

ANALYSE DU STOCK ET DES VENTES DU SITE BOTTLENECK

Double-cliquez (ou appuyez sur Entrée) pour modifier

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

```
import pandas as pd
import openpyxl
import numpy as np
```

```
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
```

```
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
```

1.2 - Chargements des fichiers

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

```
drive.mount( /content/drive )
```

```
Mounted at /content/drive
```

```
path="drive/MyDrive/Projet6/"
```

```
df_erp= pd.read_excel(path+'erp.xlsx')
```

```
df_web=pd.read_excel(path+'web.xlsx')
```

```
df_liaison=pd.read_excel(path+'liaison.xlsx')
```

```
/usr/local/lib/python3.11/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: Unknown extension is not supported
warn(msg)
/usr/local/lib/python3.11/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: Unknown extension is not supported
warn(msg)
/usr/local/lib/python3.11/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: Unknown extension is not supported
warn(msg)
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier erp.xlsx

```
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)
```

```
df_erp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
<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
```

df_erp.head()

	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

Étapes suivantes :

[Générer du code avec df_erp](#)[Afficher les graphiques recommandés](#)[New interactive sheet](#)

```
nombre_doublons_product_id = df_erp['product_id'].duplicated().sum()
print(f"Le nombre de doublon pour la colonne 'product_id' est de : {nombre_doublons_product_id}")
```

Le nombre de doublon pour la colonne 'product_id' est de : 0

```
valeurs_stock_status = df_erp['stock_status'].unique() # Par exemple
```

```
print(f"La colonne stock_status peut prendre comme valeur: {'', ' '.join(str(i) for i in valeurs_

print("La colonne 'stock_status' est relié à la colonne 'stock_quantity'.\nLorsque 'stock_quar

La colonne stock_status peut prendre comme valeur: instock, outofstock
La colonne 'stock_status' est relié à la colonne 'stock_quantity'.
Lorsque 'stock_quantity' est supérieur à 0, le produit est instock sinon il est outofstock.
```

```
df_erp["stock_status_2"] = df_erp["stock_quantity"].apply(lambda x: "outofstock" if x <= 0 else
```

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

	0
0	True
1	True
2	True
3	True
4	False
...	...
820	True
821	True
822	True
823	True
824	True

```
df_erp
```

```
825 rows × 1 columns
```

```
dtype: bool
```

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

```
print(f"Nous trouvons {nombre_valeurs_différentes} valeurs identiques entre les 2 colonnes stock_status et stock_status_2.")
```

```
Nous trouvons 823 valeurs identiques entre les 2 colonnes stock_status et stock_status_2.
Par conséquent 2 lignes sont différentes.
```

```
lignes_différentes_corrigees = df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
print("Valeur stock_status & stock_status_2 différentes :")
print(lignes_différentes_corrigees)
```

```
Valeur stock_status & stock_status_2 différentes :
  product_id  onsale_web  price  stock_quantity  stock_status \
4          4039         1   46.0              3  outofstock
398         4885         1   18.7              0    instock

  purchase_price  stock_status_2
4           23.77    instock
398           9.66  outofstock
```

```
df_erp.loc[df_erp["stock_quantity"] <= 0, "stock_status"] = "outofstock"
```

```
df_erp.loc[df_erp["stock_quantity"] > 0, "stock_status"] = "instock"
```

```
lignes_différentes_corrigees = df_erp[df_erp["stock_status"] != df_erp["stock_status_2"]]
print("Lignes avec des incohérences après correction :")
```

```
print( Lignes avec des incohérences après correction : )
print(lignes_différentes_corrigees)

Lignes avec des incohérences après correction :
Empty DataFrame
Columns: [product_id, onsale_web, price, stock_quantity, stock_status, purchase_price, stock_status_2]
Index: []
```

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

2.1.1.1 - Analyse de la variable PRIX

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

prix_non_renseigne = df_erp["price"].isna().sum()
print("Nombre d'articles avec un prix non renseigné: {}".format(prix_non_renseigne))#Saisir 1

prix_min=df_erp["price"].min()
print(f"prix maximum article: {prix_min}")

prix_max=df_erp['price'].max()
print(f"prix maximum article: {prix_max}")
prix_inferieur_zero = df_erp[df_erp["price"] < 0]
print("Articles avec des prix inférieurs à 0 :")
print(prix_inferieur_zero)

df_erp=df_erp[df_erp["price"]>0]
prix_inferieur_zero_corrige = df_erp[df_erp["price"] < 0]
```

```
print("Articles avec des prix inférieurs à 0 après correction :")
print(prix_inferieur_zero_corrige)
```

Nombre d'articles avec un prix non renseigné: 0

prix maximum article: -20.0

prix maximum article: 225.0

Articles avec des prix inférieurs à 0 :

	product_id	onsale_web	price	stock_quantity	stock_status	\
151	4233	0	-20.0	0	outofstock	
469	5017	0	-8.0	0	outofstock	
739	6594	0	-9.1	19	instock	

	purchase_price	stock_status_2
151	10.33	outofstock
469	4.34	outofstock
739	4.61	instock

Articles avec des prix inférieurs à 0 après correction :

Empty DataFrame

Columns: [product_id, onsale_web, price, stock_quantity, stock_status, purchase_price, stock_status_2]

Index: []

2.1.1.2 - Analyse de la variable STOCK

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

```
print(f"Quantité minimum de la colonne 'stock_quantity' : {df_erp['stock_quantity'].min()}")
```

```
print(f"Quantité maximum de la colonne 'stock_quantity' : {df_erp['stock_quantity'].max()}")
```




```
stock_inferieur_a_zero=df_erp[df_erp["stock_quantity"]<0]
```

```
stock_inferieur_a_zero=df_erp[df_erp['stock_quantity']<0]
print("Stock inférieur à zéro :")
stock_inferieur_a_zero
```

Quantité minimum de la colonne 'stock_quantity' : -10

Quantité maximum de la colonne 'stock_quantity' : 145

Stock inférieur à zéro :

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	stock_status_2	
449	4973	0	10.0	-10	outofstock	4.96	outofstock	
573	5700	1	44.5	-1	outofstock	22.30	outofstock	

Étapes
suivantes :

[Générer du code avec stock_inferieur_a_zero](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

```
df_erp.loc[df_erp['stock_quantity'] < 0, 'stock_quantity'] = 0
```

```
stock_inferieur_a_zero=df_erp[df_erp["stock_quantity"]<0]
print("Stock inférieur à zéro :")
stock_inferieur_a_zero
```

Stock inférieur à zéro :

product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	stock_status_2	
------------	------------	-------	----------------	--------------	----------------	----------------	---



2.1.1.3 - Analyse de la variable ONSALE_WEB

```
valeurs_onsale_web = df_erp['onsale_web'].unique()
print("Valeurs distinctes dans la colonne 'onsale_web' :", valeurs_onsale_web)
```



```
print(" La valeur '1' signifie que le produit n'est pas en vente sur le web et '0' qu'il l'est
```

```
Valeurs distinctes dans la colonne 'onsale_web' : [1 0]
```

```
La valeur '1' signifie que le produit n'est pas en vente sur le web et '0' qu'il l'est
```

```
print("Les colonnes à conserver sont : \n-product_id,\n-onsale_web,\n-price,\n-stock_quantity'
```

```
Les colonnes à conserver sont :
```

```
-product_id,  
-onsale_web,  
-price,  
-stock_quantity  
-purchase_price
```

```
#  
df_erp.drop(columns=['stock_status_2'], inplace=True)  
df_erp.head()
```

```
<ipython-input-32-1579933063>:2: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
df_erp.drop(columns=['stock_status_2'], inplace=True)

	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
---	------	---	------	---	---------	-------

Étapes suivantes :

[Générer du code avec df_erp](#)[Afficher les graphiques recommandés](#)[New interactive sheet](#)

2.1.1.4 - Analyse de la variable prix d'achat

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

```
nb_purchase_price_non_renseignes = df_erp['purchase_price'].isna().sum()
print(f"Nombre d'articles avec un prix non renseigné : {nb_purchase_price_non_renseignes}")
```

```
purchase_price_min = df_erp['purchase_price'].min(skipna=True)
print(f"Prix minimal de la colonne 'purchase_price' : {purchase_price_min}")
```

```
purchase_price_max = df_erp['purchase_price'].max(skipna=True)
print(f"Prix maximal de la colonne 'purchase_price' : {purchase_price_max}")
```

```
Nombre d'articles avec un prix non renseigné : 0
Prix minimal de la colonne 'purchase_price' : 2.74
Prix maximal de la colonne 'purchase_price' : 137.81
```

2.2 - Analyse exploratoire du fichier web.xlsx

```
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]))
```

```
print( Le tableau comporte {} colonne(s) .format(df_web.shape[1]))
```

```
Le tableau comporte 1513 observation(s) ou article(s)
```

```
Le tableau comporte 29 colonne(s)
```

```
df_web.info()
```

```
df_web.head()
```

```
<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
```

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	tax_class	post_author	post_
0	11862	0	0	0	0.0	3.0	NaN	NaN	2.0	2018-13:
1	16057	0	0	0	0.0	5.0	NaN	NaN	2.0	2018-15:
2	14692	0	0	0	0.0	5.0	taxable	NaN	2.0	2019-10:
3	16295	0	0	0	0.0	14.0	NaN	NaN	2.0	2018-14:
4	15328	0	0	0	0.0	2.0	taxable	NaN	2.0	2019-18:

#Selon vous. auelles sont les colonnes à conserver ?

```

colonnes_a_conserver = ["sku", "total_sales", "post_title", "product_type",
                        "post_date", "post_modified", "guid", "post_type"]
colonnes_a_supprimer = ["virtual", "downloadable", "tax_status", "rating_count", "tax_class", "average_rating", "post_author", "post_date", "post_date_gmt", "post_content", "post_excerpt", "post_status", "comment_status", "ping_status", "post_password", "post_name", "post_modified_gmt"]
print("Avec les objectifs de l'analyse que je dois construire, je dois conserver les colonnes")
for c in colonnes_a_conserver: print(f"-{c}")
print()
print("Les colonnes qu'il convient de supprimer car inutile pour l'analyse sont :")
for c in colonnes_a_supprimer: print(f"-{c}")

```

Avec les objectifs de l'analyse que je dois construire, je dois conserver les colonnes suivantes :



- sku
- total_sales
- post_title
- product_type
- post_date
- post_modified
- guid
- post_type

Les colonnes qu'il convient de supprimer car inutile pour l'analyse sont :

- virtual
- downloadable
- tax_status
- rating_count
- tax_class
- average_rating
- tax_class
- post_author
- post_date
- post_date_gmt
- post_content
- post_excerpt
- post_status
- comment_status
- ping_status
- post_password
- post_name
- post_modified
- post_modified_gmt

- post_modified_gmt
- post_content_filtered
- post_parent
- menu_order
- post_mime_type
- comment_count

```
df_web=df_web[colonnes_a_conserver]
df_web.head()
```

	sku	total_sales	post_title	product_type	post_date	post_modified	guid	post_type	
0	11862	3.0	Gilles Robin Hermitage Rouge 2012	Vin	2018-02-12 13:46:23	2019-01-31 12:12:56	https://www.bottle- neck.fr/wp-content/ uploads/...	attachment	
1	16057	5.0	Domaine Pellé Sancerre Rouge La Croix Au Garde... Château	Vin	2018-04-17 15:29:17	2020-07-07 10:05:02	https://www.bottle- neck.fr/wp-content/ uploads/...	attachment	




Étapes suivantes :

Générer du code avec df_web

 Afficher les graphiques recommandés

New interactive sheet

```
df_anomalies_non_decimal = df_web[~df_web["sku"].astype(str).str.match(r"^\d+$")]
df_anomalies_non_decimal
```

	sku	total_sales	post_title	product_type	post_date	post_modified	guid	post_type	
8	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN	
20	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN	
30	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN	

37	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN
41	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN
...
1387	bon- cadeau-25- euros	7.0	Bon cadeau de 25€	NaN	2018-06-01 13:53:46	2018-06-01 14:13:57	https://www.bottle- neck.fr/? post_type=product&...	product
1429	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN
1432	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN
1445	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN
1457	NaN	NaN	NaN	NaN	NaT	NaT	NaN	NaN

Étapes
suivantes :

[Générer du code avec df_anomalies_non_decimal](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

```
valeurs_anormales = df_anomalies_non_decimal["sku"].unique()
print("Liste des erreurs de la colonne sku :")
for e in valeurs_anormales:print(f"-{e}")
```

```
Liste des erreurs de la colonne sku :
-nan
-13127-1
-bon-cadeau-25-euros
```

Les erreurs de codifications je les laisse tel quel car peut être qu'il y a une correspondance dans le fichier erp

```
nombre_nan_sku = df_web["sku"].isna().sum()



print(f"Nombre de lignes avec un SKU manquant : {nombre_nan_sku}")
```

Nombre de lignes avec un SKU manquant : 85

```
df_web = df_web[~df_web["sku"].isin(valeurs_anormales[0:1])]
df_web = df_web.reset_index(drop=True)

nombre_nan_sku = df_web["sku"].isna().sum()
print(f"Nombre de lignes avec un SKU manquant : {nombre_nan_sku}")
df_web.head(10)
```

Nombre de lignes avec un SKU manquant : 0

	sku	total_sales	post_title	product_type	post_date	post_modified	guid	post_type	
0	11862	3.0	Gilles Robin Hermitage Rouge 2012	Vin	2018-02-12 13:46:23	2019-01-31 12:12:56	https://www.bottle- neck.fr/wp-content/ uploads/...	attachment	 
1	16057	5.0	Domaine Pellé Sancerre Rouge La Croix Au Garde...	Vin	2018-04-17 15:29:17	2020-07-07 10:05:02	https://www.bottle- neck.fr/wp-content/ uploads/...	attachment	
2	14692	5.0	Château Fonréaud Bordeaux Blanc Le Cygne 2016	Vin	2019-03-19 10:06:47	2020-04-25 21:40:31	https://www.bottle- neck.fr/? post_type=product&...	product	
3	16295	14.0	Moulin de Gassac IGP Pays d'Hérault Guilhem Ro...	Vin	2018-02-15 14:05:06	2020-08-27 18:55:03	https://www.bottle- neck.fr/wp-content/ uploads/...	attachment	

Étapes suivantes :

Générer du code avec df_web

 Afficher les graphiques recommandés

New interactive sheet


```
sku_counts = df_web["sku"].value_counts()
```

```
print(f"Nombre de SKU modifié :{sku_counts[sku_counts > 1].sum()}")
```

```
Nombre de SKU modifié :1428
```

```
df_sans_code_article = df_web[df_web["sku"].isna() | (df_web["sku"] == "")]
```

```
df_sans_code_article.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   sku              0 non-null      object
1   total_sales      0 non-null      float64
2   post_title       0 non-null      object
3   product_type     0 non-null      object
4   post_date        0 non-null      datetime64[ns]
5   post_modified    0 non-null      datetime64[ns]
6   guid             0 non-null      object
7   post_type        0 non-null      object
dtypes: datetime64[ns](2), float64(1), object(5)
memory usage: 0.0+ bytes
```

```
print("Il convient de supprimer les lignes où la colonne 'post_type'=='attachement' et garder
```

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



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

```
sku_counts = df_web["sku"].value_counts()
```

```
print(f"Nombre de doublons sku : {sku_counts[sku_counts > 1].sum()}")
```

```
df_web.head(10)
```

Il convient de supprimer les lignes où la colonne 'post_type'=='attachement' et garder celle où elles sont égales à
Nombre de doublons sku : 0

	sku	total_sales	post_title	product_type	post_date	post_modified	guid	post_type	
0	14692	5.0	Château Fonréaud Bordeaux Blanc Le Cygne 2016	Vin	2019-03-19 10:06:47	2020-04-25 21:40:31	https://www.bottle-neck.fr/?post_type=product&...	product	
1	15328	2.0	Agnès Levet Côte Rôtie Maestria 2017	Vin	2019-03-27 18:05:09	2020-07-25 15:45:02	https://www.bottle-neck.fr/?post_type=product&...	product	
2	16515	10.0	Château Turcaud Bordeaux Rouge Cuvée Majeure 2018	Vin	2018-06-02 09:31:31	2020-08-27 10:11:12	https://www.bottle-neck.fr/?post_type=product&...	product	
3	16585	15.0	Xavier Frissant Touraine Sauvignon 2019	Vin	2018-02-16 14:03:16	2020-08-27 09:30:36	https://www.bottle-neck.fr/?post_type=product&...	product	

Étapes suivantes :

[Générer du code avec df_web](#)
[Afficher les graphiques recommandés](#)
[New interactive sheet](#)

2.3 - Analyse exploratoire du fichier liaison.xlsx

```
print("Le tableau comporte {} observation(s) ou article(s)".format(df_liaison.shape[0]))
```

```
print("Le tableau comporte {} colonne(s)".format(df_liaison.shape[1]))
```

```
Le tableau comporte 825 observation(s) ou article(s)  
Le tableau comporte 2 colonne(s)
```

```
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
```

```
nb_total_lignes = df_liaison.shape[0]  
nb_uniques_product_id = df_liaison["product_id"].nunique()  
  
if nb_total_lignes == nb_uniques_product_id:  
    print("Toutes les valeurs de la colonne 'product_id' sont uniques.")  
else:  
    print(f"Il y a des doublons. {nb_total_lignes - nb_uniques_product_id} valeurs sont dupliées")  
  
    Toutes les valeurs de la colonne 'product_id' sont uniques.  
  
nb_uniques_id_web=df_liaison["id_web"].nunique()  
  
if nb_total_lignes == nb_uniques_id_web :  
    print("Toutes les valeurs de la colonne 'id_web' sont uniques.")  
else:
```

```
else:
```

```
    print(f"Il y a des doublons. {nb_total_lignes - nb_uniques_id_web} valeurs sont dupliquée:
```

```
        Il y a des doublons. 91 valeurs sont dupliquées.
```

```
articles_sans_correspondance=df_liaison["id_web"].isna().sum()
```

```
if articles_sans_correspondance==0:
```

```
    print("Tout les articles ont une correspondance")
```

```
else:
```

```
    print(f"Il manque des correspondances pour des articles : {articles_sans_correspondance}.\n"
```

```
        Il manque des correspondances pour des articles : 91.
```



```
        Par conséquent les doublons sont seulement des articles sans correspondance et non des valeurs duppliqués.
```

Etape 3 - Jonction des fichiers

Etape 3.1 - Jonction du fichier df_erp et df_liaison

```
df_erp_liaison=df_erp.merge(df_liaison,left_on="product_id",right_on="product_id",how="outer")
```

```
df_erp_liaison.head()
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	
0	3847	1.0	24.2	16.0	instock	12.88	15298	
1	3849	1.0	34.3	10.0	instock	17.54	15296	
2	3850	1.0	20.8	0.0	outofstock	10.64	15300	
3	4032	1.0	14.1	26.0	instock	6.92	19814	

4

4039

1.0

46.0

3.0

instock

23.77

19815

Étapes suivantes :

Générer du code avec df_erp_liaison

 Afficher les graphiques recommandés

New interactive sheet

```
non_match_erp = df_erp[~df_erp["product_id"].isin(df_liaison["product_id"])]

if non_match_erp.empty:
    print("Toutes les données ont une correspondance entre elles.")
else:
    print("Certains product_id ne correspondent pas.")
```

Toutes les données ont une correspondance entre elles.

Etape 3.2 - Jonction du fichier df_merge et df_web

```
df_merge = df_erp_liaison.merge(df_web, how='outer', left_on="id_web", right_on="sku",indicator=True)
df_merge[df_merge["_merge"] != "both"]
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	sku	total_sales	post_title
81	4741	0.0	12.4	0.0	outofstock	6.66	12601	NaN	NaN	Ne pas acheter
127	5957	0.0	39.0	0.0	outofstock	20.75	13577	NaN	NaN	Ne pas acheter
139	4289	0.0	22.8	0.0	outofstock	11.90	13771	NaN	NaN	Ne pas acheter
180	4869	0.0	17.2	0.0	outofstock	9.33	14360	NaN	NaN	Ne pas acheter
185	5955	0.0	27.3	0.0	outofstock	13.68	14377	NaN	NaN	Ne pas acheter
...

820	7196	0.0	31.0	55.0	instock	31.20	NaN	NaN	NaN	NaN
821	7200	0.0	31.0	6.0	instock	15.54	NaN	NaN	NaN	NaN
822	7201	0.0	31.0	18.0	instock	16.02	NaN	NaN	NaN	NaN
823	7203	0.0	45.0	30.0	instock	23.48	NaN	NaN	NaN	NaN
824	7204	0.0	45.0	9.0	instock	24.18	NaN	NaN	NaN	NaN

111 rows × 16 columns

```
lignes_sans_correspondance = df_merge[df_merge["id_web"].isna() | df_merge["sku"].isna()]
nb_lignes_sans_correspondance=len(lignes_sans_correspondance)
total_lignes=len(df_merge)

print(f"Nombre de lignes sans correspondance : {nb_lignes_sans_correspondance}")
print(f"Pourcentage du nombre de lignes sans correspondance : {100*(nb_lignes_sans_correspondance/total_lignes)}%")

lignes_sans_correspondance_erp = df_merge[df_merge["id_web"].isna()]
lignes_sans_correspondance_web=df_merge[df_merge["sku"].isna()]
print(f"{len(lignes_sans_correspondance_web)} lignes sans correspondance proviennent du df_web")
print(f"{len(lignes_sans_correspondance_erp)} lignes sans correspondance proviennent du df_erp")

Nombre de lignes sans correspondance : 111
Pourcentage du nombre de lignes sans correspondance : 13.45%
111 lignes sans correspondance proviennent du df_web. Cela représente 100.00% du total des lignes sans correspondance
91 lignes sans correspondance proviennent du df_erp. Cela représente 81.98% du total des lignes sans correspondance.
```

```
df_merge=df_merge.dropna(subset=["id_web", "sku"])

print("Lignes sans correspondances supprimées.")
print(f"il reste :{len(df_merge)} lignes.")

Lignes sans correspondances supprimées.
..
```

```
11 reste :714 lignes.
```

Etape 4 - Analyse univarié des prix

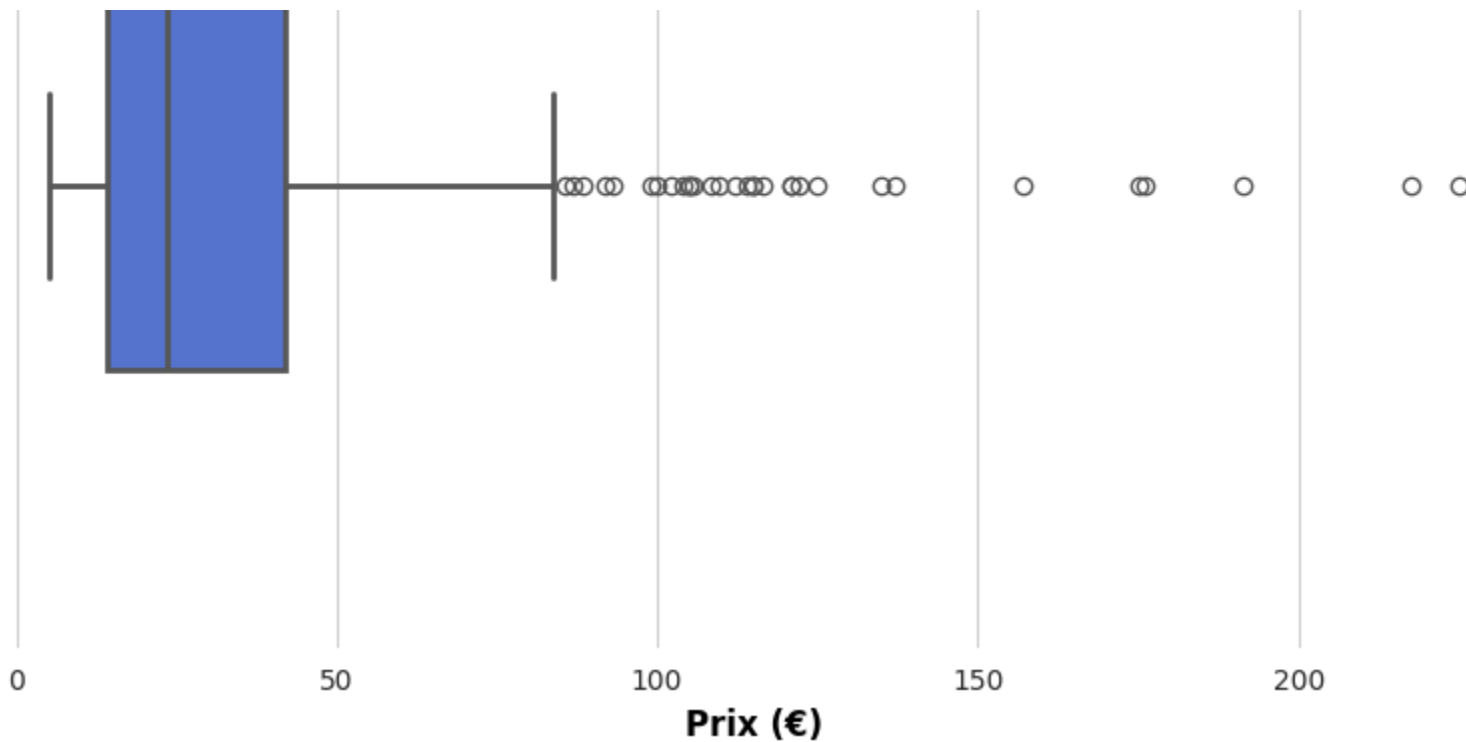
Etape 4.1 - Exploration par la visualisation de données

```
sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))

ax = sns.boxplot(
    x=df_merge["price"],
    color="royalblue",
    width=0.4,
    flierprops={"marker": "o", "color": "red", "markersize": 6},
    linewidth=2
)

plt.title("Répartition des prix", fontsize=14, fontweight="bold", color="darkblue")
plt.xlabel("Prix (€)", fontsize=12, fontweight="bold", color="black")
sns.despine(left=True, bottom=True)
plt.show()
```





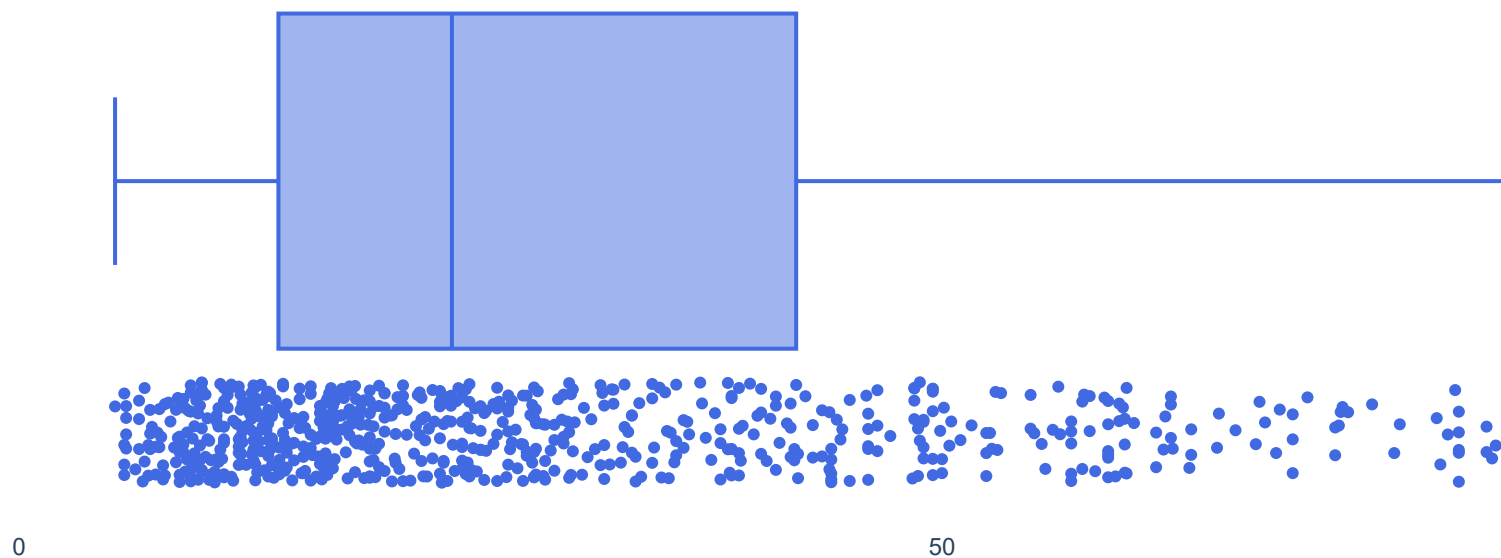
```
fig = px.box(  
    df_merge,  
    x="price",  
    points="all",  
    title="Répartition des prix",  
    color_discrete_sequence=["royalblue"]  
)
```

```
fig.update_layout(  
    title_font_size=16,  
    title_x=0.5,  
    xaxis_title="Prix (€)",  
    font=dict(family="Arial", size=12),  
    plot_bgcolor="white"
```



```
    plot_bgcolor = white ,  
    showlegend=False  
)
```

```
fig.show()
```



Etape 4 2 - Exploration par l'utilisation de méthodes statistique

Etape 4.2 - Exploration par l'application de méthodes statistiques

Etape 4.2.1 - Identification par le Z-index

```
moyenne_prix = df_merge["price"].mean()
print(f"Moyenne du prix : {moyenne_prix:.2f} €")

ecart_type_prix = df_merge["price"].std()
print(f"Écart-type du prix : {ecart_type_prix:.2f}")

df_merge["z_score_price"] = (df_merge["price"] - moyenne_prix) / ecart_type_prix
print(df_merge[["price", "z_score_price"]])
```

Moyenne du prix : 32.33 €

Écart-type du prix : 27.60

	price	z_score_price
0	8.6	-0.860030
1	41.0	0.314039
2	39.0	0.241565
3	59.9	0.998912
4	22.5	-0.356340
..
728	58.0	0.930063
729	58.0	0.930063
730	92.0	2.162110
731	54.8	0.814105
733	25.0	-0.265748

[714 rows x 2 columns]

```
seuil_prix_zscore_3 = df_merge[df_merge["z_score_price"] > 3]["price"]
print(f"Nombre de prix où zscore > 3 : {len(seuil_prix_zscore_3)} ")
print("Prix dont le Z-score est supérieur à 3 :")
print(seuil_prix_zscore_3)
```

```

fig = px.box(df_merge, x="price", title="Répartition des prix avec seuil Z-score > 3", points=
fig.add_vline(
    x=df_merge[df_merge["z_score_price"] > 3]["price"].min(),
    line=dict(color="red", dash="dash"),
    annotation_text="Seuil Z-score = 3",
    annotation_position="top right"
)

fig.update_layout(
    title_font_size=16,
    title_x=0.5,
    xaxis_title="Prix (€)",
    font=dict(family="Arial", size=12),
    plot_bgcolor="white",
)

fig.show()

```

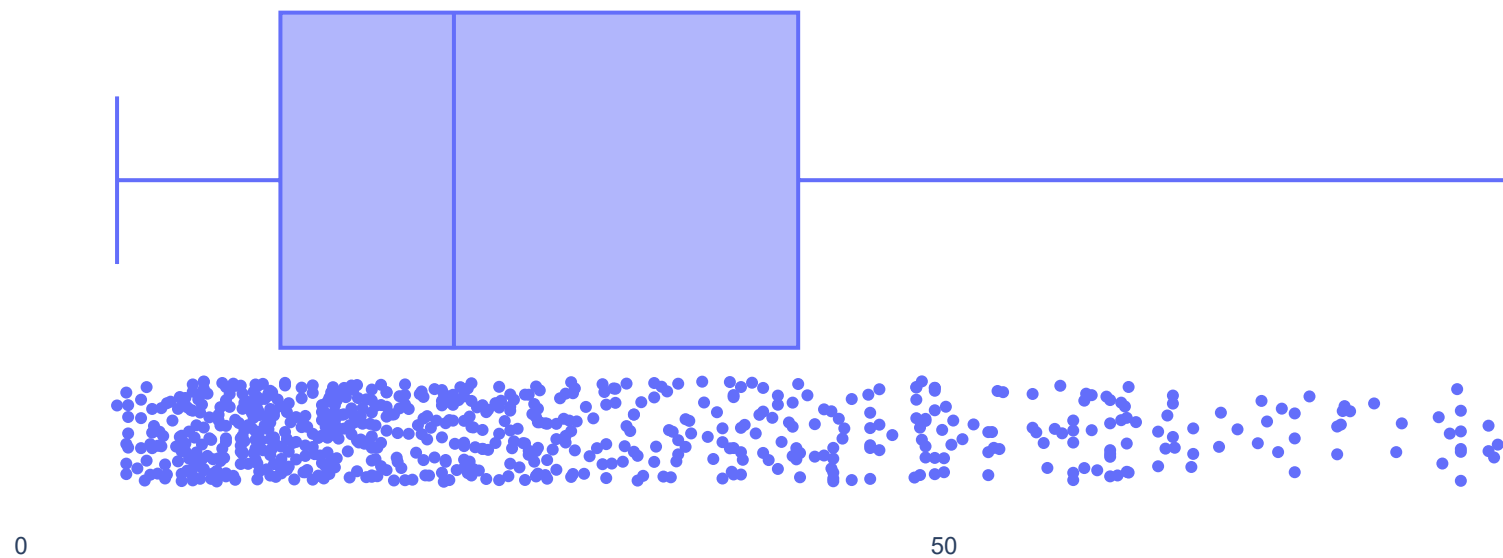
Nombre de prix où zscore > 3 : 13

Prix dont le Z-score est supérieur à 3 :

24	176.0
33	157.0
171	137.0
204	217.5
235	122.0
256	124.8
257	135.0
269	191.3
281	121.0
282	121.0
291	116.4
313	175.0
567	225.0

Nombre de prix dont le Z-score est supérieur à 3 : 13

```
name: price, dtype: float64
```



Etape 4.2.2 - Identification par l'interval interquartile

```
statistiques=df_erp["price"].describe()
print(statistiques)
Q1 = statistiques["25%"]
Q3 = statistiques["75%"]
```

```
qs = statistiques[ : ]
```

```
IQR = Q3 - Q1
```

```
count    822.000000
mean     32.350304
std      26.622453
min       5.200000
25%      14.600000
50%      24.400000
75%      42.000000
max      225.000000
Name: price, dtype: float64
```

```
borne_inferieure = Q1 - 1.5 * IQR
```

```
borne_superieure = Q3 + 1.5 * IQR
```

```
print(f"Borne inférieure : {borne_inferieure:.2f}")
```

```
print(f"Borne supérieure (seuil des outliers): {borne_superieure:.2f}")
```

```
Borne inférieure : -26.50
Borne supérieure (seuil des outliers): 83.10
```

```
outliers = df_erp[(df_erp["price"] < borne_inferieure) | (df_erp["price"] > borne_superieure)]
```

```
nombre_outliers = len(outliers)
```

```
nombre_total = len(df_erp)
```

```
proportion_outliers = (nombre_outliers / nombre_total) * 100
```

```
print(f"Nombre d'articles considérés comme outliers : {nombre_outliers}")
```

```
print(f"Proportion d'outliers dans l'ensemble du catalogue : {proportion_outliers:.2f}%")
```

Nombre d'articles considérés comme outliers : 36
Proportion d'outliers dans l'ensemble du catalogue : 4.38%

```
outliers = df_merge[df_merge["z_score_price"] > 3]
outliers
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	sku	total_sales	post
24	4402	1.0	176.0	11.0	instock	78.25	3510	3510	3.0	Cognac
33	4406	1.0	157.0	12.0	instock	69.08	7819	7819	4.0	Cognac Châ Fontpin 2
171	4904	1.0	137.0	9.0	instock	67.95	14220	14220	3.0	Doma Croix Charles Gran
204	5001	1.0	217.5	18.0	instock	116.87	14581	14581	2.0	David Cl Cha Gr
235	5917	1.0	122.0	12.0	instock	54.24	14775	14775	3.0	Wemy Sing Scotch C

```
print(f"Catégories des articles Z-Score > 3 :{df_merge[df_merge['z_score_price'] > 3]['produ']
print(f"Prix d'achat moyen des articles Z-Score > 3 : {round(df_merge[df_merge['z_score_price']
print("Etant donné la nature des produits vendus, certains peuvent être luxueux.Leur prix moye
```

```
Catégories des articles Z-Score > 3 :['Cognac', 'Vin', 'Whisky', 'Champagne'].
Prix d'achat moyen des articles Z-Score > 3 : 81.83 euros
Etant donné la nature des produits vendus, certains peuvent être luxueux. Leur prix moyen d'achat relativement élevé,
Par conséquent, dans ce contexte, il convient de les conserver.
```

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

```
dupliques = df_merge[df_merge.duplicated(keep=False)]
print("Lignes dupliquées :")
dupliques
```

Lignes dupliquées :

product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	sku	total_sales	post_title
------------	------------	-------	----------------	--------------	----------------	--------	-----	-------------	------------

Etape 5.1 - Analyse des ventes en CA

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

```
df_merge["ca_par_article"] = df_merge["total_sales"] * df_merge["price"]
ca_total = df_merge["ca_par_article"].sum()
print(f"Le chiffre d'affaires total du site web est : {ca_total:.2f} $")
```

Le chiffre d'affaires total du site web est : 143680.10 \$

```
df_groupe_produit = df_merge.groupby("product_id", as_index=False).agg({
    "post_title": "first",
    "total_sales": "sum" })

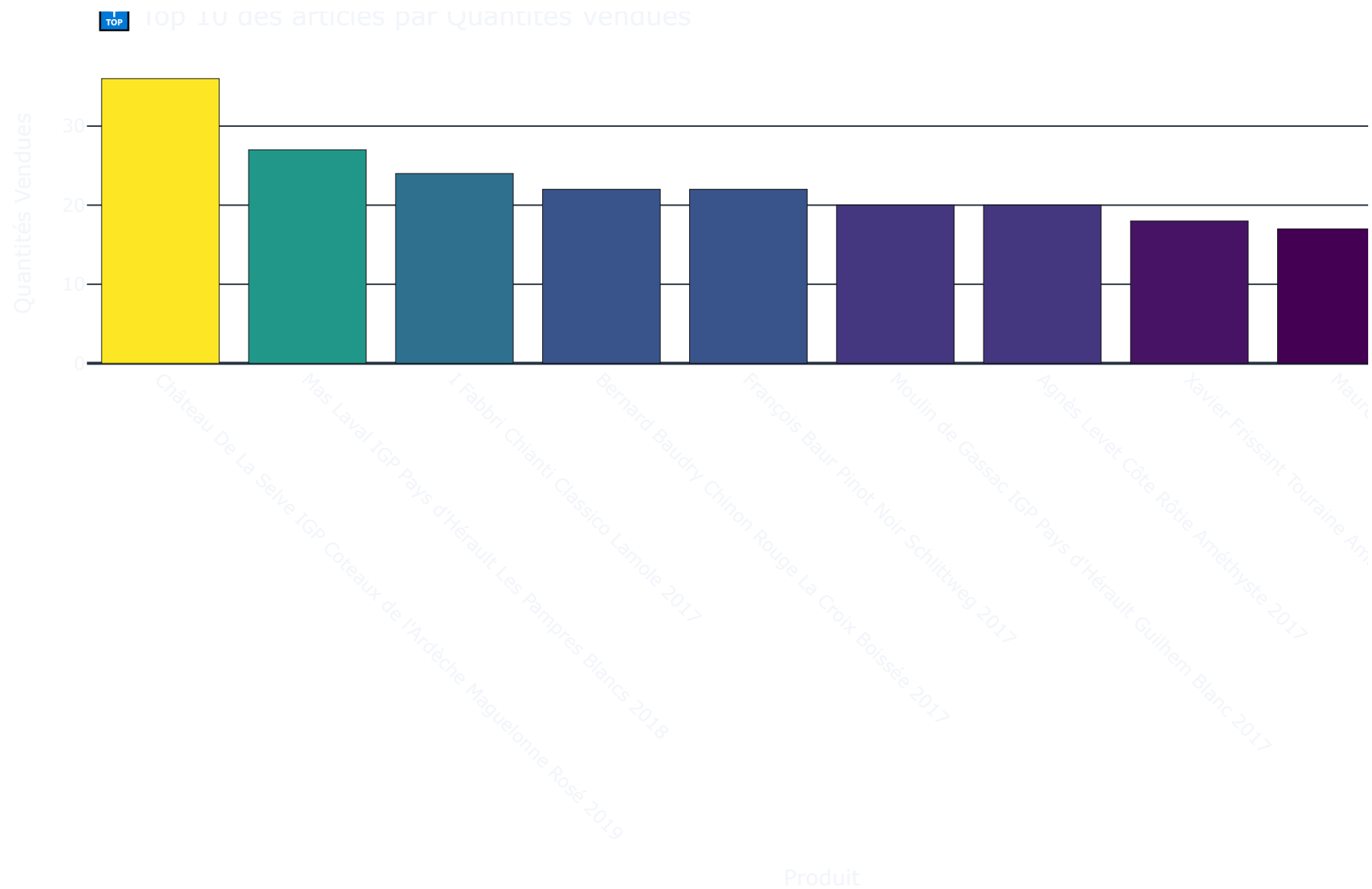
df_top10_quantites = df_groupe_produit.sort_values(by="total_sales", ascending=False).head(10)

fig = px.bar(
    df_top10_quantites,
    x="post_title",
    y="total_sales",
    labels={"post_title": "Produit", "total_sales": "Quantités Vendues"},
    title="📈 Top 10 des articles par Quantités Vendues",
    color="total_sales",
    color_continuous_scale="Viridis",
    template='plotly_dark'
)
fig.update_layout(
    width=1200,
    height=600,
    margin=dict(l=50, r=50, t=50, b=200),
)

fig.update_xaxes(tickangle=45, tickfont=dict(size=12))

fig.show()
```

📈 Top 10 des articles par Quantités Vendues



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

```
df_menge["part_CA"] = df_menge["ca_non_article"] / df_menge["ca_non_article"].sum()
```

```

df_merge["part_CA"] = df_merge["ca_par_article"] / df_merge["ca_par_article"].sum()

df_merge_sorted = df_merge.sort_values(by="ca_par_article", ascending=False, ignore_index=True)

df_merge_sorted["cumulative_CA"] = df_merge_sorted["part_CA"].cumsum()

nb_articles_80_CA = (df_merge_sorted["cumulative_CA"] <= 0.80).sum()

proportion_catalogue = nb_articles_80_CA / len(df_merge)

print(f"Nombre d'articles représentant 80% du CA : {nb_articles_80_CA}")
print(f"Proportion du catalogue concernée : {proportion_catalogue:.2%}")

plt.figure(figsize=(12, 6))
sns.barplot(x=df_merge_sorted["ca_par_article"][:20], y=df_merge_sorted["post_title"][:20], palette="magma")
plt.xlabel("CA par article")
plt.ylabel("Nom article")
plt.title("Top 20 articles par chiffre d'affaires")
plt.gca().invert_yaxis()
plt.show()

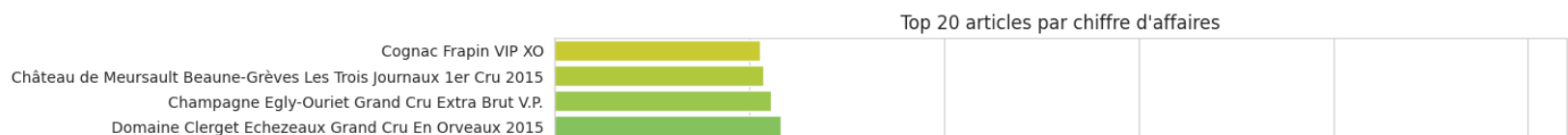
```

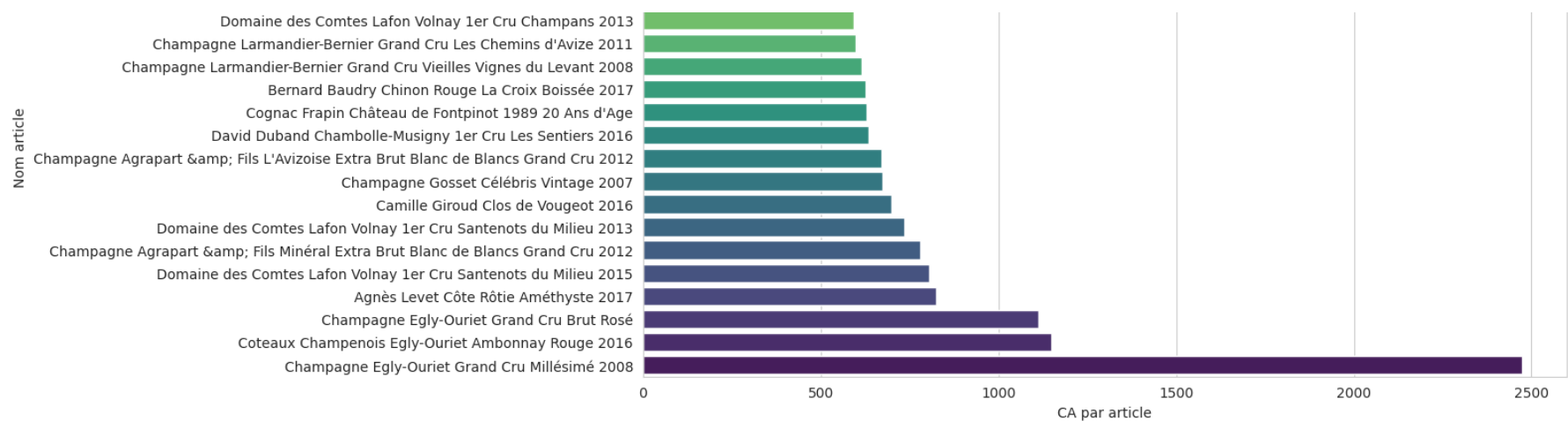
```

Nombre d'articles représentant 80% du CA : 434
Proportion du catalogue concernée : 60.78%
<ipython-input-67-1131223765>:19: FutureWarning:

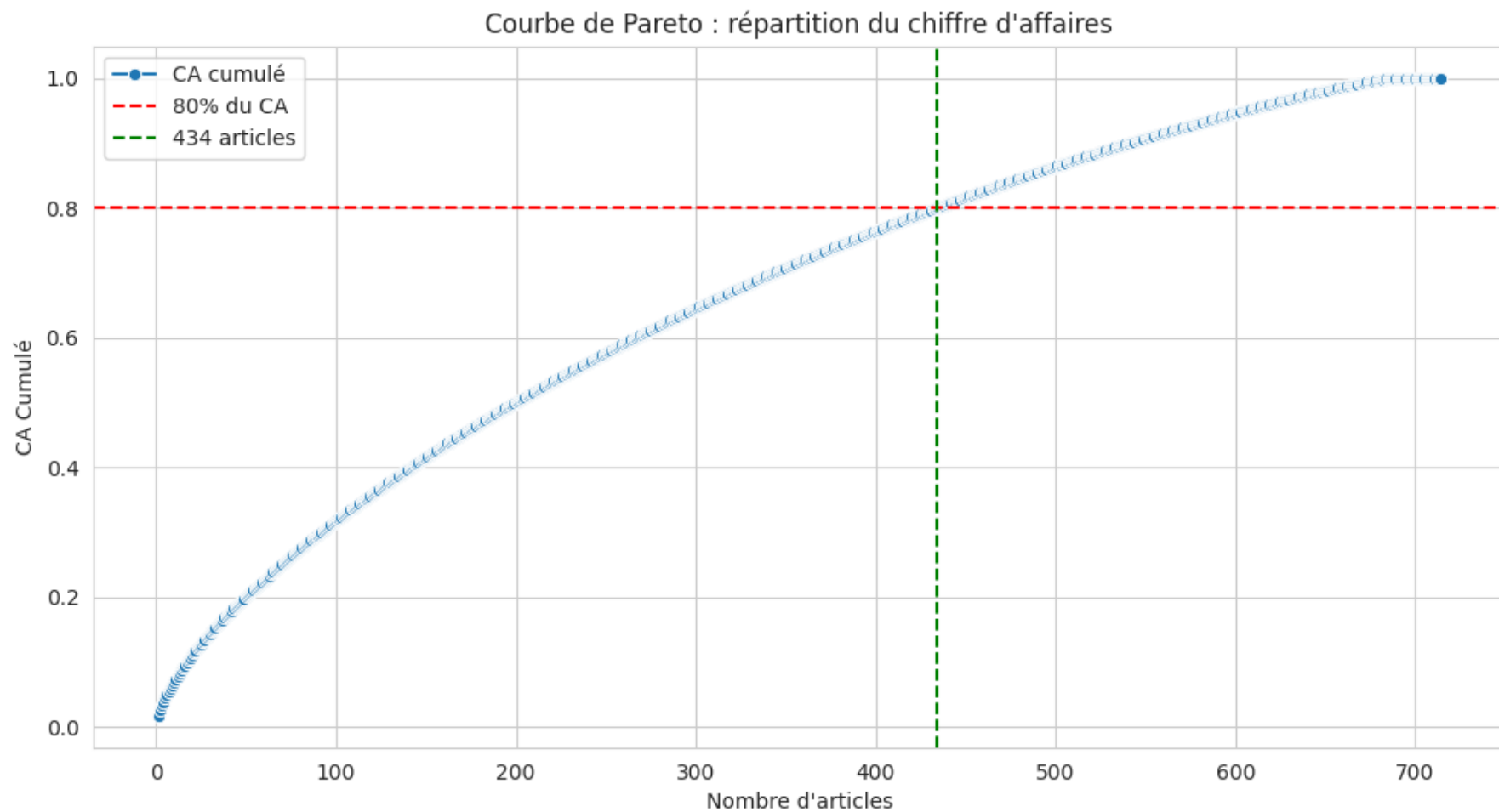
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` to preserve the current behavior.





```
plt.figure(figsize=(12, 6))
sns.lineplot(x=range(1, len(df_merge_sorted) + 1), y=df_merge_sorted["cumulative_CA"], marker=
plt.axhline(y=0.80, color="r", linestyle="--", label="80% du CA")
plt.axvline(x=nb_articles_80_CA, color="g", linestyle="--", label=f"{nb_articles_80_CA} artic:
plt.xlabel("Nombre d'articles")
plt.ylabel("CA Cumulé")
plt.title("Courbe de Pareto : répartition du chiffre d'affaires")
plt.legend()
plt.show()
```



Etape 5.2 - Analyse des ventes en Quantités

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

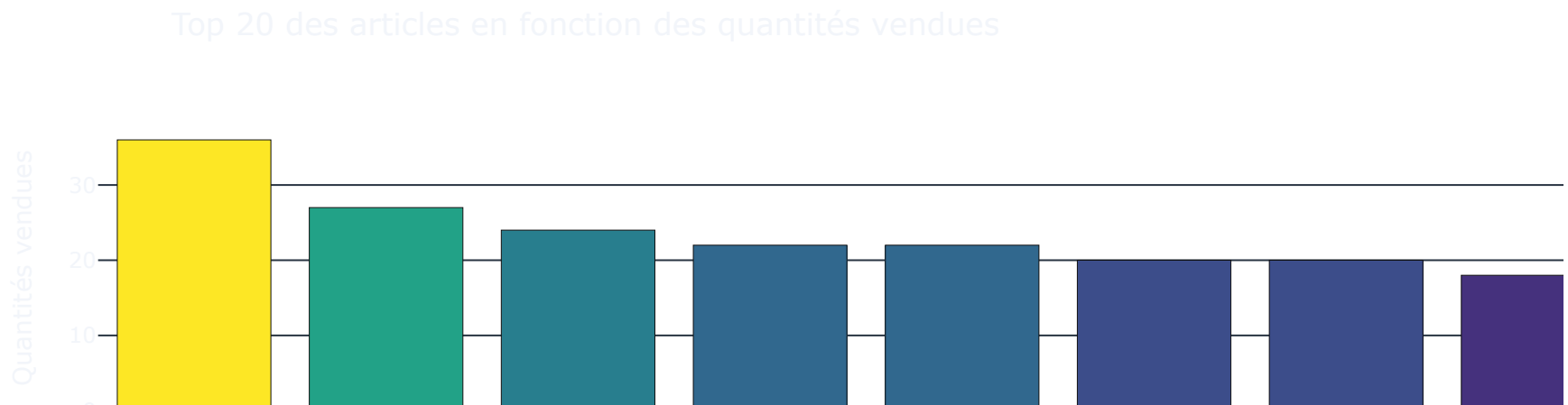
```
.....

df_merge_triees_par_quantites_vendues=df_merge.sort_values(by="total_sales",ascending=False)

df_merge_triees_par_quantites_vendues.reset_index()
top_20_articles_quantitees_vendues=df_merge_triees_par_quantites_vendues.head(20)

fig = px.bar(top_20_articles_quantitees_vendues,
              x='post_title',
              y='total_sales',
              title="Top 20 des articles en fonction des quantités vendues",
              labels={'post_title': 'Nom de l\'article', 'total_sales': 'Quantités vendues'},
              color='total_sales',
              color_continuous_scale='Viridis',
              template='plotly_dark')

fig.show()
```



```
#####
# Calculer le 20 / 80 du total des ventes #
#####

df_merge_sorted_vente = df_merge.sort_values(by='total_sales', ascending=False, ignore_index=True)

df_merge_sorted_vente['part_quantite'] = df_merge_sorted_vente['total_sales'] / df_merge_sorted_vente['total_sales'].sum()

df_merge_sorted_vente['somme_cumulative'] = df_merge_sorted_vente['part_quantite'].cumsum()

df_80 = df_merge_sorted_vente[df_merge_sorted_vente['somme_cumulative'] <= 0.80]

nb_articles_80 = df_80.shape[0]

nb_articles_total = df_merge_sorted_vente.shape[0]
proportion_articles_80 = nb_articles_80 / nb_articles_total

print(f"La proportion d'articles représentant 80% des ventes est de {(proportion_articles_80 * 100):.1f}%")
```

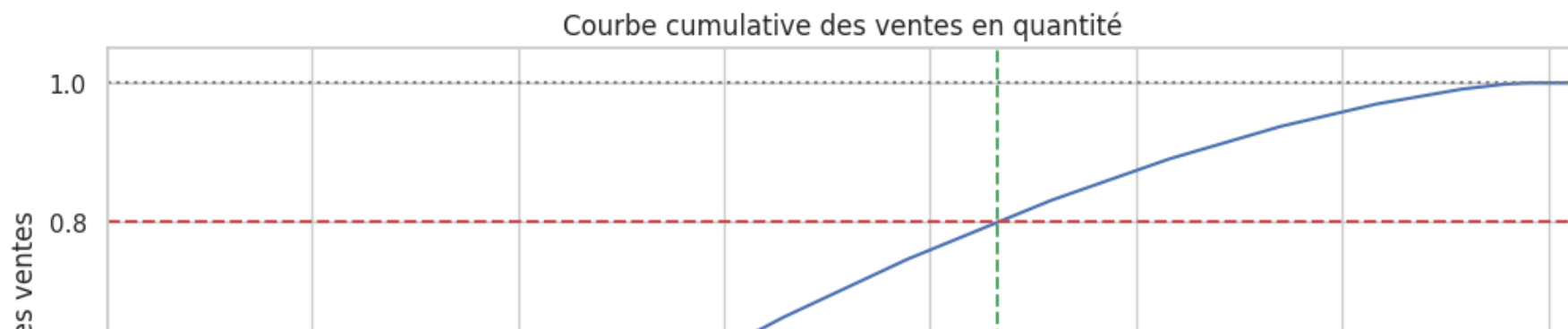
```
print(f"Somme cumulative des ventes pour les {nb_articles_80} articles : {df_80['somme_cumula
```

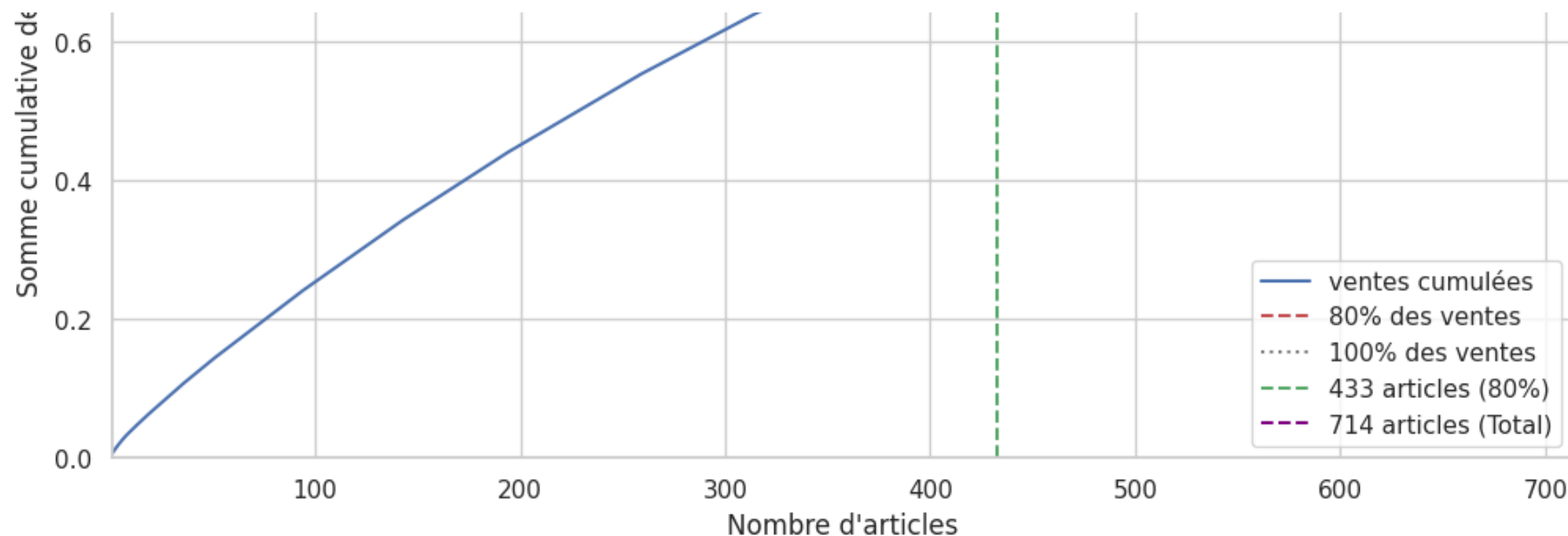
La proportion d'articles représentant 80% des ventes est de 60.64%

Somme cumulative des ventes pour les 433 articles : 0.7999

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.set(style="whitegrid")
plt.figure(figsize=(12, 6))
sns.lineplot(x=range(1, len(df_merge_sorted_vente) + 1), y=df_merge_sorted_vente['somme_cumula
plt.axhline(y=0.80, color="r", linestyle="--", label="80% des ventes")
plt.axhline(y=1.00, color="gray", linestyle=":", label="100% des ventes")
plt.axvline(x=nb_articles_80, color="g", linestyle="--", label=f"{nb_articles_80} articles (80
plt.axvline(x=len(df_merge_sorted_vente), color="purple", linestyle="--", label=f"{len(df_mer
plt.ylim(0, 1.05)
plt.xlim(1, len(df_merge_sorted_vente))
plt.xlabel("Nombre d'articles")
plt.ylabel("Somme cumulative des ventes")
plt.title("Courbe cumulative des ventes en quantité")
plt.legend()
plt.show()
```





Etape 5.3 - Analyse des stocks

```
df_merge["rotation_stock"]=df_merge["stock_quantity"]/df_merge["total_sales"]

df_merge["rotation_stock"].replace(np.inf, 0, inplace=True)

df_merge_sorted_stock = df_merge.sort_values(by="rotation_stock", ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x=df_merge_sorted_stock["rotation_stock"][:20], y=df_merge_sorted_stock["post_tit:
plt.xlabel("Rotation du stock")
plt.ylabel("Article ID")
```



```
plt.title("Top 20 articles par rotation de stock")
plt.gca().invert_yaxis()
plt.show()
```

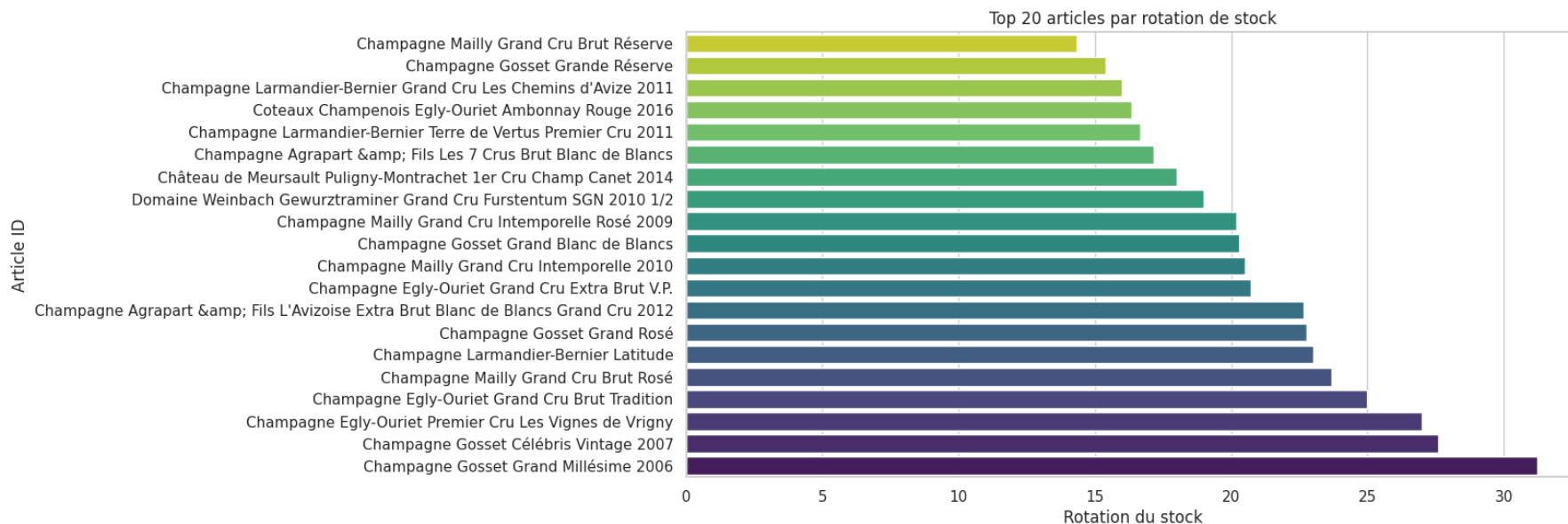
<ipython-input-72-3106065086>:3: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method. The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or

<ipython-input-72-3106065086>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `



```
#####  
# Valorisation des stocks en euros #  
#####  
  
df_merge["Valorisation_stocks_en_euros"]=df_merge["stock_quantity"]*df_merge["purchase_price"]  
  
print(f"Les stocks de marchandise ont une valeur totale de : {df_merge['Valorisation_stocks_en_euros'].sum()} euros.")  
  
Les stocks de marchandise ont une valeur totale de : 277328.07 euros.  
  
#####  
# Valorisation du nombre de produit en stock #  
#####  
  
print(f"Le nombre total de produits en stock est de : {df_merge['stock_quantity'].sum()}")  
  
Le nombre total de produits en stock est de : 16740.0.
```

Etape 5.4 - Analyse du taux de marge

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

```
df_merge["taux_marge"]=((df_merge["price"])-(df_merge['purchase_price']))/(df_merge['purchase
```

```
print(f"Le taux de marge minimale (HT) fait sur un produit vaut: {df_merge['taux_marge'].min(
```

```
print(f"Le taux de marge maximale(HT) effectué sur un produit est de : {df_merge['taux_marge']
```

```
Le taux de marge minimale (HT) fait sur un produit vaut: -83.67%
```

```
Le taux de marge maximale(HT) effectué sur un produit est de : 129.69%
```

```
df_merge[df_merge["taux_marge"]<0]
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	id_web	sku	total_sales	post_tit
79	4355	1.0	12.65	97.0	instock	77.48	12589	12589	0.0	Champag Egly-Oui Grand C Blanc

```
df_taux_marge_positif=df_merge[df_merge['taux_marge']>0]
```

```
print(f"Le prix minimum des produits ayant un taux de marge positif est : {df_taux_marge_posit
```

```
print(f"Le prix maximum des produits ayant un taux de marge positif est : {df_taux_marge_posit
```

```
Le prix minimum des produits ayant un taux de marge positif est : 5.20 euros.
```

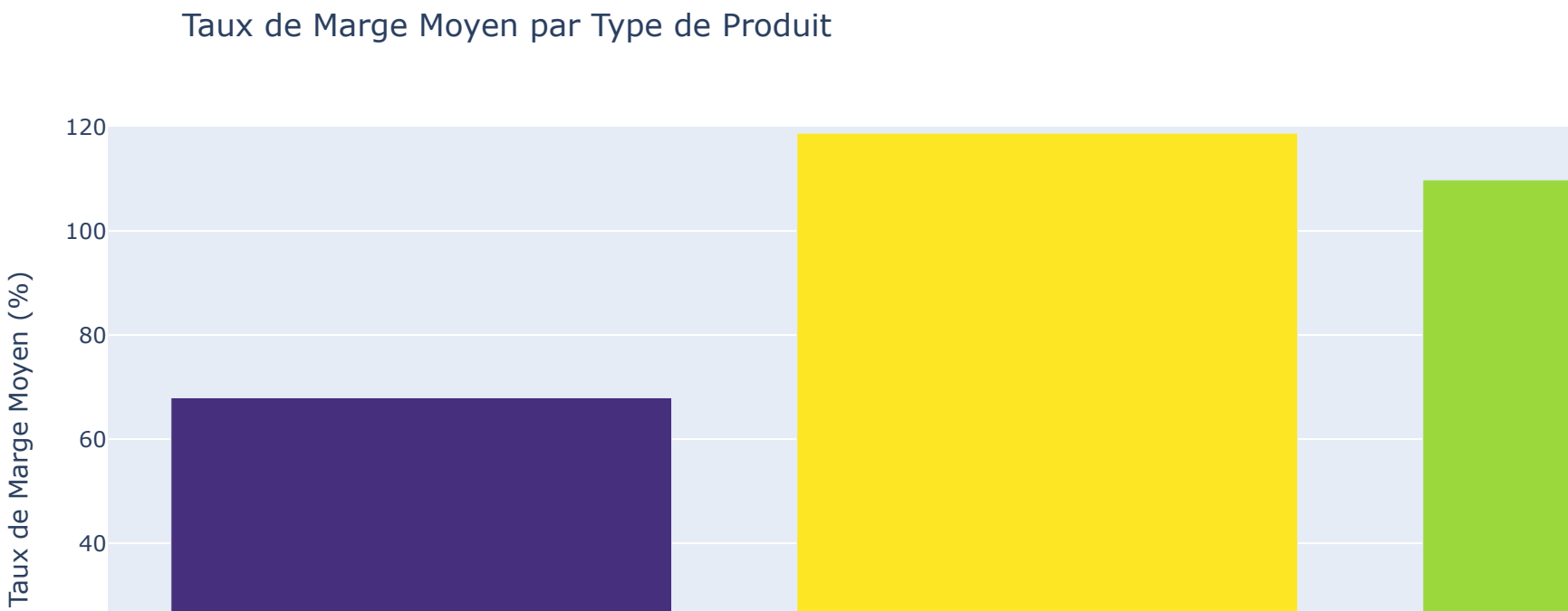
```
Le prix maximum des produits ayant un taux de marge positif est : 225.00 euros.
```

```
#création d'un dataframe avec le taux de marge moyen par type de produit
```

Création d'un dataframe avec le taux de marge moyen par type de produit

```
df_taux_marge_type_produit = df_taux_marge_positif.groupby("product_type")["taux_marge"].mean()
```

```
fig = px.bar(df_taux_marge_type_produit,  
             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.update_layout(yaxis_range=[0, 120])  
fig.show()
```



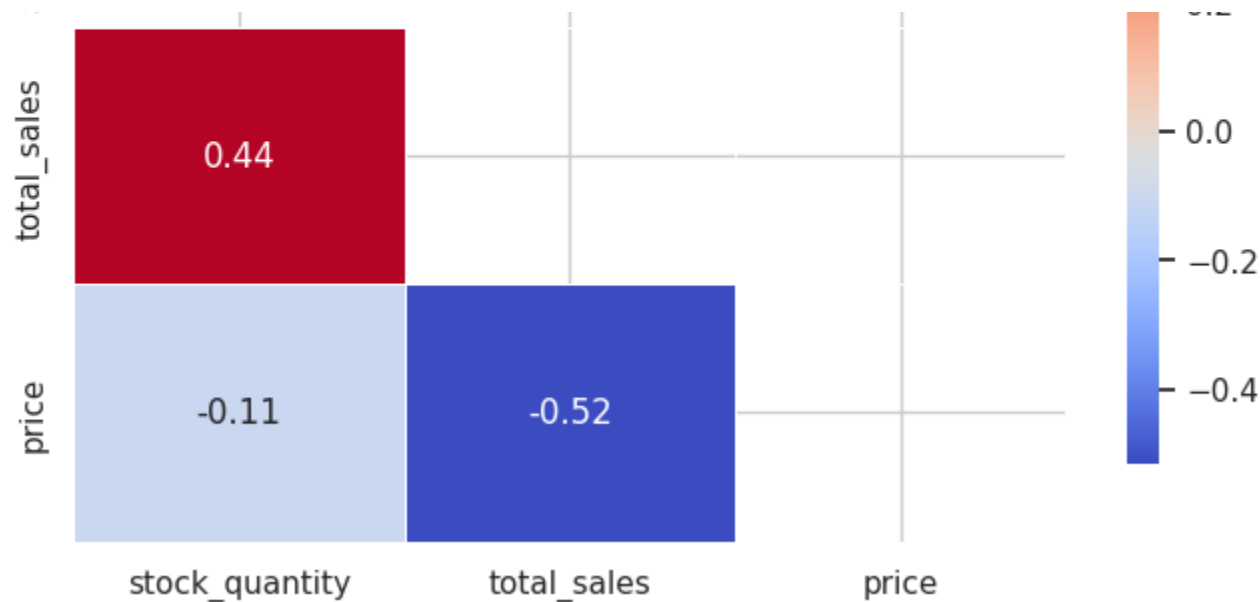


Etape 5.5 - Analyse des correlations entre les variables stock, sales et price

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

```
df_corr = df_merge[["stock_quantity", "total_sales", "price"]]
matrice_correlation = df_corr.corr()
plt.figure(figsize=(8, 5))
mask = np.triu(np.ones_like(matrice_correlation, dtype=bool))
sns.heatmap(matrice_correlation, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, mask=
plt.title("Matrice de corrélation : Sales, Stock et Price")
plt.show()
```





#Que peut-on conclure des correlations ?

```
print("Il n'y a pas de corrélation ou négligable (-0.11) entre le prix et la quantité en stock.  
print("Il y a une corrélation négative entre les ventes totales et la quantité en stock (-0.44  
print("Il y a une corrélation négative entre le prix et les ventes totales (-0.52). \nPar consé
```

Il n'y a pas de corrélation ou négligable (-0.11) entre le prix et la quantité en stock.
Cela suggère que le prix n'a pas d'impact significatif sur la quantité en stock.

Il y a une corrélation négative entre les ventes totales et la quantité en stock (-0.44).
Par conséquent, si les ventes totales augmentent, la quantité en stock a tendance à diminuer, ce qui pourrait indiqu

Il y a une corrélation négative entre le prix et les ventes totales (-0.52).
Par conséquent, si le prix augmente, les ventes totales ont tendance à diminuer, ce qui pourrait refléter une sensib

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

```
df_merge.to_excel("Projet6.xlsx", index=False)
```

