ANALYSE DU STOCK ET DES VENTES DU SITE BOTTLENECK

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

In [4]:
```
#Importation des librairies
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
#Importation de la librairie plotly express
!pip install plotly
import plotly.express as px
```

```
Requirement already satisfied: plotly in c:\users\pc\anaconda3\lib\site-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\pc\anaconda3\lib\site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\pc\anaconda3\lib\site-packages (from plotly) (24.1)
```

1.2 - Chargements des fichiers

In [8]:
```
#Importation du fichier web.xlsx
df_web = pd.read_excel("web.xlsx")
#Importation du fichier erp.xlsx
df_erp = pd.read_excel("erp.xlsx")
#importation du fichier liaison.xlsx
df_liaison = pd.read_excel("liaison.xlsx")
```

```
C:\Users\PC\anaconda3\Lib\site-packages\openpyxl\worksheet\_read_only.py:85: UserWarning: Unknown extension is not supported and will be removed
  for idx, row in parser.parse():
C:\Users\PC\anaconda3\Lib\site-packages\openpyxl\worksheet\_read_only.py:85: UserWarning: Unknown extension is not supported and will be removed
  for idx, row in parser.parse():
C:\Users\PC\anaconda3\Lib\site-packages\openpyxl\worksheet\_read_only.py:85: UserWarning: Unknown extension is not supported and will be removed
  for idx, row in parser.parse():
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier erp.xlsx

In [11]:
```
#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)
```

In [12]:
```
#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
```

In [13]:
```
# Conversion de la colonne stock status en type 'category'
df_erp['stock_status'] = df_erp['stock_status'].astype('category')
```

In [14]:
```
#Afficher les 5 premières lignes de la table
df_erp.head()
```

Out[14]:

	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

In [15]:
```
#Vérifier si il y a les lignes en doublons dans la colonne product_id
doublons = df_erp[df_erp['product_id'].duplicated()]

# Afficher les doublons (si existants)
print("Nombre de lignes dupliquées dans product_id:", len(doublons))
```

```
Nombre de lignes dupliquées dans product_id: 0
```

In [16]:
```
#Afficher les valeurs distinctes de la colonne stock_status
print(df_erp["stock_status"].unique())
#À quelle(s) autre(s) colonne(s) sont-elles liées ?
df_erp[["stock_quantity", "stock_status"]].drop_duplicates()
```

```
['instock', 'outofstock']
Categories (2, object): ['instock', 'outofstock']
```

Out[16]:

	stock_quantity	stock_status
0	16	instock
1	10	instock
2	0	outofstock
3	26	instock
4	3	outofstock
...	...	...
697	138	instock
700	68	instock
730	58	instock
799	53	instock
817	55	instock

86 rows × 2 columns

Remarque

Les colonnes stock_quantity et stock_status sont liées, lorsque la quantité de stock est à 0, "outofstock" sera affiché.

In [18]:
```
# Vérification des valeurs NAN
print(df_erp[["stock_quantity", "stock_status"]].isna().sum())
```

```
stock_quantity    0
stock_status      0
dtype: int64
```

In [19]:
```
#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 mettre "instock"
df_erp["stock_status_2"] = df_erp["stock_quantity"].apply(lambda x: "outofstock" if x == 0 else "instock")
```

In [20]:
```
#Vérifions que les 2 colonnes sont identiques:
#Les 2 colonnes sont strictement identiques si les valeurs de chaque ligne sont strictement identiques 2 à 2
#La comparaison de 2 colonnes peut se réaliser simplement avec l'instruction ci-dessous:
Comparaison = df_erp["stock_status"] == df_erp["stock_status_2"]
print(Comparaison)
#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
```

In [21]:
```
#Mais il est possible de synthétiser ce résultat en effectuant la somme de cette colonne:
#True vaut 1 et False 0
nb_identiques = Comparaison.sum()
# Affichage du nombre de lignes identiques
print(f"Nombre de lignes identiques : {nb_identiques}")
#Nous devrions obtenir la somme de 825 qui correspond au nombre de lignes dans ce dataset
```

```
Nombre de lignes identiques : 821
```

Ici il y a un problème car le nombre de lignes qui s'affiche est de 821 alors que l'on devrait avoir 825 lignes identitiques, soit, le nombre de lignes dans ce dataset. Nous allons donc afficher les lignes qui présente des différences.

In [23]:
```
#Si les colonnes ne sont absolument pas identiques ligne à ligne alors identifier la ligne en écart
##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
# Filtre des lignes différentes entre les deux colonnes
df_ecarts = df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
# Affichage du nombre de lignes différentes
print(f"Nombre de lignes différentes : {len(df_ecarts)}")
```

```
Nombre de lignes différentes : 4
```

In [24]:
```
# Affichage des lignes avec des incohérences
df_ecarts
```

Out[24]:

	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	instock
573	5700	1	44.5	-1	outofstock	22.30	instock

4 lignes incohérentes

Il y 4 lignes incohérentes, ce qui explique que nous trouvons que 821 lignes au lieu de 825. Pour remédier à cela nous allons corriger les valeurs incohérentes afin qu'elles correspondent à la bonne quantité de stock. Pour cela nous allons remplacer en premier les valeurs négatives de la colonne stock_quantity en 0 puis mettre à jour les colonnes stock_status et stock_status_2 pour qu'elles correspondent à stock_quantity.

In [26]:
```
#Remplacer les valeurs négatives de stock_quantity par 0
df_erp["stock_quantity"] = df_erp["stock_quantity"].apply(lambda x: x if x >= 0 else 0)

#Corriger la ou les données incohérentes
# Correction de la colonne stock_status pour qu'elle corresponde à stock_quantity
df_erp["stock_status"] = df_erp["stock_quantity"].apply(lambda x: "outofstock" if x == 0 else "instock")
df_erp["stock_status_2"] = df_erp["stock_quantity"].apply(lambda x: "outofstock" if x == 0 else "instock")
```

In [27]:
```
#Verification en utilisant le même code que plus haut pour afficher les problèmes
# Vérifier que les deux colonnes sont maintenant identiques
df_ecarts_final = df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
print(f"Nombre de lignes incohérentes restantes : {len(df_ecarts_final)}")
```

```
Nombre de lignes incohérentes restantes : 0
```

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

2.1.1.1 - Analyse de la variable PRIX

In [30]:
```
#Vérification des prix: Y a t-il des prix non renseignés, négatif ou nul?
nb_prix_non_renseignes = df_erp["price"].isna().sum()
#Afficher le ou les prix non renseignés dans la colonne "price"
print("Nombre d'articles avec un prix non renseignés: {}".format(nb_prix_non_renseignes))
```

```
Nombre d'articles avec un prix non renseignés: 0
```

In [31]:
```
#Afficher le prix minimum de la colonne "price"
prix_min = df_erp["price"].min()
print(f"Prix minimum : {prix_min}")
#Afficher le prix maximum de la colonne "price"
prix_max = df_erp["price"].max()
print(f"Prix maximum : {prix_max}")
```

```
Prix minimum : -20.0
Prix maximum : 225.0
```

In [32]:
```
###############
## LES PRIX  ##
###############

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

Out[32]:

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	stock_status_2
151	4233	0	-20.0	0	outofstock	10.33	outofstock
469	5017	0	-8.0	0	outofstock	4.34	outofstock
739	6594	0	-9.1	19	instock	4.61	instock

In [33]:
```
#Remplacement des valeurs négatives par des NaN
df_erp.loc[df_erp['price'] < 0, 'price'] = None
```

In [34]:
```
#Vérification du résultat
df_erp_min = df_erp['price'].min()
print("Prix minimum dans la colonne 'price' : {}".format(df_erp_min))
```

```
Prix minimum dans la colonne 'price' : 5.2
```

2.1.1.2 - Analyse de la variable STOCK

In [36]:
```
########################
### stock_quantity  ###
########################

#Vérification de la colonne stock quantity
#Afficher la quantité minimum de la colonne "stock_quantity"
qte_min_stock = df_erp['stock_quantity'].min()
print(f"La quantité minimum de stock est de :", qte_min_stock)
```

```
La quantité minimum de stock est de : 0
```

In [37]:
```
#Afficher la quantité maximum de la colonne "stock_quantity"
qte_max_stock = df_erp['stock_quantity'].max()
print(f"La quantité minimum de stock est de :", qte_max_stock)
```

```
La quantité minimum de stock est de : 145
```

In [38]:
```
#Affichier les stocks inférieurs à 0
stocks_negatifs = df_erp[df_erp['stock_quantity'] < 0].shape[0]
print(f"La quantitée de stock inférieur à 0 est de :", stocks_negatifs)
```

```
La quantitée de stock inférieur à 0 est de : 0
```

2.1.1.3 - Analyse de la variable ONSALE_WEB

Vérification de la colonne onsale_web et des valeurs qu'elle contient. Que signifient-elles?

=> La colonne onsale_web représente les produits vente sur le site web. Si c'est 1 le produit est en vente, si c'est 0 il n'est pas en vente.

Quelles sont les colonnes à conserver?

=> Les colonnes à conserver sont product_id, onsale_web, price, stock_quantity, stock_status et purchase_price

In [42]:
```
# Supprimer la colonne "stock_status_2" si elle existe, car elle est redondante avec "stock_status"
df_erp = df_erp.drop(columns=['stock_status_2'])
# Afficher le DataFrame mis à jour
df_erp
```

Out[42]:

	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	instock	23.77
...	...	...	...	...	...	...
820	7203	0	45.0	30	instock	23.48
821	7204	0	45.0	9	instock	24.18
822	7247	1	54.8	6	instock	27.18
823	7329	0	26.5	14	instock	13.42
824	7338	1	16.3	40	instock	8.00

825 rows × 6 columns

Produits pas en vente sur le site web mais en stock

Dans ce dataframe on remarque que certains produits ne sont pas en vente sur le site web mais ont un stock positif. Par exemple avec les product_id 7203 et le 7204 avec respectivement 30 et 9 de stock quantity ne sont pas disponible à la vente sur le site web. Comment expliquer cela ?

2.1.1.4 - Analyse de la variable prix d'achat

In [45]:
```
######################
##   prix d'achat   ##
######################

# Afficher les lignes où le prix d'achat n'est pas renseigné
print("Nombre de valeurs manquantes :", df_erp['purchase_price'].isna().sum())

#Afficher le prix minimum de la colonne "purchase_price"
prix_min = df_erp['purchase_price'].min()
print(f"\nPrix minimum dans 'purchase_price' : {prix_min}")

#Afficher le prix maximum de la colonne "purchase_price"
prix_max = df_erp['purchase_price'].max()
print(f"Prix maximum dans 'purchase_price' : {prix_max}")
```

```
Nombre de valeurs manquantes : 0

Prix minimum dans 'purchase_price' : 2.74
Prix maximum dans 'purchase_price' : 137.81
```

2.2 - Analyse exploratoire du fichier web.xlsx

In [47]:
```
#Dimension du dataset
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
```

In [48]:
```python
#Consulter le nombre de colonnes
nb_colonnes = df_web.shape[1]
print(f"Nombre de colonnes dans la table 'web' : {nb_colonnes}")
#La nature des données dans chacune des colonnes
print("\nTypes de données dans chaque colonne :")
print(df_web.dtypes)
```

```
Nombre de colonnes dans la table 'web' : 29

Types de données dans chaque colonne :
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
product_type                   object
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
```

In [49]:
```python
#Le nombre de valeurs présentes dans chacune des colonnes
print("\nNombre de valeurs présentes (non nulles) par colonne :")
print(df_web.count())
```

```
Nombre de valeurs présentes (non nulles) par colonne :
sku                       1428
virtual                   1513
downloadable              1513
rating_count              1513
average_rating            1430
total_sales               1430
tax_status                 716
tax_class                    0
post_author               1430
post_date                 1430
post_date_gmt             1430
post_content                 0
product_type              1429
post_title                1430
post_excerpt               716
post_status               1430
comment_status            1430
ping_status               1430
post_password                0
post_name                 1430
post_modified             1430
post_modified_gmt         1430
post_content_filtered        0
post_parent               1430
guid                      1430
menu_order                1430
post_type                 1430
post_mime_type             714
comment_count             1430
dtype: int64
```

In [50]:
```python
#Afficher le nombre de valeurs uniques pour chaque colonne
df_web_valeurs_uniques = df_web.nunique()
print("Nombre de valeurs uniques pour chaque colonne : {}", df_web_valeurs_uniques)
```

```
Nombre de valeurs uniques pour chaque colonne : {} sku                       714
virtual                     1
downloadable                1
rating_count                1
average_rating              1
total_sales                29
tax_status                  1
tax_class                   0
post_author                 2
post_date                 716
post_date_gmt             716
post_content                0
product_type                7
post_title                713
post_excerpt              679
post_status                 1
comment_status              1
ping_status                 1
post_password               0
post_name                 716
post_modified             589
post_modified_gmt         589
post_content_filtered       0
post_parent                 1
guid                     1430
menu_order                  1
post_type                   2
post_mime_type              1
comment_count               1
dtype: int64
```

In [51]:
```python
#Selon vous, quelles sont les colonnes à conserver ?
# Identifier les colonnes vides (toutes valeurs NaN)
colonnes_vides = df_web.columns[df_web.isna().all()].tolist()

# Identifier les colonnes avec une seule valeur unique
colonnes_valeur_unique = df_web.columns[df_web.nunique() == 1].tolist()

# Combiner les deux listes
colonnes_inutiles = colonnes_vides + colonnes_valeur_unique

# Afficher le résultat
print("Colonnes entièrement vides :", colonnes_vides)
print("Colonnes avec une seule valeur unique :", colonnes_valeur_unique)
print("Liste complète des colonnes inutiles :", colonnes_inutiles)
```

```
Colonnes entièrement vides : ['tax_class', 'post_content', 'post_password', 'post_content_filtered']
Colonnes avec une seule valeur unique : ['virtual', 'downloadable', 'rating_count', 'average_rating', 'tax_status', 'post_status', 'comme
nt_status', 'ping_status', 'post_parent', 'menu_order', 'post_mime_type', 'comment_count']
Liste complète des colonnes inutiles : ['tax_class', 'post_content', 'post_password', 'post_content_filtered', 'virtual', 'downloadable',
'rating_count', 'average_rating', 'tax_status', 'post_status', 'comment_status', 'ping_status', 'post_parent', 'menu_order', 'post_mime_t
ype', 'comment_count']
```

In [52]:
```
#Supprimer les colonnes inutiles
df_web = df_web.drop(columns=colonnes_inutiles)
```

In [53]:
```
#Afficher le dataset
df_web.head()
```

Out[53]:

	sku	total_sales	post_author	post_date	post_date_gmt	product_type	post_title	post_excerpt	post_name	post_modified	post_modified_gm
0	11862	3.0	2.0	2018-02-12 13:46:23	2018-02-12 12:46:23	Vin	Gilles Robin Hermitage Rouge 2012	NaN	gilles-robin-hermitage-2012	2019-01-31 12:12:56	2019-01-31 11:12:5
1	16057	5.0	2.0	2018-04-17 15:29:17	2018-04-17 13:29:17	Vin	Domaine Pellé Sancerre Rouge La Croix Au Garde...	NaN	pelle-sancerre-rouge-la-croix-au-garde-2017	2020-07-07 10:05:02	2020-07-07 08:05:0
2	14692	5.0	2.0	2019-03-19 10:06:47	2019-03-19 09:06:47	Vin	Château Fonréaud Bordeaux Blanc Le Cygne 2016	Grâce à la complémentarité des 3 cépages ...	fonreaud-bordeaux-blanc-le-cygne-2016	2020-04-25 21:40:31	2020-04-25 19:40:3
3	16295	14.0	2.0	2018-02-15 14:05:06	2018-02-15 13:05:06	Vin	Moulin de Gassac IGP Pays d'Hérault Guilhem Ro...	NaN	moulin-de-gassac-igp-pays-dherault-guilhem-ros...	2020-08-27 18:55:03	2020-08-27 16:55:0
4	15328	2.0	2.0	2019-03-27 18:05:09	2019-03-27 17:05:09	Vin	Agnès Levet Côte Rôtie Maestria 2017	<span style="float: none; background-color: tr...	agnes-levet-cote-rotie-maestria-2017	2020-07-25 15:45:02	2020-07-25 13:45:0

In [54]:
```
#Afficher un aperçu des données
df_web.describe()
```

Out[54]:

	total_sales	post_author	post_date	post_date_gmt	post_modified	post_modified_gmt
count	1430.000000	1430.000000	1430	1430	1430	1430
mean	8.223077	1.998601	2018-08-22 03:22:17.090909184	2018-08-22 01:53:30.097902080	2020-06-20 13:59:29.781818112	2020-06-20 12:06:02.509090816
min	-56.000000	1.000000	2018-02-08 12:58:52	2018-02-08 11:58:52	2018-02-20 15:19:23	2018-02-20 14:19:23
25%	5.000000	2.000000	2018-02-27 20:01:12.500000	2018-02-27 19:01:12.500000	2020-06-18 10:45:05.249999872	2020-06-18 08:45:05.249999872
50%	8.000000	2.000000	2018-04-19 14:56:05	2018-04-19 12:56:05	2020-08-04 09:30:06	2020-08-04 07:30:06
75%	11.000000	2.000000	2019-01-31 14:35:47	2019-01-31 13:35:47	2020-08-25 10:32:32	2020-08-25 08:32:32
max	122.000000	2.000000	2020-07-20 11:00:00	2020-07-20 09:00:00	2020-08-27 18:55:03	2020-08-27 16:55:03
std	6.721899	0.037385	NaN	NaN	NaN	NaN

In [55]:
```
#Visualisation des valeurs de la colonne sku
df_web['sku']
```

Out[55]:
```
0       11862
1       16057
2       14692
3       16295
4       15328
        ...
1508    16326
1509    15662
1510    15329
1511    14827
1512    16004
Name: sku, Length: 1513, dtype: object
```

In [56]:
```
#Quelles sont les valeurs qui ne semblent pas respecter la règle de codification?
df_web['sku'].unique()
```

Out[56]:
```
array([11862, 16057, 14692, 16295, 15328, 15471, 16515, 16246, nan, 13572,
       16513, 16585, 16269, 15526, 12869, 15575, 11586, 14338, 15425,
       16560, 15361, 13809, 11587, 15022, 14323, 16342, 16029, 15475,
       13754, 14680, 15875, 9636, 13849, 13662, 16564, 13557, 15429,
       14712, 15032, 15481, 15448, 16580, 15441, 804, 15300, 13958, 16071,
       15678, 13895, 15711, 12882, 16053, 13766, 16247, 12640, 15292,
       15476, 15670, 16189, 16038, 14864, 16044, 15324, 15531, 15953,
       15413, 15733, 14366, 15895, 15892, 16472, 15185, 16010, 15793,
       15849, 12315, 15741, 15934, 15148, 15781, 15659, 15106, 15490,
       14507, 14149, 16307, 13736, 14090, 16037, 15758, 14661, 12587,
       15337, 15489, 15201, 16305, 16131, 13435, 15747, 12203, 14509,
       14768, 16262, 14561, 16505, 15717, 16129, 13460, 15871, 15940,
       11602, 13127, 13520, 15480, 13032, 15436, 15269, 15910, 19821,
       16263, 15138, 15146, 15126, 15482, 16186, 13905, 16540, 15856,
       15677, 14700, 15325, 19815, 3506, 16056, 14975, 15341, 15204,
       15415, 16065, 15479, 16151, 15127, 15140, 15779, 15473, 15530,
       14805, 14106, 9937, 15281, 16553, 15315, 15668, 13211, 15161,
       11258, 16296, 12588, 15792, 15921, 15690, 15775, 15577, 15870, 802,
       15163, 15707, 15561, 15539, 4679, 16320, 15923, 3509, 2534, 16238,
       14819, 15705, 15254, 15554, 14451, 13215, 14302, 12476, 16274,
       14696, 14573, 15343, 791, 14474, 12586, 15346, 15834, 15342, 15141,
       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,
       13096, 15857, 14632, 11641, 15736, 15615, 15574, 16096, 15564,
       15774, 14930, 15941, 16045, 16230, 14429, 15769, 16265, 14657,
       15261, 16239, 15715, 13453, 16244, 15147, 16280, 14508, 15465,
       3510, 14374, 15880, 13416, 15183, 15860, 11467, 11585, 14485,
       16138, 19822, 15155, 11996, 14626, 16330, 15035, 15404, 8365, 3507,
       14773, 15967, 15812, 16319, 16004, 15310, 16097, 14856, 15629,
       1364, 15116, 14101, 14912, 15036, 16047, 14184, 14983, 16030,
       16283, 16324, 15801, 10014, 12881, 15660, 16005, 15306, 16277,
       15527, 15428, 13957, 15647, 12496, 15399, 13969, 15525, 13914,
       12790, 15720, 13313, 15928, 15432, 15655, 16539, 12589, 15648,
       16011, 16146, 15672, 15030, 15735, 14800, 16190, 13217, 13854,
       16317, 16565, 15891, 15730, 16497, 16124, 10459, 19814, 15863,
       15786, 15791, 16135, 16147, 16066, 15818, 15790, 8193, 16255,
       15047, 15966, 14736, 16120, 16013, 798, 1662, 16416, 7032, 14581,
       15783, 16144, 5646, 8344, 13514, 15848, 15658, 14897, 12194, 10775,
       16304, 14980, 14192, 15952, 14099, 15125, 13089, 16062, 14923,
       14839, 15576, 15657, 1360, 15583, 15457, 16275,
       'bon-cadeau-25-euros', 15797, 12366, 15764, 16072, 15070, 16323,
       15753, 15755, 15582, 13379, 15704, 13072, 15862, 15533, 16028,
       14725, 13647, 16261, 15944, 15256, 523, 15452, 16322, 16014, 14600,
       15316, 15776, 16211, 14899, 14955, 13910, 15134, 15434, 15813,
       15794, 15946, 16213, 14855, 14827, 14089, 15283, 14647, 16504,
       15674, 16537, 41, 13762, 14570, 16041, 15202, 15180, 15375, 16046,
       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)
```

In [57]:
```
#Quelles sont les valeurs qui ne semblent pas respecter la régle de codification?
web_sku_valeurs_nonconformes = df_web[df_web['sku'].str.len() > 6]

#Afficher les valeurs non conformes
web_sku_valeurs_nonconformes
```

Out[57]:

	sku	total_sales	post_author	post_date	post_date_gmt	product_type	post_title	post_excerpt	post_name	post_modified	post_mo
272	13127-1	4.0	2.0	2020-06-09 15:42:04	2020-06-09 13:42:04	Vin	Clos du Mont-Olivet Châteauneuf-du-Pape 2007	Nez gracieux, très élégant avec une touche flo...	clos-du-mont-olivet-chateauneuf-du-pape-2007-2	2020-07-20 17:09:06	2020-07-
842	bon-cadeau-25-euros	7.0	1.0	2018-06-01 13:53:46	2018-06-01 11:53:46	Autre	Bon cadeau de 25€	NaN	bon-cadeau-de-25-euros	2018-06-01 14:13:57	2018-06-
1117	13127-1	4.0	2.0	2020-06-09 15:42:04	2020-06-09 13:42:04	Vin	Clos du Mont-Olivet Châteauneuf-du-Pape 2007	NaN	clos-du-mont-olivet-chateauneuf-du-pape-2007-2	2020-07-20 17:09:06	2020-07-
1387	bon-cadeau-25-euros	7.0	1.0	2018-06-01 13:53:46	2018-06-01 11:53:46	NaN	Bon cadeau de 25€	Parlons ...	bon-cadeau-de-25-euros	2018-06-01 14:13:57	2018-06-

In [58]:
```
#Identifier les lignes sans code articles
df_web[df_web['sku'].isna()]
```

Out[58]:

	sku	total_sales	post_author	post_date	post_date_gmt	product_type	post_title	post_excerpt	post_name	post_modified	post_modified_gmt
8	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
20	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
30	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
37	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
41	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
...	...	...	...	...	...	...	...	...	...	...	...
1384	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1429	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1432	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1445	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1457	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT

85 rows × 13 columns

Pour les codes articles identifiés, réalisé une analyse et définissez l'action à entreprendre

=> Plusieurs lignes contiennent des colonnes avec des valeurs NaN. Ces données ne sont pas exploitables et doivent être supprimées du DataFrame df_web.

In [60]:
```
#Trouver les lignes où TOUTES les valeurs sont NaN
lignes_vides = df_web[df_web.isna().all(axis=1)]

# Afficher le résultat
print(f"Nombre de lignes entièrement vides : {len(lignes_vides)}")
```

```
Nombre de lignes entièrement vides : 83
```

In [61]:
```
#Afficher les lignes vides
lignes_vides
```

Out[61]:

	sku	total_sales	post_author	post_date	post_date_gmt	product_type	post_title	post_excerpt	post_name	post_modified	post_modified_gmt
8	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
20	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
30	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
37	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
41	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
...	...	...	...	...	...	...	...	...	...	...	...
1384	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1429	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1432	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1445	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT
1457	NaN	NaN	NaN	NaT	NaT	NaN	NaN	NaN	NaN	NaT	NaT

83 rows × 13 columns

In [62]:
```
# Supprimer du DataFrame df_web les lignes identifiées comme vides (contenues dans lignes_vides)
df_web = df_web.dropna(how='all')

# Vérification
print(f"Nombre de lignes après suppression : {len(df_web)}")
```

```
Nombre de lignes après suppression : 1430
```

In [63]:
```
#La clé pour chaque ligne est-elle uniques? ou autrement dit, y a-t-il des doublons?
df_web['sku'].duplicated().any()
```

Out[63]: True

Ici "True" est retourné donc il y a au moins 1 doublon

In [65]:
```
# Nombre de doublons (lignes identiques sur toutes les colonnes)
nombre_doublons = df_web.duplicated().sum()
nombre_doublons
```

Out[65]: 0

In [66]:
```
#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
web_nan = df_web[df_web['sku'].isna()]
#2 - utiliser la fonction df.info() sur ce nouveau dataframe pour observer le nombre de valeur reseigner dans chacune des colonnes
web_nan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2 entries, 1084 to 1087
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sku                0 non-null      object
 1   total_sales        2 non-null      float64
 2   post_author        2 non-null      float64
 3   post_date          2 non-null      datetime64[ns]
 4   post_date_gmt      2 non-null      datetime64[ns]
 5   product_type       2 non-null      object
 6   post_title         2 non-null      object
 7   post_excerpt       2 non-null      object
 8   post_name          2 non-null      object
 9   post_modified      2 non-null      datetime64[ns]
 10  post_modified_gmt  2 non-null      datetime64[ns]
 11  guid               2 non-null      object
 12  post_type          2 non-null      object
dtypes: datetime64[ns](4), float64(2), object(7)
memory usage: 224.0+ bytes
```

In [67]:
```
#Affiche du dataframe web_nan
web_nan
```

Out[67]:

	sku	total_sales	post_author	post_date	post_date_gmt	product_type	post_title	post_excerpt	post_name	post_modified	post_modified_gmt
1084	NaN	-56.0	2.0	2018-08-08 11:23:43	2018-08-08 09:23:43	Vin	Pierre Jean Villa Condrieu Jardin Suspendu 2018	Le nez séduit par ses parf...	pierre-jean-villa-condrieu-suspendu-2018	2019-11-02 13:24:01	2019-11-02 12:24:01
1087	NaN	-17.0	2.0	2018-07-31 12:07:23	2018-07-31 10:07:23	Vin	Pierre Jean Villa Côte Rôtie Fongeant 2017	Fongeant 2017 explose sur un fruit brillant, p...	pierre-jean-villa-cote-rotie-fongeant-2017	2019-11-02 13:24:15	2019-11-02 12:24:15

2 lignes avec des sku non renseignés:

Il y a 2 lignes avec des sku non renseignés, il faudra les renseigner car tous les autres éléments des colonnes sont bien présents. Afin de les retrouver facilement on va leur assigner des sku fictifs sku_null1 et sku_null2. ceci va permettre de modifier ces lignes avec le bon sku plus tard.

In [69]:
```
#Assigner des 'sku' fictifs à ces deux lignes
df_web.loc[1084, 'sku'] = 'sku_null1'
df_web.loc[1087, 'sku'] = 'sku_null2'

#Remplacer leurs valeurs négatives dans 'total_sales' par NaN
df_web.loc[[1084, 1087], 'total_sales'] = None

#Vérifier le résultat
print(df_web.loc[[1084, 1087]])
```

```
           sku  total_sales  post_author           post_date  \
1084  sku_null1          NaN          2.0 2018-08-08 11:23:43
1087  sku_null2          NaN          2.0 2018-07-31 12:07:23

          post_date_gmt product_type  \
1084 2018-08-08 09:23:43         Vin
1087 2018-07-31 10:07:23         Vin

                                      post_title  \
1084  Pierre Jean Villa Condrieu Jardin Suspendu 2018
1087       Pierre Jean Villa Côte Rôtie Fongeant 2017

                                      post_excerpt  \
1084  <span id="u1194-83">Le nez séduit par ses parf...
1087  Fongeant 2017 explose sur un fruit brillant, p...

                                   post_name       post_modified  \
1084    pierre-jean-villa-condrieu-suspendu-2018 2019-11-02 13:24:01
1087  pierre-jean-villa-cote-rotie-fongeant-2017 2019-11-02 13:24:15

       post_modified_gmt                                          guid  \
1084 2019-11-02 12:24:01  https://www.bottle-neck.fr/?post_type=product&...
1087 2019-11-02 12:24:15  https://www.bottle-neck.fr/?post_type=product&...

     post_type
1084   product
1087   product
```

In [70]:
```
#Affichage du nombre de sku en doubon dans le dataframe
web_sku_doublons = df_web["sku"].duplicated().sum()
if web_sku_doublons > 0:
    print(f"Il y a {web_sku_doublons} doublons dans la colonne 'sku'")
else:
    print("Il n'y a pas de doublon dans la colonne 'sku'")
```

```
Il y a 714 doublons dans la colonne 'sku'
```

Dans le fichier df_web on remarque qu'il y a :

- 85 sku non renseignés au total
- dont 2 erreurs dans la colonne 'total_sales' avec des valeurs négatives
- et 2 erreurs avec des sku manquants
- 2 sku non conformes avec '13127-1' et 'bon-cadeau-25-euros' dupliqué
- 714 sku en doublons

On va donc supprimer ces doublons

In [72]:
```
df_web[df_web.duplicated(subset=['sku'], keep=False)]
#Supprimer les doublons
df_web.drop_duplicates(subset=['sku'], keep='first')
```

Out[72]:

	sku	total_sales	post_author	post_date	post_date_gmt	product_type	post_title	post_excerpt	post_name	post_modified	post_modi
0	11862	3.0	2.0	2018-02-12 13:46:23	2018-02-12 12:46:23	Vin	Gilles Robin Hermitage Rouge 2012	NaN	gilles-robin-hermitage-2012	2019-01-31 12:12:56	2019-01-3
1	16057	5.0	2.0	2018-04-17 15:29:17	2018-04-17 13:29:17	Vin	Domaine Pellé Sancerre Rouge La Croix Au Garde...	NaN	pelle-sancerre-rouge-la-croix-au-garde-2017	2020-07-07 10:05:02	2020-07-0
2	14692	5.0	2.0	2019-03-19 10:06:47	2019-03-19 09:06:47	Vin	Château Fonréaud Bordeaux Blanc Le Cygne 2016	Grâce à la complémentarité des 3 cépages ...	fonreaud-bordeaux-blanc-le-cygne-2016	2020-04-25 21:40:31	2020-04-2
3	16295	14.0	2.0	2018-02-15 14:05:06	2018-02-15 13:05:06	Vin	Moulin de Gassac IGP Pays d'Hérault Guilhem Ro...	NaN	moulin-de-gassac-igp-pays-dherault-guilhem-ros...	2020-08-27 18:55:03	2020-08-2
4	15328	2.0	2.0	2019-03-27 18:05:09	2019-03-27 17:05:09	Vin	Agnès Levet Côte Rôtie Maestria 2017	<span style="float: none; background-color: tr...	agnes-levet-cote-rotie-maestria-2017	2020-07-25 15:45:02	2020-07-2
...	...	...	...	...	...	...	...	...	...	...	
1325	16043	10.0	2.0	2018-02-12 10:41:11	2018-02-12 09:41:11	Vin	Pierre Gaillard Saint-Joseph Rouge 2018	Un Saint-Joseph plein de fruits et de gourmand...	pierre-gaillard-saint-joseph-2018	2020-08-26 14:05:02	2020-08-2
1327	15845	9.0	2.0	2020-04-25 12:43:23	2020-04-25 10:43:23	Vin	Château Jean Faure Saint-Emilion Grand Cru 2016	NaN	chateau-jean-faure-saint-emilion-grand-cru-2016	2020-07-20 17:09:23	2020-07-2
1339	15951	9.0	2.0	2019-05-16 15:54:52	2019-05-16 13:54:52	Vin	Jacqueson Rully Rouge 1er Cru Les Preaux 2018	NaN	jacqueson-rully-rouge-1er-cru-les-preaux-2018	2020-08-07 15:55:03	2020-08-0
1353	13074	4.0	2.0	2018-02-12 14:25:28	2018-02-12 13:25:28	Vin	Château de Vaudieu Châteauneuf-du-Pape L'Avenu...	NaN	chateau-de-vaudieu-chateauneuf-du-pape-lavenue...	2019-12-09 10:40:03	2019-12-0
1391	14569	13.0	2.0	2019-06-28 17:56:00	2019-06-28 15:56:00	Vin	Moulin de Gassac IGP Pays d'Hérault Blanc Faun...	NaN	moulin-de-gassac-igp-pays-dherault-faune-2017	2020-08-14 17:35:02	2020-08-1

716 rows × 13 columns

2.3 - Analyse exploratoire du fichier liaison.xlsx

In [74]:
```
#Dimension du dataset
dimensions = df_liaison.shape
#Nombre d'observations
nb_observations = dimensions[0]
#Nombre de caractéristiques
nb_caracteristiques = dimensions[1]
# Affichage
print(f"Le dataset contient {nb_observations} observations et {nb_caracteristiques} caractéristiques.")
```

```
Le dataset contient 825 observations et 2 caractéristiques.
```

In [75]:
```
#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
```

In [76]:
```python
#Les valeurs de la colonne "product_id" sont elles toutes uniques?
productid_unique = df_liaison['product_id'].is_unique

print(productid_unique)
```

```
True
```

Ici True est retourné donc les valeurs de product_id sont uniques.

In [78]:
```python
#Les valeurs de la colonne "id_web" sont-elles toutes uniques?
idweb_unique = df_liaison['id_web'].is_unique

print(idweb_unique)
```

```
False
```

Ici False est retourné donc les valeurs de la colonne id_web ne sont pas uniques

In [80]:
```python
#Affichage du nombre de doublons dans la colonne id_web
nb_doublons = df_liaison['id_web'].duplicated().sum()
print(f"Nombre de doublons dans 'id_web' : {nb_doublons}")
```

```
Nombre de doublons dans 'id_web' : 90
```

Il ya 90 doublons dans la colonnes id_web donc on va supprimer ces doublons

In [82]:
```python
#Suppression des doublons dans la colonne id_web
df_liaison = df_liaison.drop_duplicates(subset='id_web', keep='first')
```

In [83]:
```python
#Affichage du nombre de doublons dans la colonne id_web après suppression des doublons
nb_doublons = df_liaison['id_web'].duplicated().sum()
print(f"Nombre de doublons dans 'id_web' : {nb_doublons}")
```

```
Nombre de doublons dans 'id_web' : 0
```

Etape 3 - Jonction des fichiers

Etape 3.1 - Jonction du fichier df_erp et df_liaison

In [86]:
```python
#Fusion des fichiers df_erp et df_liaison
df_merge = pd.merge(df_erp, df_liaison, on='product_id', how='inner')
```

In [87]:
```python
#Y a t-il des lignes ne "matchant" entre les 2 fichiers?
articles_sans_correspondance = df_erp[~df_erp['product_id'].isin(df_liaison['product_id'])]

if not articles_sans_correspondance.empty:
    print("Il y a des articles sans correspondances :")
    print(articles_sans_correspondance)
else:
    print("Tous les articles ont une correspondance.")
```

```
Il y a des articles sans correspondances :
     product_id  onsale_web  price  stock_quantity stock_status  \
49         4090           0   73.0               0   outofstock
50         4092           0   47.0               0   outofstock
119        4195           0   14.1               0   outofstock
131        4209           0   73.5               0   outofstock
151        4233           0    NaN               0   outofstock
..          ...         ...    ...             ...          ...
817        7196           0   31.0              55      instock
818        7200           0   31.0               6      instock
819        7201           0   31.0              18      instock
820        7203           0   45.0              30      instock
821        7204           0   45.0               9      instock

     purchase_price
49            33.79
50            25.25
119            7.36
131           33.01
151           10.33
..              ...
817           31.20
818           15.54
819           16.02
820           23.48
821           24.18

[90 rows x 6 columns]
```

In [88]:
```
#Affichage des articles sans correspondance
articles_sans_correspondance
```

Out[88]:

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price
49	4090	0	73.0	0	outofstock	33.79
50	4092	0	47.0	0	outofstock	25.25
119	4195	0	14.1	0	outofstock	7.36
131	4209	0	73.5	0	outofstock	33.01
151	4233	0	NaN	0	outofstock	10.33
...	...	...	...	...	...	...
817	7196	0	31.0	55	instock	31.20
818	7200	0	31.0	6	instock	15.54
819	7201	0	31.0	18	instock	16.02
820	7203	0	45.0	30	instock	23.48
821	7204	0	45.0	9	instock	24.18

90 rows × 6 columns

Etape 3.2 - Jonction du fichier df_merge et df_web

In [90]:
```
#Changement du nom de colonne : 'sku' devient 'id_web' dans df_web
df_web = df_web.rename(columns={'sku': 'id_web'})
```

In [193…
```
#Fusion des datasets df_merge et df_web
df_merge = pd.merge(df_merge, df_web, on='id_web', how='inner')
```

In [92]:
```
#Affichage de df_merge
df_merge
```

Out[92]:

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	total_sales	post_author	post_date	post_date_gmt	produc
0	3847	1	24.2	16	instock	12.88	15298	6.0	2.0	2018-02-08 12:58:52	2018-02-08 11:58:52	
1	3847	1	24.2	16	instock	12.88	15298	6.0	2.0	2018-02-08 12:58:52	2018-02-08 11:58:52	
2	3849	1	34.3	10	instock	17.54	15296	9.0	2.0	2018-02-08 13:49:41	2018-02-08 12:49:41	
3	3849	1	34.3	10	instock	17.54	15296	9.0	2.0	2018-02-08 13:49:41	2018-02-08 12:49:41	
4	3850	1	20.8	0	outofstock	10.64	15300	0.0	2.0	2018-02-08 14:08:36	2018-02-08 13:08:36	
...	...	...	...	...	...	...	...	...	...	...	...	
1423	7025	1	69.0	8	instock	34.22	15887	5.0	2.0	2020-05-02 15:00:54	2020-05-02 13:00:54	
1424	7247	1	54.8	6	instock	27.18	13127-1	4.0	2.0	2020-06-09 15:42:04	2020-06-09 13:42:04	
1425	7247	1	54.8	6	instock	27.18	13127-1	4.0	2.0	2020-06-09 15:42:04	2020-06-09 13:42:04	
1426	7338	1	16.3	40	instock	8.00	16230	13.0	2.0	2020-07-20 11:00:00	2020-07-20 09:00:00	
1427	7338	1	16.3	40	instock	8.00	16230	13.0	2.0	2020-07-20 11:00:00	2020-07-20 09:00:00	

1428 rows × 19 columns

In [93]:
```
#Avons-nous des lignes sans correspondances?
lignes_sans_correspondance = df_merge[~df_merge['id_web'].isin(df_web['id_web'])]

if not articles_sans_correspondance.empty:
    print("Il y a des articles sans correspondances :")
    print(articles_sans_correspondance)
else:
    print("Tous les articles ont une correspondance.")
```

```
Il y a des articles sans correspondances :
     product_id  onsale_web  price  stock_quantity stock_status  \
49         4090           0   73.0               0   outofstock
50         4092           0   47.0               0   outofstock
119        4195           0   14.1               0   outofstock
131        4209           0   73.5               0   outofstock
151        4233           0    NaN               0   outofstock
..          ...         ...    ...             ...          ...
817        7196           0   31.0              55      instock
818        7200           0   31.0               6      instock
819        7201           0   31.0              18      instock
820        7203           0   45.0              30      instock
821        7204           0   45.0               9      instock

     purchase_price
49            33.79
50            25.25
119            7.36
131           33.01
151           10.33
..              ...
817           31.20
818           15.54
819           16.02
820           23.48
821           24.18

[90 rows x 6 columns]
```

In [94]:
```
#Suppression des doublons dans la colonne (post_title)
df_merge = df_merge.drop_duplicates(subset='post_title')
```

In [95]:
```
#Affichage de df_merge après suppression des doublons
df_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 711 entries, 0 to 1426
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   product_id         711 non-null    int64
 1   onsale_web         711 non-null    int64
 2   price              711 non-null    float64
 3   stock_quantity     711 non-null    int64
 4   stock_status       711 non-null    object
 5   purchase_price     711 non-null    float64
 6   id_web             711 non-null    object
 7   total_sales        711 non-null    float64
 8   post_author        711 non-null    float64
 9   post_date          711 non-null    datetime64[ns]
 10  post_date_gmt      711 non-null    datetime64[ns]
 11  product_type       711 non-null    object
 12  post_title         711 non-null    object
 13  post_excerpt       360 non-null    object
 14  post_name          711 non-null    object
 15  post_modified      711 non-null    datetime64[ns]
 16  post_modified_gmt  711 non-null    datetime64[ns]
 17  guid               711 non-null    object
 18  post_type          711 non-null    object
dtypes: datetime64[ns](4), float64(4), int64(3), object(8)
memory usage: 111.1+ KB
```

Conversion des colonnes

Maintenant que les données ont été nettoyées et fusionnées, nous allons ajuster les types des colonnes pour garantir leur cohérence et optimiser leur traitement.

In [97]:
```
# Conversion de la colonne id_web en type int
df_merge.loc[:, 'price'] = df_merge['price'].astype(int)

# Conversion de la colonne stock_status en type category
df_merge.loc[:, 'stock_status'] = df_merge['stock_status'].astype('category')
```

In [98]:
```
print(f"Nombre total d'articles dans df_merge : {len(df_merge)}")
print(f"Nombre d'articles sans correspondance : {len(articles_sans_correspondance)}")
```

```
Nombre total d'articles dans df_merge : 711
Nombre d'articles sans correspondance : 90
```

In [99]:
```
#Affichage des articles sans correspondance
articles_sans_correspondance
```

Out[99]:

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price
49	4090	0	73.0	0	outofstock	33.79
50	4092	0	47.0	0	outofstock	25.25
119	4195	0	14.1	0	outofstock	7.36
131	4209	0	73.5	0	outofstock	33.01
151	4233	0	NaN	0	outofstock	10.33
...	...	...	...	...	...	...
817	7196	0	31.0	55	instock	31.20
818	7200	0	31.0	6	instock	15.54
819	7201	0	31.0	18	instock	16.02
820	7203	0	45.0	30	instock	23.48
821	7204	0	45.0	9	instock	24.18

90 rows × 6 columns

In [100…
```
#Vérification des articles non présent sur le site web dans le fichier sans correspondance onsale= 0
article_non_commercialisé = (articles_sans_correspondance['onsale_web'] == 0).sum()
print(f"Nombre de valeurs égales à 0 dans 'onsale_web': {article_non_commercialisé}")
```

```
Nombre de valeurs égales à 0 dans 'onsale_web': 87
```

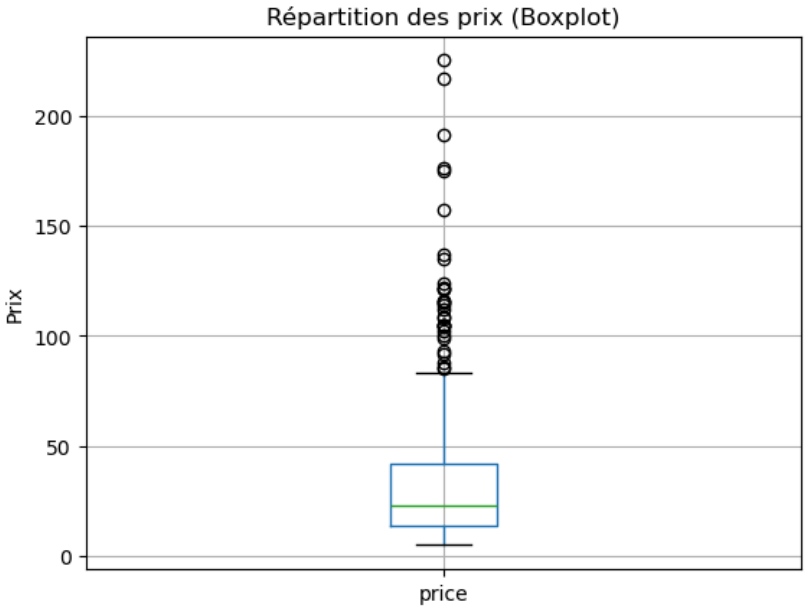
Remarque

Ces valeurs semblent être des articles qui ne sont plus en vente sur le site web avec 'onsale_web' à 0. Est ce qu'il s'agit d'articles qui ont été rétirés de la vente? Il faudra mettre à jour le fichier erp.

Etape 4 - Analyse univarié des prix

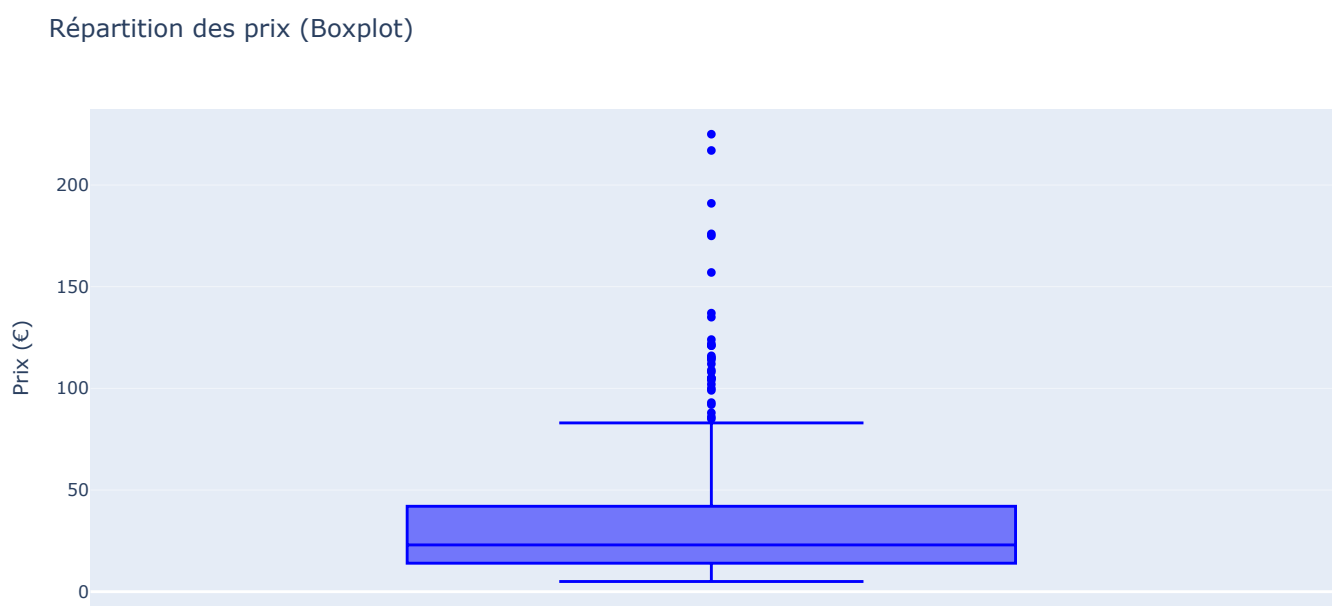
Etape 4.1 - Exploration par la visualisation de données

In [104…
```
#Création d'une Boite à moustache (boxplot) de la répartition des prix grâce à Pandas
df_merge.boxplot(column='price')

# Affichage du graphique
plt.title("Répartition des prix (Boxplot)")
plt.ylabel("Prix")
plt.show()
```



In [105…
```
#Autre méthode avec plotly express

import plotly.express as px
fig = px.box(
    df_merge,
    y='price',
    title='Répartition des prix (Boxplot)',
```

```
    labels={'price': 'Prix (€)'},
    color_discrete_sequence=['blue'])

# Affichage du graphique
fig.show()
```



Etape 4.2 - Exploration par l'utisation de méthodes statistique

Etape 4.2.1 - Identification par le Z-index

In [108…
```
#Calculer la moyenne du prix
moy_prix = df_merge['price'].mean()
print("Moyenne des prix : {:.2f}".format(moy_prix))
#Calculer l'écart-type du prix
ecart_type_prix = df_merge['price'].std()
print("Écart-type des prix : {:.2f}".format(ecart_type_prix))
```

```
Moyenne des prix : 31.85
Écart-type des prix : 27.70
```

In [141…
```
#Calculer le Z-score
df_merge['z_score_price'] = (df_merge['price'] - df_merge['price'].mean()) / df_merge['price'].std()
#Affichage de la colonne
df_merge['z_score_price'].head()
```

Out[141…
```
0    0.980329
1    6.974126
2   -0.716710
3    0.980329
4   -0.752818
Name: z_score_price, dtype: float64
```

In [112…
```
# Quel est le seuil de prix dont le z-score est supérieur à 3
seuil_prix = df_merge.loc[df_merge['z_score_price'] > 3, 'price'].min()

print(f"Le seuil de prix avec un z-score > 3 est de: {seuil_prix:.2f}€")
```

```
Le seuil de prix avec un z-score > 3 est de: 115.00€
```

Etape 4.2.2 - Identification par l'interval interquartile

In [115…
```
#Utilisation de la fonction describe de Pandas pour l'étude des mesures de dispersions
df_merge.describe()
```

Out[115…

	product_id	onsale_web	price	stock_quantity	purchase_price	total_sales	post_author	post_date	post_date_gmt	post_
count	711.000000	711.000000	711.000000	711.000000	711.000000	711.000000	711.000000	711	711	
mean	5026.427567	0.998594	31.849508	23.513361	16.872250	8.540084	1.998594	2018-08-20 07:58:32.623066112	2018-08-20 06:29:46.040787456	2(21:23:01.5
min	3847.000000	0.000000	5.000000	0.000000	2.740000	0.000000	1.000000	2018-02-08 12:58:52	2018-02-08 11:58:52	2(
25%	4278.500000	1.000000	14.000000	9.000000	7.230000	5.000000	2.000000	2018-02-27 14:05:46.500000	2018-02-27 13:05:46.500000	2(
50%	4794.000000	1.000000	23.000000	21.000000	12.280000	8.000000	2.000000	2018-04-19 14:41:18	2018-04-19 12:41:18	2(
75%	5708.000000	1.000000	42.000000	30.500000	22.025000	11.000000	2.000000	2019-01-31 14:19:45.500000	2019-01-31 13:19:45.500000	2(
max	7338.000000	1.000000	225.000000	145.000000	137.810000	122.000000	2.000000	2020-07-20 11:00:00	2020-07-20 09:00:00	2(
std	785.163566	0.037503	27.695296	22.237249	14.843267	8.159727	0.037503	NaN	NaN	

In [117…

```
#Définissez un seuil pour les articles "outliers" en prix
# Récupération des quartiles pour la colonne "price"
Q1 = df_merge['price'].quantile(0.25)
Q3 = df_merge['price'].quantile(0.75)
IQR = Q3 - Q1

# Définition du seuil
seuil_haut = Q3 + 1.5 * IQR

print(f"Seuil haut : {seuil_haut:.2f}")
```

```
Seuil haut : 84.00
```

In [189…

```
# Détection des outliers
outliers = df_merge[df_merge['price'] > seuil_haut]

# Affichage des outliers
outliers.head()
```

Out[189…

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	total_sales	post_author	post_date	…	ca_par_article	part_(
534	6126.0	1.0	135.0	138.0	instock	80.33	14923	5.0	2.0	2019-06-28 17:22:27	…	675.0	0.(
487	5025.0	1.0	112.0	136.0	instock	68.60	13914	6.0	2.0	2018-07-18 10:39:43	…	672.0	0.(
682	5612.0	1.0	124.0	19.0	instock	66.41	14915	1.0	2.0	2019-01-15 15:30:49	…	124.0	0.(
472	5892.0	1.0	191.0	98.0	instock	116.06	14983	6.0	2.0	2019-03-28 10:21:36	…	1146.0	0.(
436	4359.0	1.0	85.0	112.0	instock	51.93	13853	7.0	2.0	2018-03-02 11:11:48	…	595.0	0.(

5 rows × 30 columns

In [121…

```
#Définissez le nombre d'articles et la proportion de l'ensemble du catalogue "outliers"
# Nombre d'articles outliers
nb_outliers = outliers.shape[0]

# Proportion par rapport à l'ensemble du catalogue
proportion_outliers = nb_outliers / df_merge.shape[0]

# Affichage
print(f"Nombre d'outliers : {nb_outliers}")
print(f"Proportion d'outliers : {proportion_outliers:.2%}")
```

```
Nombre d'outliers : 31
Proportion d'outliers : 4.35%
```

In [123…

```
#Affichage des outliers
outliers
```

Out[123...

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	total_sales	post_author	post_date	post_date_gmt	produc
126	4115.0	1.0	100.0	12.0	instock	52.70	15382	1.0	2.0	2018-02-13 11:08:45	2018-02-13 10:08:45	
130	4132.0	1.0	88.0	7.0	instock	44.30	11668	5.0	2.0	2018-02-13 11:43:55	2018-02-13 10:43:55	
398	4352.0	1.0	225.0	0.0	outofstock	137.81	15940	11.0	2.0	2018-03-02 10:30:04	2018-03-02 09:30:04	Cham
410	4359.0	1.0	85.0	112.0	instock	51.93	13853	7.0	2.0	2018-03-02 11:11:48	2018-03-02 10:11:48	Cham
436	4402.0	1.0	176.0	11.0	instock	78.25	3510	3.0	2.0	2018-03-22 11:21:05	2018-03-22 10:21:05	C
438	4404.0	1.0	108.0	17.0	instock	52.22	3507	4.0	2.0	2018-03-22 11:32:55	2018-03-22 10:32:55	C
442	4406.0	1.0	157.0	12.0	instock	69.08	7819	4.0	2.0	2018-03-22 11:42:48	2018-03-22 10:42:48	C
444	4407.0	1.0	104.0	14.0	instock	46.71	3509	5.0	2.0	2018-03-22 11:49:53	2018-03-22 10:49:53	C
454	4582.0	1.0	109.0	18.0	instock	53.80	12857	1.0	2.0	2018-04-12 17:56:13	2018-04-12 15:56:13	
760	4903.0	1.0	102.0	12.0	instock	51.80	14805	2.0	2.0	2018-05-15 10:10:57	2018-05-15 08:10:57	
762	4904.0	1.0	137.0	9.0	instock	67.95	14220	3.0	2.0	2018-05-15 10:23:41	2018-05-15 08:23:41	
852	5001.0	1.0	217.0	18.0	instock	116.87	14581	2.0	2.0	2018-07-17 09:45:39	2018-07-17 07:45:39	
862	5007.0	1.0	105.0	15.0	instock	55.88	12791	3.0	2.0	2018-07-17 10:36:03	2018-07-17 08:36:03	
864	5008.0	1.0	105.0	12.0	instock	56.42	11602	7.0	2.0	2018-07-17 10:52:41	2018-07-17 08:52:41	
874	5025.0	1.0	112.0	136.0	instock	68.60	13914	6.0	2.0	2018-07-18 10:39:43	2018-07-18 08:39:43	Cham
876	5026.0	1.0	86.0	101.0	instock	50.13	13913	9.0	2.0	2018-07-18 10:46:30	2018-07-18 08:46:30	Cham
1004	5565.0	1.0	92.0	0.0	outofstock	46.11	19822	1.0	2.0	2018-11-26 10:59:10	2018-11-26 09:59:10	
1022	5612.0	1.0	124.0	19.0	instock	66.41	14915	1.0	2.0	2019-01-15 15:30:49	2019-01-15 14:30:49	

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	total_sales	post_author	post_date	post_date_gmt	produc
1106	5767.0	1.0	175.0	12.0	instock	90.42	15185	4.0	2.0	2019-03-13 14:43:22	2019-03-13 13:43:22	
1174	5892.0	1.0	191.0	98.0	instock	116.06	14983	6.0	2.0	2019-03-28 10:21:36	2019-03-28 09:21:36	Cham
1204	5916.0	1.0	93.0	1.0	instock	40.49	14774	1.0	2.0	2019-04-04 16:39:24	2019-04-04 14:39:24	
1206	5917.0	1.0	122.0	12.0	instock	54.24	14775	3.0	2.0	2019-04-04 16:49:37	2019-04-04 14:49:37	
1208	5918.0	1.0	114.0	12.0	instock	52.25	14773	3.0	2.0	2019-04-04 17:01:54	2019-04-04 15:01:54	
1284	6126.0	1.0	135.0	138.0	instock	80.33	14923	5.0	2.0	2019-06-28 17:22:27	2019-06-28 15:22:27	Cham
1294	6201.0	1.0	105.0	16.0	instock	57.29	14596	6.0	2.0	2019-07-23 10:37:14	2019-07-23 08:37:14	
1296	6202.0	1.0	116.0	12.0	instock	63.15	15126	5.0	2.0	2019-07-23 10:50:24	2019-07-23 08:50:24	
1306	6212.0	1.0	115.0	16.0	instock	59.42	13996	7.0	2.0	2019-07-25 09:09:17	2019-07-25 07:09:17	
1308	6213.0	1.0	121.0	9.0	instock	63.14	15072	3.0	2.0	2019-07-25 09:10:32	2019-07-25 07:10:32	
1310	6214.0	1.0	99.0	9.0	instock	49.62	11601	6.0	2.0	2019-07-25 09:15:41	2019-07-25 07:15:41	
1312	6215.0	1.0	115.0	14.0	instock	56.45	12790	4.0	2.0	2019-07-25 09:30:16	2019-07-25 07:30:16	
1314	6216.0	1.0	121.0	14.0	instock	60.02	15070	2.0	2.0	2019-07-25 09:31:09	2019-07-25 07:31:09	

Remarque

Il y a 31 produits qui depassent le seuil de 84 euros. Mais ces prix semblent dans l'ensemble cohérent par rapport au prix d'achat. Ces outliers sont donc justifiés dans le sens où nous commercialisont des vins, certains d'exeptions qui coûtent plus cher que d'autres. Si on regarde le prix d'achat "purchase price" on remarque que le prix d'achat est cohérent avec le prix de vente. Par exemple pour le product ID 4055 le purchase price est de 37.88 et il a été vendu à 86,1.

In [137…

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

# Créez une colonne calculant le CA par article avec .loc
df_merge.loc[:, 'ca_par_article'] = round(df_merge['price'] * df_merge['total_sales'], 0)

# Calcul de la somme de la colonne "ca_par_article"
ca_total = df_merge['ca_par_article'].sum()
```

```
# Formatage du CA total pour avoir un espace
formatted_ca = f"{ca_total:,.0f}".replace(',', ' ')

# Affichage du résultat
print(f"Chiffre d'affaires total du site : {formatted_ca}€")
```

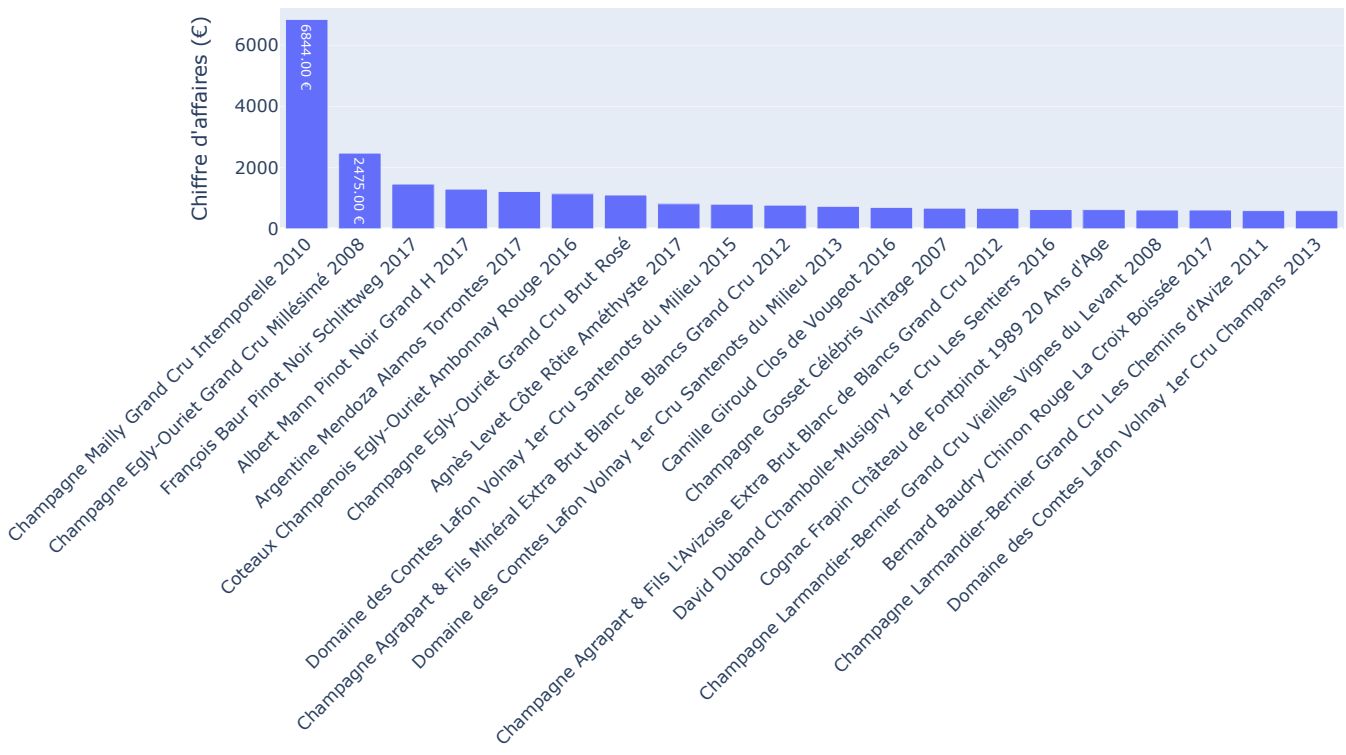
```
Chiffre d'affaires total du site : 150 752€
```

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

In [139…

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


# Trier par CA décroissant et réinitialiser l'index
top20 = df_merge.sort_values('ca_par_article', ascending=False).reset_index(drop=True).head(20)

#Graphique en barre des 20 premiers articles avec plotly express
fig = px.bar(
    top20,
    x='post_title',
    y='ca_par_article',
    title='Top 20 articles par chiffre d\'affaires',
    text='ca_par_article',
    labels={'post_title': 'Article', 'ca_par_article': 'CA (€)'})

fig.update_traces(
    texttemplate='%{text:.2f} €',)

fig.update_layout(
    xaxis_tickangle=-45,
    yaxis_title='Chiffre d\'affaires (€)',
    xaxis_title='Article',
    uniformtext_minsize=8,
    uniformtext_mode='hide',
    height=600,
    margin=dict(t=50, b=150),
)

fig.show()
```



In [132…
```
# Afficher les 20 premiers articles
print(top20[['post_title', 'ca_par_article']])
```

	post_title	ca_par_article
0	Champagne Mailly Grand Cru Intemporelle 2010	6844.0
1	Champagne Egly-Ouriet Grand Cru Millésimé 2008	2475.0
2	François Baur Pinot Noir Schlittweg 2017	1464.0
3	Albert Mann Pinot Noir Grand H 2017	1298.0
4	Argentine Mendoza Alamos Torrontes 2017	1221.0
5	Coteaux Champenois Egly-Ouriet Ambonnay Rouge ...	1146.0
6	Champagne Egly-Ouriet Grand Cru Brut Rosé	1106.0
7	Agnès Levet Côte Rôtie Améthyste 2017	820.0
8	Domaine des Comtes Lafon Volnay 1er Cru Santen...	805.0
9	Champagne Agrapart & Fils Minéral Extra Br...	774.0
10	Domaine des Comtes Lafon Volnay 1er Cru Santen...	735.0
11	Camille Giroud Clos de Vougeot 2016	700.0
12	Champagne Gosset Célébris Vintage 2007	675.0
13	Champagne Agrapart & Fils L'Avizoise Extra...	672.0
14	David Duband Chambolle-Musigny 1er Cru Les Sen...	630.0
15	Cognac Frapin Château de Fontpinot 1989 20 Ans...	628.0
16	Champagne Larmandier-Bernier Grand Cru Vieille...	616.0
17	Bernard Baudry Chinon Rouge La Croix Boissée 2017	616.0
18	Champagne Larmandier-Bernier Grand Cru Les Che...	595.0
19	Domaine des Comtes Lafon Volnay 1er Cru Champa...	594.0

In [134…
```
##############################
# Calculer le 20 / 80 en CA #
##############################

# Trier df_merge par CA décroissant
df_merge = df_merge.sort_values(by='ca_par_article', ascending=False).reset_index(drop=True)

#Créer une colonne calculant la part du CA de la ligne dans le dataset
df_merge['part_CA_pct'] = df_merge['ca_par_article'] / df_merge['ca_par_article'].sum()

#Créer une colonne réalisant la somme cumulative de la colonne précedemment créée
df_merge['cumul_part_pct'] = df_merge['part_CA_pct'].cumsum()

#Grâce au deux colonnes créées précedemment, calculer le nombre d'articles représentant 80% du CA
nb_articles_80 = (df_merge['cumul_part_pct'] <= 0.8).sum()

#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier du site web
proportion_80 = nb_articles_80 / len(df_merge)

# Affichage des résultats
print(f"Nombre d'articles représentant 80% du CA : {nb_articles_80}")
print(f"Proportion dans le catalogue : {proportion_80:.2%}")
```

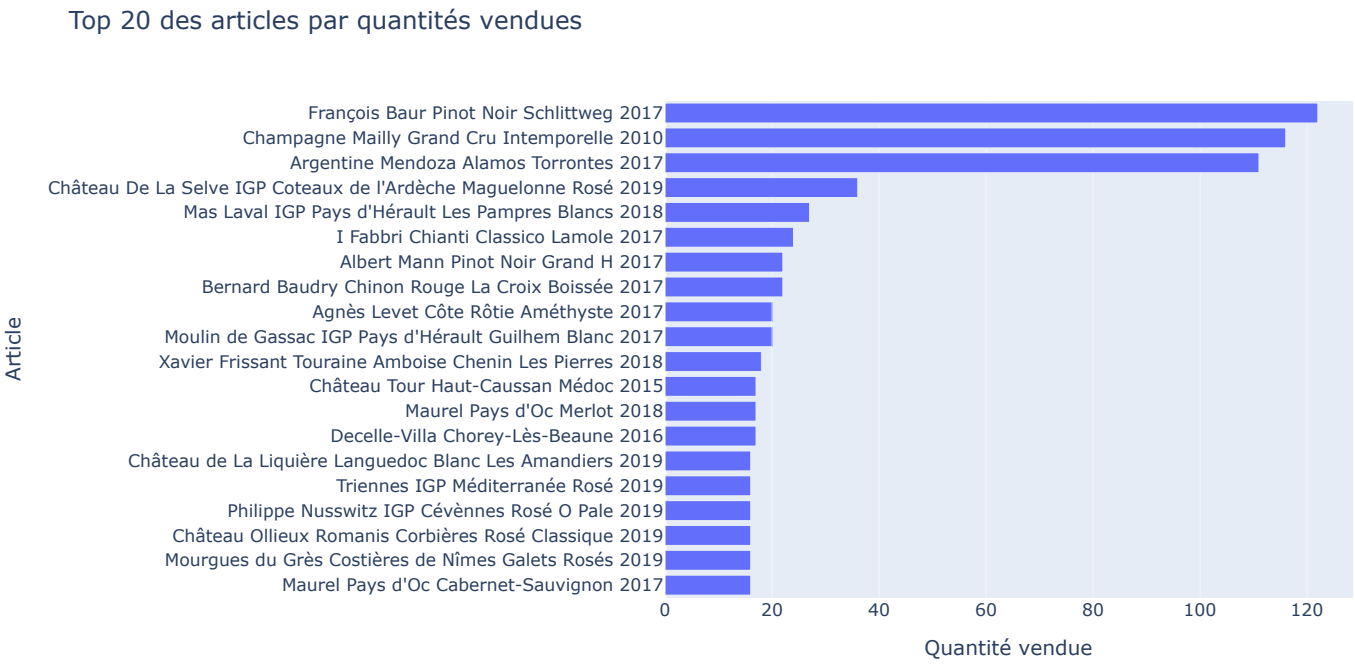
```
Nombre d'articles représentant 80% du CA : 415
Proportion dans le catalogue : 58.29%
```

Etape 5.2 - Analyse des ventes en Quantités

In [143…
```
####################################
# 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
df_quantites = df_merge.sort_values(by='total_sales', ascending=False).reset_index(drop=True)

# Afficher les 20 premiers articles en quantité
df_merge_top_20_qte = df_quantites.head(20)
print(df_merge_top_20_qte[['post_title', 'total_sales']])
```

	post_title	total_sales
0	François Baur Pinot Noir Schlittweg 2017	122.0
1	Champagne Mailly Grand Cru Intemporelle 2010	116.0
2	Argentine Mendoza Alamos Torrontes 2017	111.0
3	Château De La Selve IGP Coteaux de l'Ardèche M...	36.0
4	Mas Laval IGP Pays d'Hérault Les Pampres Blanc...	27.0
5	I Fabbri Chianti Classico Lamole 2017	24.0
6	Albert Mann Pinot Noir Grand H 2017	22.0
7	Bernard Baudry Chinon Rouge La Croix Boissée 2017	22.0
8	Agnès Levet Côte Rôtie Améthyste 2017	20.0
9	Moulin de Gassac IGP Pays d'Hérault Guilhem Bl...	20.0
10	Xavier Frissant Touraine Amboise Chenin Les Pi...	18.0
11	Château Tour Haut-Caussan Médoc 2015	17.0
12	Maurel Pays d'Oc Merlot 2018	17.0
13	Decelle-Villa Chorey-Lès-Beaune 2016	17.0
14	Château de La Liquière Languedoc Blanc Les Ama...	16.0
15	Triennes IGP Méditerranée Rosé 2019	16.0
16	Philippe Nusswitz IGP Cévènnes Rosé O Pale 2019	16.0
17	Château Ollieux Romanis Corbières Rosé Classiq...	16.0
18	Mourgues du Grès Costières de Nîmes Galets Ros...	16.0
19	Maurel Pays d'Oc Cabernet-Sauvignon 2017	16.0

In [145…
```python
#Graphique en barre des 20 premiers articles avec plotly express
fig = px.bar(
    df_merge_top_20_qte,
    x='total_sales',
    y='post_title',
    orientation='h',
    title='Top 20 des articles par quantités vendues',
    labels={
        'post_title': 'Article',
        'total_sales': 'Quantité vendue'})

# Inverser l'ordre pour avoir le plus vendu en haut
fig.update_layout(yaxis=dict(autorange="reversed"))

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



In [147…
```python
##############################
# Calculer le 20 / 80 en quantité #
##############################

# trie par quantités vendues décroissantes
df_merge = df_merge.sort_values(by='total_sales', ascending=False).reset_index(drop=True)
#Créer une colonne calculant la part en quantité de la ligne dans le dataset
df_merge['part_qte'] = df_merge['total_sales'] / df_merge['total_sales'].sum()

#Créer une colonne réalisant la somme cumulative de la colonne précedemment créée
df_merge['cumul_part_qte'] = df_merge['part_qte'].cumsum()

#Grâce au deux colonnes créées précedemment, calculer le nombre d'articles représentant 80% des ventes en quantité
nb_articles_80_qte = (df_merge['cumul_part_qte'] <= 0.8).sum()

# Calculer le nombre total d'articles dans le catalogue
total_articles = len(df_merge)

#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier du site web
proportion_80_qte = nb_articles_80_qte / len(df_merge)

# Calculer le total des quantités vendues correspondant à ces articles
total_qte_80 = df_merge.loc[df_merge['cumul_part_qte'] <= 0.8, 'total_sales'].sum()


# Afficher les résultats
print(f"Nombre total d'articles dans le catalogue : {total_articles}")
print(f"Nombre d'articles représentant 80% des quantités vendues : {nb_articles_80_qte}")
print(f"Proportion dans le catalogue : {proportion_80_qte:.2%}")
print(f"Quantité totale vendue par ces articles : {total_qte_80}")
```

```
Nombre total d'articles dans le catalogue : 712
Nombre d'articles représentant 80% des quantités vendues : 423
Proportion dans le catalogue : 59.41%
Quantité totale vendue par ces articles : 4854.0
```

In [149…
```python
#Quantités vendues totales
total_qte_vendues = df_merge['total_sales'].sum()
print(f"Le total des quantités vendues est de: {total_qte_vendues}")
```

```
Le total des quantités vendues est de: 6072.0
```

Etape 5.3 - Analyse des stocks

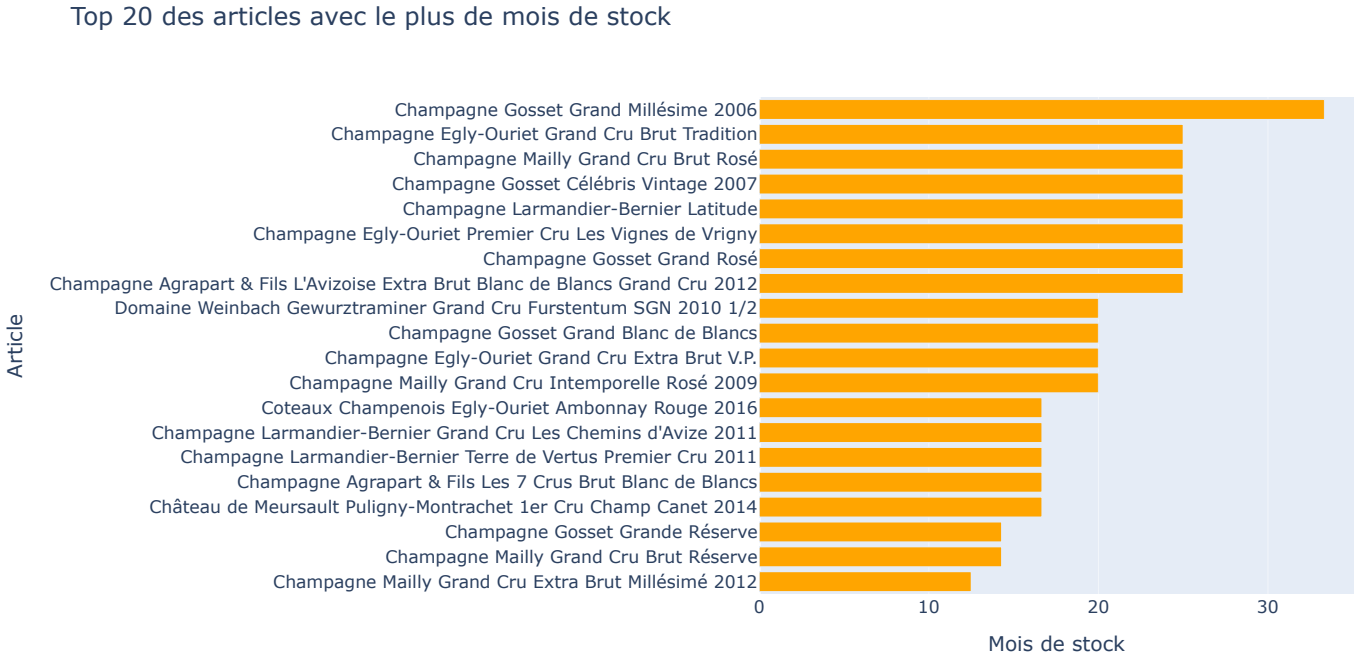
In [191…

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

# Calcul de la rotation de stock
df_merge['rotation_stock'] = round(df_merge['total_sales'] / df_merge['stock_quantity'], 2)

# Remplacement des infinis par 0
df_merge['rotation_stock'] = df_merge['rotation_stock'].replace([np.inf, -np.inf], 0)

# Calcul du nombre de mois de stock
# Pour éviter la division par zéro, on gère les cas où rotation_stock est 0
df_merge['mois_stock'] = df_merge.apply(
    lambda row: round(1 / row['rotation_stock'], 2) if row['rotation_stock'] > 0 else 0,
    axis=1
)

# Trier par mois_stock décroissant
df_merge = df_merge.sort_values(by='mois_stock', ascending=False)

# TOP 20 des produits avec le plus de mois de stock
top20_mois_stock = df_merge.head(20).reset_index(drop=True)
print(top20_mois_stock[['post_title', 'mois_stock']])
```

```
                                   post_title  mois_stock
0              Champagne Gosset Grand Millésime 2006       33.33
1   Champagne Egly-Ouriet Premier Cru Les Vignes d...       25.00
2   Champagne Agrapart &amp;amp; Fils L'Avizoise Extra...       25.00
3                         Champagne Gosset Grand Rosé       25.00
4      Champagne Egly-Ouriet Grand Cru Brut Tradition       25.00
5               Champagne Larmandier-Bernier Latitude       25.00
6                Champagne Mailly Grand Cru Brut Rosé       25.00
7              Champagne Gosset Célébris Vintage 2007       25.00
8   Domaine Weinbach Gewurztraminer Grand Cru Furs...       20.00
9             Champagne Gosset Grand Blanc de Blancs       20.00
10    Champagne Egly-Ouriet Grand Cru Extra Brut V.P.       20.00
11  Champagne Mailly Grand Cru Intemporelle Rosé 2009       20.00
12  Champagne Larmandier-Bernier Terre de Vertus P...       16.67
13  Champagne Agrapart &amp;amp; Fils Les 7 Crus Brut ...       16.67
14  Château de Meursault Puligny-Montrachet 1er Cr...       16.67
15  Coteaux Champenois Egly-Ouriet Ambonnay Rouge ...       16.67
16  Champagne Larmandier-Bernier Grand Cru Les Che...       16.67
17                    Champagne Gosset Grande Réserve       14.29
18            Champagne Mailly Grand Cru Brut Réserve       14.29
19  Champagne Mailly Grand Cru Extra Brut Millésim...       12.50
```

In [155…

```python
#Graphique en barre du top 20 des produits qui ont le plus de mois de stock
fig = px.bar(
    top20_mois_stock,
    x='mois_stock',
    y='post_title',
    orientation='h',
    title='Top 20 des articles avec le plus de mois de stock',
    labels={'mois_stock': 'Mois de stock', 'post_title': 'Article'},
    color_discrete_sequence=['orange']
)

# Inverser pour que le produit avec le plus de mois de stock soit en haut
fig.update_layout(yaxis=dict(autorange='reversed'))

fig.show()
```



In [157…

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

#Création de la colonne Valorisation des stocks en euros
df_merge['valorisation_stock_euros'] = df_merge['stock_quantity'] * df_merge['price']
#Calculer la somme de la colonne "Valorisation_stock_euros"
total_valorisation_stock = df_merge['valorisation_stock_euros'].sum()
# Afficher la somme
print(f"Valorisation totale des stocks : {total_valorisation_stock:,.0f} €".replace(',', ' '))
```

```
Valorisation totale des stocks : 486 881 €
```

In [159…

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

#Calculer la somme de la colonne stock quantity
total_produits_en_stock = df_merge['stock_quantity'].sum()

# Affichage du résultat
print(f"Nombre total de produits en stock : {total_produits_en_stock:,}".replace(',', ' '))
```

```
Nombre total de produits en stock : 16 718.0
```

Etape 5.4 - Analyse du taux de marge

In [162…

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

# Création de la colonne prix HT
tva = 0.2
df_merge['prix_ht'] = round(df_merge['price'] / (1 + tva), 2)

# Formule CORRECTE du taux de marge (marge brute)
df_merge['taux_marge'] = round(
    (df_merge['prix_ht'] - df_merge['purchase_price']) / df_merge['purchase_price'] * 100,
    2
)

# Vérification des valeurs aberrantes
print("Produits vendus à perte:")
print(df_merge[df_merge['taux_marge'] < 0][['product_id', 'price', 'purchase_price', 'taux_marge']])

# Statistiques
print(f"\nTaux de marge minimum: {df_merge['taux_marge'].min():.2f}%")
print(f"Taux de marge moyen: {df_merge['taux_marge'].mean():.2f}%")
print(f"Taux de marge maximum: {df_merge['taux_marge'].max():.2f}%")
```

```
Produits vendus à perte:
     product_id  price  purchase_price  taux_marge
697      4355.0   12.0           77.48      -87.09

Taux de marge minimum: -87.09%
Taux de marge moyen: 57.14%
Taux de marge maximum: 91.41%
```

Il semble qu'il y a une erreur de saisie pour un produit ayant un taux de marge de -87.09 %.

In [165…
```
#affichage de la ligne avec un taux de marge inférieur à 0
produits_marge_neg = df_merge[df_merge['taux_marge'] < 0]
# Afficher ces lignes
produits_marge_neg
```

Out[165…

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	total_sales	post_author	post_date	...	ca_par_article	part_(
697	4355.0	1.0	12.0	97.0	instock	77.48	12589	0.0	2.0	2018-03-02 10:46:10	...	0.0	

1 rows × 30 columns

Remarque

L'erreur de saisie concerne le "product_id" 4355 avec un prix de 12.65 pour un prix d'achat de 77.48, ce qui est incohérent. Il faudra corriger le prix de vente pour résoudre ce problème.

In [168…
```
#affichage de la ligne avec un taux de marge superieur à 90
produits_marge_high = df_merge[df_merge['taux_marge'] > 90 ]
produits_marge_high
```

Out[168…

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	total_sales	post_author	post_date	...	ca_par_article	part_(
679	5916.0	1.0	93.0	1.0	instock	40.49	14774	1.0	2.0	2019-04-04 16:39:24	...	93.0	0.(

1 rows × 30 columns

Remarque

Concernant le taux maximum de 91%, il correspond à un produit acheté environ 40€ et revendu 93€, générant ainsi une marge importante

In [171…
```
#création d'un dataframe avec les taux positifs
df_marge_positive = df_merge[df_merge['taux_marge'] > 0]

#Afficher le prix minimum de la colonne "taux_marge"
min_marge_positive = df_marge_positive['taux_marge'].min()
print(f"Taux de marge minimum positif : {min_marge_positive:.2f} %")

#Afficher le prix maximum de la colonne "taux_marge"
max_marge_positive = df_marge_positive['taux_marge'].max()
print(f"Taux de marge maximum positif : {max_marge_positive:.2f} %")
```

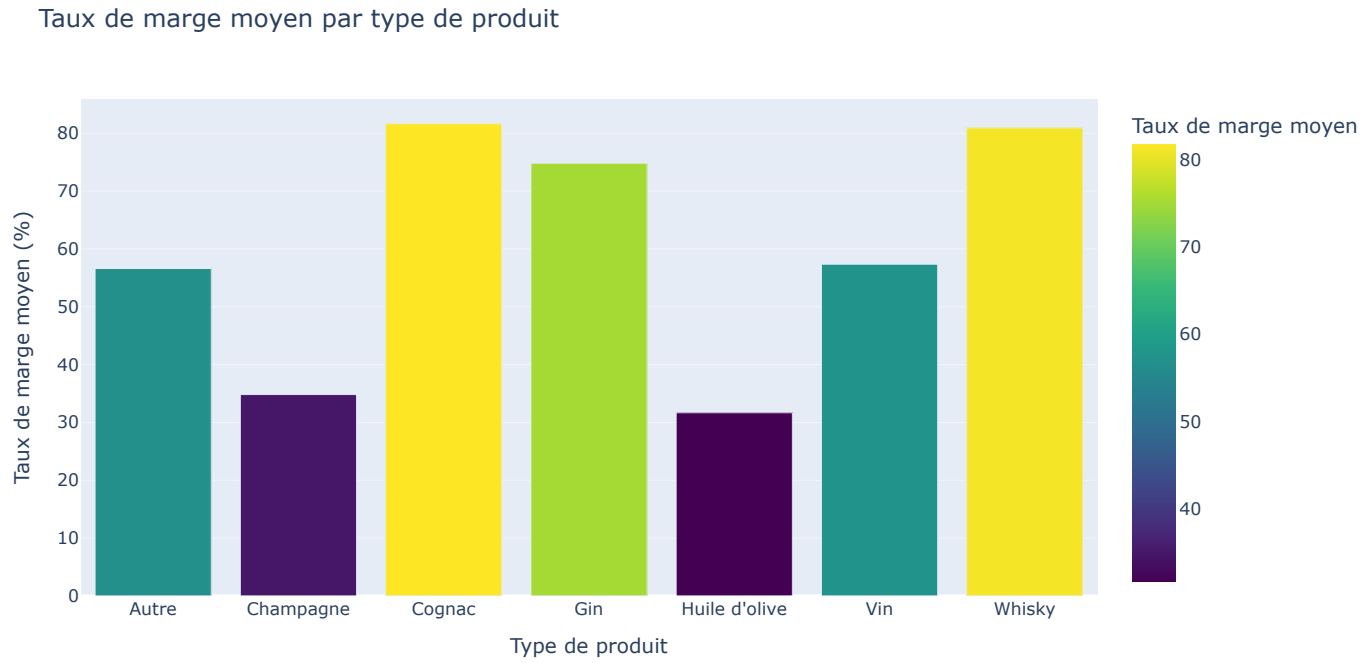
```
Taux de marge minimum positif : 28.47 %
Taux de marge maximum positif : 91.41 %
```

In [173…
```
#création d'un dataframe avec le taux de marge moyen par type de produit
marge_moyenne_par_type = df_merge.groupby('product_type')['taux_marge'].mean().reset_index()

# Afficher le DataFrame pour vérification
print(marge_moyenne_par_type)
```

```
   product_type  taux_marge
0         Autre   56.620000
1     Champagne   34.815357
2        Cognac   81.656250
3           Gin   74.830000
4  Huile d'olive   31.673333
5           Vin   57.354718
6        Whisky   80.946429
```

In [175…
```
# Graphique en barre du taux de marge moyen par type de produit
fig = px.bar(
    marge_moyenne_par_type,
    x='product_type',
    y='taux_marge',
    title='Taux de marge moyen par type de produit',
    labels={'product_type': 'Type de produit', 'taux_marge': 'Taux de marge moyen (%)'},
```

```
    color='taux_marge',
    color_continuous_scale='Viridis')

fig.show()
```



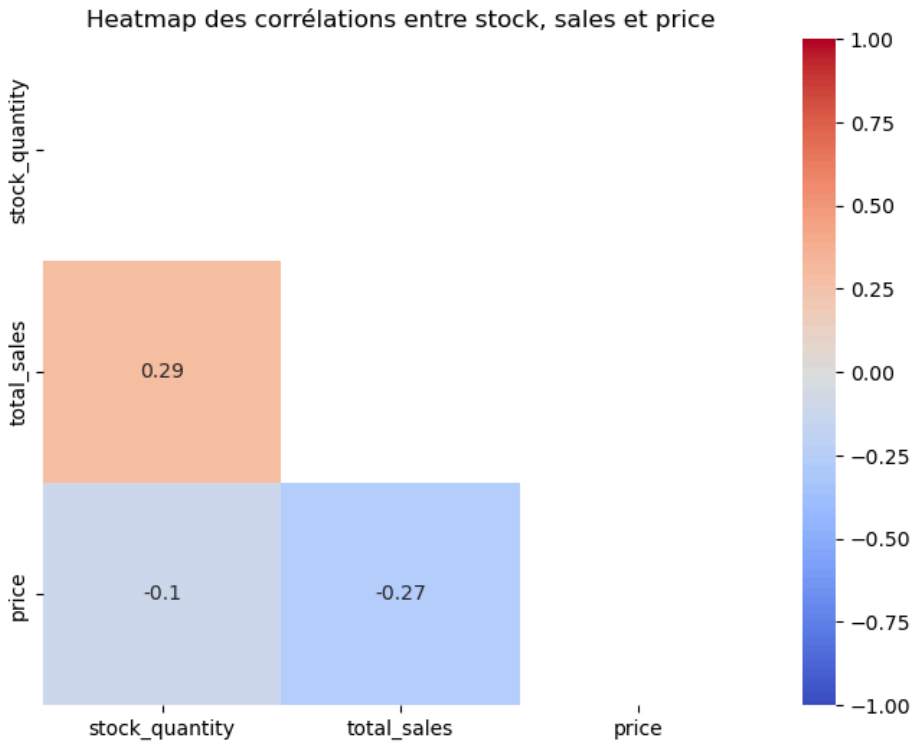
Le Cognac, le Whisky et le Gin ont les marges moyennes les plus élevées, ce qui les rend très rentables.

Ce graphique en barres illustre le taux de marge moyen (%) par type de produit. On observe que le Cognac et le Whisky ont les marges les plus élevées (autour de 80%), suivis du Gin et du vin. Les produits comme le champagne et l'Huile d'olive affichent des marges plus modestes (environ 30%).

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

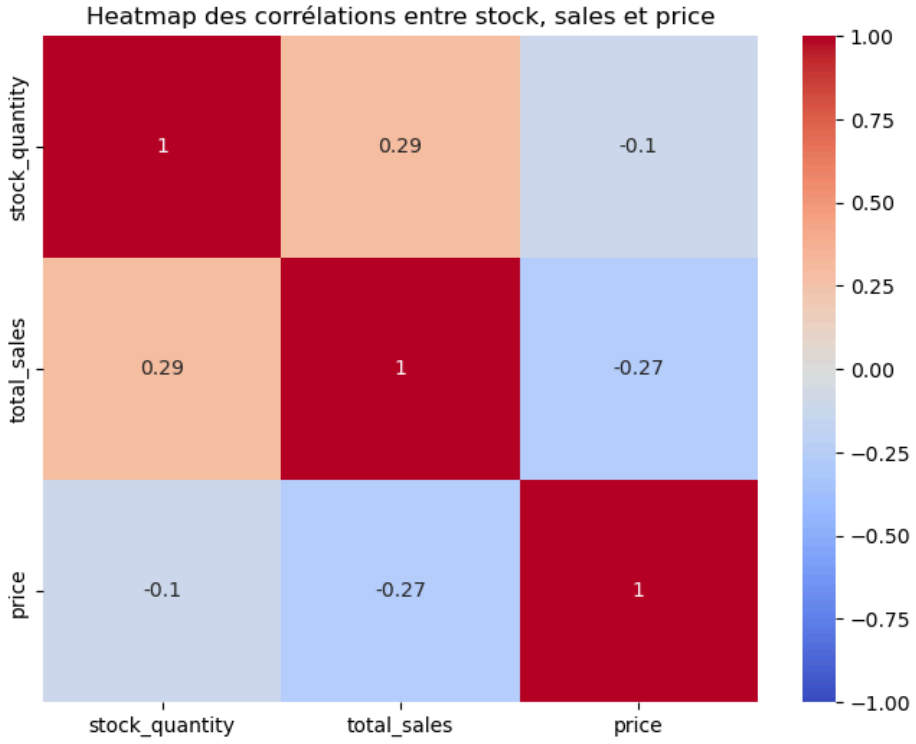
In [179…

```
############################
#Analyse des correlations #
############################

#Importation de Seaborn
import seaborn as sns

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

#Calcul de la matrice de corrélation sur df_merge
corr = df_merge[['stock_quantity', 'total_sales', 'price']].corr()

#Création d'un mask pour ne montrer que la moitié supérieure
mask = np.triu(np.ones_like(corr, dtype=bool))

#Création de la heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr, mask=mask, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Heatmap des corrélations entre stock, sales et price')
plt.show()
```



In [180…

```
#Calcul de la matrice de corrélation sur df_merge
corr = df_merge[['stock_quantity', 'total_sales', 'price']].corr()

#Création de la heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Heatmap des corrélations entre stock, sales et price')
plt.show()
```



Que peut-on conclure des corrélations ?

Les corrélations montrent que le prix a un effet négatif modéré sur les ventes, plus le prix augmente moins les ventes sont élevées. La quantité en stock a une faible influence positive sur les ventes. Le lien entre prix et stock est très faible et peu significatif. Les ventes semblent donc sensibles au prix.

Etape 5.6 - Mettre à disposition la nouvelle table sur un fichier Excel

In [185…

```
#Mettre le dataset df_merge sur un fichier Excel
df_merge.to_excel("df_merge.xlsx", index=False)
```

```
Requirement already satisfied: pandas in c:\users\pc\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: openpyxl in c:\users\pc\anaconda3\lib\site-packages (3.1.5)
Requirement already satisfied: numpy>=1.26.0 in c:\users\pc\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pc\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\pc\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\pc\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: et-xmlfile in c:\users\pc\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Requirement already satisfied: six>=1.5 in c:\users\pc\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```