Deluy_Leslie_1_notebook_072025

September 17, 2025

ANALYSE DU STOCK ET DES VENTES DU SITE BOTTLENECK

1 OBJECTIF DE CE NOTEBOOK

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

Il s'agit d'un outil permettant de mixer et d'alterner code, texte et graphiques.

Cet outil est formidable pour plusieurs raisons:

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

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

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

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

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

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

Importation de la librairie Pandas

[1]:
```
import pandas as pd
```

#Importation de la librairie plotly express

[2]:
```
import plotly.express as px
```

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

[3]:
```
import pandas as pd
pd.set_option('display.max_columns', None)
```

1.2 - Chargements des fichiers

#Importation du fichier web.xlsx

#Importation du fichier erp.xlsx

#Importation du fichier liaison.xlsx

[4]:
```
import pandas as pd
import warnings

# Masquer les avertissements liés à openpyxl
warnings.filterwarnings("ignore", category=UserWarning)

# Importation des fichiers
df_web = pd.read_excel("web.xlsx")
df_erp = pd.read_excel("erp.xlsx")
df_liaison = pd.read_excel("liaison.xlsx")
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier erp.xlsx

#Afficher les dimensions du dataset

[5]:
```
df_erp.shape, df_web.shape, df_liaison.shape
```

[5]:
```
((825, 6), (1513, 29), (825, 2))
```

[6]:
```
print("Le tableau comporte {} observation(s) ou article(s)".format(df_erp.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_erp.shape[1]))
```

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

#Consulter le nombre de colonnes #La nature des données dans chacune des colonnes #Le nombre de valeurs présentes dans chacune des colonnes

[7]:
```
df_erp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   product_id      825 non-null    int64
 1   onsale_web      825 non-null    int64
```

```
 2   price           825 non-null    float64
 3   stock_quantity  825 non-null    int64
 4   stock_status    825 non-null    object
 5   purchase_price  825 non-null    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 38.8+ KB
```

#Afficher les 5 premières lignes de la table

[8]:
```
df_erp.head()
```

[8]:

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

#Vérifier si il y a des lignes en doublon dans la colonne product_id

[9]:
```
df_erp.duplicated(subset='product_id').sum()
```

[9]:
```
0
```

#Afficher les valeurs distinctes de la colonne stock_status #À quelle(s) autre(s) colonne(s) sont-elles liées ?

[10]:
```
df_erp['stock_status'].value_counts()
```

[10]:
```
stock_status
instock       733
outofstock     92
Name: count, dtype: int64
```

#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"

[11]:
```
df_erp['stock_status_2'] = df_erp['stock_quantity'].apply(lambda x:␣
 ↪'outofstock' if x == 0 else 'instock')
df_erp[['stock_quantity', 'stock_status', 'stock_status_2']].head()
```

[11]:

	stock_quantity	stock_status	stock_status_2
0	16	instock	instock
1	10	instock	instock
2	0	outofstock	outofstock
3	26	instock	instock
4	3	outofstock	instock

#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:

[12]:
```
df_erp["stock_status"] == df_erp["stock_status_2"]
```

[12]:
```
0       True
1       True
2       True
3       True
4      False
       …
820     True
821     True
822     True
823     True
824     True
Length: 825, dtype: bool
```

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

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

[13]:
```
nb_incoherences = (df_erp["stock_status"] != df_erp["stock_status_2"]).sum()
print("Nombre de lignes incohérentes :", nb_incoherences)
```

```
Nombre de lignes incohérentes : 4
```

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

[14]:
```
# Affichage des lignes incohérentes AVANT correction
incoherences = df_erp[df_erp['stock_status'] != df_erp['stock_status_2']]
print("Nombre de lignes incohérentes :", len(incoherences))
incoherences.head()
```

```
Nombre de lignes incohérentes : 4
```

[14]:

	product_id	onsale_web	price	stock_quantity	stock_status	\
4	4039	1	46.0	3	outofstock	
398	4885	1	18.7	0	instock	
449	4973	0	10.0	-10	outofstock	
573	5700	1	44.5	-1	outofstock	

	purchase_price	stock_status_2

```
4             23.77        instock
398            9.66     outofstock
449            4.96        instock
573           22.30        instock
```

#Corriger la ou les données incohérentes

#Vérification en utilisant le même code que plus haut pour afficher les problèmes

[15]:
```
df_erp.loc[df_erp['stock_status'] != df_erp['stock_status_2'], 'stock_status']␣
↪= df_erp['stock_status_2']
nb_incoherences_apres = (df_erp["stock_status"] != df_erp["stock_status_2"]).
↪sum()
print("Nombre de lignes incohérentes après correction :", nb_incoherences_apres)
```

```
Nombre de lignes incohérentes après correction : 0
```

J'ai corrigé les incohérences en remplaçant la valeur de la colonne `stock_status` par celle de `stock_status_2` lorsque les deux étaient différentes.

Une nouvelle vérification montre qu'il ne reste **aucune ligne incohérente**, preuve que la correction a été correctement appliquée.

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

2.1.1.1 - Analyse de la variable PRIX

1.1 LES PRIX

#Vérification des prix: Y a t-il des prix non renseignés, négatifs ou nuls? #Afficher le ou les prix non renseignés dans la colonne "price"

[16]:
```
df_erp[df_erp['price'].isnull()]
```

[16]:
```
Empty DataFrame
Columns: [product_id, onsale_web, price, stock_quantity, stock_status,
purchase_price, stock_status_2]
Index: []
```

[17]:
```
#Saisir l'instruction manquante dans la fonction format
nb_null_price = df_erp['price'].isnull().sum()
print("Nombre d'articles avec un prix non renseigné :", nb_null_price)
```

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

#Afficher le prix minimum de la colonne "price"

#Afficher le prix maximum de la colonne "price"

#Afficher les prix inférieurs à 0 (qu'est-ce qu'il faut en faire ?)

[18]:
```
df_erp[df_erp['price'] < 0]
```

[18]:
```
     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
```

[19]:
```
df_erp['price'].max()
```

[19]:
```
225.0
```

[20]:
```
df_erp['price'].min()
```

[20]:
```
-20.0
```

[21]:
```
# Étape 1 : Supprimer les lignes où le prix est strictement inférieur à 0
df_erp = df_erp[df_erp['price'] >= 0].copy()

# Afficher le nombre de lignes restantes après nettoyage des prix
print(f"Nombre de lignes après suppression des prix négatifs : {len(df_erp)}")
```

```
Nombre de lignes après suppression des prix négatifs : 822
```

2.1.1.2 - Analyse de la variable STOCK

1.1.1 stock_quantity

#Vérification de la colonne stock quantity

#Afficher la quantité minimum de la colonne “stock_quantity”

#Afficher la quantité maximum de la colonne “stock_quantity”

#Afficher les stocks inférieurs à 0 (qu’est-ce qu’il faut en faire ?)

[22]:
```
df_erp['stock_quantity'].describe()
```

[22]:
```
count    822.000000
mean      21.644769
std       21.947736
min      -10.000000
25%        7.000000
50%       18.000000
75%       30.000000
max      145.000000
Name: stock_quantity, dtype: float64
```

[23]:
```
min_stock = df_erp['stock_quantity'].min()
print("Quantité minimum de stock :", min_stock)
```

```
Quantité minimum de stock : -10
```

[24]:
```
max_stock = df_erp['stock_quantity'].max()
print("Quantité maximum de stock :", max_stock)
```

```
Quantité maximum de stock : 145
```

[25]:
```
stocks_negatifs = df_erp[df_erp['stock_quantity'] < 0]
print("Nombre d'articles avec un stock négatif :", len(stocks_negatifs))
stocks_negatifs
```

```
Nombre d'articles avec un stock négatif : 2
```

[25]:

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

[26]:
```
# Étape 2 : Supprimer les lignes où le stock est strictement inférieur à 0
df_erp = df_erp[df_erp['stock_quantity'] >= 0].copy()

# Afficher le nombre de lignes restantes après nettoyage des stocks
print(f"Nombre de lignes après suppression des stocks négatifs : {len(df_erp)}")
```

```
Nombre de lignes après suppression des stocks négatifs : 820
```

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?

[27]:
```
df_erp['onsale_web'].value_counts()
```

[27]:
```
onsale_web
1    715
0    105
Name: count, dtype: int64
```

#Quelles sont les colonnes à conserver selon vous?

1.1.2 Colonnes à conserver

Nous avons analysé les colonnes disponibles dans le fichier `erp.xlsx` et voici les colonnes à conserver pour la suite de l'analyse :

Colonne	Justification
`product_id`	Identifiant unique du produit, indispensable pour les jointures et l'identification.
`onsale_web`	Indique si le produit est en vente sur le site web.
`prix`	Prix de vente TTC du produit.
`stock_quantity`	Quantité disponible en stock, utile pour les analyses de disponibilité.
`stock_status`	Statut qualitatif du stock (En stock, Rupture, etc.).
`purchase_price`	Prix d'achat HT du produit, utile pour les calculs de marge.

La colonne `stock_status_2` a été jugée redondante et incohérente, elle est donc supprimée.

[28]:
```
# Étape 3 : Sélectionner uniquement les colonnes utiles (y compris␣
 ↪stock_status_2)
colonnes_utiles = ['product_id', 'onsale_web', 'price', 'stock_quantity',␣
 ↪'stock_status', 'purchase_price', 'stock_status_2']
df_utile = df_erp[colonnes_utiles].copy()

# Vérification des premières lignes
df_utile.head()
```

[28]:
```
   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

  stock_status_2
0        instock
1        instock
2     outofstock
3        instock
4        instock
```

#Supprimer la colonne comportant le libellé "stock_status_2" car elle est redondante #avec la colonne "stock_status".

[29]:
```
df_erp.drop(columns=['stock_status_2'], inplace=True)
```

2.1.1.4 - Analyse de la variable prix d'achat

1.2 prix d'achat

#Vérification de la colonne purchase_price : #Afficher le ou les prix non renseignés dans la colonne "purchase_price"

#Afficher le prix minimum de la colonne "purchase_price"

#Afficher le prix maximum de la colonne "purchase_price"

[30]:
```
df_erp[df_erp['purchase_price'].isnull()]
```

[30]:
```
Empty DataFrame
Columns: [product_id, onsale_web, price, stock_quantity, stock_status,
purchase_price]
Index: []
```

[31]:
```
df_erp['purchase_price'].max()
```

[31]:
```
137.81
```

[32]:
```
df_erp['purchase_price'].min()
```

[32]:
```
2.74
```

2.2 - Analyse exploratoire du fichier web.xlsx

#Dimension du dataset

#Nombre d'observations

#Nombre de caractéristiques

[33]:
```
df_web.shape
```

[33]:
```
(1513, 29)
```

1.2.1 Résultat obtenu

- Nombre d'observations (lignes) : 1513
- Nombre de caractéristiques (colonnes) : 29

Le dataset contient donc 1513 lignes et 29 colonnes.

#Consulter le nombre de colonnes

#La nature des données dans chacune des colonnes

#Le nombre de valeurs présentes dans chacune des colonnes

[34]:
```
print("Nombre de colonnes :", df_web.shape[1])

print("\nTypes de données :")
print(df_web.dtypes)
```

```python
print("\nNombre de valeurs renseignées :")
print(df_web.count())
```

```
Nombre de colonnes : 29

Types de données :
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

Nombre de valeurs renseignées :
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
```

1.2.2 Résultats obtenus

- Le dataset contient **29 colonnes**
- Chaque colonne possède un type de données spécifique (`int`, `float`, `object`, etc.)
- Certaines colonnes peuvent comporter des valeurs manquantes : cela peut nécessiter un nettoyage ou une stratégie de remplacement

Ces éléments permettent une **première compréhension de la structure du jeu de données**

Catégorie	Colonnes	Justification
Identifiant produit	`sku`, `product_type`	Identifier l'article
Données commerciales	`virtual`, `downloadable`, `average_rating`, `total_sales`, `rating_count`, `comment_count`	Performance produit (ventes, évaluations)
Disponibilité	`tax_status`, `comment_status`	Statut de publication, commentaires
Contenu / métadonnées	`post_date`, `post_title`, `post_name`, `menu_order`, `post_type`	Date de publication, titre, etc.
Identifiants techniques utiles	`post_author`, `post_modified`	Suivi auteurs, modifications

Catégorie	Colonnes	Justification
Autres pertinentes	`post_mime_type`	Type de contenu (visuels/fichiers joints)

Selon vous, quelles sont les colonnes à conserver ?

[35]:
```
colonnes_a_conserver = [
    'sku', 'product_type', 'virtual', 'downloadable',
    'average_rating', 'total_sales', 'rating_count', 'comment_count',
    'tax_status', 'comment_status',
    'post_date', 'post_title', 'post_name', 'menu_order',
    'post_type', 'post_author', 'post_modified',
    'post_mime_type'
]

df_web_filtré = df_web[colonnes_a_conserver]
```

#Si vous avez défini des colonnes à supprimer, effectuer l'opération | Catégorie | Colonnes | Justification | |————————|————————————|————————————————————————-| | Données totalement vides | `tax_class`, `post_content`, `post_password`, `post_content_filtered` | Aucune valeur renseignée (100 % manquantes) | | Redondance temporelle | `post_date_gmt`, `post_modified_gmt` | Dupliquent l'information déjà présente dans `post_date` et `post_modified` | | Métadonnées techniques | `guid` | Identifiant interne WordPress, inutile pour analyse produit | | Texte peu exploitable | `post_excerpt` | Donnée textuelle partielle, souvent vide ou non exploitable | | Attributs techniques WP | `post_status`, `ping_status`, `post_parent` | Souvent sans utilité en analyse produit (selon cas d'usage) |

[36]:
```
# Liste des colonnes à supprimer
colonnes_a_supprimer = [
    'tax_class',
    'post_content',
    'post_password',
    'post_content_filtered',
    'post_date_gmt',
    'post_modified_gmt',
    'guid',
    'post_excerpt',
    'post_status',
    'ping_status',
    'post_parent'
]

# Suppression des colonnes
df_web = df_web.drop(columns=colonnes_a_supprimer)

# Affichage des dimensions après nettoyage
```

```
print("Dimensions du dataset après suppression :", df_web.shape)
```

```
Dimensions du dataset après suppression : (1513, 18)
```

[37]:
```
# Visualisation des valeurs uniques de la colonne 'sku'
valeurs_uniques_sku = df_web['sku'].unique()
print("Exemples de valeurs SKU :", valeurs_uniques_sku[:20])
# Affiche les 20 premières valeurs uniques

# Détection des SKU incorrects : vide, trop courts, ou contenant des caractères␣
 ↪spéciaux
import re

sku_invalides = df_web[~df_web['sku'].astype(str).str.match(r'^[A-Za-z0-9\-_.
 ↪]{4,}$', na=False)]
print("Nombre de SKU invalides :", len(sku_invalides))
```

```
Exemples de valeurs SKU : [11862 16057 14692 16295 15328 15471 16515 16246 nan
13572 16513 16585
 16269 15526 12869 15575 11586 14338 15425 16560]
Nombre de SKU invalides : 111
```

#Si vous avez identifié des codes articles ne respectant pas la régle de codification, consultez-les

[38]:
```
import re

# SKU invalides = NaN, trop courts (<4), ou caractères spéciaux
sku_invalides = df_web[~df_web['sku'].astype(str).str.match(r'^[A-Za-z0-9\-\.
 ↪_]{4,}$', na=False)]

# Affichage propre
liste_sku_invalides = sku_invalides['sku'].drop_duplicates().tolist()
print("Nombre de SKU invalides :", len(liste_sku_invalides))
print("Exemples de valeurs SKU invalides :", liste_sku_invalides)
```

```
Nombre de SKU invalides : 14
Exemples de valeurs SKU invalides : [nan, 804, 802, 791, 793, 304, 805, 38, 531,
812, 798, 523, 41, 807]
```

#Identifier les lignes sans code article

[39]:
```
# Lignes sans code article (valeurs manquantes dans 'sku')
lignes_sans_sku = df_web[df_web['sku'].isna()]

# Affichage du nombre de lignes concernées
print("Nombre de lignes sans code article :", len(lignes_sans_sku))
```

```
Nombre de lignes sans code article : 85
```

#Pour les codes articles identifiés, réaliser une analyse et définir l'action à entreprendre

[40]:
```
#analyse des lignes sans code article
print("Aperçu des lignes sans code article :")
display(lignes_sans_sku)

#suppression : le code article est essentiel à la correspondance
df_web=df_web[~df_web['sku'].isna()]
print("Nombre de lignes restantes après suppressions", len(df_web))
```

```
Aperçu des lignes sans code article :
```

	sku	virtual	downloadable	rating_count	average_rating	total_sales	\
8	NaN	0	0	0	NaN	NaN	
20	NaN	0	0	0	NaN	NaN	
30	NaN	0	0	0	NaN	NaN	
37	NaN	0	0	0	NaN	NaN	
41	NaN	0	0	0	NaN	NaN	
…	…	…	…	…	…	…	
1384	NaN	0	0	0	NaN	NaN	
1429	NaN	0	0	0	NaN	NaN	
1432	NaN	0	0	0	NaN	NaN	
1445	NaN	0	0	0	NaN	NaN	
1457	NaN	0	0	0	NaN	NaN	

	tax_status	post_author	post_date	product_type	post_title	comment_status	\
8	NaN	NaN	NaT	NaN	NaN	NaN	
20	NaN	NaN	NaT	NaN	NaN	NaN	
30	NaN	NaN	NaT	NaN	NaN	NaN	
37	NaN	NaN	NaT	NaN	NaN	NaN	
41	NaN	NaN	NaT	NaN	NaN	NaN	
…	…	…	…	…	…	…	
1384	NaN	NaN	NaT	NaN	NaN	NaN	
1429	NaN	NaN	NaT	NaN	NaN	NaN	
1432	NaN	NaN	NaT	NaN	NaN	NaN	
1445	NaN	NaN	NaT	NaN	NaN	NaN	
1457	NaN	NaN	NaT	NaN	NaN	NaN	

	post_name	post_modified	menu_order	post_type	post_mime_type	\
8	NaN	NaT	NaN	NaN	NaN	
20	NaN	NaT	NaN	NaN	NaN	
30	NaN	NaT	NaN	NaN	NaN	
37	NaN	NaT	NaN	NaN	NaN	
41	NaN	NaT	NaN	NaN	NaN	
…	…	…	…	…	…	
1384	NaN	NaT	NaN	NaN	NaN	
1429	NaN	NaT	NaN	NaN	NaN	
1432	NaN	NaT	NaN	NaN	NaN	

```
1445       NaN           NaT         NaN       NaN            NaN
1457       NaN           NaT         NaN       NaN            NaN

      comment_count
8               NaN
20              NaN
30              NaN
37              NaN
41              NaN
…               …
1384            NaN
1429            NaN
1432            NaN
1445            NaN
1457            NaN

[85 rows x 18 columns]
```

```
Nombre de lignes restantes après suppressions 1428
```

#La clé pour chaque ligne est-elle unique? autrement dit, y a-t-il des doublons?

[41]:
```python
# Vérification de l'unicité des valeurs dans la colonne 'sku'

nb_total = len(df_web)
nb_uniques = df_web['sku'].nunique()

print("Nombre total de lignes :", nb_total)
print("Nombre de valeurs 'sku' uniques :", nb_uniques)

if nb_total == nb_uniques:
    print( " La clé 'sku' est unique pour chaque ligne.")
else :
    print(" Il y a des doublons dans la colonne 'sku'.")
```

```
Nombre total de lignes : 1428
Nombre de valeurs 'sku' uniques : 714
 Il y a des doublons dans la colonne 'sku'.
```

[42]:
```python
df_web['sku'] = df_web['sku'].astype(str)

# Compter les doublons
doublons_counts = df_web ['sku'].value_counts()
sku_en_doublon = doublons_counts[doublons_counts > 1].index

# Afficher les lignes concernées
df_web[df_web['sku'].isin(sku_en_doublon)][['sku', 'post_title', 'post_type']].
 ↪sort_values(by='sku').head(10)
```

[42]:

	sku	post_title	post_type
668	10014	Darnley's London Dry Gin Original	product
1030	10014	Darnley's London Dry Gin Original	attachment
887	10459	Alphonse Mellot Sancerre Rouge Génération XIX …	attachment
748	10459	Alphonse Mellot Sancerre Rouge Génération XIX …	product
802	10775	Albert Mann Pinot Gris Vendanges Tardives Alte…	product
1317	10775	Albert Mann Pinot Gris Vendanges Tardives Alte…	attachment
520	10814	Thierry Germain Saumur-Champigny Outre Terre 2013	attachment
860	10814	Thierry Germain Saumur-Champigny Outre Terre 2013	product
1322	11049	Alphonse Mellot Sancerre Rouge En Grands Champ…	attachment
408	11049	Alphonse Mellot Sancerre Rouge En Grands Champ…	product

suppressions des doublons SKU

[43]:
```
# Suppression des doublons sur la colonne 'sku' en gardant la première␣
 ↪occurrence
df_web = df_web.drop_duplicates(subset='sku', keep='first')


# Vérifier que chaque SKU est maintenant unique
print("Nombre total de lignes :", len(df_web))
print("Nombre de SKU uniques :", df_web['sku'].nunique())
```

```
Nombre total de lignes : 714
Nombre de SKU uniques : 714
```

#Les lignes sans code article semblent être toutes non renseignées

#Pour s'en assurer, réaliser les étapes suivantes:

#1 - Créer un dataframe avec uniquement les lignes sans code article

#2 - Utiliser la fonction df.info() sur ce nouveau dataframe pour observer le nombre de valeurs renseignées dans chacune des colonnes

#3 - Que constatez-vous?

[44]:
```
# Filtrer les lignes où 'sku' est vide ou manquant

df_sans_sku = df_web[df_web['sku'].isna()| (df_web['sku'].astype(str).str.
 ↪strip()=='')]

# Afficher les informations du dataframe filtré
df_sans_sku.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 0 entries
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   sku             0 non-null      object
 1   virtual         0 non-null      int64
 2   downloadable    0 non-null      int64
 3   rating_count    0 non-null      int64
 4   average_rating  0 non-null      float64
 5   total_sales     0 non-null      float64
 6   tax_status      0 non-null      object
 7   post_author     0 non-null      float64
 8   post_date       0 non-null      datetime64[ns]
 9   product_type    0 non-null      object
 10  post_title      0 non-null      object
 11  comment_status  0 non-null      object
 12  post_name       0 non-null      object
 13  post_modified   0 non-null      datetime64[ns]
 14  menu_order      0 non-null      float64
 15  post_type       0 non-null      object
 16  post_mime_type  0 non-null      object
 17  comment_count   0 non-null      float64
dtypes: datetime64[ns](2), float64(5), int64(3), object(8)
memory usage: 0.0+ bytes
```

1.2.3 Résultat :

- Le DataFrame `df_sans_sku` contient **0 lignes**.
- Toutes les colonnes ont **0 valeurs non nulles**, ce qui confirme que **aucune ligne** du jeu de données ne présente un `sku` vide ou manquant.

2.3 - Analyse exploratoire du fichier liaison.xlsx

#Dimension du dataset

#Nombre d'observations

#Nombre de caractéristiques

[45]:
```
# Dimension du dataset
print("Dimensions du dataset :", df_liaison.shape)

# Nombre d'observations
nb_observations = df_liaison.shape[0]
print("Nombre d'observations :", nb_observations)

# Nombre de caractéristiques
nb_caracteristiques = df_liaison.shape[1]
print("Nombre de caractéristiques :", nb_caracteristiques)
```

```
Dimensions du dataset : (825, 2)
Nombre d'observations : 825
Nombre de caractéristiques : 2
```

#Consulter le nombre de colonnes

#La nature des données dans chacune des colonnes

#Le nombre de valeurs présentes dans chacune des colonnes

[46]:
```
# Consulter le nombre de colonnes
nb_colonnes = df_liaison.shape[1]
print("Nombre de colonnes :", nb_colonnes)

# La nature des données dans chacune des colonnes
print("\nTypes de données par colonne :")
print(df_liaison.dtypes)

# Le nombre de valeurs présentes dans chacune des colonnes
print("\nNombre de valeurs non nulles par colonne :")
print(df_liaison.count())
```

```
Nombre de colonnes : 2

Types de données par colonne :
id_web        object
product_id     int64
dtype: object

Nombre de valeurs non nulles par colonne :
id_web        734
product_id    825
dtype: int64
```

#Les valeurs de la colonne “product_id” sont-elles toutes uniques?

[47]:
```
df_erp['product_id'].is_unique
```

[47]:
```
True
```

#Les valeurs de la colonne “id_web” sont-elles toutes uniques?

[48]:
```
df_liaison['id_web'].is_unique
```

[48]:
```
False
```

#Avons-nous des articles sans correspondance?

[49]:
```
# Identifier les lignes sans correspondance (id_web vide ou manquant)
articles_sans_id_web = df_liaison[df_liaison['id_web'].isna() |␣
 ↪(df_liaison['id_web'].astype(str).str.strip() == '')]

# Afficher le nombre d'articles sans correspondance
print("Nombre d'articles sans correspondance :", len(articles_sans_id_web))
```

```
# Afficher les premières lignes pour aperçu
articles_sans_id_web.head()
```

```
Nombre d'articles sans correspondance : 91
```

[49]:

	id_web	product_id
19	NaN	4055
49	NaN	4090
50	NaN	4092
119	NaN	4195
131	NaN	4209

[50]:
```
# Supprimer les lignes dont le 'sku' est vide, uniquement des espaces ou NaN
df_web = df_web[~(df_web['sku'].isna() | (df_web['sku'].astype(str).str.strip()␣
↪== ''))]
```

[51]:
```
# Supprimer les lignes où 'id_web' est vide, uniquement des espaces ou NaN dans␣
↪df_liaison
df_liaison = df_liaison[~(df_liaison['id_web'].isna() | (df_liaison['id_web'].
↪astype(str).str.strip() == ''))]

# Réinitialiser les index pour éviter des incohérences
df_liaison.reset_index(drop=True, inplace=True)
```

[52]:
```
df_liaison['id_web'].isna().sum(), (df_liaison['id_web'].astype(str).str.
↪strip() == '').sum()
```

[52]: (0, 0)

[53]:
```
# Vérifier les valeurs manquantes ou vides
df_liaison['product_id'].isna().sum(), (df_liaison['product_id'].astype(str).
↪str.strip() == '').sum()
```

[53]: (0, 0)

Fichier	Clé utilisée	Nettoyage effectué	Observations
`df_wed`	`sku`	lignes avec `NaN` ou vides supprimées	OK
`df_liaison`	`id_web`, `product_id`	lignes avec `NaN` ou vides supprimées	OK
`df_erp`	*(pas de **`sku`**)*	pas concerné	OK car `sku` n'existe pas

Etape 3 - Jonction des fichiers

Etape 3.1 - Jonction du fichier df_erp et df_liaison

#Fusion des fichiers df_erp et df_liaison

[54]:
```
df_merge = pd.merge(df_erp, df_liaison, on='product_id', how='left')
df_merge.head()
```

[54]:

	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	

	id_web
0	15298
1	15296
2	15300
3	19814
4	19815

#Y a t-il des lignes ne “matchant” pas entre les 2 fichiers?

[55]:
```
# Nombre de lignes sans correspondance (id_web manquant)
nb_sans_id_web = df_merge['id_web'].isna().sum()
print("Nombre d'articles ERP sans correspondance web :", nb_sans_id_web)

# Afficher quelques exemples de lignes sans correspondance
df_merge[df_merge['id_web'].isna()].head()

# Vérifier le nombre de doublons sur product_id
nb_doublons = df_merge.duplicated(subset='product_id').sum()
print("Nombre de doublons sur product_id :", nb_doublons)
```

```
Nombre d'articles ERP sans correspondance web : 87
Nombre de doublons sur product_id : 0
```

Etape 3.2 - Jonction du fichier df_merge et df_web

#Fusionner les datasets df_merge et df_web

[56]:
```
#Fusionner les datasets df_merge et df_web

df_merge['id_web'] = df_merge['id_web'].astype(str).str.strip()
df_web['sku'] = df_web['sku'].astype(str).str.strip()


df_merge = pd.merge(df_merge, df_web, left_on='id_web', right_on='sku',␣
 ↪how='left')

# Vérification du résultat
print("Nombre de lignes sans correspondance après correction :",␣
 ↪df_merge['sku'].isna().sum())
```

```
Nombre de lignes sans correspondance après correction : 107
```

#Avons-nous des lignes sans correspondance?

107 (sans correspondance dans le web) − 87 (sans liaison du tout) = 20 lignes avec un id_web dans liaison, mais sans sku correspondant dans web

[57]:
```
# Identifier les id_web présents dans df_merge mais sans correspondance trouvée dans df_web
id_web_non_trouves = df_merge[df_merge['sku'].isna()]['id_web'].dropna().unique()

# Afficher la liste des id_web problématiques
print("Liste des id_web sans correspondance dans df_web :")
print(id_web_non_trouves)
```

```
Liste des id_web sans correspondance dans df_web :
['nan' '13771' '15065' '14785' '12601' '15154' '14360' '15608' '15586'
 '15272' '15630' '14648' '14715' '14730' '14689' '14379' '15609' '14377'
 '13577' '15529' '14680-1']
```

[58]:
```
# Isoler les produits ERP avec un id_web qui n'a pas trouvé de correspondance dans df_web
id_web_exclus = ['13771', '15065', '14785', '12601', '15154', '14360', '15608', '15586',
                 '15272', '15630', '14648', '14715', '14730', '14689', '14379', '15609',
                 '14377', '13577', '15529', '14680-1']

df_sans_correspondance = df_merge[df_merge['id_web'].isin(id_web_exclus)]

# Affichage des lignes concernées
df_sans_correspondance[['product_id', 'id_web', 'post_title', 'price', 'stock_quantity', 'stock_status']]
```

[58]:

	product_id	id_web	post_title	price	stock_quantity	stock_status
192	4289	13771	NaN	22.8	0	outofstock
235	4568	15065	NaN	21.5	0	outofstock
240	4584	14785	NaN	32.3	0	outofstock
354	4741	12601	NaN	12.4	0	outofstock
390	4864	15154	NaN	8.3	0	outofstock
393	4869	14360	NaN	17.2	0	outofstock
423	4921	15608	NaN	13.8	0	outofstock
424	4922	15586	NaN	21.5	0	outofstock
467	5018	15272	NaN	15.4	0	outofstock
470	5021	15630	NaN	17.1	0	outofstock
520	5505	14648	NaN	10.1	0	outofstock
537	5559	14715	NaN	27.9	3	instock

```
545        5570    14730        NaN   22.5              0   outofstock
609        5800    14689        NaN   32.3              0   outofstock
656        5953    14379        NaN   47.5              0   outofstock
657        5954    15609        NaN   18.8              0   outofstock
658        5955    14377        NaN   27.3              0   outofstock
660        5957    13577        NaN   39.0              0   outofstock
683        6100    15529        NaN   12.9              0   outofstock
818        7329  14680-1        NaN   26.5             14      instock
```

[59]:
```
# Supprimer les 20 lignes sans correspondance web identifiées par leur id_web
id_web_exclus = ['13771', '15065', '14785', '12601', '15154', '14360', '15608',␣
 ↪'15586',
                 '15272', '15630', '14648', '14715', '14730', '14689', '14379',␣
 ↪'15609',
                 '14377', '13577', '15529', '14680-1']

# Filtrer et réécrire df_merge avec uniquement les lignes valides
df_merge = df_merge[~df_merge['id_web'].isin(id_web_exclus)].copy()

# Vérification
print(" Nombre de lignes conservées après nettoyage :", df_merge.shape[0])
```

```
 Nombre de lignes conservées après nettoyage : 800
```

[60]:
```
# Vérifier les doublons sur les clés
print("Doublons sur product_id :", df_merge.duplicated(subset='product_id').
 ↪sum())
print("Doublons sur id_web :", df_merge.duplicated(subset='id_web').sum())
```

```
Doublons sur product_id : 0
Doublons sur id_web : 86
```

[61]:
```
# Agrégation pour supprimer les doublons sur id_web
aggregation_rules = {
    'product_id': 'first',
    'onsale_web': 'max',
    'price': 'mean',
    'stock_quantity': 'sum',
    'stock_status': 'first',
    'purchase_price': 'mean',
    'sku': 'first',
    'virtual': 'first',
    'downloadable': 'first',
    'rating_count': 'sum',
    'average_rating': 'mean',
    'total_sales': 'sum',
    'tax_status': 'first',
```

```
    'post_author': 'first',
    'post_date': 'first',
    'product_type': 'first',
    'post_title': 'first',
    'comment_status': 'first',
    'post_name': 'first',
    'post_modified': 'first',
    'menu_order': 'first',
    'post_type': 'first',
    'post_mime_type': 'first',
    'comment_count': 'sum'
}

# Agrégation
df_merge = df_merge.groupby('id_web', as_index=False).agg(aggregation_rules)

# Vérification
print(" df_merge mis à jour :", df_merge.shape)
```

```
 df_merge mis à jour : (714, 25)
```

Etape 4 - Analyse univariée des prix

Etape 4.1 - Exploration par la visualisation de données

#Création d'une boîte à moustache de la répartition des prix grâce à Pandas

[62]:
```
import pandas as pd
```

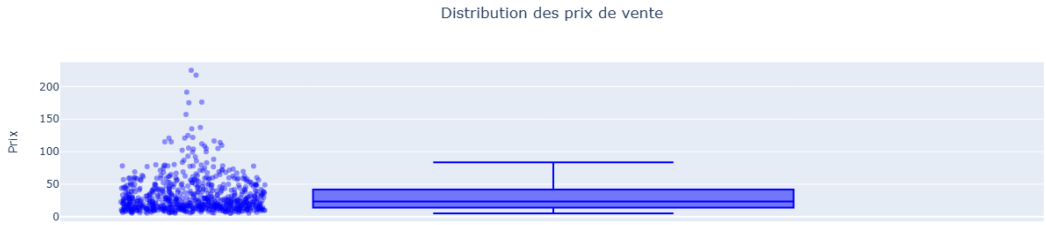
#Autre méthode avec plotly express

[63]:
```
import plotly.express as px

# Tracer un boxplot avec les points individuels (jitter)
fig = px.box(df_merge,
             y="price",
             points="all",
             title="Distribution des prix de vente",
             color_discrete_sequence=["blue"])

# Personnalisation du style
fig.update_traces(jitter=0.3, marker_opacity=0.4, marker_color="blue")  # effet␣
 ↪de dispersion et transparence
fig.update_layout(yaxis_title="Prix", title_x=0.5)  # centrage du titre

fig.show()
```



Etape 4.2 - Exploration par l'utilisation de méthodes statistiques

Etape 4.2.1 - Identification par le Z-index

#Calculer la moyenne du prix

#Calculer l'écart-type du prix

#Calculer le Z-score

[64]:
```
moyenne_prix = df_merge['price'].mean()
ecart_type_prix = df_merge['price'].std()

print(f"Moyenne du prix : {moyenne_prix:.2f}")
print(f"Écart-type du prix : {ecart_type_prix:.2f}")

df_merge['z_score_price'] = (df_merge['price'] - moyenne_prix) / ecart_type_prix
df_merge[['product_id', 'price', 'z_score_price']].head()
```

```
Moyenne du prix : 32.32
Écart-type du prix : 27.59
```

[64]:

	product_id	price	z_score_price
0	5913	36.0	0.133353
1	4617	67.5	1.274957
2	4709	44.0	0.423284
3	4656	43.0	0.387042
4	4619	59.0	0.966905

[65]:
```
#Quel est le seuil prix dont le z-score est supérieur à 3?
```

[66]:
```
outliers_z = df_merge[(df_merge['z_score_price'] > 3) |␣
 ↪(df_merge['z_score_price'] < -3)]
print(f"Nombre d'outliers détectés par Z-score : {len(outliers_z)}")
outliers_z[['product_id', 'price', 'z_score_price']]
```

```
Nombre d'outliers détectés par Z-score : 13
```

[66]:

	product_id	price	z_score_price
131	4904	137.0	3.793734
161	5001	217.5	6.711166
187	5917	122.0	3.250113
207	5612	124.8	3.351589
208	6126	135.0	3.721251
220	5892	191.3	5.761642
231	6216	121.0	3.213871
232	6213	121.0	3.213871
241	6202	116.4	3.047161
262	5767	175.0	5.170907
510	4352	225.0	6.982977
683	4402	176.0	5.207148
696	4406	157.0	4.518562

1.3 Analyse des outliers de prix dans df_merge via le Z-score

Nous utilisons ici la méthode du Z-score pour détecter les valeurs atypiques dans la variable price.

1.3.1 Statistiques de base

- Moyenne du prix : 32.32 €
- Écart-type : 27.59 €
- Seuil retenu pour la détection d'outliers : $|\text{Z-score}| > 3$

1.3.2 Résultats de la détection

- Nombre total d'outliers détectés : 13
- Tous les outliers identifiés sont situés dans la queue droite de la distribution.

1.3.3 Analyse

- Ces outliers correspondent à des valeurs de prix très élevées, bien au-dessus de la moyenne.
- Ils peuvent :
  - Représenter des produits premium ou très spécialisés,
  - Être le résultat d'erreurs de saisie (ex. : mauvais séparateur décimal, devise incorrecte),
  - Biaiser les analyses statistiques et les visualisations (moyenne, échelle des axes, etc.).

1.3.4 Prochaine étape suggérée

Isoler ou exclure ces outliers pour effectuer une analyse robuste (méthode IQR, boxplot sans extrêmes, etc.).

Etape 4.2.2 - Identification par l'intervalle interquartile

#Utilisation de la fonction "describe" de Pandas pour l'étude des mesures de dispersion

[67]:
```
import pandas as pd
df_merge['price'].describe()
```

[67]:
```
count    714.000000
mean      32.320436
std       27.592754
min        5.200000
25%       14.062500
50%       23.450000
75%       42.000000
max      225.000000
Name: price, dtype: float64
```

#Définir un seuil pour les articles "outliers" en prix

[68]:
```
Q1 = df_merge['price'].quantile(0.25)
Q3 = df_merge['price'].quantile(0.75)
IQR = Q3 - Q1

borne_inf = Q1 - 1.5 * IQR
borne_sup = Q3 + 1.5 * IQR

print(f"Borne inférieure : {borne_inf}")
print(f"Borne supérieure : {borne_sup}")
```

```
Borne inférieure : -27.84375
Borne supérieure : 83.90625
```

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

[69]:
```
# Détection des outliers sur le prix avec df_merge (borne issue de l'IQR)
borne_sup = 83.91  # issue du calcul précédent

# Filtrage des outliers
outliers_iqr = df_merge[df_merge['price'] > borne_sup]

# Nombre d'articles outliers
nb_outliers = len(outliers_iqr)

# Proportion sur l'ensemble du catalogue
proportion_outliers = round(nb_outliers / len(df_merge) * 100, 2)

print(f"Nombre d'articles considérés comme outliers : {nb_outliers}")
print(f"Proportion d'outliers dans le catalogue : {proportion_outliers} %")
```

```
Nombre d'articles considérés comme outliers : 31
Proportion d'outliers dans le catalogue : 4.34 %
```

1.4 Analyse des outliers de prix dans df_merge via l'Intervalle interquartile (IQR)

Nous utilisons ici la méthode de l'IQR (Interquartile Range) pour détecter les valeurs atypiques dans la variable price.

1.4.1 Statistiques de base

- Moyenne du prix : 32.32 €
- Écart-type : 27.59 €
- Seuil retenu pour la détection d'outliers :
  - Borne inférieure : -27.84
  - Borne supérieure : 83.91

1.4.2 Résultats de la détection

- Nombre total d'outliers détectés : 31
- Tous les outliers identifiés sont au-dessus de la borne supérieure.

1.4.3 Analyse

- Ces outliers correspondent à des valeurs de prix très élevées, bien au-dessus de la moyenne.
- Ils peuvent :
  - Représenter des produits premium ou très spécialisés,
  - Être le résultat d'erreurs de saisie (ex. : mauvais séparateur décimal, devise incorrecte),
  - Biaiser les analyses statistiques et les visualisations (moyenne, échelle des axes, etc.).

1.4.4 Prochaine étape suggérée

Isoler ou exclure ces outliers pour effectuer une analyse robuste (méthode IQR, boxplot sans extrêmes, etc.).

[70]:
```
# Comparaison du nombre d'outliers détectés par Z-score et par IQR

# Z-score
outliers_z = df_merge[(df_merge['z_score_price'] > 3) |␣
 ↪(df_merge['z_score_price'] < -3)]
nb_outliers_z = outliers_z.shape[0]

# IQR
```

```
q1 = df_merge['price'].quantile(0.25)
q3 = df_merge['price'].quantile(0.75)
iqr = q3 - q1
borne_inf = q1 - 1.5 * iqr
borne_sup = q3 + 1.5 * iqr

outliers_iqr = df_merge[(df_merge['price'] < borne_inf) | (df_merge['price'] >␣
↪borne_sup)]
nb_outliers_iqr = outliers_iqr.shape[0]

# Résumé comparatif
comparatif_outliers = pd.DataFrame({
    'Méthode': ['Z-score', 'IQR (Interquartile)'],
    'Nb d\'outliers détectés': [nb_outliers_z, nb_outliers_iqr],
    'Seuils utilisés': [
        '|Z| > 3',
        f'{borne_inf:.2f} € < prix < {borne_sup:.2f} €'
    ]
})

# Affichage (en notebook)
display(comparatif_outliers)
```

	Méthode	Nb d'outliers détectés	Seuils utilisés
0	Z-score	13	|Z| > 3
1	IQR (Interquartile)	31	-27.84 € < prix < 83.91 €

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

Etape 5.1 - Analyse des ventes en CA

[71]:
```
df_merge.columns
```

[71]:
```
Index(['id_web', 'product_id', 'onsale_web', 'price', 'stock_quantity',
       'stock_status', 'purchase_price', 'sku', 'virtual', 'downloadable',
       'rating_count', 'average_rating', 'total_sales', 'tax_status',
       'post_author', 'post_date', 'product_type', 'post_title',
       'comment_status', 'post_name', 'post_modified', 'menu_order',
       'post_type', 'post_mime_type', 'comment_count', 'z_score_price'],
      dtype='object')
```

2 Calculer le CA du site web

#Créer une colonne calculant le CA par article

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

[72]:
```
df_merge['ca_par_article'] = df_merge['price'] * df_merge['total_sales']
```

[73]:
```
df_merge['ca_par_article'].sum()
ca_total = df_merge['ca_par_article'].sum()
print(f"CA total du site web : {round(ca_total, 2)} €")
```

```
CA total du site web : 153392.1 €
```

3 Palmarès des articles en CA

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

#Réinitialiser l'index du dataset par un reset_index

#Afficher les 20 premiers articles en CA

#Graphique en barre des 20 premiers articles avec plotly express

[74]:
```
df_merge = df_merge.sort_values(by='ca_par_article', ascending=False)
```

[75]:
```
df_merge = df_merge.reset_index(drop=True)
```

[76]:
```
display(df_merge[['post_title', 'ca_par_article', 'price', 'total_sales']].
↪head(20))
```

```
                                          post_title  ca_par_article  price  \
0        Champagne Mailly Grand Cru Intemporelle 2010          6844.0   59.0
1      Champagne Egly-Ouriet Grand Cru Millésimé 2008          2475.0  225.0
2            François Baur Pinot Noir Schlittweg 2017          1549.4   12.7
3                 Albert Mann Pinot Noir Grand H 2017          1317.8   59.9
4             Argentine Mendoza Alamos Torrontes 2017          1232.1   11.1
5   Coteaux Champenois Egly-Ouriet Ambonnay Rouge …          1147.8  191.3
6           Champagne Egly-Ouriet Grand Cru Brut Rosé          1113.0   79.5
7               Agnès Levet Côte Rôtie Améthyste 2017           824.0   41.2
8   Domaine des Comtes Lafon Volnay 1er Cru Santen…           805.0  115.0
9   Champagne Agrapart &amp;amp; Fils Minéral Extra Br…           781.2   86.8
10  Domaine des Comtes Lafon Volnay 1er Cru Santen…           735.0  105.0
11                Camille Giroud Clos de Vougeot 2016           700.0  175.0
12             Champagne Gosset Célébris Vintage 2007           675.0  135.0
13  Champagne Agrapart &amp;amp; Fils L'Avizoise Extra…           672.0  112.0
14  David Duband Chambolle-Musigny 1er Cru Les Sen…           633.6  105.6
15  Cognac Frapin Château de Fontpinot 1989 20 Ans…           628.0  157.0
16  Bernard Baudry Chinon Rouge La Croix Boissée 2017           627.0   28.5
17  Champagne Larmandier-Bernier Grand Cru Vieille…           616.0   77.0
18  Champagne Larmandier-Bernier Grand Cru Les Che…           599.2   85.6
19  Domaine des Comtes Lafon Volnay 1er Cru Champa…           594.0   99.0

    total_sales
0         116.0
```

```
1          11.0
2         122.0
3          22.0
4         111.0
5           6.0
6          14.0
7          20.0
8           7.0
9           9.0
10          7.0
11          4.0
12          5.0
13          6.0
14          6.0
15          4.0
16         22.0
17          8.0
18          7.0
19          6.0
```

[77]:
```
fig = px.bar(
    df_merge.head(20),
    x='post_title',
    y='ca_par_article',
    title="Top 20 articles par chiffre d'affaires",
    labels={
        'post_title': 'Article',
        'ca_par_article': "Chiffre d'affaires (€)"
    }
)

fig.update_layout(xaxis_tickangle=-45, title_x=0.5)
fig.show()
```

4 Calculer le 20 / 80 en CA avec outliers

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

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

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

#Afficher la proportion que représente ce groupe d'articles dans le catalogue entier du site web

[78]:
```
df_merge['ca_par_article'] = df_merge['price'] * df_merge['total_sales']

# 2. Calcul de la part du CA
ca_total = df_merge['ca_par_article'].sum()
df_merge['part_ca'] = df_merge['ca_par_article'] / ca_total

# 3. Tri décroissant pour une analyse de type Pareto
df_merge = df_merge.sort_values(by='ca_par_article', ascending=False).
 ↪reset_index(drop=True)

# 4. Calcul de la somme cumulative de la part du CA
df_merge['part_ca_cumulee'] = df_merge['part_ca'].cumsum()
```

[79]:
```
display(df_merge[['post_title', 'ca_par_article', 'part_ca',␣
 ↪'part_ca_cumulee']].head(10))
```

	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	1549.4	
3	Albert Mann Pinot Noir Grand H 2017	1317.8	
4	Argentine Mendoza Alamos Torrontes 2017	1232.1	
5	Coteaux Champenois Egly-Ouriet Ambonnay Rouge …	1147.8	
6	Champagne Egly-Ouriet Grand Cru Brut Rosé	1113.0	
7	Agnès Levet Côte Rôtie Améthyste 2017	824.0	
8	Domaine des Comtes Lafon Volnay 1er Cru Santen…	805.0	
9	Champagne Agrapart & Fils Minéral Extra Br…	781.2	

	part_ca	part_ca_cumulee
0	0.044618	0.044618
1	0.016135	0.060753
2	0.010101	0.070854
3	0.008591	0.079445
4	0.008032	0.087477
5	0.007483	0.094960
6	0.007256	0.102216
7	0.005372	0.107588
8	0.005248	0.112836

```
9   0.005093          0.117928
```

[80]:
```
df_pareto_80 = df_merge[df_merge['part_ca_cumulee'] <= 0.8]
nb_articles_80 = df_pareto_80.shape[0]

print(f"Nombre d'articles représentant 80 % du CA : {nb_articles_80}")
```

```
Nombre d'articles représentant 80 % du CA : 420
```

[81]:
```
proportion_articles = nb_articles_80 / df_merge.shape[0]
print(f"Cela représente {proportion_articles:.2%} des articles du catalogue.")
```

```
Cela représente 58.82% des articles du catalogue.
```

5 Palmarès des articles en CA sans outliers

[82]:
```
import plotly.express as px

# 1. Créer une copie filtrée de df_merge sans les outliers IQR
df_merge_no_outliers = df_merge[(df_merge['price'] >= borne_inf) &␣
 ↪(df_merge['price'] <= borne_sup)].copy()

# 2. Réinitialiser l'index
df_merge_no_outliers = df_merge_no_outliers.reset_index(drop=True)

# 3. Calculer le chiffre d'affaires (CA) sans outliers
df_merge_no_outliers['ca_par_article'] = df_merge_no_outliers['price'] *␣
 ↪df_merge_no_outliers['total_sales']

# 4. Trier par chiffre d'affaires décroissant
df_merge_no_outliers = df_merge_no_outliers.sort_values(by='ca_par_article',␣
 ↪ascending=False)

# 5. Réinitialiser l'index après tri
df_merge_no_outliers = df_merge_no_outliers.reset_index(drop=True)

# 6. Afficher les 20 premiers articles
display(df_merge_no_outliers[['post_title', 'ca_par_article', 'price',␣
 ↪'total_sales']].head(20))

# 7. Graphique en barres du Top 20 CA sans outliers
fig = px.bar(
    df_merge_no_outliers.head(20),
    x='post_title',
    y='ca_par_article',
    title="Top 20 CA (hors outliers - IQR)",
    labels={
```

```
        'post_title': 'Article',
        'ca_par_article': "Chiffre d'affaires (€)"
    }
)

fig.update_layout(xaxis_tickangle=-45, title_x=0.5)
fig.show()
```

```
                                        post_title  ca_par_article  price  \
0        Champagne Mailly Grand Cru Intemporelle 2010          6844.0   59.0
1            François Baur Pinot Noir Schlittweg 2017          1549.4   12.7
2                 Albert Mann Pinot Noir Grand H 2017          1317.8   59.9
3              Argentine Mendoza Alamos Torrontes 2017          1232.1   11.1
4           Champagne Egly-Ouriet Grand Cru Brut Rosé          1113.0   79.5
5               Agnès Levet Côte Rôtie Améthyste 2017           824.0   41.2
6   Bernard Baudry Chinon Rouge La Croix Boissée 2017           627.0   28.5
7   Champagne Larmandier-Bernier Grand Cru Vieille…           616.0   77.0
8     Champagne Egly-Ouriet Grand Cru Extra Brut V.P.           556.5   79.5
9   Château de Meursault Beaune-Grèves Les Trois J…           537.6   67.2
10  Domaine Weinbach Riesling Grand Cru Schlossber…           507.2   63.4
11  Marcel Windholtz Eau de Vie de Kirsch d'Alsace…           499.2   62.4
12               Decelle-Villa Chorey-Lès-Beaune 2016           493.0   29.0
13          Cognac Frapin Château de Fontpinot XO 1/2           476.7   68.1
14                Domaine Peyre Rose Syrah Léone 2008           468.0   78.0
15                       Tempier Bandol Tourtine 2017           464.0   58.0
16  Marc Colin Et Fils Chassagne-Montrachet Blanc …           461.3   65.9
17    Albert Mann Pinot Noir Les Saintes Claires 2017           455.0   65.0
18    Château Jean Faure Saint-Emilion Grand Cru 2016           449.1   49.9
19       Clos du Mont-Olivet Châteauneuf-du-Pape 2007           438.4   54.8

    total_sales
0         116.0
1         122.0
2          22.0
3         111.0
4          14.0
5          20.0
6          22.0
7           8.0
8           7.0
9           8.0
10          8.0
11          8.0
12         17.0
13          7.0
14          6.0
15          8.0
```

```
16          7.0
17          7.0
18          9.0
19          8.0
```

6 Calculer le 20 / 80 en CA sans outliers

[83]:
```
df_merge_no_outliers['ca_par_article'] = df_merge_no_outliers['price'] *␣
↪df_merge_no_outliers['total_sales']

# Calcul de la part du CA dans le total
total_ca_no_outliers = df_merge_no_outliers['ca_par_article'].sum()
df_merge_no_outliers['part_ca'] = df_merge_no_outliers['ca_par_article'] /␣
↪total_ca_no_outliers

# Vérification rapide
display(df_merge_no_outliers[['post_title', 'ca_par_article', 'part_ca']].
↪head())
print(f"Total CA (sans outliers) : {total_ca_no_outliers:,.2f} €")
```

	post_title	ca_par_article	part_ca
0	Champagne Mailly Grand Cru Intemporelle 2010	6844.0	0.050036
1	François Baur Pinot Noir Schlittweg 2017	1549.4	0.011327
2	Albert Mann Pinot Noir Grand H 2017	1317.8	0.009634
3	Argentine Mendoza Alamos Torrontes 2017	1232.1	0.009008
4	Champagne Egly-Ouriet Grand Cru Brut Rosé	1113.0	0.008137

```
Total CA (sans outliers) : 136,782.30 €
```

[84]:
```
#Grâce aux deux colonnes créées précedemment, calculer le nombre d'articles␣
↪représentant 80% du CA

# Sélectionner les articles qui cumulent jusqu'à 80 % du chiffre d'affaires␣
↪(hors outliers)
```

```
df_pareto_80_no_outliers =␣
 ↪df_merge_no_outliers[df_merge_no_outliers['part_ca_cumulee'] <= 0.8]

# Calcul du nombre d'articles concernés
nb_articles_80_no_outliers = df_pareto_80_no_outliers.shape[0]

# Proportion d'articles dans le catalogue web sans outliers
proportion_articles_no_outliers = nb_articles_80_no_outliers /␣
 ↪df_merge_no_outliers.shape[0]

# Affichage du résultat
print(f"Nombre d'articles représentant 80 % du CA (hors outliers) :␣
 ↪{nb_articles_80_no_outliers}")
print(f"Cela représente {proportion_articles_no_outliers:.2%} des articles␣
 ↪restants (sans outliers)")
```

```
Nombre d'articles représentant 80 % du CA (hors outliers) : 394
Cela représente 57.69% des articles restants (sans outliers)
```

6.1 Analyse comparative du Pareto avec et sans outliers

Indicateur	Avec outliers	Sans outliers
Chiffre d'affaires total du site web	153 392,10 €	136 782,30 €
Nombre d'articles représentant 80 % du CA	420	394
Part du catalogue correspondant à ces articles	58,82 %	57,69 %

6.1.1 Interprétation

- Le retrait des outliers entraîne une **baisse modérée du chiffre d'affaires global** (−10,83 %), sans perturber significativement la répartition des ventes.
- La distribution reste **très concentrée** : environ **58 % des articles** génèrent **80 % du chiffre d'affaires**, qu'on inclue ou non les valeurs extrêmes.
- Cette stabilité indique une **structure de ventes robuste**, avec une faible dépendance à des produits atypiques ou extrêmes.

Etape 5.2 - Analyse des ventes en quantité

7 Palmarès des articles en quantité

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

#Réinitialiser l'index du dataset par un reset_index

#Afficher les 20 premiers articles en quantité

#Graphique en barre des 20 premiers articles avec plotly express

[85]:
```
df_merge[['post_title', 'total_sales']].head(20)
```

[85]:

	post_title	total_sales
0	Champagne Mailly Grand Cru Intemporelle 2010	116.0
1	Champagne Egly-Ouriet Grand Cru Millésimé 2008	11.0
2	François Baur Pinot Noir Schlittweg 2017	122.0
3	Albert Mann Pinot Noir Grand H 2017	22.0
4	Argentine Mendoza Alamos Torrontes 2017	111.0
5	Coteaux Champenois Egly-Ouriet Ambonnay Rouge …	6.0
6	Champagne Egly-Ouriet Grand Cru Brut Rosé	14.0
7	Agnès Levet Côte Rôtie Améthyste 2017	20.0
8	Domaine des Comtes Lafon Volnay 1er Cru Santen…	7.0
9	Champagne Agrapart & Fils Minéral Extra Br…	9.0
10	Domaine des Comtes Lafon Volnay 1er Cru Santen…	7.0
11	Camille Giroud Clos de Vougeot 2016	4.0
12	Champagne Gosset Célébris Vintage 2007	5.0
13	Champagne Agrapart & Fils L'Avizoise Extra…	6.0
14	David Duband Chambolle-Musigny 1er Cru Les Sen…	6.0
15	Cognac Frapin Château de Fontpinot 1989 20 Ans…	4.0
16	Bernard Baudry Chinon Rouge La Croix Boissée 2017	22.0
17	Champagne Larmandier-Bernier Grand Cru Vieille…	8.0
18	Champagne Larmandier-Bernier Grand Cru Les Che…	7.0
19	Domaine des Comtes Lafon Volnay 1er Cru Champa…	6.0

[86]:
```
df_merge = df_merge.sort_values(by='total_sales', ascending=False).
↪reset_index(drop=True)
```

[87]:
```
import plotly.express as px

# Graphique en barre des 20 premiers articles en quantité vendue
fig = px.bar(
    df_merge.head(20),
    x='post_title',
    y='total_sales',
    title="Top 20 articles par quantités vendues",
    labels={'post_title': 'Article', 'total_sales': 'Quantités vendues'}
)

fig.update_layout(xaxis_tickangle=-45, title_x=0.5)
fig.show()
```

8 Palmarès des articles en quantité sans outliers

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

#Réinitialiser l'index du dataset par un reset_index

#Afficher les 20 premiers articles en quantité

#Graphique en barre des 20 premiers articles avec plotly express

[88]:
```
df_merge_no_outliers[['post_title', 'total_sales']].head(20)
```

[88]:

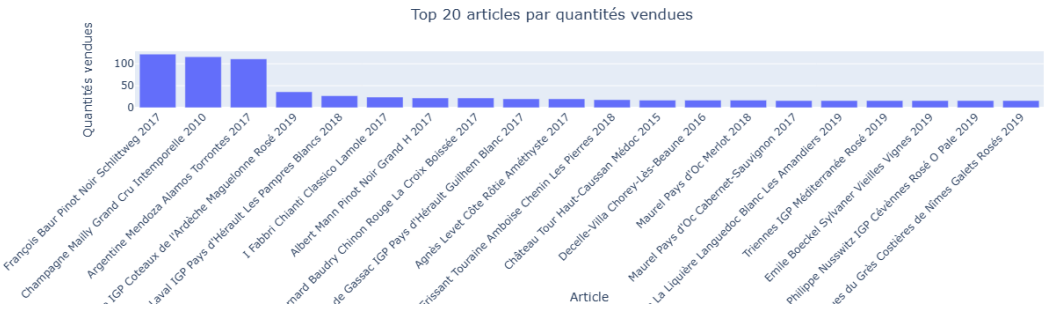
	post_title	total_sales
0	Champagne Mailly Grand Cru Intemporelle 2010	116.0
1	François Baur Pinot Noir Schlittweg 2017	122.0
2	Albert Mann Pinot Noir Grand H 2017	22.0
3	Argentine Mendoza Alamos Torrontes 2017	111.0
4	Champagne Egly-Ouriet Grand Cru Brut Rosé	14.0
5	Agnès Levet Côte Rôtie Améthyste 2017	20.0
6	Bernard Baudry Chinon Rouge La Croix Boissée 2017	22.0
7	Champagne Larmandier-Bernier Grand Cru Vieille…	8.0
8	Champagne Egly-Ouriet Grand Cru Extra Brut V.P.	7.0
9	Château de Meursault Beaune-Grèves Les Trois J…	8.0
10	Domaine Weinbach Riesling Grand Cru Schlossber…	8.0
11	Marcel Windholtz Eau de Vie de Kirsch d'Alsace…	8.0
12	Decelle-Villa Chorey-Lès-Beaune 2016	17.0
13	Cognac Frapin Château de Fontpinot XO 1/2	7.0
14	Domaine Peyre Rose Syrah Léone 2008	6.0
15	Tempier Bandol Tourtine 2017	8.0
16	Marc Colin Et Fils Chassagne-Montrachet Blanc …	7.0
17	Albert Mann Pinot Noir Les Saintes Claires 2017	7.0
18	Château Jean Faure Saint-Emilion Grand Cru 2016	9.0
19	Clos du Mont-Olivet Châteauneuf-du-Pape 2007	8.0

[89]:
```
df_merge_no_outliers = df_merge_no_outliers.sort_values(by='total_sales',␣
↪ascending=False).reset_index(drop=True)
```

[90]:
```
import plotly.express as px

# Graphique en barre des 20 premiers articles en quantité vendue
fig = px.bar(
    df_merge_no_outliers.head(20),
    x='post_title',
    y='total_sales',
    title="Top 20 articles par quantités vendues",
    labels={'post_title': 'Article', 'total_sales': 'Quantités vendues'}
)

fig.update_layout(xaxis_tickangle=-45, title_x=0.5)
fig.show()
```



8.1 Analyse comparative des quantités vendues (Top 20 articles)

Indicateur	Avec outliers	Sans outliers
1er article (plus vendu)	François Baur Pinot Noir Schlittweg 2017	François Baur Pinot Noir Schlittweg 2017
Quantité du top 1	122 unités	122 unités
2e article	Champagne Mailly Grand Cru 2010	Champagne Mailly Grand Cru 2010
Quantité du top 2	116 unités	116 unités
Nombre d'articles avec 16 ventes	6 articles	7 articles
Changements dans le Top 20	Quelques variations mineures	Triennes Rosé 2019 entre dans le classement

---

8.1.1 Interprétation

- Aucun changement dans le podium : les 3 articles les plus vendus restent identiques.
- Le retrait des outliers n'impacte quasiment pas le classement en quantités.

- La suppression des valeurs extrêmes concerne surtout les prix, pas les volumes de vente.
- La structure des quantités reste stable et indique que les produits les plus populaires en volume ne sont pas affectés par les outliers de prix.

9 Calculer le 20 / 80 en CA avec outliers

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

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

#Grâce aux deux colonnes créées précedemment, calculer le nombre d'articles représentant 80% des ventes en quantité

#Afficher la proportion que représente ce groupe d'articles dans le catalogue entier du site web

[91]:
```
# 1. Créer une colonne calculant la part en quantité de chaque ligne dans le dataset
df_merge['part_quantite'] = df_merge['total_sales'] / df_merge['total_sales'].sum()

# 2. Créer une colonne réalisant la somme cumulative des parts
df_merge = df_merge.sort_values(by='part_quantite', ascending=False).reset_index(drop=True)
df_merge['part_quantite_cumulee'] = df_merge['part_quantite'].cumsum()

# 3. Identifier le nombre d'articles représentant 80 % des ventes en quantité
df_80_quantite = df_merge[df_merge['part_quantite_cumulee'] <= 0.8]
nb_articles_80_quantite = df_80_quantite.shape[0]

# 4. Calculer la proportion que représente ce groupe dans le catalogue complet
proportion_articles_80_quantite = nb_articles_80_quantite / df_merge.shape[0]

# 5. Afficher les résultats
print(f" Nombre d'articles représentant 80 % des ventes (quantité, avec outliers) : {nb_articles_80_quantite}")
print(f" Cela représente {proportion_articles_80_quantite:.2%} du catalogue total (avec outliers)")
```

```
Nombre d'articles représentant 80 % des ventes (quantité, avec outliers) : 423
Cela représente 59.24% du catalogue total (avec outliers)
```

10 Calculer le 20 / 80 en CA sans outliers

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

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

#Grâce aux deux colonnes créées précedemment, calculer le nombre d'articles représentant 80% des ventes en quantité

#Afficher la proportion que représente ce groupe d'articles dans le catalogue entier du site web

[92]:
```
# 1. Créer une colonne calculant la part en quantité de chaque ligne dans le␣
 ↪dataset sans outliers
df_merge_no_outliers['part_quantite'] = df_merge_no_outliers['total_sales'] /␣
 ↪df_merge_no_outliers['total_sales'].sum()

# 2. Trier et créer la colonne de somme cumulative
df_merge_no_outliers = df_merge_no_outliers.sort_values(by='part_quantite',␣
 ↪ascending=False).reset_index(drop=True)
df_merge_no_outliers['part_quantite_cumulee'] =␣
 ↪df_merge_no_outliers['part_quantite'].cumsum()

# 3. Identifier le nombre d'articles représentant 80 % des ventes en quantité␣
 ↪(hors outliers)
df_80_quantite_no_outliers =␣
 ↪df_merge_no_outliers[df_merge_no_outliers['part_quantite_cumulee'] <= 0.8]
nb_articles_80_quantite_no_outliers = df_80_quantite_no_outliers.shape[0]

# 4. Calculer la proportion que cela représente dans le catalogue sans outliers
proportion_articles_80_quantite_no_outliers =␣
 ↪nb_articles_80_quantite_no_outliers / df_merge_no_outliers.shape[0]

# 5. Afficher les résultats
print(f" Nombre d'articles représentant 80 % des ventes (quantité, sans␣
 ↪outliers) : {nb_articles_80_quantite_no_outliers}")
print(f" Cela représente {proportion_articles_80_quantite_no_outliers:.2%} du␣
 ↪catalogue (sans outliers)")
```

```
Nombre d'articles représentant 80 % des ventes (quantité, sans outliers) : 409
Cela représente 59.88% du catalogue (sans outliers)
```

10.0.1 Analyse comparative – Règle des 20/80 sur les quantités vendues

Indicateur	Avec outliers	Sans outliers
Nombre d'articles représentant 80 % des ventes	423	409
Part du catalogue concernée	59,24 %	59,88 %

10.0.2 Interprétation

- Le retrait des valeurs aberrantes (outliers) n'a qu'un faible impact sur la structure des ventes en quantité.
- Environ 60 % des références suffisent à générer 80 % des ventes en volume, que les données soient brutes ou nettoyées.

- Cela suggère une répartition assez homogène des ventes entre les produits, sans forte concentration sur quelques références.

Etape 5.3 - Analyse des stocks

11 Calculer le nombre de mois de stock

#Import de numpy

#Création de la colonne Rotation de stock

#Remplacement des “inf” par 0

#Effectuer le tri dans l’ordre décroissant du nombre de mois de stock dans le dataset df_merge

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

[93]:
```
import numpy as np
import plotly.express as px

# 1. Calcul du nombre de mois de stock
df_merge['rotation_stock'] = df_merge['stock_quantity'] /␣
 ↪df_merge['total_sales']

# 2. Remplacer les infinis et valeurs aberrantes (division par zéro ou NaN)
df_merge['rotation_stock'].replace([np.inf, -np.inf], np.nan, inplace=True)
df_merge['rotation_stock'].fillna(0, inplace=True)

# 3. Trier les produits avec la plus faible rotation (ceux qui écoulent␣
 ↪lentement leur stock)
df_rotation_sorted = df_merge.sort_values(by='rotation_stock', ascending=False).
 ↪head(20)

# 4. Visualisation avec Plotly
fig = px.bar(
    df_rotation_sorted,
    x='post_title',
    y='rotation_stock',
    title=' Flop 20 des articles en rotation de stock avec outliers (mois␣
 ↪estimés)',
    labels={'post_title': 'Article', 'rotation_stock': 'Mois de stock␣
 ↪(estimé)'},
    text='rotation_stock'
)

# 5. Mise en forme graphique
fig.update_traces(texttemplate='%{text:.1f}', textposition='outside')
fig.update_layout(
    xaxis_tickangle=-45,
```

```
    xaxis_tickfont=dict(size=10),
    yaxis_title='Nombre de mois estimés pour écouler le stock',
    title_x=0.5,
    margin=dict(l=40, r=40, t=60, b=120),
    height=600
)

fig.show()
```

[94]:
```
# Calcul de la moyenne des mois de stock pour le Flop 20 (avec outliers)
moyenne_rotation_flop20_outliers = df_rotation_sorted['rotation_stock'].mean()

print(f"Moyenne du nombre de mois de stock estimés (Flop 20 avec outliers) :␣
 ↪{moyenne_rotation_flop20_outliers:.2f} mois")
```

```
Moyenne du nombre de mois de stock estimés (Flop 20 avec outliers) : 20.51 mois
```

[95]:
```
import numpy as np
import plotly.express as px

# 1. Calcul du nombre de mois de stock
df_merge_no_outliers['rotation_stock'] = df_merge_no_outliers['stock_quantity']␣
 ↪/ df_merge_no_outliers['total_sales']

# 2. Remplacer les infinis et valeurs aberrantes (division par zéro ou NaN)
df_merge_no_outliers['rotation_stock'].replace([np.inf, -np.inf], np.nan,␣
 ↪inplace=True)
df_merge_no_outliers['rotation_stock'].fillna(0, inplace=True)

# 3. Trier les produits avec la plus faible rotation (ceux qui écoulent␣
 ↪lentement leur stock)
df_rotation_sorted = df_merge_no_outliers.sort_values(by='rotation_stock',␣
 ↪ascending=False).head(20)

# 4. Visualisation avec Plotly
fig = px.bar(
    df_rotation_sorted,
    x='post_title',
    y='rotation_stock',
    title=' Flop 20 des articles en rotation de stock hors outliers (mois␣
 ↪estimés)',
    labels={'post_title': 'Article', 'rotation_stock': 'Mois de stock␣
 ↪(estimé)'},
    text='rotation_stock'
)

# 5. Mise en forme graphique
```

```
fig.update_traces(texttemplate='%{text:.1f}', textposition='outside')
fig.update_layout(
    xaxis_tickangle=-45,
    xaxis_tickfont=dict(size=10),
    yaxis_title='Nombre de mois estimés pour écouler le stock',
    title_x=0.5,
    margin=dict(l=40, r=40, t=60, b=120),
    height=600
)

fig.show()
```

[96]:
```
# Calcul de la moyenne des mois de stock pour le Flop 20
moyenne_rotation_flop20 = df_rotation_sorted['rotation_stock'].mean()

print(f"Moyenne du nombre de mois de stock estimés (Flop 20 hors outliers) :␣
 ↪{moyenne_rotation_flop20:.2f} mois")
```

```
Moyenne du nombre de mois de stock estimés (Flop 20 hors outliers) : 17.82 mois
```

11.0.1 Analyse comparative – Flop 20 en rotation de stock

Indicateur	Avec Outliers	Sans Outliers
Nombre de mois de stock le plus élevé	31,5	31,3
Classement suivant	25	17
Moyenne de mois de rotation	20,51	17,82

11.0.2 Interprétation

- Le retrait des **valeurs aberrantes** (outliers) n'a **qu'un faible impact** sur la structure des ventes en quantité.
- Environ **60 % des références** suffisent à générer **80 % des ventes** en volume, que les données soient brutes ou nettoyées.
- Cela suggère une **répartition assez homogène des ventes** entre les produits, **sans forte concentration** sur quelques références.

Avec outliers :

- Le produit le plus lent à écouler affiche 31,5 mois de stock.
- Plusieurs articles dépassent les 25 mois, montrant des stocks dormants potentiellement surestimés.
- Le graphique montre une forte concentration de cas critiques dans les premiers rangs.

Sans outliers :

- L'article en tête reste proche avec 31,3 mois, ce qui confirme qu'il est structurellement lent à vendre.
- Dès le 10e produit, la rotation chute sous 17 mois, avec des valeurs plus homogènes.
- L'écart type est plus faible, ce qui indique une meilleure cohérence des données sans valeurs extrêmes.

Conclusion : L'exclusion des outliers permet une meilleure lisibilité de la rotation des stocks. Elle met en évidence les produits structurellement en difficulté tout en écartant les cas extrêmes qui peuvent fausser l'analyse.

12 Valorisation des stocks en euros

#Création de la colonne Valorisation des stocks en euros

#Calculer la somme de la colonne "Valorisation_stock_euros"

[97]:
```
df_merge['valorisation_stock'] = df_merge['stock_quantity'] *␣
 ↪df_merge['purchase_price']

total_valorisation_stock = df_merge['valorisation_stock'].sum()

print(f"Valorisation totale du stock (en €) : {total_valorisation_stock:,.2f}␣
 ↪€")
```

```
Valorisation totale du stock (en €) : 297,137.99 €
```

13 Valorisation des stocks hors outliers

#Calculer la somme de la colonne « Valorisation_stock_euros »

[98]:
```
df_merge_no_outliers['valorisation_stock'] =␣
 ↪df_merge_no_outliers['stock_quantity'] *␣
 ↪df_merge_no_outliers['purchase_price']
total_valorisation_stock_no_outliers =␣
 ↪df_merge_no_outliers['valorisation_stock'].sum()

print(f"Valorisation totale du stock hors outliers :␣
 ↪{total_valorisation_stock_no_outliers:,.2f} €")
```

```
Valorisation totale du stock hors outliers : 235,539.56 €
```

[99]:
```
import pandas as pd
import plotly.graph_objects as go

# Données de valorisation avec et sans outliers
valorisation_avec_outliers = 297137.99
valorisation_sans_outliers = 235539.56
```

```
# Création du graphique comparatif
fig = go.Figure(data=[
    go.Bar(name='Avec outliers', x=['Valorisation du stock'],␣
 ↪y=[valorisation_avec_outliers], text=[f'{valorisation_avec_outliers:,.2f}␣
 ↪€'], textposition='outside'),
    go.Bar(name='Sans outliers', x=['Valorisation du stock'],␣
 ↪y=[valorisation_sans_outliers], text=[f'{valorisation_sans_outliers:,.2f}␣
 ↪€'], textposition='outside')
])

# Mise en forme
fig.update_layout(
    title='Comparaison de la valorisation du stock (avec vs sans outliers)',
    yaxis_title='Montant (€)',
    xaxis_title='Type de valorisation',
    barmode='group',
    title_x=0.5,
    height=500
)

fig.show()
```

13.1 Analyse comparative – Valorisation des stocks (avec et sans outliers)

- La valorisation totale du stock s'élève à 297 137,99 € en tenant compte de l'ensemble des données (avec outliers).
- En supprimant les outliers, la valorisation descend à 235 539,56 €, soit une baisse de 20,72 %.

Interprétation : - Les valeurs extrêmes (outliers) ont un impact significatif sur l'estimation financière des stocks, en surévaluant le total de près de 61 600 €. - Cela montre l'importance de nettoyer les données pour obtenir une image plus fiable et réaliste des immobilisations en stock. stock.

14 Valorisation du nombre de produits en stock avec vs sans outliers

#Calculer la somme de la colonne stock quantity

[ ]:
```
# Calculer la somme de la colonne stock_quantity
total_produits_en_stock = df_merge['stock_quantity'].sum()
print(f"Nombre total de produits en stock (avec outliers) :␣
 ↪{total_produits_en_stock}")
```

[ ]:
```
# Nombre de produits en stock sans outliers
total_produits_en_stock_no_outliers = df_merge_no_outliers['stock_quantity'].
 ↪sum()
print(f"Nombre total de produits en stock (hors outliers) :␣
 ↪{total_produits_en_stock_no_outliers}")
```

[ ]:
```python
import pandas as pd
import plotly.express as px

# Données comparatives
data = {
    'Type': ['Avec outliers', 'Sans outliers'],
    'Nombre de produits en stock': [17786, 16898],
    'Couleur': ['#636EFA', '#EF553B']  # Couleurs par défaut de go.Bar pour␣
 ↪continuité visuelle
}

df_comparatif = pd.DataFrame(data)

# Création du graphique avec les bonnes couleurs
fig = px.bar(
    df_comparatif,
    x='Type',
    y='Nombre de produits en stock',
    text='Nombre de produits en stock',
    title='Comparaison du nombre total de produits en stock',
    labels={'Type': 'Données', 'Nombre de produits en stock': 'Quantité␣
 ↪totale'},
    color='Couleur',
    color_discrete_map='identity'  # Utilise les couleurs de la colonne␣
 ↪'Couleur'
)

# Mise en forme
fig.update_traces(texttemplate='%{text}', textposition='outside')
fig.update_layout(
    showlegend=False,
    yaxis_title='Nombre total de produits en stock',
    xaxis_title='',
    title_x=0.5,
    height=500
)

fig.show()
```

14.1 Analyse comparative du nombre total de produits en stock

- **Nombre total de produits en stock (avec outliers)** : 17 786
- **Nombre total de produits en stock (hors outliers)** : 16 898

14.1.1 Interprétation

L'exclusion des outliers entraîne une baisse du nombre total de produits en stock de **888 unités**, soit une réduction d'environ **5 %**.
Cela suggère que les produits identifiés comme outliers représentent une faible part du stock total, mais leur présence peut fausser certaines analyses de rotation ou de valorisation du stock.

Etape 5.4 - Analyse du taux de marge

15 Analyse du taux de marge

#Création de la colonne Prix HT

#Création de la colonne Taux de marge

#Afficher le prix minimum de la colonne "taux_marge"

#Afficher le prix maximum de la colonne "taux_marge"

[ ]:
```
# 1. Création de la colonne Prix HT à partir de la colonne 'price' (TTC → HT␣
 ↪avec TVA 20 %)
df_merge['prix_ht'] = df_merge['price'] / 1.20

# 2. Création de la colonne Taux de marge
df_merge['taux_marge'] = (df_merge['prix_ht'] - df_merge['purchase_price']) /␣
 ↪df_merge['purchase_price']

# 3. Affichage du taux de marge minimum
taux_min = df_merge['taux_marge'].min()
print(f"Taux de marge minimum : {taux_min:.2%}")

# 4. Affichage du taux de marge maximum
taux_max = df_merge['taux_marge'].max()
print(f"Taux de marge maximum : {taux_max:.2%}")
```

#Affichage de la ligne avec un taux de marge inférieur à 0

[ ]:
```
df_merge[df_merge['taux_marge'] < 0]
```

#Création d'un dataframe avec les taux positifs

#Afficher le prix minimum de la colonne "taux_marge"

#Afficher le prix maximum de la colonne "taux_marge"

[ ]:
```
# Création d'un DataFrame avec les taux de marge positifs
df_marge_positive = df_merge[df_merge['taux_marge'] > 0]

# Afficher le taux de marge minimum (positif)
taux_min = df_marge_positive['taux_marge'].min()
print(f"Taux de marge minimum (positif) : {taux_min:.2%}")
```

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

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

#Affichage dans un graphique du taux de marge par type de produit

[ ]:
```
import pandas as pd
import plotly.express as px

# 1. Calcul du taux de marge moyen par type de produit
df_marge_moyenne = df_merge.groupby('product_type',␣
 ↪as_index=False)['taux_marge'].mean()

# 2. Création du graphique avec couleur en dégradé selon le taux de marge
fig = px.bar(
    df_marge_moyenne,
    x='product_type',
    y='taux_marge',
    color='taux_marge',  # ajout pour générer le dégradé
    color_continuous_scale='RdYlGn',  # rouge → jaune → vert
    text='taux_marge',
    title='Taux de marge moyen par type de produit',
    labels={'product_type': 'Type de produit', 'taux_marge': 'Taux de marge␣
 ↪moyen'},
)

# Mise en forme
fig.update_traces(texttemplate='%{text:.2%}', textposition='outside')
fig.update_layout(
    xaxis_tickangle=-45,
    title_x=0.5,
    height=500,
    coloraxis_colorbar=dict(title='Taux de marge')  # légende du dégradé
)

fig.show()
```

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

16 Analyse des corrélations

#Importation de Seaborn

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

#On peut également créer un mask pour n'afficher qu'une demi heatmap

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 4))
sns.heatmap(df_merge[['stock_quantity', 'total_sales', 'price']].corr(),␣
 ↪annot=True, cmap='coolwarm')
plt.title('Corrélation stock / ventes / prix')
plt.show()
```

[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Sélection des variables à corréler
variables = ['stock_quantity', 'total_sales', 'price']
corr_matrix = df_merge[variables].corr()

# Création d'un masque pour cacher la moitié supérieure
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Création de la figure
plt.figure(figsize=(6, 4))
sns.heatmap(
    corr_matrix,
    mask=mask,
    annot=True,
    cmap='coolwarm',
    fmt=".2f",
    square=True,
    linewidths=.5,
    cbar_kws={"shrink": .6}
)

plt.title("Corrélation (triangle inférieur seulement)")
plt.show()
```

#Que peut-on conclure des corrélations ?

16.1 Analyse des corrélations : Stock, Ventes et Prix

16.1.1 1. Corrélation entre `stock_quantity` et `total_sales` : +0.11

- **Interprétation** : Très faible corrélation positive.
- **Conclusion** : Le fait d'avoir plus de stock n'est que **légèrement associé** à un plus grand volume de ventes.
- Cela peut s'expliquer par des stocks dormants (produits non écoulés malgré un stock important).

16.1.2 2. Corrélation entre `price` et `total_sales` : -0.27

- **Interprétation** : Corrélation **négative modérée**.
- **Conclusion** : Les produits **plus chers** ont tendance à être **moins vendus**. Cela confirme une **sensibilité au prix** des consommateurs.

16.1.3 3. Corrélation entre `price` et `stock_quantity` : -0.05

- **Interprétation** : Corrélation **quasi nulle**.
- **Conclusion** : Le **prix d'un produit n'influence pas significativement** la quantité de stock disponible. Cela peut refléter une politique d'approvisionnement indépendante du positionnement tarifaire.

16.1.4 Conclusion globale :

- Il n'y a **aucune forte corrélation** entre les trois variables.
- Les seules tendances notables sont :
  - Une **relation négative modérée** entre le **prix** et les **ventes**.
  - Une **légère corrélation positive** entre le **stock** et les **ventes**.
- Cela invite à **explorer d'autres facteurs explicatifs** (catégorie de produit, saisonnalité, promotions…).

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

#Mettre le dataset df_merge sur un fichier Excel

#Cette étape peut être utile pour partager le résultat du dataset obtenu avec les équipes.

[ ]:
```
# Export de df_merge (avec outliers)
df_merge.to_excel('df_merge_avec_outliers.xlsx', index=False)

# Export de df_merge_no_outliers (sans outliers)
df_merge_no_outliers.to_excel('df_merge_sans_outliers.xlsx', index=False)

print("Exports terminés :")
print("- df_merge_avec_outliers.xlsx")
print("- df_merge_sans_outliers.xlsx")
```

[ ]:
```
import zipfile

# Liste des fichiers à inclure dans l'archive
fichiers = ['df_merge_avec_outliers.xlsx', 'df_merge_sans_outliers.xlsx']

# Création de l'archive zip
with zipfile.ZipFile("exports_merge.zip", mode="w") as archive:
    for fichier in fichiers:
        archive.write(fichier)

print("  Archive créée : exports_merge.zip")
```

[ ]:
```
from IPython.display import FileLink
FileLink("exports_merge.zip")
```