

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'une 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

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

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

```
#Trouver dans Google l'instruction permettant d'afficher toutes les colonnes d'un data
#Saisir, dans Google, les mots clés "display all columns dataframe Pandas", par exemple
#Dans les résultats de la recherche, privilégiez les solutions provenant de Stack Overflow
```

1.2 - Chargements des fichiers

Montage du Drive Google pour accéder aux fichiers Excel qui y sont stockés.

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
#Importation du fichier web.xlsx
df_web = pd.read_excel("/content/drive/MyDrive/Datasets/Bottleneck/web.xlsx")
#Importation du fichier erp.xlsx
df_erp = pd.read_excel("/content/drive/MyDrive/Datasets/Bottleneck/erp.xlsx")
#Importation du fichier liaison.xlsx
df_liaison = pd.read_excel("/content/drive/MyDrive/Datasets/Bottleneck/liaison.xlsx")
```

```
Warning: /usr/local/lib/python3.10/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: warn(msg)
Warning: /usr/local/lib/python3.10/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: warn(msg)
Warning: /usr/local/lib/python3.10/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: warn(msg)
```

Le warning message ne semble pas avoir d'incidence sur l'importation des données.

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier erp.xlsx

```
#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]))
```

Le tableau comporte 825 observation(s) ou article(s)
Le tableau comporte 6 colonne(s)

```
#Consulter le nombre de colonnes
print("Le tableau erp comporte {} colonne(s)".format(df_erp.shape[1]))
```

Le tableau erp comporte 6 colonne(s)

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  -
0   product_id            825 non-null   int64
1   onsale_web            825 non-null   int64
2   price                 825 non-null   float64
3   stock_quantity        825 non-null   int64
4   stock_status          825 non-null   object
5   purchase_price        825 non-null   float64
dtypes: float64(2), int64(3), object(1)
memory usage: 38.8+ KB
```

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

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price
0	3847	1	24.2	16	instock	12.88
1	3849	1	34.3	10	instock	17.54
2	3850	1	20.8	0	outofstock	10.64
3	4032	1	14.1	26	instock	6.92
4	4039	1	46.0	3	outofstock	23.77

```
#Vérifier si il y a les lignes en doublons dans la colonne product_id
print("Le tableau comporte {} doublon(s) dans la colonne product_id".format(df_erp.duplicated().sum()))
```

Le tableau comporte 0 doublon(s) dans la colonne product_id

```
print("Absence de doublons dans la table: {}".format(df_erp["product_id"].unique().size == df_erp.shape[0]))
```

Absence de doublons dans la table: True

```
#Afficher les valeurs distinctes de la colonne stock_status
print("Les valeurs distinctes de la colonne stock_status sont les suivantes :", df_erp["stock_status"].unique())
```

Les valeurs distinctes de la colonne stock_status sont les suivantes : ['instock', 'outofstock']

```
# À quelle(s) autre(s) colonne(s) sont-elles liées ?
```

Le stock_status est lié au stock_quantity. Quand le stock_quantity est = 0, le stock_status = outofstock. Sinon stock_status = instock.

```
#Création d'une colonne "stock_status_2"
#La valeur de cette deuxième colonne sera fonction de la valeur dans la colonne "stock_quantity"
#si la valeur de la colonne "stock_quantity" est nulle renseigner "outofstock" sinon "instock"
df_erp["stock_status_2"] = df_erp["stock_quantity"].apply(lambda x: "outofstock" if x == 0 else "instock")
```



Imnou Macodou Ba
14 juin 2024



df_erp.info()



Alexandre Duvernay
25 juin 2024



Fait



Imnou Macodou Ba
14 juin 2024



```
#Vérifier si il y a les lignes en doublons dans la colonne product_id
print("1er méthode - Absence de doublons dans la table: {}".format(df_erp["product_id"].drop_duplicates().size == df_erp.shape[0]))
#autre méthode
print("2eme méthode - Absence de doublons dans la table: {}".format(df_erp["product_id"].unique().size == df_erp.shape[0]))
```



Alexandre Duvernay
25 juin 2024



modifié

```
#Vérifions que les 2 colonnes sont identiques:
#Les 2 colonnes sont strictement identiques si les valeurs de chaque ligne sont strict
#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
```

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

```
#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
print("La somme de cette colonne indique une égalité entre {} lignes.".format(df_erp["stock_status"].sum()))
print("Il y a donc {} lignes inégales.".format(df_erp["stock_status"] != df_erp["stock_status_2"].sum()))
```

```
La somme de cette colonne indique une égalité entre 821 lignes.
Il y a donc 4 lignes inégales.
```

```
#Si les colonnes ne sont absolument pas identiques ligne à ligne alors identifier la ligne à l'origine du problème
##Dans ce cas je vous propose ce lien pour apprendre à réaliser des filtres dans Pandas:
##https://bitbucket.org/hrojas/learn-pandas/src/master/
##Lesson 3
print("Les lignes comportant un écart sont les suivantes :")
df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
```

```
Les lignes comportant un écart sont les suivantes :
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	stock_status_2
4	4039	1	46.0	3	outofstock	23.77	instock
398	4885	1	18.7	0	instock	9.66	outofstock
449	4973	0	10.0	-10	outofstock	4.96	outofstock
573	5700	1	44.5	-1	outofstock	22.30	outofstock

```
#Corriger la ou les données incohérentes
```

```
# Remplacement de la valeur "stock_status" pour le product_id 4039
df_erp.loc[df_erp['product_id'] == 4039, 'stock_status'] = 'instock'
```

Le produit 4039 est en stock comme le confirme la stock_quantity qui est de 3. Donc on passe le stock_status à instock.

```
# Remplacement de la valeur "stock_status" pour le product_id 4885
df_erp.loc[df_erp['product_id'] == 4885, 'stock_status'] = 'outofstock'
```

Le produit 4885 n'est pas en stock comme le confirme la stock_quantity qui est à 0. Donc on passe le stock_status à outofstock.

```
# Remplacement de la valeur "stock_status_2" pour le product_id 4973
df_erp.loc[df_erp['product_id'] == 4973, 'stock_status_2'] = 'outofstock'
```

```
# Remplacement de la valeur "stock_status_2" pour le product_id 5700
df_erp.loc[df_erp['product_id'] == 5700, 'stock_status_2'] = 'outofstock'
```

Les produits 4973 et 5700 indiquent un stock_quantity négatif ce qui pourrait indiquer des produits en commande. On passe donc leur stock_status à outofstock.

```
#Vérification en utilisant le même code que plus haut pour afficher les problèmes
print("Après ce nettoyage, il nous reste {} lignes inégales.".format(df_erp["stock_status"] != df_erp["stock_status_2"].sum()))
```

```
Après ce nettoyage, il nous reste 0 lignes inégales.
```

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

2.1.1.1 - Analyse de la variable PRIX

```
#####
## 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"
print("Nombres d'article avec un prix non renseignés: {}".format(df_erp['price'].isnul
```

```
↳ Nombres d'article avec un prix non renseignés: 0
```

```
#Afficher le prix minimum de la colonne "price"
print("Le prix minimum dans la colonne 'price' est de: {}".format(df_erp['price'].min()
```

```
↳ Le prix minimum dans la colonne 'price' est de: -20.0
```

```
#Afficher le prix maximum de la colonne "price"
print("Le prix maximum dans la colonne 'price' est de: {}".format(df_erp['price'].max()
```

```
↳ Le prix maximum dans la colonne 'price' est de: 225.0
```

```
#Affichier les prix inférieurs à 0 (qu'est ce qu'il faut en faire ?)
df_erp[df_erp['price'] < 0]
```

```
↳
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	st
151	4233	0	-20.0	0	outofstock	10.33	
469	5017	0	-8.0	0	outofstock	4.34	
739	6594	0	-9.1	19	instock	4.61	

Que faire de ces valeurs négatives ? Les supprimer, les mettre à 0, les passer en positif ?

Les prix de vente étant corrélés aux prix d'achat, il faut donc les comparer afin de voir la relation qui les unie.

Usuellement une marge est appliquée entre le prix d'achat et le prix de vente (un coefficient multiplicateur).

Sur les 3 lignes concernées, on s'aperçoit que si l'on retire le signe "-", il y a pratiquement un coef 2, ce qui est pertinent.

*Je préconise donc le passage de ces 3 prix en positif. *

```
# Modification de toutes les valeurs de la colonne "price" pour les passer en valeur a
df_erp["price"] = abs(df_erp["price"])
```

```
# Vérification qu'il ne reste pas de prix inférieur à 0
print("Le nombre de prix inférieurs à 0 restant est de : {}".format(df_erp['price'].lt
```

```
↳ Le nombre de prix inférieurs à 0 restant est de : 0
```

2.1.1.2 - Analyse de la variable STOCK

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

```
#Vérification de la colonne stock quantity
print("Nombre d'articles ayant une quantité non renseignée: {}".format(df_erp['stock_c
```

```
↳ Nombre d'articles ayant une quantité non renseignée: 0
```

```
#Afficher la quantité minimum de la colonne "stock_quantity"
print("Le stock minimum dans la colonne 'stock_quantity' est de: {}".format(df_erp['st
```

```
↳ Le stock minimum dans la colonne 'stock_quantity' est de: -10
```

```
#Afficher la quantité maximum de la colonne "stock_quantity"
print("Le stock maximum dans la colonne 'stock_quantity' est de: {}".format(df_erp['st
```

```
↗ Le stock maximum dans la colonne 'stock_quantity' est de: 145
```

```
#Affichier les stocks inférieurs à 0 (qu'est ce qu'il faut en faire ?)
df_erp[df_erp['stock_quantity'] < 0]
```

```
↗
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	st
449	4973	0	10.0	-10	outofstock	4.96	
573	5700	1	44.5	-1	outofstock	22.30	

Que faire d'un stock inférieur à 0 ?

Un stock négatif pourrait indiquer des produits vendus qui n'étaient plus en stock. Donc des produits potentiellements en commande.

Si c'est le cas, les valeurs devraient rester telles quelles. Sinon il pourrait s'agir d'erreurs de stock, auquel cas les valeurs devraient être corrigées lors d'un inventaire.

2.1.1.3 - Analyse de la variable ONSALE_WEB

```
#Vérification de la colonne onsale_web et des valeurs qu'elle contient? Que signifient
print("La colonne 'onsale_web' contient les valeurs suivantes :", df_erp['onsale_web'])
```

```
↗ La colonne 'onsale_web' contient les valeurs suivantes : [1 0]
```

La colonne 'onsale_web' semble spécifier si le produit est à la vente sur le site web ou non (de façon booléenne): 1 = oui et 0 = non.

```
#Quelles sont les colonnes à conserver selon vous?
```

Les colonnes à conserver sont :

- product_id
- purchase_price
- price
- stock_quantity

Les autres colonnes n'ont pas d'intérêt pour l'analyse attendue ou sont redondantes (correlation entre 'stock_quantity', 'stock_status' et 'stock_status_2').

```
#Supprimer la colonnecomportant le libellé "stock_status_2" car elle est redondante
#avec la colonne "stock_status".
del df_erp['stock_status_2']
```

```
df_erp.head(0)
```

```
↗
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price
--	------------	------------	-------	----------------	--------------	----------------

2.1.1.4 - Analyse de la variable prix d'achat

```
#####
##  prix d'achat  ##
#####
```

```
#Vérification de la colonne purchase_price :
```

```
#Afficher le ou les prix non renseignés dans la colonne "purchase_price"
print("Nombres d'article ayant un prix d'achat non renseignés: {}".format(df_erp['purc
```

```
↗ Nombres d'article ayant un prix d'achat non renseignés: 0
```

```
#Afficher le prix minimum de la colonne "purchase_price"
print("Le prix d'achat minimum dans la colonne 'purchase_price' est de: {}".format(df_
```

```
↗ Le prix d'achat minimum dans la colonne 'purchase_price' est de: 2.74
```

```
#Afficher le prix maximum de la colonne "purchase_price"
print("Le prix d'achat maximum dans la colonne 'purchase_price' est de: {}".format(df_

↵ Le prix d'achat maximum dans la colonne 'purchase_price' est de: 137.81
```

2.2 - Analyse exploratoire du fichier web.xlsx

```
#Dimension du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_web.shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_web.shape[1]))
#Nombre d'observations

#Nombre de caractéristiques

↵ Le tableau comporte 1513 observation(s) ou article(s)
Le tableau comporte 29 colonne(s)

#Consulter le nombre de colonnes
print("Le tableau web comporte {} colonne(s)".format(df_web.shape[1]))

↵ Le tableau web comporte 29 colonne(s)

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

```
↵ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1513 entries, 0 to 1512
Data columns (total 29 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                ---
0   sku                                  1428 non-null   object
1   virtual                             1513 non-null   int64
2   downloadable                         1513 non-null   int64
3   rating_count                        1513 non-null   int64
4   average_rating                       1430 non-null   float64
5   total_sales                         1430 non-null   float64
6   tax_status                           716 non-null    object
7   tax_class                           0 non-null      float64
8   post_author                         1430 non-null   float64
9   post_date                           1430 non-null   datetime64[ns]
10  post_date_gmt                       1430 non-null   datetime64[ns]
11  post_content                         0 non-null      float64
12  product_type                        1429 non-null   object
13  post_title                           1430 non-null   object
14  post_excerpt                         716 non-null    object
15  post_status                         1430 non-null   object
16  comment_status                      1430 non-null   object
17  ping_status                         1430 non-null   object
18  post_password                       0 non-null      float64
19  post_name                           1430 non-null   object
20  post_modified                       1430 non-null   datetime64[ns]
21  post_modified_gmt                   1430 non-null   datetime64[ns]
22  post_content_filtered               0 non-null      float64
23  post_parent                         1430 non-null   float64
24  guid                                1430 non-null   object
25  menu_order                          1430 non-null   float64
26  post_type                           1430 non-null   object
27  post_mime_type                      714 non-null    object
28  comment_count                       1430 non-null   float64
dtypes: datetime64[ns](4), float64(10), int64(3), object(12)
memory usage: 342.9+ KB
```

```
df_web.iloc[:, 15:20].head()
```

```
↵
```

	post_status	comment_status	ping_status	post_password	post_name
0	publish	closed	closed	NaN	gilles-robin-hermitage-2012
1	publish	closed	closed	NaN	pelle-sancerre-rouge-la-croix-au-garde-2017
2	publish	closed	closed	NaN	fonreaud-bordeaux-blanc-le-cygne-2016
3	publish	closed	closed	NaN	moulin-de-gassac-igp-pays-dherault-auilhem-ros...

```
#Selon vous, quelles sont les colonnes à conserver ?
```

A

Alexandre Duvernay

25 juin 2024

✓⋮

df_web.info() mis à jour

Les valeurs à conserver sont :

- sku
- total_sales
- product_type
- post_date
- post_modified
- post_title
- post_type (permet de vérifier s'il s'agit d'un produit ou d'un fichier joint)

```
#Si vous avez défini des colonnes à supprimer, effectuer l'opération
# Liste des colonnes à supprimer
columns_to_drop = ['virtual', 'downloadable', 'rating_count', 'average_rating', 'tax_s

# Suppression des colonnes spécifiées
df_web = df_web.drop(columns=columns_to_drop)
```

```
df_web.head()
```



	sku	total_sales	post_date	product_type	post_title	post_modified	post_type
0	11862	3.0	2018-02-12 13:46:23	Vin	Gilles Robin Hermitage Rouge 2012	2019-01-31 12:12:56	attachment
1	16057	5.0	2018-04-17 15:29:17	Vin	Domaine Pellé Sancerre Rouge La Croix Au Gardon	2020-07-07 10:05:02	attachment

```
#Visualisation des valeurs de la colonne sku
# Visualisation des valeurs uniques de la colonne SKU
print("Les valeurs uniques de la colonne SKU sont les suivantes :")
df_web['sku'].unique()
```



15038, 14265, 15238, 16586, 13073, 15664, 14527, 14845, 13172, 13996, 15864, 15440, 15759, 15706, 11277, 15676, 15731, 16525, 15785, 14699, 16063, 15718, 14372, 15881, 793, 14775, 12771, 15426, 13291, 16023, 15713, 15675, 15282, 15927, 15351, 12791, '13127-1', 13853, 15613, 11849, 15478, 16121, 16237, 15461, 15745, 15621, 15683, 12639, 15466, 15184, 13078, 13904, 14828, 14000, 15612, 304, 15811, 15649, 14599, 15033, 13959, 15026, 16449, 15095, 15829, 13965, 16306, 16276, 15729, 15303, 14679, 15318, 13599, 15206, 19816, 16153, 16067, 15241, 14981, 16264, 14915, 15737, 16289, 15229, 7033, 15654, 15879, 16256, 15710, 16498, 15349, 11933, 16462, 14596, 15296, 16042, 15887, 15264, 15307, 15369, 14095, 15868, 14371, 12657, 15808, 15787, 13604, 15605, 15795, 15566, 16210, 15732, 15661, 13814, 16130, 16003, 14676, 13516, 14977, 14945, 12641, 14469, 15237, 14905, 15462, 16031, 16191, 3568, 15080, 15353, 15861, 16529, 16068, 16081, 14729, 16077, 15373, 11049, 16154, 14809, 15073, 15345, 15756, 13531, 15614, 7086, 15784, 14332, 14300, 14141, 15734, 15456, 805, 15766, 15667, 16152, 16318, 15360, 13517, 7819, 15740, 15930, 16155, 15339, 14950, 2179, 15688, 13982, 38, 15178, 16209, 14982, 8463, 13412, 15298, 15770, 15487, 15205, 13230, 14220, 15227, 16094, 15859, 15179, 14774, 14944, 16022, 15767, 15338, 13913, 14756, 15746, 16192, 15763, 12365, 9562, 16039, 16328, 15196, 531, 15810, 2361, 15402, 15945, 15807, 14506, 14100, 15075, 13765, 16093, 10814, 12942, 1366, 12857, 12045, 13515, 11601, 16148, 12339, 16567, 15240, 15414, 812, 16292, 11736, 15714, 15120, 13117, 3383, 15631, 15136, 15145,
--

```
13074, 10331, 41, 13702, 14370, 10041, 13202, 13100, 13373, 10040,
16034, 14092, 14241, 11997, 15922, 14844, 15444, 15486, 15378, 807,
13567, 16281, 15213, 16166, 11225, 15773, 16160, 19820, 7818,
14751, 15280, 15933, 13627, 15403, 14580, 16024, 14797, 12494,
16229, 13659, 16578, 14802, 15850, 16159, 15839, 13052, 15869,
14604, 16273, 15329, 14253, 14941, 15567, 16132, 14746, 15656,
16527, 15162, 16326, 16501, 16119, 15662, 15344, 6616, 15958,
14865, 11669, 14395, 15149, 12585, 11668, 11847, 12599, 16133,
15665, 15382, 13209, 14461, 15748, 15663, 15072, 15004, 16069,
16180, 15949, 16149, 16043, 15845, 15951, 13074, 14569],
dtype=object)
```

#Quelles sont les valeurs qui ne semblent pas respecter la règle de codification?

Le SKU semble être codé sur la base d'un entier absolu. Nous allons donc rechercher les valeurs qui ne respectent pas cette condition.

```
# Fonction pour vérifier si une valeur n'est pas un entier absolu
def is_not_absolute_integer(value):
    try:
        # Vérifier si la conversion en entier est possible et si c'est un entier positif
        return not (isinstance(int(value), int) and int(value) >= 0)
    except (ValueError, TypeError):
        # Si la conversion échoue, ce n'est pas un entier
        return True
```

```
# Appliquer la fonction pour filtrer les valeurs
non_absolute_integers = df_web['sku'].apply(is_not_absolute_integer)
```

```
# Obtenir les valeurs uniques qui ne sont pas des entiers absolus
unique_non_absolute_integers = df_web['sku'][non_absolute_integers].unique()
```

```
print("Valeurs uniques de 'sku' qui ne sont pas des entiers absolus :")
print(unique_non_absolute_integers)
```

```
↗ Valeurs uniques de 'sku' qui ne sont pas des entiers absolus :
[nan '13127-1' 'bon-cadeau-25-euros']
```

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

```
# Fonction pour filtrer les lignes
filtered_df = df_web[df_web['sku'].apply(is_not_absolute_integer)]
```

```
print("Lignes avec un 'sku' qui n'est pas un entier absolu :")
print(filtered_df)
```

```
↗ Lignes avec un 'sku' qui n'est pas un entier absolu :
```

	sku	total_sales	post_date	product_type	\
8	NaN	NaN	NaT	NaN	
20	NaN	NaN	NaT	NaN	
30	NaN	NaN	NaT	NaN	
37	NaN	NaN	NaT	NaN	
41	NaN	NaN	NaT	NaN	
...	...	...	...	...	
1387	bon-cadeau-25-euros	7.0	2018-06-01 13:53:46	NaN	
1429	NaN	NaN	NaT	NaN	
1432	NaN	NaN	NaT	NaN	
1445	NaN	NaN	NaT	NaN	
1457	NaN	NaN	NaT	NaN	

	post_title	post_modified	post_type
8	NaN	NaT	NaN
20	NaN	NaT	NaN
30	NaN	NaT	NaN
37	NaN	NaT	NaN
41	NaN	NaT	NaN
...	...	...	...
1387	Bon cadeau de 25€	2018-06-01 14:13:57	product
1429	NaN	NaT	NaN
1432	NaN	NaT	NaN
1445	NaN	NaT	NaN
1457	NaN	NaT	NaN

[89 rows x 7 columns]

On obtient 89 lignes qui ne respectent pas la règle de codification.


```
#Identifier les lignes sans code articles
filtered_df = df_web[df_web['sku'].isna()]
```

```
print("Lignes avec un 'sku' égal à NaN :")
print(filtered_df)
```

```
↳ Lignes avec un 'sku' égal à NaN :
```

	sku	total_sales	post_date	product_type	post_title	post_modified	\
8	NaN	NaN	NaT	NaN	NaN	NaN	NaT
20	NaN	NaN	NaT	NaN	NaN	NaN	NaT
30	NaN	NaN	NaT	NaN	NaN	NaN	NaT
37	NaN	NaN	NaT	NaN	NaN	NaN	NaT
41	NaN	NaN	NaT	NaN	NaN	NaN	NaT
...	...	...	...	...	...	...	...
1384	NaN	NaN	NaT	NaN	NaN	NaN	NaT
1429	NaN	NaN	NaT	NaN	NaN	NaN	NaT
1432	NaN	NaN	NaT	NaN	NaN	NaN	NaT
1445	NaN	NaN	NaT	NaN	NaN	NaN	NaT
1457	NaN	NaN	NaT	NaN	NaN	NaN	NaT

	post_type
8	NaN
20	NaN
30	NaN
37	NaN
41	NaN
...	...
1384	NaN
1429	NaN
1432	NaN
1445	NaN
1457	NaN

[85 rows x 7 columns]

On obtient 85 lignes sans code article.

```
#Pour les codes articles identifiés, réalisé une analyse et définissez l'action à entr
# Valeurs uniques de chaque colonne dans le DataFrame filtré
unique_values = {col: filtered_df[col].unique() for col in filtered_df.columns}
```

```
print("Valeurs uniques des colonnes dans les lignes où 'sku' est NaN :")
for col, values in unique_values.items():
    print(f"{col} : {values}")
```

```
↳ Valeurs uniques des colonnes dans les lignes où 'sku' est NaN :
```

```
sku : [nan]
total_sales : [ nan -56. -17.]
post_date : <DatetimeArray>
['NaT', '2018-08-08 11:23:43', '2018-07-31 12:07:23']
Length: 3, dtype: datetime64[ns]
product_type : [nan 'Vin']
post_title : [nan 'Pierre Jean Villa Condrieu Jardin Suspendu 2018'
 'Pierre Jean Villa Côte Rôtie Fongean 2017']
post_modified : <DatetimeArray>
['NaT', '2019-11-02 13:24:01', '2019-11-02 13:24:15']
Length: 3, dtype: datetime64[ns]
post_type : [nan 'product']
```

Le SKU étant la clé permettant de relier les valeurs de df_web avec df_erp via la table de liaison, si celui-ci est manquant dans le df_web, il sera impossible de faire le lien avec df_erp. Ces lignes pourraient donc être écartées à ce stade. Pour autant elles pourront l'être aussi naturellement suite à une jonction interne qui les écartera par défaut de correspondance avec la table de liaison.

```
#La clé pour chaque ligne est-elle uniques? ou autrement dit, y a-t-il des doublons?
df_web_non_null = df_web[df_web['sku'].notna()] # On Filtre les valeurs nulles dans la
num_duplicates = df_web_non_null['sku'].duplicated().sum() # On compte les doublons de
```

```
print("Nombre de doublons dans la colonne 'sku' (valeurs nulles exclues) :")
print(num_duplicates)
```

```
↳ Nombre de doublons dans la colonne 'sku' (valeurs nulles exclues) :
714
```

```
#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
df_web_sans_sku = df_web[df_web['sku'].isna()]
```

#2 - utiliser la fonction `df.info()` sur ce nouveau dataframe pour observer le nombre c
`df_web_sans_sku.info()`

```
<class 'pandas.core.frame.DataFrame'>
Index: 85 entries, 8 to 1457
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   sku                    0 non-null     object
1   total_sales            2 non-null     float64
2   post_date              2 non-null     datetime64[ns]
3   product_type           2 non-null     object
4   post_title             2 non-null     object
5   post_modified          2 non-null     datetime64[ns]
6   post_type              2 non-null     object
dtypes: datetime64[ns](2), float64(1), object(4)
memory usage: 5.3+ KB
```



Alexandre Duvernay
18 juin 2024



Il faudra revenir sur ce point pour voir si l'on peut récupérer les infos des 2 sku qui ne sont pas des entiers.

#3 - Que constatez-vous?

On constate que plusieurs lignes (2) sans sku contiennent des informations.

Ajout Alex

En arrivant à l'analyse du CA, on s'aperçoit de l'impact des doublons dans le résultat. Je reviens donc sur ce point pour traiter les doublons, les sku 'nuls' ou qui ne sont pas des nombres entiers. Ci-dessous.

```
df_web_filtered = df_web[pd.to_numeric(df_web['sku'], errors='coerce').notnull()] # or
df_web_filtered.loc[:, 'sku'] = df_web_filtered['sku'].astype(int) # on écarte les val

df_web_filtered = df_web_filtered[df_web_filtered['post_type'] == 'product'] # on écar

# df_web_sorted = df_web_filtered.sort_values(by='sku', ascending=True) # décommenter

# print(df_web_sorted) # Décommenter pour afficher le DataFrame trié

df_web = df_web_filtered # df_web prend la valeur df_web_filtered
```

2.3 - Analyse exploratoire du fichier liaison.xlsx

```
#Dimension du dataset
print("Dimensions du DataSet : ", df_liaison.shape)
```

```
Dimensions du DataSet : (825, 2)
```

```
#Nombre d'observations
print("Le fichier liaison comporte {} observations (ou lignes)".format(df_liaison.shap
```

```
Le fichier liaison comporte 825 observations (ou lignes)
```

```
#Nombre de caractéristiques
print("Le fichier liaison comporte {} caractéristiques (ou colonnes)".format(df_liaisc
```

```
Le fichier liaison comporte 2 caractéristiques (ou colonnes)
```

```
#Consulter le nombre de colonnes
```

Il y a donc 2 colonnes dans le DataFrame.

```
#La nature des données dans chacune des colonnes
#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):
#   Column      Non-Null Count  Dtype
---  ---
0   id_web      734 non-null    object
1   product_id  825 non-null    int64
dtypes: int64(1), object(1)
memory usage: 13.0+ KB
```



Alexandre Duvernay
25 juin 2024



[df_liaison.info\(\)](#) mis à jour

```
#Les valeurs de la colonne "product_id" sont elles toutes uniques?
print("La colonne 'product_id' comporte {} valeurs uniques sur 825 au total.".format(c

↵ La colonne 'product_id' comporte 825 valeurs uniques sur 825 au total.

#Les valeurs de la colonne "id_web" sont-elles toutes uniques?
print("La colonne 'id_web' comporte {} valeurs uniques sur 825 au total.".format(df_li

↵ La colonne 'id_web' comporte 734 valeurs uniques sur 825 au total.

#Avons-nous des articles sans correspondances?
print("Nous avons {} articles sans correspondance".format(df_liaison['id_web'].isnull(

↵ Nous avons 91 articles sans correspondance
```

Etape 3 - Jonction des fichiers

Etape 3.1 - Jonction du fichier df_erp et df_liaison

```
#Fusion des fichiers df_erp et df_liaison
df_erp_liaison = pd.merge(df_erp, df_liaison) # jointure naturelle

#Y a t-il des lignes ne "matchant" entre les 2 fichiers?
df_erp_liaison_oj = pd.merge(df_erp, df_liaison, how='outer', indicator=True) #jonctio
non_matching_rows = df_erp_liaison_oj[df_erp_liaison_oj['_merge'] != 'both']
print("Nombre de lignes ne matchant pas suite à la jonction :", non_matching_rows['_me

↵ Nombre de lignes ne matchant pas suite à la jonction : 0
```

Etape 3.2 - Jonction du fichier df_merge et df_web

```
#Fusionnez les datasets df_merge et df_web
df_erp_liaison_web = pd.merge(df_web, df_erp_liaison, left_on='sku', right_on='id_web'

#Avons-nous des lignes sans correspondances?
df_merged = pd.merge(df_web, df_erp_liaison, left_on='sku', right_on='id_web', how='ou
non_matching_rows = df_merged[df_merged['_merge'] != 'both'] # Identifier les lignes s
num_non_matching_rows = non_matching_rows.shape[0] # Compter le nombre de lignes ne ma
print("Nombre de lignes ne matchant pas suite à la jonction :", num_non_matching_rows)

↵ Nombre de lignes ne matchant pas suite à la jonction : 113

print("Les lignes sans correspondance lors de la jointure sont les suivantes :")
print(non_matching_rows)
```

```
↵ Les lignes sans correspondance lors de la jointure sont les suivantes :
   sku  total_sales post_date product_type post_title post_modified \
712  NaN          NaN      NaT          NaN          NaN          NaT
713  NaN          NaN      NaT          NaN          NaN          NaT
714  NaN          NaN      NaT          NaN          NaN          NaT
715  NaN          NaN      NaT          NaN          NaN          NaT
716  NaN          NaN      NaT          NaN          NaN          NaT
..   ...          ...      ...          ...          ...          ...
820  NaN          NaN      NaT          NaN          NaN          NaT
821  NaN          NaN      NaT          NaN          NaN          NaT
822  NaN          NaN      NaT          NaN          NaN          NaT
823  NaN          NaN      NaT          NaN          NaN          NaT
824  NaN          NaN      NaT          NaN          NaN          NaT

   post_type  product_id  onsale_web  price  stock_quantity  stock_status \
712      NaN          4055          0    86.1              0  outofstock
713      NaN          4090          0    73.0              0  outofstock
714      NaN          4092          0    47.0              0  outofstock
715      NaN          4195          0   14.1              0  outofstock
716      NaN          4209          0   73.5              0  outofstock
..   ...          ...      ...          ...          ...          ...
820      NaN          5955          0   27.3              0  outofstock
821      NaN          5957          0   39.0              0  outofstock
822      NaN          6100          0   12.9              0  outofstock
823      NaN          7247          1   54.8              6   instock
824      NaN          7329          0   26.5              14   instock

   purchase_price  id_web  _merge
712         37.88     NaN  right_only
713         33.79     NaN  right_only
714         25.25     NaN  right_only
```

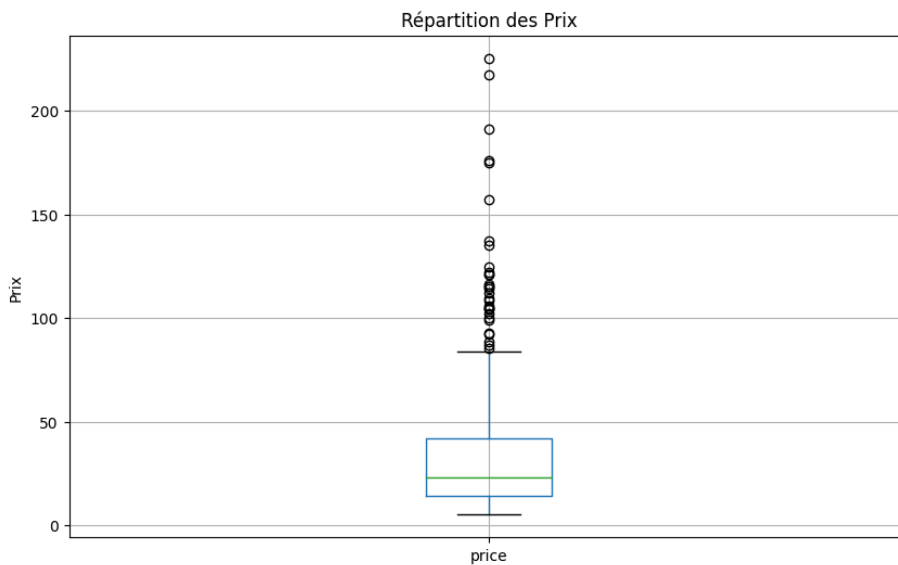
715	7.36	NaN	right_only
716	33.01	NaN	right_only
..	...	...	...
820	13.68	14377	right_only
821	20.75	13577	right_only
822	6.47	15529	right_only
823	27.18	13127-1	right_only
824	13.42	14680-1	right_only

[113 rows x 15 columns]

Etape 4 - Analyse univarié des prix

Etape 4.1 - Exploration par la visualisation de données

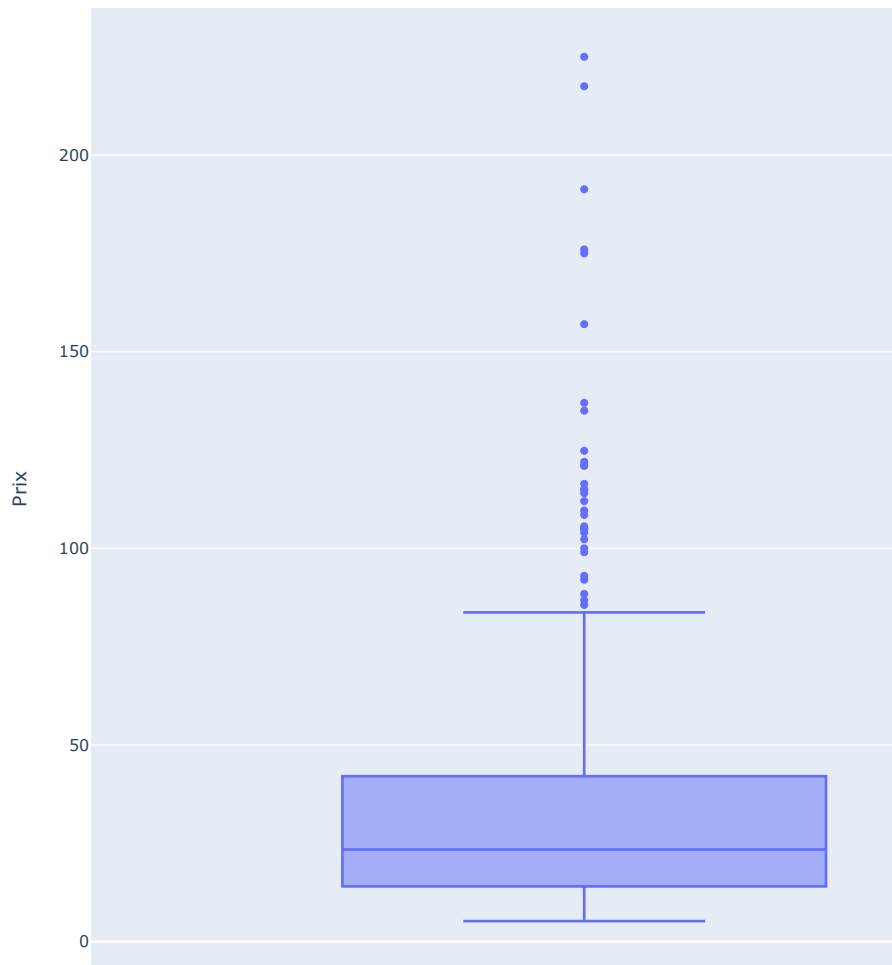
```
#Création d'une Boite à moustache de la répartition des prix grâce à Pandas
import matplotlib.pyplot as plt # importation de la bibliothèque pour la création de graphiques
plt.figure(figsize=(10, 6))
df_erp_liaison_web.boxplot(column='price')
plt.title('Répartition des Prix')
plt.ylabel('Prix')
plt.grid(True)
plt.show()
```



```
#Autre méthode avec plotly express
fig = px.box(df_erp_liaison_web, y='price', title='Répartition des Prix', labels={'price': 'Prix'}, width=900, height=900)
fig.show()
```



Répartition des Prix



Etape 4.2 - Exploration par l'utilisation de méthodes statistique

Etape 4.2.1 - Identification par le Z-index

```
#Calculer la moyenne du prix
average_price = df_erp_liaison_web['price'].mean()
print("Moyenne des prix :", round(average_price, 2))
```

➤ Moyenne des prix : 32.31

```
#Calculer l'écart-type du prix
std_price = df_erp_liaison_web['price'].std()
print("Écart type des prix :", round(std_price, 2))
```

➤ Écart type des prix : 27.62

```
#Calculer le Z-score
df_erp_liaison_web['z_score'] = (df_erp_liaison_web['price'] - average_price) / std_pr

print("DataFrame avec Z-score :", df_erp_liaison_web.head(5))
```

➤ DataFrame avec Z-score :

	sku	total_sales	post_date	product_type	\
0	14692	5.0	2019-03-19 10:06:47	Vin	
1	15328	2.0	2019-03-27 18:05:09	Vin	
2	16515	10.0	2018-06-02 09:31:31	Vin	
3	16585	15.0	2018-02-16 14:03:16	Vin	
4	12869	7.0	2019-03-28 14:29:35	Vin	

post_title post_modified \



Imnou Macodou Ba
5 juil. 2024



```
z_score_price = moyenne_prix /
ecart_type_prix
print("Le z-score est de: ", z_score_price)
```

```

0      Château Fonréaud Bordeaux Blanc Le Cygne 2016 2020-04-25 21:40:31
1      Agnès Levet Côte Rôtie Maestria 2017 2020-07-25 15:45:02
2      Château Turcaud Bordeaux Rouge Cuvée Majeure 2018 2020-08-27 10:11:12
3      Xavier Frissant Touraine Sauvignon 2019 2020-08-27 09:30:36
4      Stéphane Tissot Arbois D.D. 2016 2019-12-13 15:40:01

```

```

post_type  product_id  onsale_web  price  stock_quantity  stock_status  \
0  product      5794         1  21.70             15      instock
1  product      5827         1  55.00             4      instock
2  product      4964         1  12.10            23      instock
3  product      4223         1   9.70            42      instock
4  product      5900         1  18.25            21      instock

```

```

purchase_price  id_web  z_score
0      10.65  14692 -0.384217
1      29.55  15328  0.821392
2       6.50  16515 -0.731780
3       4.81  16585 -0.818671
4       8.96  12869 -0.509123

```

```

#Quel est le seuil prix dont z-score est supérieur à 3?
high_z_score_prices = df_erp_liaison_web[df_erp_liaison_web['z_score'] > 3]
print("Seuil de prix avec Z-score > 3 :", high_z_score_prices['price'].min())

```

```

↪ Seuil de prix avec Z-score > 3 : 116.4

```

Etape 4.2.2 - Identification par l'interval interquartile

```

#Utilisation de la fonction describe de Pandas pour l'etude des mesures de dispersions
description = df_erp_liaison_web['price'].describe()
print("Statistiques descriptives de la colonne 'price' :", description)

```

```

↪ Statistiques descriptives de la colonne 'price' : count      712.000000
mean         32.312430
std          27.620894
min           5.200000
25%          14.037500
50%          23.400000
75%          42.025000
max          225.000000
Name: price, dtype: float64

```

```

#Définissez un seuil pour les articles "outliers" en prix
Q1 = df_erp_liaison_web['price'].quantile(0.25) # 1er Quartile
Q3 = df_erp_liaison_web['price'].quantile(0.75) # 3e Quartile

```

```

IQR = Q3 - Q1 # IQR - Etendue interquartile

```

```

lower_bound = Q1 - 1.5 * IQR # Borne inférieure
upper_bound = Q3 + 1.5 * IQR # Borne supérieure
print("Le seuil bas est", lower_bound, "et le seuil haut est", upper_bound, ". Le prix

```

```

↪ Le seuil bas est -27.943749999999994 et le seuil haut est 84.00625 . Le prix de ve

```

```

#Définissez le nombre d'articles et la proportion de l'ensemble du catalogue "outliers"

```

```

outliers = df_erp_liaison_web[(df_erp_liaison_web['price'] > upper_bound)] # Filtre de
num_outliers = outliers.shape[0] # Nombre d'outliers
total_articles = df_erp_liaison_web.shape[0] # Nombre d'articles au total
proportion_outliers = num_outliers / total_articles # Proportion des outliers
print("Le catalogue comporte", num_outliers, "outliers correspondant à", round(proport

```

```

↪ Le catalogue comporte 31 outliers correspondant à 0.04 % de l'ensemble du catalogue

```

```

#Selon vous, ces outliers sont-ils justifiés ? Comment le démontrer si cela est possib

```

Dans le cas de la vente de vin il est tout à fait possible d'imaginer que certains grands crus puissent avoir des prix de vente qui se situent bien au dela de la moyenne. Cela peut se vérifier en affichant les 10 outliers les plus grands par ex. (cf ci-dessous)

```

# Trier le DataFrame par la colonne 'Valeurs' en ordre décroissant
outlier_sorted = outliers.sort_values(by='price', ascending=False)

```

```

# Afficher le DataFrame trié
outlier_sorted.head(10)

```



	sku	total_sales	post_date	product_type	post_title	post_modified	post_.
597	15940	11.0	2018-03-02 10:30:04	Champagne	Champagne Egly-Ouriat Grand Cru Millésimé 2008	2020-03-07 11:18:45	pr
687	14581	2.0	2018-07-17 09:45:39	Vin	David Duband Charmes- Chambertin Grand Cru 2014	2020-05-16 09:00:05	pr
318	14983	6.0	2019-03-28 10:21:36	Champagne	Coteaux Champenois Egly-Ouriat Ambonnay Rouge ...	2020-04-01 09:30:09	pr
447	3510	3.0	2018-03-22 11:21:05	Cognac	Cognac Frapin VIP XO	2020-08-22 11:35:03	pr
209	15185	4.0	2019-03-13 14:43:22	Vin	Camille Giroud Clos de Vougeot 2016	2020-06-11 15:25:04	pr
211	7819	4.0	2018-03-22 11:42:48	Cognac	Cognac Frapin Château de Fontpinot 1989 20 Ans...	2020-03-14 16:05:04	pr
225	14220	3.0	2018-05-15 10:23:41	Vin	Domaine Des Croix Corton Charlemagne Grand Cru...	2020-05-19 17:15:02	pr
391	14923	5.0	2019-06-28 17:22:27	Champagne	Champagne Gosset Célébris Vintage 2007	2020-08-27 11:45:02	pr
152	14915	1.0	2019-01-15 15:30:49	Vin	Domaine Weinbach Gewurztraminer Grand Cru Furs...	2019-01-23 09:33:57	pr
418	14775	3.0	2019-04-04 16:49:37	Whisky	Wemyss Malts Single Cask Scotch Whisky Choc 'n...	2020-03-11 09:30:09	pr

Etape 5 - Analyse univarié du CA, des quantités vendues, des stocks et de la marge ainsi qu'une analyse multivarié

Etape 5.1 - Analyse des ventes en CA

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

```
#Créez une colonne calculant le CA par article
df_erp_liaison_web['ca_par_article'] = df_erp_liaison_web['price'] * df_erp_liaison_we

df_erp_liaison_web.head()
```



	sku	total_sales	post_date	product_type	post_title	post_modified	post_type
0	14692	5.0	2019-03-19 10:06:47	Vin	Château Fonréaud Bordeaux Blanc Le Cygne 2016	2020-04-25 21:40:31	product
1	15328	2.0	2019-03-27 18:05:09	Vin	Agnès Levet Côte Rôtie Maestria 2017	2020-07-25 15:45:02	product
2	16515	10.0	2018-06-02 09:31:31	Vin	Château Turcaud Bordeaux Rouge Cuvée Majeure 2018	2020-08-27 10:11:12	product
3	16585	15.0	2018-02-16 14:03:16	Vin	Xavier Frissant Touraine Sauvignon 2019	2020-08-27 09:30:36	product
4	12869	7.0	2019-03-28 14:29:35	Vin	Stéphane Tissot Arbois D.D. 2016	2019-12-13 15:40:01	product

Remarque : A ce stade, je constate que les 'sku' et 'product_id' sont dupliqués avec des 'post_type' différents, 'attachment' et 'product'. Il faudrait ne conserver que les 'product' qui sont donc les produits. C'est une action qui aurait du être entreprise lors de l'exploration et du nettoyage du dataframe df_web. On pourrait la corriger à ce stade, mais les première analyses de prix ci-dessus pourraient être faussées. Je vais donc créer un nouveau dataframe filtré sur le post_type 'product' pour écarter les doublons.

```
#Calculez la somme de la colonne "ca_par_article"
ca_par_article_sum = df_erp_liaison_web['ca_par_article'].sum()
#Ce résultat correspond au chiffre d'affaire du site web
print("Le CA du site web est de :", round(ca_par_article_sum))
```



Le CA du site web est de : 143286

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

```
#Effectuer le tri dans l'ordre décroissant du CA du dataset df_merge
df_erp_liaison_web_sorted = df_erp_liaison_web.sort_values(by='ca_par_article', ascenc
```

```
#Réinitialiser l'index du dataset par un reset_index
df_erp_liaison_web_sorted_reset = df_erp_liaison_web_sorted.reset_index(drop=True)
```

```
#Afficher les 20 premier articles en CA
df_erp_liaison_web_sorted_reset.head(20)
```




	sku	total_sales	post_date	product_type	post_title	post_modified	post_typ
0	15940	11.0	2018-03-02 10:30:04	Champagne	Champagne Egly-Ouriet Grand Cru Millésimé 2008	2020-03-07 11:18:45	produ
1	14983	6.0	2019-03-28 10:21:36	Champagne	Coteaux Champenois Egly-Ouriet Ambonnay Rouge ...	2020-04-01 09:30:09	produ
2	12587	14.0	2018-03-02 10:37:26	Champagne	Champagne Egly-Ouriet Grand Cru Brut Rosé	2020-08-22 11:45:02	produ
3	15325	20.0	2019-03-27 17:59:49	Vin	Agnès Levat Côte Rôtie Améthyste 2017	2020-05-21 14:00:02	produ
4	13996	7.0	2019-07-25 09:09:17	Vin	Domaine des Comtes Lafon Volnay 1er Cru Santen...	2020-06-16 09:30:16	produ
5	13913	9.0	2018-07-18 10:46:30	Champagne	Champagne Agrapart & Fils Minéral Extra Br...	2020-05-11 14:35:02	produ
6	11602	7.0	2018-07-17 10:52:41	Vin	Domaine des Comtes Lafon Volnay 1er Cru Santen...	2020-06-23 15:35:02	produ
7	15185	4.0	2019-03-13 14:43:22	Vin	Camille Giroud Clos de Vougeot 2016	2020-06-11 15:25:04	produ
8	14923	5.0	2019-06-28 17:22:27	Champagne	Champagne Gosset Célébris Vintage 2007	2020-08-27 11:45:02	produ
9	13914	6.0	2018-07-18 10:39:43	Champagne	Champagne Agrapart & Fils L'Avizoise Extra...	2020-07-09 17:05:02	produ

David

```
#Graphique en barre des 20 premiers articles avec plotly express
df_top_20 = df_erp_liaison_web_sorted_reset.head(20)
```

```
# Créer le graphique en barres avec Plotly Express
fig = px.bar(df_top_20, x='post_title', y='ca_par_article', title='Top 20 Articles par
fig.update_xaxes(title='Produit')
fig.update_yaxes(title='CA')
```

```
# Afficher le graphique
fig.show()
```



Top 20 Articles par CA



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

```
#Créer une colonne calculant la part du CA de la ligne dans le dataset
total_ca = df_erp_liaison_web['ca_par_article'].sum() # Calculer le total du chiffre c
df_erp_liaison_web['part_du_ca'] = df_erp_liaison_web['ca_par_article'] / total_ca # C
```

```
#Créer une colonne réalisant la somme cumulative de la colonne précédemment créée
df_erp_liaison_web_sorted = df_erp_liaison_web.sort_values(by='part_du_ca', ascending=
df_erp_liaison_web_sorted['somme_part_du_ca'] = df_erp_liaison_web_sorted['part_du_ca'
# df_erp_liaison_web_sorted.head()
```

```
#Grâce au deux colonnes créées précédemment, calculer le nombre d'articles représentar
seuil = 0.80
num_articles_ca = (df_erp_liaison_web_sorted['somme_part_du_ca'] <= seuil).sum()
```

```
print("Nombre d'articles représentant 80% du CA :", num_articles_ca)
```

```
➡ Nombre d'articles représentant 80% du CA : 432
```

```
#Afficher la proportion que représente ce groupe d'articles dans le catalogue entier c
total_articles = len(df_erp_liaison_web)
proportion = num_articles_ca / total_articles
print(f"Ce groupe d'articles représente {proportion:.2%} de l'ensemble du catalogue.")
```

```
➡ Ce groupe d'articles représente 60.67% de l'ensemble du catalogue.
```

Etape 5.2 - Analyse des ventes en Quantités

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

```
#Effectuer le tri dans l'ordre décroissant de quantités vendues du dataset df_merge
df_erp_liaison_web_sorted = df_erp_liaison_web.sort_values(by='total_sales', ascending=False)
```

```
#Réinitialiser l'index du dataset par un reset_index
df_erp_liaison_web_sorted_reset = df_erp_liaison_web_sorted.reset_index(drop=True)
```

```
#Afficher les 20 premier articles en quantité
df_erp_liaison_web_sorted_reset.head(20)
```

	sku	total_sales	post_date	product_type	post_title	post_modified	post_typ
0	16148	36.0	2018-05-03 13:20:05	Vin	Château De La Selve IGP Coteaux de l'Ardèche M...	2020-08-27 09:30:15	produit
1	15415	27.0	2018-02-15 14:33:42	Vin	Mas Laval IGP Pays d'Hérault Les Pampres Blanc...	2020-07-11 16:45:03	produit
2	14864	24.0	2018-02-27 13:33:54	Vin	I Fabbri Chianti Classico Lamole 2017	2020-08-22 14:35:02	produit
3	16525	22.0	2018-04-17 09:28:58	Vin	Bernard Baudry Chinon Rouge La Croix Boissée 2017	2020-07-31 09:31:39	produit
4	14950	22.0	2018-04-18 11:53:51	Vin	François Baur Pinot Noir Schlittweg 2017	2020-05-06 11:35:01	produit
5	15325	20.0	2019-03-27 17:59:49	Vin	Agnès Levet Côte Rôtie Améthyste 2017	2020-05-21 14:00:02	produit
6	14570	20.0	2019-06-28 18:01:06	Vin	Moulin de Gassac IGP Pays d'Hérault Guilhem Bl...	2020-08-26 15:55:02	produit
7	15758	18.0	2018-02-16 10:54:27	Vin	Xavier Frissant Touraine Amboise Chenin Les Pi...	2020-08-27 11:45:02	produit
8	15561	17.0	2019-03-15 10:20:59	Vin	Maurel Pays d'Oc Merlot 2018	2020-08-14 10:55:02	produit
9	13572	17.0	2019-03-19 11:33:39	Vin	Château Tour Haut- Caussan Médoc 2015	2020-08-26 16:55:02	produit

```
# Ajout Alex pour compléter présentation
ca_par_article_sum = df_erp_liaison_web['total_sales'].sum()
#Nombre total d'articles vendus
print("Le nombre total d'articles vendus sur le site web est de :", round(ca_par_article_sum, 2))
```

```
Le nombre total d'articles vendus sur le site web est de : 5740
nimes
```

```
#Graphique en barre des 20 premiers articles avec plotly express
df_top_20 = df_erp_liaison_web_sorted_reset.head(20)
```

```
# Créer le graphique en barres avec Plotly Express
fig = px.bar(df_top_20, x='post_title', y='total_sales', title='Top 20 Articles par Qu
fig.update_xaxes(title='Produit')
fig.update_yaxes(title='Quantités vendues')

# Afficher le graphique
fig.show()
```



Top 20 Articles par Quantités vendue



```
#####
```

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

```
#####
```

```
#Créer une colonne calculant la part en quantité de la ligne dans le dataset
total_sales_quant = df_erp_liaison_web['total_sales'].sum() # Total des qté vendues
df_erp_liaison_web['part_en_qte'] = df_erp_liaison_web['total_sales'] / total_sales_q
#df_erp_liaison_web.head(5)
```

```
#Créer une colonne réalisant la somme cumulative de la colonne précédemment créée
df_erp_liaison_web_sorted = df_erp_liaison_web.sort_values(by='part_en_qte', ascending
df_erp_liaison_web_sorted['somme_part_en_qte'] = df_erp_liaison_web_sorted['part_en_qt
#df_erp_liaison_web_sorted.head()
```

```
#Grâce au deux colonnes créées précédemment, calculer le nombre d'articles représentar
seuil = 0.8
num_articles_qte = (df_erp_liaison_web_sorted['somme_part_en_qte'] <= seuil).sum()
```

```
print("Nombre d'articles représentant 80% des ventes en quantité :", num_articles_qte)
```



Nombre d'articles représentant 80% des ventes en quantité : 431

```
#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier
total_articles = len(df_erp_liaison_web)
proportion = num_articles_qte / total_articles
print(f"Ce groupe d'articles représente {proportion:.2%} de l'ensemble du catalogue.")
```

🔄 Ce groupe d'articles représente 60.53% de l'ensemble du catalogue.

Etape 5.3 - Analyse des stocks

```
#####
# Calcule le nombre de mois de stock #
#####
```

```
#Import de numpy
import numpy as np
```

```
#Création de la colonne Rotation de stock
```

En général, la rotation de stock est calculée comme le rapport entre le coût des marchandises vendues (CMV) et le stock moyen. Cependant, dans notre cas, avec les données disponibles, nous pouvons calculer une approximation de la rotation de stock en utilisant le nombre total de ventes et les niveaux de stock.

```
df_erp_liaison_web['rotation_de_stock'] = df_erp_liaison_web['total_sales'] / df_erp_l
df_erp_liaison_web.head(5)
```

🔄

	sku	total_sales	post_date	product_type	post_title	post_modified	post_type
0	14692	5.0	2019-03-19 10:06:47	Vin	Château Fonréaud Bordeaux Blanc Le Cygne 2016	2020-04-25 21:40:31	product
1	15328	2.0	2019-03-27 18:05:09	Vin	Agnès Levet Côte Rôtie Maestria 2017	2020-07-25 15:45:02	product
2	16515	10.0	2018-06-02 09:31:31	Vin	Château Turcaud Bordeaux Rouge Cuvée Majeure 2018	2020-08-27 10:11:12	product
3	16585	15.0	2018-02-16 14:03:16	Vin	Xavier Frissant Touraine Sauvignon 2019	2020-08-27 09:30:36	product
4	12869	7.0	2019-03-28 14:29:35	Vin	Stéphane Tissot Arbois D.D. 2016	2019-12-13 15:40:01	product

```
#Remplacement des "inf" par 0
df_erp_liaison_web.replace([np.inf, -np.inf], 0, inplace=True)
```

```
#Effectuer le tri dans l'ordre décroissant du nombre de mois de stock dans le dataset
df_erp_liaison_web_sorted = df_erp_liaison_web.sort_values(by='stock_quantity', ascenc
#df_erp_liaison_web_sorted.head(5)
```

```
#Graphique en barre du flop 20 des produits qui ont le plus de mois de stock
```

```
df_erp_liaison_web['months_of_stock'] = df_erp_liaison_web['stock_quantity'] / (df_erp
df_erp_liaison_web.replace([np.inf, -np.inf], 0, inplace=True) #Remplacement des "inf"
df_erp_liaison_web_sorted= df_erp_liaison_web.sort_values(by='months_of_stock', ascenc
df_erp_liaison_web_sorted_reset = df_erp_liaison_web_sorted.reset_index(drop=True) #Ré
```

```
#df_erp_liaison_web_sorted_reset.head(20) #Afficher les 20 premier articles en mois de
```

```
#Graphique en barre du flop 20 avec plotly express
df_flop_20 = df_erp_liaison_web_sorted_reset.head(20)
```

```
# Créer le graphique en barre
fig = px.bar(df_flop_20, x='post_title', y='months_of_stock', title='Flop 20 des produ
```

```
# Afficher le graphique
fig.show()
```



Flop 20 des produits avec le plus de



```
#####
# Valorisation des stocks en euros #
#####
```

```
#Création de la colonne Valorisation des stocks en euros
df_erp_liaison_web['valorisation_des_stocks'] = df_erp_liaison_web['stock_quantity'] *
df_erp_liaison_web.head(5)
```

	sku	total_sales	post_date	product_type	post_title	post_modified	post_type
0	14692	5.0	2019-03-19 10:06:47	Vin	Château Fonréaud Bordeaux Blanc Le Cygne 2016	2020-04-25 21:40:31	product
1	15328	2.0	2019-03-27 18:05:09	Vin	Agnès Levet Côte Rôtie Maestria 2017	2020-07-25 15:45:02	product
2	16515	10.0	2018-06-02 09:31:31	Vin	Château Turcaud Bordeaux Rouge Cuvée Majeure 2018	2020-08-27 10:11:12	product
3	16585	15.0	2018-02-16 14:03:16	Vin	Xavier Frissant Touraine Sauvignon 2019	2020-08-27 09:30:36	product
4	12869	7.0	2019-03-28 14:29:35	Vin	Stéphane Tissot Arbois D.D. 2016	2019-12-13 15:40:01	product

5 rows × 21 columns

```
#Calculer la somme de la colonne "Valorisation_stock_euros"
somme_valorisation_stocks = df_erp_liaison_web['valorisation_des_stocks'].sum() # Calc
```

```
# Afficher la somme
print(f"La somme de la valorisation des stocks en euros est de {somme_valorisation_stc
```

```
La somme de la valorisation des stocks en euros est de 493689.60€
```

```
#####
# Valorisation du nombre de produit en stock #
#####
```

```
#Calculer la somme de la colonne stock quantity
somme_stock_quantity = df_erp_liaison_web['stock_quantity'].sum()

print("Le niveau de stock est de", somme_stock_quantity, "produits.")
```

```
Le niveau de stock est de 16710 produits.
```

Etape 5.4 - Analyse du taux de marge

```
#####
# Analyse du taux de marge #
#####
```

```
#Création de la colonne prix HT
```

```
taux_tva = 0.20 # Définir le taux de TVA
df_erp_liaison_web['prix_HT'] = df_erp_liaison_web['price'] / (1 + taux_tva) # Calcule
#print(df_erp_liaison_web.head()) # Afficher les premières lignes du DataFrame pour vé
```

```
#Création de la colonne Taux de marge
# Calculer le taux de marge et ajouter une colonne 'taux_de_marge' au DataFrame
df_erp_liaison_web['taux_de_marge_nette'] = (df_erp_liaison_web['prix_HT'] - df_erp_li

# Afficher les premières lignes du DataFrame pour vérifier
# print(df_erp_liaison_web[['purchase_price', 'prix_HT', 'taux_de_marge_nette']].head(5))
```

La marge brute est calculée en utilisant les prix de vente TTC et la marge nette en utilisant le prix de vente HT. Dans la mesure où nous avons précédemment calculé le prix HT, nous partons du postulat qu'il est attendu la marge nette.

```
#Afficher le prix minimum de la colonne "taux_marge"
taux_de_marge_minimum = df_erp_liaison_web['taux_de_marge_nette'].min()
print(f"Le taux de marge minimum est de : {taux_de_marge_minimum:.2f}")
```

```
↗ Le taux de marge minimum est de : -6.35
```

```
#Afficher le prix maximum de la colonne "taux_marge"
taux_de_marge_maximum = df_erp_liaison_web['taux_de_marge_nette'].max()
print(f"Le taux de marge maximum est de : {taux_de_marge_maximum:.2f}")
```

```
↗ Le taux de marge maximum est de : 0.48
```

```
#affichage de la ligne avec un taux de marge inférieur à 0
# Filtrer les lignes avec un taux de marge inférieur à 0
lignes_taux_de_marge_negatif = df_erp_liaison_web[df_erp_liaison_web['taux_de_marge_ne
```

```
# Afficher les lignes correspondantes
print(lignes_taux_de_marge_negatif)
```

```
↗
```

	sku	total_sales	post_date	product_type	\
343	12589	0.0	2018-03-02 10:46:10	Champagne	

	post_title	post_modified	\
343	Champagne Egly-Ouriet Grand Cru Blanc de Noirs	2020-08-13 10:15:02	

	post_type	product_id	onsale_web	price	...	id_web	z_score	\
343	product	4355	1	12.65	...	12589	-0.711868	

	ca_par_article	part_du_ca	part_en_qte	rotation_de_stock	\
343	0.0	0.0	0.0	0.0	

	months_of_stock	valorisation_des_stocks	prix_HT	taux_de_marge_nette
343	0.0	1227.05	10.541667	-6.349881

[1 rows x 23 columns]

```
#création d'un dataframe avec les taux positifs
df_taux_de_marge_positif = df_erp_liaison_web[df_erp_liaison_web['taux_de_marge_nette']
```

```
#Afficher le prix minimum de la colonne "taux_marge"
taux_de_marge_minimum = df_taux_de_marge_positif['taux_de_marge_nette'].min()
print(f"Le taux de marge minimum est de : {taux_de_marge_minimum:.2f}")
```

```
↗ Le taux de marge minimum est de : 0.23
```

```
#Afficher le prix maximum de la colonne "taux_marge"
taux_de_marge_maximum = df_taux_de_marge_positif['taux_de_marge_nette'].max()
print(f"Le taux de marge maximum est de : {taux_de_marge_maximum:.2f}")
```

```
↗ Le taux de marge maximum est de : 0.48
```

```
#création d'un dataframe avec le taux de marge moyen par type de produit (remarque Al
# Calculer le taux de marge moyen par type de produit
df_taux_de_marge_moyen_par_type = df_erp_liaison_web.groupby('product_type')['taux_de_
```

```
# Renommer la colonne pour plus de clarté
df_taux_de_marge_moyen_par_type.rename(columns={'taux_de_marge_nette': 'taux_de_marge_

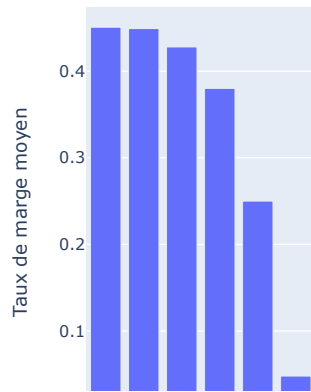
# Afficher le nouveau DataFrame
# print(df_taux_de_marge_moyen_par_type.head(5))
```

```
#Affichage dans un graphique du taux de marge par type de produit
df_taux_de_marge_moyen_par_type_sorted= df_taux_de_marge_moyen_par_type.sort_values(by
# Créer le graphique avec Plotly Express
fig = px.bar(df_taux_de_marge_moyen_par_type_sorted, x='product_type', y='taux_de_marg
title='Taux de marge moyen par type de produit',
labels={'taux_de_marge_moyen': 'Taux de marge moyen', 'product_type': 'Ty
```

```
# Afficher le graphique
fig.show()
```




Taux de marge moyen par type de pr



Etape 5.5 - Analyse des correlations entre les variables stock, sales et price

```
#####
```

```
# Analyse des correlations #
```

```
#####
```

```
#Importation de Seaborn
```

```
import seaborn as sns
```

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

```
#Création d'un heatmap de corrélation avec les variables stock, sales et price
```

```
# Calculer la corrélation entre les variables
```

```
correlation_matrix = df_erp_liaison_web[['stock_quantity', 'total_sales', 'price']].corr()
```

```
# Afficher la heatmap de corrélation avec Seaborn
```

```
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
plt.title('Heatmap de corrélation entre stock, sales et price')
```

```
plt.show()
```

