object
post_mime_type       0 non-null object
comment_count        0 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(11)
memory usage: 12.6+ KB
```

Entrée [65]:

```
df_web.drop(df_web[df_web['sku'].isnull()].index,inplace=True)
```

Entrée [66]:

```
df_web[df_web['sku'].isnull()]
```

Out[66]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_author

0 rows × 24 columns

Entrée [67]:

```
print(df_web.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1430 entries, 0 to 1512
Data columns (total 24 columns):
sku                 1430 non-null object
virtual             1430 non-null int64
downloadable        1430 non-null int64
rating_count        1430 non-null int64
average_rating      1430 non-null float64
total_sales         1430 non-null float64
tax_status          716 non-null object
post_author         1430 non-null float64
post_date           1430 non-null datetime64[ns]
post_date_gmt       1430 non-null datetime64[ns]
post_title          1430 non-null object
post_excerpt        716 non-null object
post_status         1430 non-null object
comment_status      1430 non-null object
ping_status         1430 non-null object
post_name           1430 non-null object
post_modified       1430 non-null datetime64[ns]
post_modified_gmt   1430 non-null datetime64[ns]
post_parent         1430 non-null float64
guid                1430 non-null object
menu_order          1430 non-null float64
post_type           1430 non-null object
post_mime_type      714 non-null object
comment_count       1430 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(11)
memory usage: 217.9+ KB
None
```

Entrée [68]:

```
df_web=df_web[df_web['sku'].notnull()]
```

Entrée [69]:

```
df_web['sku'].isnull().sum()
```

Out[69]:

```
0
```

Entrée [70]:

```
df_web.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1430 entries, 0 to 1512
Data columns (total 24 columns):
sku                 1430 non-null object
virtual             1430 non-null int64
downloadable        1430 non-null int64
rating_count        1430 non-null int64
average_rating      1430 non-null float64
total_sales         1430 non-null float64
tax_status          716 non-null object
post_author         1430 non-null float64
post_date           1430 non-null datetime64[ns]
post_date_gmt       1430 non-null datetime64[ns]
post_title          1430 non-null object
post_excerpt        716 non-null object
post_status         1430 non-null object
comment_status      1430 non-null object
ping_status         1430 non-null object
post_name           1430 non-null object
post_modified       1430 non-null datetime64[ns]
post_modified_gmt   1430 non-null datetime64[ns]
post_parent         1430 non-null float64
guid                1430 non-null object
menu_order          1430 non-null float64
post_type           1430 non-null object
post_mime_type      714 non-null object
comment_count       1430 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(11)
memory usage: 217.9+ KB
```

Entrée [71]:

```
#La clé pour chaque ligne est-elle uniques? ou autrement dit, y a-t-il des doublons?
df_web.duplicated('sku').sum()
```

Out[71]:

```
714
```

Entrée [72]:

```
#Identifier les lignes sans code articles
df_web.loc[df_web['post_type']=='attachment',:]
```

Out[72]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post
799	15298	0	0	0	0.0	6.0	NaN	
800	15296	0	0	0	0.0	0.0	NaN	
801	15300	0	0	0	0.0	0.0	NaN	
802	19814	0	0	0	0.0	3.0	NaN	
803	19815	0	0	0	0.0	0.0	NaN	
...	...	...	...	...	...	...	...	
1508	16135	0	0	0	0.0	5.0	NaN	
1509	15891	0	0	0	0.0	0.0	NaN	
1510	15887	0	0	0	0.0	0.0	NaN	
1511	131271	0	0	0	0.0	0.0	NaN	
1512	16230	0	0	0	0.0	0.0	NaN	

714 rows × 24 columns

Entrée [73]:

```
df_web.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1430 entries, 0 to 1512
Data columns (total 24 columns):
sku                 1430 non-null object
virtual             1430 non-null int64
downloadable        1430 non-null int64
rating_count        1430 non-null int64
average_rating      1430 non-null float64
total_sales         1430 non-null float64
tax_status          716 non-null object
post_author         1430 non-null float64
post_date           1430 non-null datetime64[ns]
post_date_gmt       1430 non-null datetime64[ns]
post_title          1430 non-null object
post_excerpt        716 non-null object
post_status         1430 non-null object
comment_status      1430 non-null object
ping_status         1430 non-null object
post_name           1430 non-null object
post_modified       1430 non-null datetime64[ns]
post_modified_gmt   1430 non-null datetime64[ns]
post_parent         1430 non-null float64
guid                1430 non-null object
menu_order          1430 non-null float64
post_type           1430 non-null object
post_mime_type      714 non-null object
comment_count       1430 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(11)
memory usage: 217.9+ KB
```

Entrée [74]:

```
#Les lignes sans code article semble être toutes non renseignés
#Pour s'en assurer réaliser les étapes suivantes:
#1 - Créer un dataframe avec uniquement les lignes sans code article

#2 - utiliser la fonction df.info() sur ce nouveau dataframe pour observer le nombre de val

#3 - Que constatez-vous?
```

Entrée [75]:

```
df_web = df_web[df_web["post_type"] == "product"]
```

Entrée [76]:

```
df_web.loc[df_web['post_type']=='product',:]
```

Out[76]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post
0	inconu_3	0	0	0	0.0	10.0	taxable	
1	15298	0	0	0	0.0	6.0	taxable	
2	15296	0	0	0	0.0	0.0	taxable	
3	15300	0	0	0	0.0	0.0	taxable	
4	19814	0	0	0	0.0	3.0	taxable	
...	...	...	...	...	...	...	...	
762	16135	0	0	0	0.0	5.0	taxable	
767	15891	0	0	0	0.0	0.0	taxable	
768	15887	0	0	0	0.0	0.0	taxable	
797	131271	0	0	0	0.0	0.0	taxable	
798	16230	0	0	0	0.0	0.0	taxable	

716 rows × 24 columns

Entrée [77]:

```
df_web.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 716 entries, 0 to 798
Data columns (total 24 columns):
sku                 716 non-null object
virtual             716 non-null int64
downloadable        716 non-null int64
rating_count        716 non-null int64
average_rating      716 non-null float64
total_sales         716 non-null float64
tax_status          716 non-null object
post_author         716 non-null float64
post_date           716 non-null datetime64[ns]
post_date_gmt       716 non-null datetime64[ns]
post_title          716 non-null object
post_excerpt        716 non-null object
post_status         716 non-null object
comment_status      716 non-null object
ping_status         716 non-null object
post_name           716 non-null object
post_modified       716 non-null datetime64[ns]
post_modified_gmt   716 non-null datetime64[ns]
post_parent         716 non-null float64
guid                716 non-null object
menu_order          716 non-null float64
post_type           716 non-null object
post_mime_type      0 non-null object
comment_count       716 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(11)
memory usage: 109.1+ KB
```

Entrée [78]:

```
df_web=df_web.drop(['post_mime_type'],axis=1)
```

Entrée [79]:

```
df_web.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 716 entries, 0 to 798
Data columns (total 23 columns):
sku                 716 non-null object
virtual             716 non-null int64
downloadable        716 non-null int64
rating_count        716 non-null int64
average_rating      716 non-null float64
total_sales         716 non-null float64
tax_status          716 non-null object
post_author         716 non-null float64
post_date           716 non-null datetime64[ns]
post_date_gmt       716 non-null datetime64[ns]
post_title          716 non-null object
post_excerpt        716 non-null object
post_status         716 non-null object
comment_status      716 non-null object
ping_status         716 non-null object
post_name           716 non-null object
post_modified       716 non-null datetime64[ns]
post_modified_gmt   716 non-null datetime64[ns]
post_parent         716 non-null float64
guid                716 non-null object
menu_order          716 non-null float64
post_type           716 non-null object
comment_count       716 non-null float64
dtypes: datetime64[ns](4), float64(6), int64(3), object(10)
memory usage: 106.3+ KB
```

Entrée [80]:

```
df_web.head()
```

Out[80]:

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	post_a
0	inconu_3	0	0	0	0.0	10.0	taxable	
1	15298	0	0	0	0.0	6.0	taxable	
2	15296	0	0	0	0.0	0.0	taxable	
3	15300	0	0	0	0.0	0.0	taxable	
4	19814	0	0	0	0.0	3.0	taxable	

5 rows × 23 columns

2.3 - Analyse exploratoire du fichier liaison.xlsx

Entrée [81]:

```
#Dimension du dataset
#Nombre d'observations
#Nombre de caractéristiques
print(df_liaison.shape)
```

```
(825, 2)
```

Entrée [82]:

```
#Consulter le nombre de colonnes
print(len(df_liaison.columns))
```

```
2
```

Entrée [83]:

```
#La nature des données dans chacune des colonnes
print(df_liaison.dtypes)
```

```
product_id     int64
id_web        object
dtype: object
```

Entrée [84]:

```
#Le nombre de valeurs présentes dans chacune des colonnes
df_liaison.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 2 columns):
product_id    825 non-null int64
id_web        734 non-null object
dtypes: int64(1), object(1)
memory usage: 9.7+ KB
```

Entrée [85]:

```
#Les valeurs de la colonne "product_id" sont elles toutes uniques?
df_liaison.duplicated('product_id').sum()
```

Out[85]:

```
0
```

Entrée [86]:

```
#Les valeurs de la colonne "id_web" sont-elles toutes uniques?
df_liaison.duplicated('id_web').sum()
```

Out[86]:

```
90
```

Entrée [87]:

```
#Avons-nous des articles sans correspondances?
#oui on a 91 articles sans correspondance
df_liaison[df_liaison['id_web'].isnull()]
```

Out[87]:

	product_id	id_web
19	4055	NaN
49	4090	NaN
50	4092	NaN
119	4195	NaN
131	4209	NaN
...	...	...
817	7196	NaN
818	7200	NaN
819	7201	NaN
820	7203	NaN
821	7204	NaN

91 rows × 2 columns

2.4 - Analyse exploratoire du fichier caracteristiques_vins.xlsx

Entrée [88]:

```
#Dimension du dataset
#Nombre d'observations
#Nombre de caractéristiques
print(df_car_vins.shape)
```

```
(611, 13)
```

Entrée [89]:

```
#Consulter le nombre de colonnes
print(len(df_car_vins.columns))
```

```
13
```

Entrée [90]:

```
#La nature des données dans chacune des colonnes
print(df_car_vins.dtypes)
```

```
post_name                  object
poids                      object
Région                     object
Domaine                    object
Appellation                object
Couleur                    object
Cépage                     object
Millésime                 float64
Garde                      object
Contenance                 object
Degré d'alcool             object
Température dégustation    object
Alliance mets              object
dtype: object
```

Entrée [91]:

```
#Le nombre de valeurs présentes dans chacune des colonnes
print(df_car_vins.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 611 entries, 0 to 610
Data columns (total 13 columns):
post_name                  611 non-null object
poids                      611 non-null object
Région                     586 non-null object
Domaine                    577 non-null object
Appellation                559 non-null object
Couleur                    566 non-null object
Cépage                     571 non-null object
Millésime                  541 non-null float64
Garde                      569 non-null object
Contenance                 611 non-null object
Degré d'alcool             586 non-null object
Température dégustation    574 non-null object
Alliance mets              574 non-null object
dtypes: float64(1), object(12)
memory usage: 33.5+ KB
None
```

Entrée [92]:

```
#Affichage des 5 premières lignes du dataset
df_car_vins.head()
```

Out[92]:

	post_name	poids	Région	Domaine	Appellation	Couleur	Cépage	Millésime	Garde	Coı
0	pierre-jean-villa-saint-joseph-preface-2018	1.5 kg	Rhône	Pierre Jean Villa	Saint Joseph	Rouge	100% Syrah	2020.0	4-7 ans	
1	pierre-jean-villa-saint-joseph-tilde-2017	1.5 kg	Rhône	Pierre Jean Villa	Saint Joseph	Rouge	100% Syrah	2019.0	6-8 ans	
2	pierre-jean-villa-croze-hermitage-accroche-coe...	1.5 kg	Rhône	Pierre Jean Villa	Crozes-Hermitage	Rouge	100% Syrah	2020.0	3-5 ans	
3	pierre-jean-villa-igp-gamine-2018	1.5 kg	Rhône	Pierre Jean Villa	Collines Rhodaniennes	Rouge	100% Syrah	2020.0	3-5 ans	
4	pierre-jean-villa-cote-rotie-carmina-2017	1.5 kg	Rhône	Pierre Jean Villa	Côte Rôtie	Rouge	100% Syrah	2019.0	10-20 ans	

Entrée [93]:

```
#Quels sont les produits avec des informations manquantes?
print(df_car_vins.isnull().sum())
```

```
post_name                0
poids                    0
Région                  25
Domaine                 34
Appellation             52
Couleur                 45
Cépage                  40
Millésime               70
Garde                   42
Contenance               0
Degré d'alcool          25
Température dégustation 37
Alliance mets           37
dtype: int64
```

Entrée [94]:

```
df_car_vins.isnull().sum().sum()
```

Out[94]:

```
407
```

Entrée [95]:

```
#Est-il possible de corriger les données manquantes?
#Oui, on peut modifier les données manquantes à l'aide des valeurs des autres colonnes.
```

Etape 3 - Jonction des fichiers

Entrée [96]:

```
#Etape 3.1 - Jonction du fichier df_erp et df_liaison
```

Entrée [97]:

```
#Fusion des fichiers df_erp et df_liaison
df_merge=pd.merge(df_erp,df_liaison)
print(df_merge)
```

```
     product_id  onsale_web  price  stock_quantity   id_web
0          3847           1   24.2               0    15298
1          3849           1   34.3               0    15296
2          3850           1   20.8               0    15300
3          4032           1   14.1               0    19814
4          4039           1   46.0               0    19815
..          ...         ...    ...             ...      ...
820        7203           0   45.0              30      NaN
821        7204           0   45.0               9      NaN
822        7247           1   54.8              23  13127-1
823        7329           0   26.5              14  14680-1
824        7338           1   16.3              45    16230

[825 rows x 5 columns]
```

Entrée [98]:

```
df_merge=pd.merge(df_erp,df_liaison,on='product_id',how='outer',indicator=True)
df_merge
```

Out[98]:

	product_id	onsale_web	price	stock_quantity	id_web	_merge
0	3847	1	24.2	0	15298	both
1	3849	1	34.3	0	15296	both
2	3850	1	20.8	0	15300	both
3	4032	1	14.1	0	19814	both
4	4039	1	46.0	0	19815	both
...	...	...	...	...	...	...
820	7203	0	45.0	30	NaN	both
821	7204	0	45.0	9	NaN	both
822	7247	1	54.8	23	13127-1	both
823	7329	0	26.5	14	14680-1	both
824	7338	1	16.3	45	16230	both

825 rows × 6 columns

Entrée [99]:

```
#Y a t-il des lignes ne "matchant" entre les 2 fichiers?
```

Entrée [100]:

```
df_merge[df_merge['_merge']!='both']
```

Out[100]:

product_id	onsale_web	price	stock_quantity	id_web	_merge

Entrée [101]:

```
df_merge.drop('_merge',axis=1,inplace=True)
```

Etape 3.2 - Jonction du fichier df_merge et df_web

Entrée [102]:

```
#Fusionnez les datasets df_merge et df_web
df_merge_1=pd.merge(df_merge,df_web,left_on='id_web',right_on='sku',how='outer',indicator=T
print(df_merge_1)
```

```
     product_id  onsale_web  price  stock_quantity id_web        sku  virtua
l  \
0        3847.0         1.0   24.2             0.0  15298      15298      0.
0
1        3849.0         1.0   34.3             0.0  15296      15296      0.
0
2        3850.0         1.0   20.8             0.0  15300      15300      0.
0
3        4032.0         1.0   14.1             0.0  19814      19814      0.
0
4        4039.0         1.0   46.0             0.0  19815      19815      0.
0
..          ...         ...    ...             ...    ...        ...
...
824      7338.0         1.0   16.3            45.0  16230      16230      0.
0
825         NaN         NaN    NaN             NaN    NaN   inconu_3      0.
0
826         NaN         NaN    NaN             NaN    NaN  inconnu_1      0.
0
827         NaN         NaN    NaN             NaN    NaN  inconnu_2      0.
0
828         NaN         NaN    NaN             NaN    NaN     131271      0.
0

     downloadable  rating_count  average_rating  ...  ping_status  \
0             0.0           0.0             0.0  ...       closed
1             0.0           0.0             0.0  ...       closed
2             0.0           0.0             0.0  ...       closed
3             0.0           0.0             0.0  ...       closed
4             0.0           0.0             0.0  ...       closed
..            ...           ...             ...  ...          ...
824           0.0           0.0             0.0  ...       closed
825           0.0           0.0             0.0  ...       closed
826           0.0           0.0             0.0  ...       closed
827           0.0           0.0             0.0  ...       closed
828           0.0           0.0             0.0  ...       closed

                                          post_name       post_modified
\
0          pierre-jean-villa-saint-joseph-preface-2018 2019-12-30 09:30:29
1            pierre-jean-villa-saint-joseph-tilde-2017 2019-12-21 09:00:17
2    pierre-jean-villa-croze-hermitage-accroche-coe... 2020-06-26 18:15:03
3                    pierre-jean-villa-igp-gamine-2018 2020-01-04 16:36:01
4            pierre-jean-villa-cote-rotie-carmina-2017 2020-01-04 16:36:10
..                                                 ...                 ...
824  domaine-saint-nicolas-fiefs-vendeens-blanc-les... 2020-08-13 10:45:03
825                            bon-cadeau-de-25-euros 2018-06-01 14:13:57
826          pierre-jean-villa-cote-rotie-fongeant-2017 2019-11-02 13:24:15
827            pierre-jean-villa-condrieu-suspendu-2018 2019-11-02 13:24:01
828      clos-du-mont-olivet-chateauneuf-du-pape-2007-2 2020-07-20 17:09:06

      post_modified_gmt post_parent  \
0   2019-12-30 08:30:29         0.0
1   2019-12-21 08:00:17         0.0
```

```
2   2020-06-26 16:15:03         0.0
3   2020-01-04 15:36:01         0.0
4   2020-01-04 15:36:10         0.0
..                  ...         ...
824 2020-08-13 08:45:03         0.0
825 2018-06-01 12:13:57         0.0
826 2019-11-02 12:24:15         0.0
827 2019-11-02 12:24:01         0.0
828 2020-07-20 15:09:06         0.0

                                                    guid menu_order post_type
\
0    https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
1    https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
2    https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
3    https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
4    https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
..                                                  ...         ...       ...
824  https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
825  https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
826  https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
827  https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product
828  https://www.bottle-neck.fr/?post_type=product&... (https://www.bottle-n
eck.fr/?post_type=product&...)        0.0   product

    comment_count      _merge
0             0.0        both
1             0.0        both
2             0.0        both
3             0.0        both
4             0.0        both
..            ...         ...
824           0.0        both
825           0.0  right_only
826           0.0  right_only
827           0.0  right_only
828           0.0  right_only

[829 rows x 29 columns]
```

Entrée [103]:

```
df_merge_1[df_merge_1["_merge"]=="left_only"]
```

Out[103]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	rating
19	4055.0	0.0	86.1	0.0	NaN	NaN	NaN	NaN	
20	4090.0	0.0	73.0	0.0	NaN	NaN	NaN	NaN	
21	4092.0	0.0	47.0	0.0	NaN	NaN	NaN	NaN	
22	4195.0	0.0	14.1	0.0	NaN	NaN	NaN	NaN	
23	4209.0	0.0	73.5	0.0	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	NaN	
720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	NaN	
743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	NaN	
822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	NaN	
823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	NaN	

113 rows × 29 columns

Entrée [104]:

```
df_merge_1[df_merge_1["_merge"]=="right_only"]
```

Out[104]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	ı
825	NaN	NaN	NaN	NaN	NaN	inconu_3	0.0	0.0	
826	NaN	NaN	NaN	NaN	NaN	inconnu_1	0.0	0.0	
827	NaN	NaN	NaN	NaN	NaN	inconnu_2	0.0	0.0	
828	NaN	NaN	NaN	NaN	NaN	131271	0.0	0.0	

4 rows × 29 columns

Entrée [105]:

```
df_merge_1[df_merge_1["_merge"]=="both"]
```

Out[105]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	ratin
0	3847.0	1.0	24.2	0.0	15298	15298	0.0	0.0	
1	3849.0	1.0	34.3	0.0	15296	15296	0.0	0.0	
2	3850.0	1.0	20.8	0.0	15300	15300	0.0	0.0	
3	4032.0	1.0	14.1	0.0	19814	19814	0.0	0.0	
4	4039.0	1.0	46.0	0.0	19815	19815	0.0	0.0	
...	...	...	...	...	...	...	...	...	
818	6928.0	1.0	19.0	20.0	15741	15741	0.0	0.0	
819	6930.0	1.0	8.4	83.0	16135	16135	0.0	0.0	
820	7023.0	1.0	27.5	15.0	15891	15891	0.0	0.0	
821	7025.0	1.0	69.0	2.0	15887	15887	0.0	0.0	
824	7338.0	1.0	16.3	45.0	16230	16230	0.0	0.0	

712 rows × 29 columns

Entrée [106]:

```
df_merge_1 = df_merge_1[df_merge_1["_merge"]!="right_only"]
df_merge_1
```

Out[106]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	ratin
0	3847.0	1.0	24.2	0.0	15298	15298	0.0	0.0	
1	3849.0	1.0	34.3	0.0	15296	15296	0.0	0.0	
2	3850.0	1.0	20.8	0.0	15300	15300	0.0	0.0	
3	4032.0	1.0	14.1	0.0	19814	19814	0.0	0.0	
4	4039.0	1.0	46.0	0.0	19815	19815	0.0	0.0	
...	...	...	...	...	...	...	...	...	
820	7023.0	1.0	27.5	15.0	15891	15891	0.0	0.0	
821	7025.0	1.0	69.0	2.0	15887	15887	0.0	0.0	
822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	NaN	
823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	NaN	
824	7338.0	1.0	16.3	45.0	16230	16230	0.0	0.0	

825 rows × 29 columns

Entrée [107]:

```
#Avons-nous des lignes sans correspondances?
df_merge_1[df_merge_1["_merge"]!="both"]
```

Out[107]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	rating
19	4055.0	0.0	86.1	0.0	NaN	NaN	NaN	NaN	
20	4090.0	0.0	73.0	0.0	NaN	NaN	NaN	NaN	
21	4092.0	0.0	47.0	0.0	NaN	NaN	NaN	NaN	
22	4195.0	0.0	14.1	0.0	NaN	NaN	NaN	NaN	
23	4209.0	0.0	73.5	0.0	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	NaN	
720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	NaN	
743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	NaN	
822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	NaN	
823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	NaN	

113 rows × 29 columns

Entrée [108]:

```
df_merge_1.drop('_merge',axis=1,inplace=True)
```

Etape 3.3 - Jonction du fichier df_merge et df_caracteristiques

Entrée [109]:

```
#Fusion de la table df_merge et df_caracteristiques
df_merge_2=pd.merge(df_merge_1,df_car_vins,on='post_name',how='left')
print(df_merge_2)
```

```
     product_id  onsale_web  price  stock_quantity   id_web    sku  virtua
l  \
0        3847.0         1.0   24.2             0.0    15298  15298      0.
0
1        3849.0         1.0   34.3             0.0    15296  15296      0.
0
2        3850.0         1.0   20.8             0.0    15300  15300      0.
0
3        4032.0         1.0   14.1             0.0    19814  19814      0.
0
4        4039.0         1.0   46.0             0.0    19815  19815      0.
0
..          ...         ...    ...             ...      ...    ...
...
820      7023.0         1.0   27.5            15.0    15891  15891      0.
0
821      7025.0         1.0   69.0             2.0    15887  15887      0.
0
822      7247.0         1.0   54.8            23.0  13127-1    NaN      Na
N
```

Entrée [110]:

```
df_merge_2
```

Out[110]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	ratin
0	3847.0	1.0	24.2	0.0	15298	15298	0.0	0.0	
1	3849.0	1.0	34.3	0.0	15296	15296	0.0	0.0	
2	3850.0	1.0	20.8	0.0	15300	15300	0.0	0.0	
3	4032.0	1.0	14.1	0.0	19814	19814	0.0	0.0	
4	4039.0	1.0	46.0	0.0	19815	19815	0.0	0.0	
...	...	...	...	...	...	...	...	...	
820	7023.0	1.0	27.5	15.0	15891	15891	0.0	0.0	
821	7025.0	1.0	69.0	2.0	15887	15887	0.0	0.0	
822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	NaN	
823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	NaN	
824	7338.0	1.0	16.3	45.0	16230	16230	0.0	0.0	

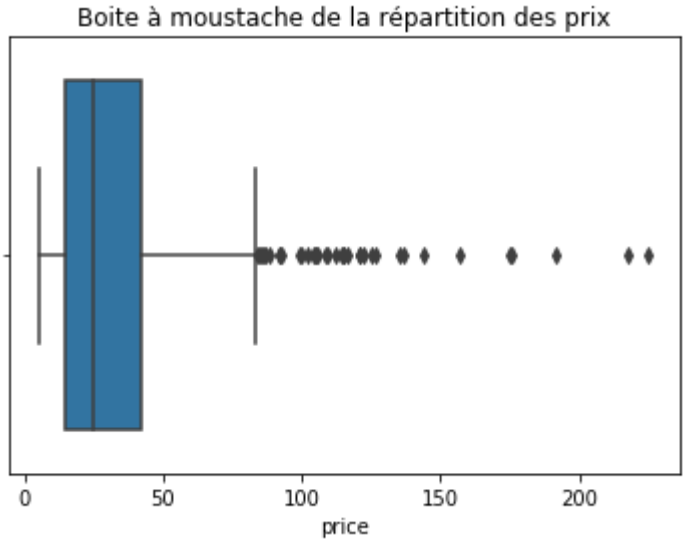
825 rows × 40 columns

Etape 4 - Analyse univarié des prix

Etape 4.1 - Exploration par la visualisation de données

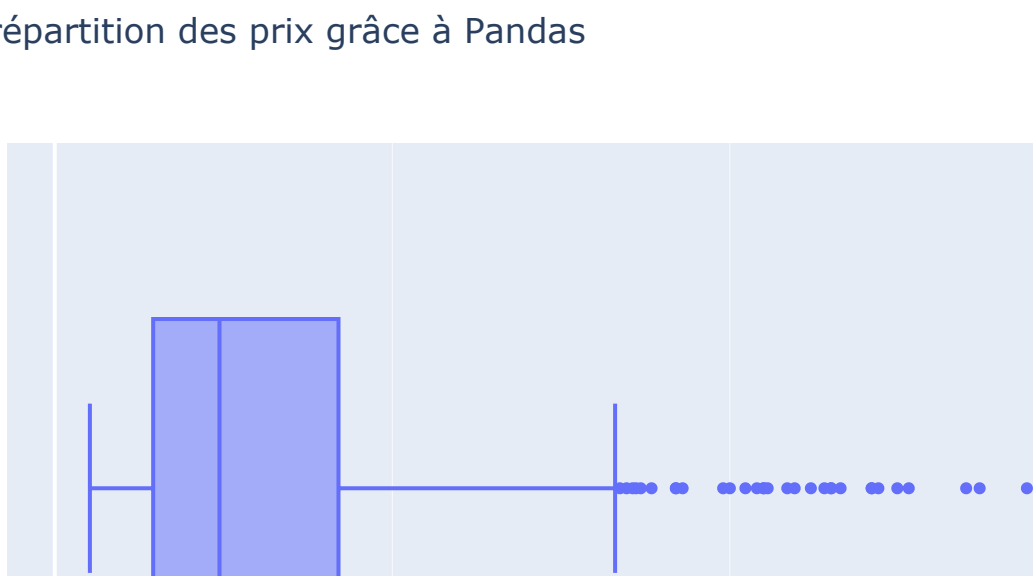
Entrée [111]:

```
#Création d'une Boite à moustache de la répartition des prix grâce à Pandas
import seaborn as sns
sns.boxplot(x=df_merge['price'])
plt.title("Boite à moustache de la répartition des prix")
plt.show()
```



Entrée [112]:

```
#Autre méthode avec plotly express
fig=px.box(df_merge,x='price',title='la répartition des prix grâce à Pandas')
fig.show()
```



Etape 4.2 - Exploration par l'utisation de méthodes statistique

Etape 4.2.1 - Identification par le Z-index

Entrée [113]:

```
#Calculer la moyenne du prix
df_merge_2['price'].mean()
```

Out[113]:

```
32.415636363636374
```

Entrée [114]:

```
#Calculer l'écart-type du prix
df_merge_2['price'].std()
```

Out[114]:

```
26.795849199710545
```

Entrée [115]:

```
import scipy.stats as stats
```

Entrée [116]:

```
#Calculer le Z-score
df_merge_2['z_score'] = stats.zscore(df_merge_2['price'])
print(df_merge_2['z_score'])
```

```
0     -0.306787
1      0.070366
2     -0.433749
3     -0.683940
4      0.507265
         ...
820   -0.183559
821    1.366128
822    0.835874
823   -0.220901
824   -0.601788
Name: z_score, Length: 825, dtype: float64
```

Entrée [117]:

```
#2 eme methode pour calculer z_score
z_score=(df_merge_2['price']-df_merge_2['price'].mean())/df_merge_2['price'].std()
z_score
```

Out[117]:

```
0     -0.306601
1      0.070323
2     -0.433486
3     -0.683525
4      0.506958
         ...
820   -0.183448
821    1.365300
822    0.835367
823   -0.220767
824   -0.601423
Name: price, Length: 825, dtype: float64
```

Entrée [118]:

```
#Quel est le seuil prix dont z-score est supérieur à 3?
df_merge_2[abs(df_merge_2['z_score']) > 3]['price']
```

Out[118]:

```
30     144.0
291    225.0
293    126.5
310    176.0
313    157.0
478    137.0
525    217.5
615    124.8
657    175.0
692    191.3
708    122.0
709    114.0
752    135.0
758    116.4
763    115.0
764    121.0
766    115.0
767    121.0
Name: price, dtype: float64
```

Entrée [119]:

```
seuil_price=df_merge_2[abs(df_merge_2['z_score']) > 3]['price'].min()
seuil_price
```

Out[119]:

```
114.0
```

Etape 4.2.2 - Identification par l'interval interquartile

Entrée [120]:

```
#Utilisation de la fonction describe de Pandas pour l'etude des mesures de dispersions
df_merge_describe=df_merge_2['price'].describe()
df_merge_describe
```

Out[120]:

```
count    825.000000
mean      32.415636
std       26.795849
min        5.200000
25%       14.600000
50%       24.400000
75%       42.000000
max      225.000000
Name: price, dtype: float64
```

Entrée [121]:

```
#Définissez un seuil pour les articles "outliers" en prix
```

Entrée [122]:

```
q1=df_merge_2['price'].quantile(0.25)
q3=df_merge_2['price'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
df_outliers=df_merge_2[(df_merge_2['price']<lower_bound)| (df_merge_2['price']>upper_bound)
df_outliers
```

Out[122]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	rating_count	ave
19	4055.0	0.0	86.1	0.0	NaN	NaN	NaN	NaN	NaN	
30	4594.0	1.0	144.0	0.0	NaN	NaN	NaN	NaN	NaN	
47	5070.0	1.0	84.7	0.0	NaN	NaN	NaN	NaN	NaN	
55	6324.0	0.0	92.0	18.0	NaN	NaN	NaN	NaN	NaN	
154	4115.0	1.0	100.0	11.0	15382	15382	0.0	0.0	0.0	
156	4132.0	1.0	88.4	5.0	11668	11668	0.0	0.0	0.0	
291	4352.0	1.0	225.0	0.0	15940	15940	0.0	0.0	0.0	

Entrée [123]:

```
#Définissez le nombre d'articles de l'ensemble du catalogue "outliers"
len(df_outliers)
```

Out[123]:

```
37
```

Entrée [124]:

```
#Définissez la proportion de l'ensemble du catalogue "outliers"
```

Entrée [125]:

```
len(df_merge_2)
```

Out[125]:

```
825
```

Entrée [126]:

```
(len(df_outliers)/len(df_merge_2))*100
```

Out[126]:

```
4.484848484848484
```

Entrée [127]:

```
#Selon vous, ces outliers sont-ils justifiés ? Comment le démontrer si cela est possible ?
caract_outliers = df_merge_2[df_merge_2["price"]>seuil_price]
caract_outliers
```

Out[127]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable
30	4594.0	1.0	144.0	0.0	NaN	NaN	NaN	NaN
291	4352.0	1.0	225.0	0.0	15940	15940	0.0	0.0
293	4355.0	1.0	126.5	2.0	12589	12589	0.0	0.0
310	4402.0	1.0	176.0	8.0	3510	3510	0.0	0.0
313	4406.0	1.0	157.0	3.0	7819	7819	0.0	0.0
478	4904.0	1.0	137.0	13.0	14220	14220	0.0	0.0
525	5001.0	1.0	217.5	20.0	14581	14581	0.0	0.0
615	5612.0	1.0	124.8	12.0	14915	14915	0.0	0.0
657	5767.0	1.0	175.0	12.0	15185	15185	0.0	0.0
692	5892.0	1.0	191.3	10.0	14983	14983	0.0	0.0
708	5917.0	1.0	122.0	4.0	14775	14775	0.0	0.0
752	6126.0	1.0	135.0	10.0	14923	14923	0.0	0.0

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable
758	6202.0	1.0	116.4	14.0	15126	15126	0.0	0.0
763	6212.0	1.0	115.0	2.0	13996	13996	0.0	0.0
764	6213.0	1.0	121.0	7.0	15072	15072	0.0	0.0
766	6215.0	1.0	115.0	4.0	12790	12790	0.0	0.0
767	6216.0	1.0	121.0	6.0	15070	15070	0.0	0.0

17 rows × 41 columns

Entrée [128]:

```
caract_outliers["Appellation"].unique()
```

Out[128]:

```
array([nan, 'Champagne', 'Cognac Grande Champagne 1er Cru',
       'Corton Charlemagne', 'Charmes-Chambertin Grand Cru',
       'Alsace Grand Cru', 'Clos de Vougeot', 'Coteaux Champenois',
       'Echezeaux Grand Cru', 'Volnay 1er Cru'], dtype=object)
```

Entrée [129]:

```
caract_outliers["post_name"].unique()
```

Out[129]:

```
array([nan, 'champagne-egly-ouriet-grand-cru-millesime-2008',
       'champagne-egly-ouriet-grand-cru-brut-blanc-de-noirs',
       'cognac-frapin-vip-xo',
       'cognac-frapin-chateau-de-fontpinot-1989-20-ans',
       'domaine-des-croix-corton-charlemagne-grand-cru-2016',
       'david-duband-charmes-chambertin-grand-cru-2014',
       'domaine-weinbach-gewurztraminer-gc-furstentum-sgn-2010',
       'camille-giroud-clos-de-vougeot-2016',
       'coteaux-champenois-egly-ouriet-ambonnay-rouge-2016',
       'wemyss-malts-single-cask-scotch-whisky-choc-n-nut-pretzel-2001',
       'champagne-gosset-celebris-vintage-2007',
       'domaine-clerget-echezeaux-en-orveaux-2015',
       'domaine-des-comtes-lafon-volnay-1er-cru-santenots-du-milieu-2015',
       'domaine-des-comtes-lafon-volnay-1er-cru-santenots-du-milieu-2016',
       'domaine-des-comtes-lafon-volnay-1er-cru-champans-2014',
       'domaine-des-comtes-lafon-volnay-1er-cru-champans-2016'],
      dtype=object)
```

Entrée [130]:

```
# A CHAQUE FOIS LES PRODUITS QUI ON A DES PRIX SUPERIEUR C'EST LES ALCOOLS CHERE
```

Etape 5 - Analyse univarié du CA et des quantités vendues

Etape 5.1 - Analyse des ventes en CA

Entrée [131]:

```
##############################
# Calculer le CA su site web #
##############################

#Créez une colonne calculant le CA par article

#Calculez la somme de la colonne "ca_par_article"
#Ce résultat correspond au chiffre d'affaire du site web
```

Entrée [132]:

```
#CHIFFRE D'AFFAIRES = PRIX DE VENTE * QUANTITÉS VENDUES
df_merge_2["ca_par_article"] = df_merge_2["price"] * df_merge_2["total_sales"]
df_merge_2["ca_par_article"]
```

Out[132]:

```
0      145.2
1        0.0
2        0.0
3       42.3
4        0.0
       ...
820      0.0
821      0.0
822      NaN
823      NaN
824      0.0
Name: ca_par_article, Length: 825, dtype: float64
```

Entrée [133]:

```
#chiffre d'affaire du site web:
ca_total=df_merge_2["ca_par_article"].sum().round(2)
ca_total
```

Out[133]:

```
70318.6
```

Entrée [134]:

```
df_merge_2
```

Out[134]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	ratin
0	3847.0	1.0	24.2	0.0	15298	15298	0.0	0.0	
1	3849.0	1.0	34.3	0.0	15296	15296	0.0	0.0	
2	3850.0	1.0	20.8	0.0	15300	15300	0.0	0.0	
3	4032.0	1.0	14.1	0.0	19814	19814	0.0	0.0	
4	4039.0	1.0	46.0	0.0	19815	19815	0.0	0.0	
...	...	...	...	...	...	...	...	...	
820	7023.0	1.0	27.5	15.0	15891	15891	0.0	0.0	
821	7025.0	1.0	69.0	2.0	15887	15887	0.0	0.0	
822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	NaN	
823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	NaN	
824	7338.0	1.0	16.3	45.0	16230	16230	0.0	0.0	

825 rows × 42 columns

Entrée [135]:

```
################################
# Palmares des articles en CA #
################################

#Effectuer le tri dans l'ordre décroissant du CA du dataset df_merge

#Réinitialiser l'index du dataset par un reset_index

#Afficher les 20 premier articles en CA

#Graphique en barre des 20 premiers articles avec plotly express
```

Entrée [136]:

```
df_merge_2=df_merge_2.sort_values('ca_par_article',ascending=False)
df_merge_2
```

Out[136]:

	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable	r
286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.0	
162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.0	
310	4402.0	1.0	176.0	8.0	3510	3510	0.0	0.0	
161	4142.0	1.0	53.0	8.0	11641	11641	0.0	0.0	
160	4141.0	1.0	39.0	1.0	304	304	0.0	0.0	
...	...	...	...	...	...	...	...	...	
718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	NaN	
720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	NaN	
743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	NaN	
822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	NaN	
823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	NaN	

825 rows × 42 columns

Entrée [137]:

```
df_merge_2.reset_index(inplace = True)
df_merge_2
```

Out[137]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadabl
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.
2	310	4402.0	1.0	176.0	8.0	3510	3510	0.0	0.
3	161	4142.0	1.0	53.0	8.0	11641	11641	0.0	0.
4	160	4141.0	1.0	39.0	1.0	304	304	0.0	0.
...	...	...	...	...	...	...	...	...	
820	718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	Na
821	720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	Na
822	743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	Na
823	822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	Na
824	823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	Na

825 rows × 43 columns

Entrée [138]:

```
df_pre=df_merge_2.sort_values('ca_par_article',ascending=False).head(20)
df_pre
```

Out[138]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.0
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.0
2	310	4402.0	1.0	176.0	8.0	3510	3510	0.0	0.0
3	161	4142.0	1.0	53.0	8.0	11641	11641	0.0	0.0
4	160	4141.0	1.0	39.0	1.0	304	304	0.0	0.0
5	293	4355.0	1.0	126.5	2.0	12589	12589	0.0	0.0
6	291	4352.0	1.0	225.0	0.0	15940	15940	0.0	0.0
7	170	4153.0	1.0	29.0	0.0	16237	16237	0.0	0.0
8	761	6206.0	1.0	25.2	120.0	16580	16580	0.0	0.0
9	121	4068.0	1.0	16.6	157.0	16416	16416	0.0	0.0
10	17	4053.0	1.0	44.3	16.0	13127	13127	0.0	0.0
11	322	4596.0	1.0	43.9	0.0	15476	15476	0.0	0.0
12	470	4891.0	1.0	27.9	0.0	15807	15807	0.0	0.0
13	546	5067.0	1.0	59.9	0.0	15346	15346	0.0	0.0

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadable
14	762	6207.0	1.0	25.2	363.0	16077	16077	0.0	0.0
15	487	4918.0	1.0	37.2	0.0	15533	15533	0.0	0.0
16	710	5922.0	1.0	48.5	0.0	15343	15343	0.0	0.0
17	18	4054.0	1.0	71.6	0.0	19816	19816	0.0	0.0
18	273	4286.0	1.0	69.8	4.0	16066	16066	0.0	0.0
19	478	4904.0	1.0	137.0	13.0	14220	14220	0.0	0.0

20 rows × 43 columns

Entrée [139]:

```
df_pre["produit"] = df_pre["product_id"].astype(str)
df_pre["produit"]
```

Out[139]:

```
0     4334.0
1     4144.0
2     4402.0
3     4142.0
4     4141.0
5     4355.0
6     4352.0
7     4153.0
8     6206.0
9     4068.0
10    4053.0
11    4596.0
12    4891.0
13    5067.0
14    6207.0
15    4918.0
16    5922.0
17    4054.0
18    4286.0
19    4904.0
Name: produit, dtype: object
```

Entrée [140]:

```
fig=px.bar(df_pre,x='produit',y='ca_par_article',title='Graphique en barre des 20 premiers
fig.show()
```

Entrée [141]:

```
fig=px.bar(df_pre,x='produit',y='price',title='Graphique en barre des 20 premiers articles'
fig.show()
```

Entrée [142]:

```
##############################
# Calculer le 20 / 80 en CA #
##############################

#Créer une colonne calculant la part du CA de la ligne dans le dataset

#Créer une colonne réalisant la somme cumulative de la colonne précedemment créée

#Grâce au deux colonnes créées précedemment, calculer le nombre d'articles représentant 80%

#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier du s
```

Entrée [143]:

```
df_merge_2["part_ca"]=(df_merge_2["ca_par_article"]/ca_total)*100
df_merge_2["part_ca"]
```

Out[143]:

```
0      6.689553
1      6.062407
2      3.253762
3      2.261137
4      2.218474
         ...
820         NaN
821         NaN
822         NaN
823         NaN
824         NaN
Name: part_ca, Length: 825, dtype: float64
```

Entrée [144]:

```
df_merge_2["part_ca"].sum()
```

Out[144]:

```
100.0
```

Entrée [145]:

```
df_merge_2["cumulative_sum"] = df_merge_2["part_ca"].cumsum()
df_merge_2["cumulative_sum"]
```

Out[145]:

```
0       6.689553
1      12.751960
2      16.005723
3      18.266860
4      20.485334
         ...
820          NaN
821          NaN
822          NaN
823          NaN
824          NaN
Name: cumulative_sum, Length: 825, dtype: float64
```

Entrée [146]:

```
df_ca_80=df_merge_2.loc[df_merge_2["cumulative_sum"]<=80]
df_ca_80
```

Out[146]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadab
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0
2	310	4402.0	1.0	176.0	8.0	3510	3510	0.0	0
3	161	4142.0	1.0	53.0	8.0	11641	11641	0.0	0
4	160	4141.0	1.0	39.0	1.0	304	304	0.0	0
...	...	...	...	...	...	...	...	...	
125	642	5722.0	1.0	7.1	118.0	15026	15026	0.0	0
126	169	4152.0	1.0	19.2	57.0	16238	16238	0.0	0
127	693	5893.0	1.0	26.6	65.0	13910	13910	0.0	0
128	136	4085.0	1.0	19.0	37.0	16462	16462	0.0	0
129	380	4674.0	1.0	19.0	41.0	16056	16056	0.0	0

130 rows × 45 columns

Entrée [147]:

```
df_ca_80.shape[0]
```

Out[147]:

```
130
```

Entrée [148]:

```
(len(df_ca_80)/len(df_merge_2))*100
```

Out[148]:

```
15.757575757575756
```

Etape 5.2 - Analyse des ventes en Quantités

Entrée [149]:

```
################################
# Palmares des articles en quantité #
################################

#Effectuer le tri dans l'ordre décroissant de quantités vendues du dataset df_merge

#Réinitialiser l'index du dataset par un reset_index

#Afficher les 20 premier articles en quantité

#Graphique en barre des 20 premiers articles avec plotly express
```

Entrée [150]:

```
df_merge_2=df_merge_2.sort_values('total_sales',ascending=False)
df_merge_2
```

Out[150]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadabl
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.
9	121	4068.0	1.0	16.6	157.0	16416	16416	0.0	0.
50	210	4200.0	1.0	5.8	190.0	16295	16295	0.0	0.
56	188	4172.0	1.0	5.7	167.0	16210	16210	0.0	0.
...	...	...	...	...	...	...	...	...	.
820	718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	Nal
821	720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	Nal
822	743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	Nal
823	822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	Nal
824	823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	Nal

825 rows × 45 columns

Entrée [151]:

```
df_merge_2.reset_index(drop= True)
```

Out[151]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadabl
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.
2	121	4068.0	1.0	16.6	157.0	16416	16416	0.0	0.
3	210	4200.0	1.0	5.8	190.0	16295	16295	0.0	0.
4	188	4172.0	1.0	5.7	167.0	16210	16210	0.0	0.
...	...	...	...	...	...	...	...	...	.
820	718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	Nal
821	720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	Nal
822	743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	Nal
823	822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	Nal
824	823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	Nal

825 rows × 45 columns

Entrée [152]:

```
df_pre=df_merge_2.sort_values('total_sales',ascending=False).head(20)
df_pre
```

Out[152]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloac
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	
9	121	4068.0	1.0	16.6	157.0	16416	16416	0.0	
50	210	4200.0	1.0	5.8	190.0	16295	16295	0.0	
56	188	4172.0	1.0	5.7	167.0	16210	16210	0.0	
25	200	4187.0	1.0	13.3	90.0	16189	16189	0.0	
8	761	6206.0	1.0	25.2	120.0	16580	16580	0.0	
4	160	4141.0	1.0	39.0	1.0	304	304	0.0	
35	734	6047.0	1.0	10.9	46.0	16264	16264	0.0	
41	418	4729.0	1.0	8.6	151.0	38	38	0.0	
14	762	6207.0	1.0	25.2	363.0	16077	16077	0.0	
12	470	4891.0	1.0	27.9	0.0	15807	15807	0.0	
7	170	4153.0	1.0	29.0	0.0	16237	16237	0.0	
42	463	4870.0	1.0	9.3	0.0	16149	16149	0.0	
26	398	4706.0	1.0	16.8	23.0	15349	15349	0.0	

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloac
3	161	4142.0	1.0	53.0	8.0	11641	11641	0.0	
23	246	4250.0	1.0	19.5	14.0	16317	16317	0.0	
55	456	4861.0	1.0	8.5	284.0	15307	15307	0.0	
83	513	4965.0	1.0	7.1	203.0	16586	16586	0.0	
122	665	5778.0	1.0	5.8	36.0	15561	15561	0.0	

20 rows × 45 columns

Entrée [153]:

```
df_pre["produit"] = df_pre["product_id"].astype(str)
df_pre["produit"]
```

Out[153]:

```
0      4334.0
1      4144.0
9      4068.0
50     4200.0
56     4172.0
25     4187.0
8      6206.0
4      4141.0
35     6047.0
41     4729.0
14     6207.0
12     4891.0
7      4153.0
42     4870.0
26     4706.0
3      4142.0
23     4250.0
55     4861.0
83     4965.0
122    5778.0
Name: produit, dtype: object
```

Entrée [154]:

```
fig=px.bar(df_pre,x='produit',y='total_sales',title='Graphique en barre des 20 premiers art
fig.show()
```

Entrée [155]:

```
##############################
# Calculer le 20 / 80 en CA #
##############################

#Créer une colonne calculant la part en quantité de la ligne dans le dataset

#Créer une colonne réalisant la somme cumulative de la colonne précedemment créée

#Grâce au deux colonnes créées précedemment, calculer le nombre d'articles représentant 80%

#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier du s
```

Entrée [156]:

```
qu_total=df_merge_2['total_sales'].sum()
qu_total
```

Out[156]:

```
2855.0
```

Entrée [157]:

```
df_merge_2["part_qu"]=(df_merge_2["total_sales"]/qu_total)*100
df_merge_2["part_qu"]
```

Out[157]:

```
0      3.362522
1      3.047285
9      2.171629
50     1.611208
56     1.506130
         ...
820         NaN
821         NaN
822         NaN
823         NaN
824         NaN
Name: part_qu, Length: 825, dtype: float64
```

Entrée [158]:

```
df_merge_2["part_qu"].sum().round(0)
```

Out[158]:

```
100.0
```

Entrée [159]:

```
df_merge_2["cumulative_qu"] = df_merge_2["part_qu"].cumsum()
df_merge_2["cumulative_qu"]
```

Out[159]:

```
0       3.362522
1       6.409807
9       8.581436
50     10.192644
56     11.698774
         ...
820          NaN
821          NaN
822          NaN
823          NaN
824          NaN
Name: cumulative_qu, Length: 825, dtype: float64
```

Entrée [160]:

```
df_qu_80=df_merge_2.loc[df_merge_2["cumulative_qu"]<=80]
df_qu_80
```

Out[160]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadabl
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.
9	121	4068.0	1.0	16.6	157.0	16416	16416	0.0	0.
50	210	4200.0	1.0	5.8	190.0	16295	16295	0.0	0.
56	188	4172.0	1.0	5.7	167.0	16210	16210	0.0	0.
...	...	...	...	...	...	...	...	...	.
103	432	4757.0	1.0	26.5	1.0	14680	14680	0.0	0.
137	183	4166.0	1.0	20.5	8.0	16124	16124	0.0	0.
108	483	4912.0	1.0	25.9	10.0	15612	15612	0.0	0.
110	158	4138.0	1.0	25.7	0.0	15341	15341	0.0	0.
115	488	4919.0	1.0	24.4	0.0	15531	15531	0.0	0.

150 rows × 47 columns

Entrée [161]:

```
df_qu_80.shape[0]
```

Out[161]:

```
150
```

Entrée [162]:

```
(len(df_qu_80)/len(df_merge_2))*100
```

Out[162]:

```
18.181818181818183
```

Entrée [163]:

```
df_merge_2
```

Out[163]:

	index	product_id	onsale_web	price	stock_quantity	id_web	sku	virtual	downloadabl
0	286	4334.0	1.0	49.0	0.0	7818	7818	0.0	0.
1	162	4144.0	1.0	49.0	11.0	1662	1662	0.0	0.
9	121	4068.0	1.0	16.6	157.0	16416	16416	0.0	0.
50	210	4200.0	1.0	5.8	190.0	16295	16295	0.0	0.
56	188	4172.0	1.0	5.7	167.0	16210	16210	0.0	0.
...	...	...	...	...	...	...	...	...	.
820	718	5955.0	0.0	27.3	0.0	14377	NaN	NaN	Nal
821	720	5957.0	0.0	39.0	0.0	13577	NaN	NaN	Nal
822	743	6100.0	0.0	12.9	0.0	15529	NaN	NaN	Nal
823	822	7247.0	1.0	54.8	23.0	13127-1	NaN	NaN	Nal
824	823	7329.0	0.0	26.5	14.0	14680-1	NaN	NaN	Nal

825 rows × 47 columns

Etape 5.3 - Mettre à disposition la nouvelle table sur un fichier Excel

Entrée [164]:

```
#Mettre le dataset df_merge sur un fichier Excel
#Cette étape peut-être utile pour partager le résultat du dataset obtenu pour le partager a
```

Entrée [165]:

```
df_merge_2.to_excel('df_final.xlsx',index=False)
```

Entrée [ ]:

```

```