BottleNeck

Résumé de la demande :

La mission consiste en trois étapes principales :

1. **Rapprochement des exports :**
   - Rapprocher les données de l'ERP (références produit, prix, stock) avec celles de la boutique en ligne (nom, description, ventes).
   - Utiliser un tableau Excel créé par Sylvie pour établir le lien entre les références du produit dans l'ERP (product_id) et celles de la boutique en ligne (SKU).
2. **Calcul du chiffre d'affaires :**
   - Une fois le rapprochement effectué, fournir le chiffre d'affaires par produit.
   - Calculer le total du chiffre d'affaires réalisé en ligne.
3. **Analyse des prix des produits :**
   - Détecter d'éventuelles valeurs aberrantes dans les prix des produits.
   - Liste des valeurs aberrantes et représentation graphique pour une meilleure lisibilité.

Les exports nécessaires ainsi qu'un tableau Excel pour le lien entre les références seront fournis. La présentation des résultats se fera lors de la prochaine réunion de COPIL, pouvant être sous forme de présentation ou d'un notebook utilisant R ou Python.

I] Installation

In [ ]:
```
#Importations des librairies nécessaires
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [ ]:
```
#chemin_fichier1_csv = 'C:\Users\Rorsharks\Desktop\P5\Fichier_erp.csv'

# Chargez les données depuis le fichier CSV dans un DataFrame
stock = pd.read_csv('C:\\Users\\Rorsharks\\Desktop\\P5\\Fichier_erp.csv', sep=';')

produits = pd.read_csv('C:\\Users\\Rorsharks\\Desktop\\P5\\Fichier_web.csv', sep=';

liaison = pd.read_csv('C:\\Users\\Rorsharks\\Desktop\\P5\\Fichier_liaison.csv', sep

# Affichez les premières lignes du DataFrame
display(stock.head())
```

```
display(produits.head())

display(liaison.head())
```

	product_id	onsale_web	price	stock_quantity	stock_status
0	3847	1	24,2	0	outofstock
1	3849	1	34,3	0	outofstock
2	3850	1	20,8	0	outofstock
3	4032	1	14,1	0	outofstock
4	4039	1	46	0	outofstock

	sku	virtual	downloadable	rating_count	average_rating	total_sales	tax_status	tax_cla
0	16004	0	0	0	0.0	5.0	NaN	Na
1	NaN	0	0	0	NaN	NaN	NaN	Na
2	15075	0	0	0	0.0	3.0	taxable	Na
3	16209	0	0	0	0.0	6.0	taxable	Na
4	15763	0	0	0	0.0	1.0	NaN	Na

5 rows × 28 columns

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

Il semble que certaines données soient marquées comme `NaN` (Not a Number), tandis que d'autres colonnes affichent des valeurs égales à zéro, notamment pour les colonnes "forerp" et "web".

Les colonnes suivantes présentent des situations particulières :**

- Downloadable**: Il est possible qu'aucun produit n'ait été téléchargé, expliquant ainsi l'absence de données dans cette colonne.
- Rating_count**: Certains produits peuvent ne pas avoir reçu de notation sur le site web, se traduisant par une valeur nulle dans cette colonne. Une autre possibilité est qu'ils aient reçu une notation de zéro.
- Post_parent**: Il est envisageable que certains éléments ne possèdent pas de poste parent ou ne soient pas associés à un système post_parent.
- Post_mime_type**: Tous les produits ne contiennent pas nécessairement de photo sur le site web, ce qui pourrait expliquer des valeurs vides ou spécifiques dans cette colonne.
- Comment_count**: Il est envisageable que certains produits n'aient reçu aucun commentaire, d'où la présence de valeurs nulles dans cette colonne.

I.I]Vérification des fichiers

In [ ]:
```
# Charger le fichier CSV dans un DataFrame
stock = pd.read_csv("fichier_erp.csv", delimiter=';')

# Vérifier les doublons
doublons = stock[stock.duplicated()]
if not doublons.empty:
    print("Doublons détectés:")
    print(doublons)
else:
    print("Aucun doublon détecté.")

# Vérifier les valeurs NaN
valeurs_nan = stock[stock.isna().any(axis=1)]
if not valeurs_nan.empty:
    print("Valeurs NaN détectées:")
    print(valeurs_nan)
else:
    print("Aucune valeur NaN détectée.")

# Afficher des statistiques générales sur le DataFrame (en excluant la colonne prod
print("\nStatistiques générales (hors product_id):")
display(stock.drop('product_id', axis=1).describe())

print("\nTypes de données par colonne:")
print(stock.dtypes)
stock.shape
```

```
Aucun doublon détecté.
Aucune valeur NaN détectée.

Statistiques générales (hors product_id):
```

	onsale_web	stock_quantity
count	825.000000	825.000000
mean	0.869091	26.583030
std	0.337506	45.875948
min	0.000000	-1.000000
25%	1.000000	1.000000
50%	1.000000	11.000000
75%	1.000000	34.000000
max	1.000000	578.000000

```
Types de données par colonne:
product_id         int64
onsale_web         int64
price             object
stock_quantity     int64
stock_status      object
dtype: object
```

Out[ ]: (825, 5)

Analyse statistique préliminaire de la DataFrame avant le nettoyage : Cette étape permet d'explorer les données, de détecter d'éventuels problèmes, de planifier le nettoyage, d'orienter le processus, et de se préparer à l'analyse ultérieure.

Résumé des résultats :

- Aucun doublon détecté, aucune valeur NaN présente.

Colonnes clés :

`onsale_web` :

- La plupart des produits sont disponibles en ligne (75e percentile à 1), avec une légère variabilité (écart-type d'environ 0.34).

`stock_quantity` :

- Une variabilité significative dans les quantités de stock (écart-type d'environ 45.88), allant de -1 à 578.
- La colonne `price` est de type objet, suggérant peut-être une conversion en type numérique.

In [ ]:
```
# Vérifier du fichier_liaison

# Charger le fichier CSV dans un DataFrame (remplacez "web.csv" par le chemin de vo
liaison = pd.read_csv("fichier_liaison.csv", delimiter=';')
```

```
# Vérifier les doublons
doublons = liaison[liaison.duplicated()]
if not doublons.empty:
    print("Doublons détectés:")
    print(doublons)
else:
    print("Aucun doublon détecté.")

# Vérifier les valeurs NaN
valeurs_nan = liaison[liaison.isna().any(axis=1)]
if not valeurs_nan.empty:
    print("Valeurs NaN détectées:")
    print(valeurs_nan)
else:
    print("Aucune valeur NaN détectée.")

# Afficher des statistiques générales sur le DataFrame (en excluant les colonnes no
print("\nStatistiques générales:")
display(liaison.describe(include='all'))

# Afficher les types de données par colonne
print("\nTypes de données par colonne:")
print(liaison.dtypes)

# Afficher les premières lignes du DataFrame
print("\nPremières lignes du DataFrame:")
print(liaison.head())
```

```
Aucun doublon détecté.
Valeurs NaN détectées:
     product_id id_web
19         4055    NaN
49         4090    NaN
50         4092    NaN
119        4195    NaN
131        4209    NaN
..          ...    ...
817        7196    NaN
818        7200    NaN
819        7201    NaN
820        7203    NaN
821        7204    NaN

[91 rows x 2 columns]

Statistiques générales:
```

	product_id	id_web
count	825.000000	734
unique	NaN	734
top	NaN	15298
freq	NaN	1
mean	5162.597576	NaN
std	902.644635	NaN
min	3847.000000	NaN
25%	4348.000000	NaN
50%	4907.000000	NaN
75%	5805.000000	NaN
max	7338.000000	NaN

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

Premières lignes du DataFrame:
   product_id id_web
0        3847  15298
1        3849  15296
2        3850  15300
3        4032  19814
4        4039  19815
```

In [ ]:
```
# on créé une nouvelle colonne 'sku' égale à la colonne 'id_web', qu'on supprime en
liaison['sku'] = liaison['id_web']
liaison = liaison.drop(columns = ['id_web'])

# on affiche les premières lignes
liaison.head()
```

Out[ ]:

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

Résumé des Résultats :

Pour la colonne **product_id :**

- Pas de doublons.
- Valeur la plus fréquente : Manquante (NaN) avec une fréquence de 1.
- Moyenne : Environ 5162.60.
- Écart-type : Environ 902.64, indiquant une variabilité autour de la moyenne.
- Valeur minimale : 3847, valeur maximale : 7338.

Pour la colonne **id_web :**

- Valeur la plus fréquente : 15298, fréquence : 1.
- Note : Des valeurs manquantes (NaN) sont présentes dans cette colonne, suggérant un traitement nécessaire.

Création d'une nouvelle colonne 'sku' dans le DataFrame 'liaison', en copiant les valeurs de la colonne 'id_web', puis supprime la colonne 'id_web', réorganisant ainsi la structure du DataFrame en remplaçant 'id_web' par 'sku'.

In [ ]:
```
# Vérifier du fichier_web

# Charger le fichier CSV dans un DataFrame (remplacez "web.csv" par le chemin de vo
produits = pd.read_csv("fichier_web.csv", encoding='ISO-8859-1', on_bad_lines='skip

# Vérifier les doublons
doublons = produits[produits.duplicated()]
if not doublons.empty:
    print("Doublons détectés:")
    print(doublons)
else:
    print("Aucun doublon détecté.")

# Vérifier les valeurs NaN
valeurs_nan = produits[produits.isna().any(axis=1)]
if not valeurs_nan.empty:
    print("Valeurs NaN détectées:")
    print(valeurs_nan)
else:
    print("Aucune valeur NaN détectée.")


# Afficher des statistiques générales sur le DataFrame
print("\nStatistiques générales:")
display(produits.describe())
produits.shape
```

```
Doublons détectés:
      sku  virtual  downloadable  rating_count  average_rating  total_sales  \
17    NaN        0             0             0             NaN          NaN
82    NaN        0             0             0             NaN          NaN
97    NaN        0             0             0             NaN          NaN
121   NaN        0             0             0             NaN          NaN
126   NaN        0             0             0             NaN          NaN
...   ...      ...           ...           ...             ...          ...
1378  NaN        0             0             0             NaN          NaN
1421  NaN        0             0             0             NaN          NaN
1448  NaN        0             0             0             NaN          NaN
1450  NaN        0             0             0             NaN          NaN
1496  NaN        0             0             0             NaN          NaN

     tax_status  tax_class  post_author post_date  ... post_name  \
17          NaN        NaN          NaN       NaN  ...       NaN
82          NaN        NaN          NaN       NaN  ...       NaN
97          NaN        NaN          NaN       NaN  ...       NaN
121         NaN        NaN          NaN       NaN  ...       NaN
126         NaN        NaN          NaN       NaN  ...       NaN
...         ...        ...          ...       ...  ...       ...
1378        NaN        NaN          NaN       NaN  ...       NaN
1421        NaN        NaN          NaN       NaN  ...       NaN
1448        NaN        NaN          NaN       NaN  ...       NaN
1450        NaN        NaN          NaN       NaN  ...       NaN
1496        NaN        NaN          NaN       NaN  ...       NaN

      post_modified post_modified_gmt post_content_filtered post_parent guid  \
17              NaN               NaN                   NaN         NaN  NaN
82              NaN               NaN                   NaN         NaN  NaN
97              NaN               NaN                   NaN         NaN  NaN
121             NaN               NaN                   NaN         NaN  NaN
126             NaN               NaN                   NaN         NaN  NaN
...             ...               ...                   ...         ...  ...
1378            NaN               NaN                   NaN         NaN  NaN
1421            NaN               NaN                   NaN         NaN  NaN
1448            NaN               NaN                   NaN         NaN  NaN
1450            NaN               NaN                   NaN         NaN  NaN
1496            NaN               NaN                   NaN         NaN  NaN

     menu_order  post_type post_mime_type comment_count
17          NaN        NaN            NaN           NaN
82          NaN        NaN            NaN           NaN
97          NaN        NaN            NaN           NaN
121         NaN        NaN            NaN           NaN
126         NaN        NaN            NaN           NaN
...         ...        ...            ...           ...
1378        NaN        NaN            NaN           NaN
1421        NaN        NaN            NaN           NaN
1448        NaN        NaN            NaN           NaN
1450        NaN        NaN            NaN           NaN
1496        NaN        NaN            NaN           NaN

[82 rows x 28 columns]
Valeurs NaN détectées:
       sku  virtual  downloadable  rating_count  average_rating  total_sales  \
```

```
0     16004        0             0             0             0.0          5.0
1       NaN        0             0             0             NaN          NaN
2     15075        0             0             0             0.0          3.0
3     16209        0             0             0             0.0          6.0
4     15763        0             0             0             0.0          1.0
...     ...      ...           ...           ...           ...          ...
1508  12881        0             0             0             0.0          2.0
1509  15663        0             0             0             0.0          3.0
1510  15910        0             0             0             0.0          0.0
1511     38        0             0             0             0.0         38.0
1512  13599        0             0             0             0.0          1.0

     tax_status  tax_class  post_author            post_date  ...  \
0           NaN        NaN           2.0  2018-06-07 16:27:25  ...
1           NaN        NaN           NaN                  NaN  ...
2       taxable        NaN           2.0  2018-02-14 15:39:43  ...
3       taxable        NaN           2.0  2018-02-14 17:15:31  ...
4           NaN        NaN           2.0  2020-01-25 14:08:16  ...
...         ...        ...           ...                  ...  ...
1508        NaN        NaN           2.0  2019-03-28 15:25:14  ...
1509    taxable        NaN           2.0  2018-02-27 10:27:01  ...
1510    taxable        NaN           2.0  2019-03-28 10:59:43  ...
1511        NaN        NaN           2.0  2018-04-18 12:25:58  ...
1512        NaN        NaN           2.0  2018-03-01 14:12:39  ...

                                              post_name        post_modified  \
0                       chateau-du-couvent-pomerol-2017  2020-08-25 18:35:02
1                                                   NaN                  NaN
2                     pares-balta-penedes-indigena-2017  2020-08-20 15:35:02
3                        maurel-cabardes-tradition-2017  2020-08-05 18:05:03
4     domaine-de-la-monardiere-vacqueyras-les-vieill...  2020-08-21 11:35:02
...                                                 ...                  ...
1508                montbourgeau-etoile-vin-jaune-2009  2019-12-30 10:30:01
1509  chermette-domaine-du-vissoux-brouilly-pierreux...  2020-08-01 09:35:02
1510     thevenet-quintaine-vire-clesse-la-bongran-2015  2020-08-14 10:45:02
1511         emile-boeckel-cremant-brut-blanc-de-blancs  2020-08-27 17:15:03
1512     champagne-mailly-grand-cru-blanc-de-pinot-noir  2020-08-26 18:05:02

        post_modified_gmt post_content_filtered post_parent  \
0     2020-08-25 16:35:02                   NaN         0.0
1                     NaN                   NaN         NaN
2     2020-08-20 13:35:02                   NaN         0.0
3     2020-08-05 16:05:03                   NaN         0.0
4     2020-08-21 09:35:02                   NaN         0.0
...                   ...                   ...         ...
1508  2019-12-30 09:30:01                   NaN         0.0
1509  2020-08-01 07:35:02                   NaN         0.0
1510  2020-08-14 08:45:02                   NaN         0.0
1511  2020-08-27 15:15:03                   NaN         0.0
1512  2020-08-26 16:05:02                   NaN         0.0

                                                   guid menu_order  \
0     https://www.bottle-neck.fr/wp-content/uploads/...        0.0
1                                                   NaN        NaN
2     https://www.bottle-neck.fr/?post_type=product&...        0.0
3     https://www.bottle-neck.fr/?post_type=product&...        0.0
```

```
4     https://www.bottle-neck.fr/wp-content/uploads/...        0.0
...                                                  ...        ...
1508  https://www.bottle-neck.fr/wp-content/uploads/...        0.0
1509  https://www.bottle-neck.fr/?post_type=product&...        0.0
1510  https://www.bottle-neck.fr/?post_type=product&...        0.0
1511  https://www.bottle-neck.fr/wp-content/uploads/...        0.0
1512  https://www.bottle-neck.fr/wp-content/uploads/...        0.0

       post_type post_mime_type comment_count
0     attachment     image/jpeg           0.0
1            NaN            NaN           NaN
2        product            NaN           0.0
3        product            NaN           0.0
4     attachment     image/jpeg           0.0
...          ...            ...           ...
1508  attachment     image/jpeg           0.0
1509     product            NaN           0.0
1510     product            NaN           0.0
1511  attachment     image/jpeg           0.0
1512  attachment     image/jpeg           0.0

[1513 rows x 28 columns]

Statistiques générales:
```

	virtual	downloadable	rating_count	average_rating	total_sales	tax_class	post_auth
count	1513.0	1513.0	1513.0	1430.0	1430.000000	0.0	1430.0000
mean	0.0	0.0	0.0	0.0	3.855245	NaN	1.9986
std	0.0	0.0	0.0	0.0	7.702346	NaN	0.0373
min	0.0	0.0	0.0	0.0	0.000000	NaN	1.0000
25%	0.0	0.0	0.0	0.0	0.000000	NaN	2.0000
50%	0.0	0.0	0.0	0.0	1.000000	NaN	2.0000
75%	0.0	0.0	0.0	0.0	4.000000	NaN	2.0000
max	0.0	0.0	0.0	0.0	96.000000	NaN	2.0000

Out[ ]: (1513, 28)

Résumé des Résultats :

- Des doublons et des valeurs NaN sont présents dans les données.

Résumé des Statistiques :

- Les colonnes telles que 'virtual', 'downloadable', 'rating_count', 'average_rating', etc., affichent des variations dans le nombre total d'entrées non manquantes.
- La moyenne de 'total_sales' est d'environ 3.86, présentant une certaine variabilité autour de la moyenne.
- Des valeurs constantes de zéro dans plusieurs colonnes suggèrent une possible absence

de diversité.

- La colonne 'average_rating' contient des valeurs NaN, indiquant des données manquantes ou indéfinies.

A.I) Analyse du Fichier ERP

In [ ]:
```
# Nettoyage du fichier_erp

# Conversion de la colonne "price" en type numérique
stock['price'] = pd.to_numeric(stock['price'], errors='coerce')

display(stock)
```

	product_id	onsale_web	price	stock_quantity	stock_status
0	3847	1	NaN	0	outofstock
1	3849	1	NaN	0	outofstock
2	3850	1	NaN	0	outofstock
3	4032	1	NaN	0	outofstock
4	4039	1	46.0	0	outofstock
...	...	...	...	...	...
820	7203	0	45.0	30	instock
821	7204	0	45.0	9	instock
822	7247	1	NaN	23	instock
823	7329	0	NaN	14	instock
824	7338	1	NaN	45	instock

825 rows × 5 columns

In [ ]:
```
# Charger le fichier CSV dans un DataFrame
stock = pd.read_csv("fichier_erp.csv", delimiter=';')

# Convertir la colonne "price" en numérique en conservant les valeurs originales si
stock['price_numeric'] = pd.to_numeric(stock['price'], errors='coerce')

# Afficher les lignes où la conversion a échoué
failed_conversion_rows = stock[stock['price_numeric'].isnull()]
print("Lignes où la conversion a échoué :")
print(failed_conversion_rows)

# Conserver les valeurs originales là où la conversion a échoué
stock['price_numeric'] = stock['price_numeric'].combine_first(stock['price'])
```

```
Lignes où la conversion a échoué :
     product_id  onsale_web price  stock_quantity stock_status  price_numeric
0          3847           1  24,2               0   outofstock            NaN
1          3849           1  34,3               0   outofstock            NaN
2          3850           1  20,8               0   outofstock            NaN
3          4032           1  14,1               0   outofstock            NaN
5          4040           1  34,3               0   outofstock            NaN
..          ...         ...   ...             ...          ...            ...
787        6930           1   8,4              83      instock            NaN
792        7023           1  27,5              15      instock            NaN
822        7247           1  54,8              23      instock            NaN
823        7329           0  26,5              14      instock            NaN
824        7338           1  16,3              45      instock            NaN

[609 rows x 6 columns]
```

In [ ]:
```python
# Remplacer les virgules par des points dans la colonne "price"
stock['price'] = stock['price'].str.replace(',', '.')

# Tenter la conversion en numérique
stock['price_numeric'] = pd.to_numeric(stock['price'], errors='coerce')

# Afficher les lignes où la conversion a échoué
failed_conversion_rows = stock[stock['price_numeric'].isnull()]
print("Lignes où la conversion a échoué :")
print(failed_conversion_rows)

# Conserver les valeurs originales là où la conversion a échoué
stock['price_numeric'] = stock['price_numeric'].combine_first(stock['price'])
display(stock)
```

```
Lignes où la conversion a échoué :
Empty DataFrame
Columns: [product_id, onsale_web, price, stock_quantity, stock_status, price_numeri
c]
Index: []
```

	product_id	onsale_web	price	stock_quantity	stock_status	price_numeric
0	3847	1	24.2	0	outofstock	24.2
1	3849	1	34.3	0	outofstock	34.3
2	3850	1	20.8	0	outofstock	20.8
3	4032	1	14.1	0	outofstock	14.1
4	4039	1	46	0	outofstock	46.0
...	...	...	...	...	...	...
820	7203	0	45	30	instock	45.0
821	7204	0	45	9	instock	45.0
822	7247	1	54.8	23	instock	54.8
823	7329	0	26.5	14	instock	26.5
824	7338	1	16.3	45	instock	16.3

825 rows × 6 columns

B.I) Analyse du Fichier liaison

In [ ]:
```
# Remplacer les NaN par 0 dans tout le DataFrame
liaison = liaison.fillna(0)
display(liaison)
```

	product_id	sku
0	3847	15298
1	3849	15296
2	3850	15300
3	4032	19814
4	4039	19815
...	...	...
820	7203	0
821	7204	0
822	7247	13127-1
823	7329	14680-1
824	7338	16230

825 rows × 2 columns

In [ ]:
```
liaison['sku'].unique()
```

Out[ ]:
```
array(['15298', '15296', '15300', '19814', '19815', '15303', '14975',
       '16042', '14980', '16041', '15269', '14977', '16044', '16043',
       '16449', '16045', '16030', '13127', '19816', 0, '16029', '16039',
       '16318', '16275', '16498', '16320', '16319', '15966', '15022',
       '15967', '15490', '16416', '11862', '15444', '15953', '12045',
       '13074', '15941', '16069', '13072', '15440', '13435', '13078',
       '13117', '16296', '16014', '16462', '16013', '16180', '15676',
       '16120', '15564', '15675', '15378', '15813', '13416', '14905',
       '15767', '16505', '15683', '16504', '15787', '14800', '15353',
       '15382', '15339', '11668', '13209', '15341', '13217', '304',
       '11641', '1662', '1360', '15648', '1364', '7086', '1366', '15140',
       '16238', '16237', '15141', '14944', '14941', '14751', '16093',
       '15668', '15373', '15375', '14474', '15482', '13453', '15075',
       '16124', '15785', '15784', '15786', '14332', '16210', '16211',
       '16209', '15629', '15583', '16160', '16166', '15783', '16560',
       '15747', '15746', '16190', '16189', '16265', '16191', '16263',
       '15605', '16529', '15441', '13032', '16256', '16322', '16295',
       '15656', '15655', '15415', '15414', '15413', '16023', '16024',
       '15720', '15714', '15717', '15718', '15480', '15213', '15672',
       '12599', '15758', '15829', '15759', '16585', '15306', '16497',
       '15261', '12657', '15403', '15461', '16269', '13905', '16567',
       '15436', '14725', '15310', '15770', '16097', '15428', '15033',
       '16317', '15032', '6616', '12203', '14253', '12476', '14485',
       '14945', '15662', '15663', '15664', '15665', '15136', '16537',
       '16307', '16244', '15839', '13460', '13089', '12942', '14864',
       '14527', '14865', '15690', '16330', '16154', '16153', '16066',
       '16065', '15292', '13771', '16246', '16501', '16578', '15567',
       '16553', '13172', '15120', '15949', '15946', '7818', '13599',
       '4679', '12586', '12588', '15940', '12587', '12589', '12585',
       '9562', '13854', '13853', '11585', '11467', '11586', '13765',
       '13766', '11587', '9636', '12639', '12641', '12640', '14768',
       '3506', '3510', '3507', '13230', '7819', '3509', '15426', '15621',
       '15457', '15065', '13604', '12857', '14785', '15476', '14000',
       '15478', '15475', '16151', '15659', '15147', '15660', '15148',
       '15149', '15146', '15145', '15801', '15452', '15038', '15030',
       '15875', '16186', '14371', '10459', '14372', '11049', '15850',
       '15849', '812', '807', '805', '802', '2534', '793', '791', '2179',
       '804', '41', '798', '2361', '15848', '16525', '16262', '16261',
       '15206', '11849', '13515', '13514', '13516', '10814', '11847',
       '13517', '16081', '15402', '15404', '13647', '14657', '16053',
       '15525', '15527', '14676', '16057', '16056', '13762', '15280',
       '15282', '15281', '15283', '15934', '15933', '15575', '16239',
       '14451', '16324', '15582', '13736', '13659', '15465', '15004',
       '14699', '15349', '15466', '14700', '10775', '16119', '15667',
       '14746', '15361', '15196', '15657', '15658', '15670', '16527',
       '16513', '15880', '15879', '16010', '14950', '16540', '15729',
       '38', '5646', '8344', '15576', '16138', '14366', '13412', '12601',
       '14632', '15315', '13627', '14184', '15429', '16132', '14680',
       '15859', '16229', '14302', '16072', '14300', '13096', '16564',
       '13754', '15734', '15448', '15881', '15731', '15316', '15732',
       '14599', '15733', '15730', '12771', '3568', '14506', '15811',
       '16342', '16292', '15307', '16047', '16255', '15154', '16274',
       '16148', '14360', '16149', '16289', '14981', '15773', '15776',
       '16037', '16038', '15807', '15952', '15808', '16062', '16063',
       '14802', '13052', '14805', '14220', '14374', '14395', '15614',
       '13809', '15612', '13814', '15613', '15615', '15533', '15531',
```

```
'15530', '15608', '15586', '15928', '16276', '16277', '15456',
'15425', '15047', '15927', '16155', '16280', '9937', '16281',
'15554', '15106', '16283', '13379', '15338', '15337',
'bon-cadeau-25-euros', '15737', '15958', '16515', '16586', '11225',
'16004', '14756', '16005', '14930', '13313', '15229', '14507',
'14509', '14508', '15868', '14581', '14580', '15869', '15871',
'15870', '12791', '11602', '15073', '14839', '15272', '14696',
'15630', '11996', '13914', '13913', '11997', '531', '13531',
'15711', '15713', '15715', '15346', '15345', '15344', '15755',
'15677', '14561', '16022', '16011', '3383', '14149', '13904',
'14141', '12494', '15462', '15095', '14626', '12496', '12315',
'15649', '14809', '15155', '12194', '16328', '14469', '16034',
'14679', '15526', '16305', '16306', '15138', '15753', '15756',
'16131', '16130', '16129', '14712', '15481', '16146', '14648',
'14192', '15860', '15863', '15861', '15862', '15864', '14819',
'14828', '14827', '15202', '13959', '13965', '13958', '13957',
'13520', '13969', '14715', '19820', '19821', '15748', '19822',
'16192', '14730', '14729', '8463', '13982', '15944', '15930',
'14912', '15945', '14915', '14855', '14856', '15923', '14845',
'14844', '15921', '15922', '12366', '8365', '12365', '14647',
'15812', '14661', '16304', '15797', '16094', '14736', '11736',
'15036', '15360', '15674', '13557', '15035', '16121', '14241',
'14982', '15026', '15116', '15369', '15566', '16003', '15127',
'15125', '14323', '15631', '16147', '7033', '11258', '13849',
'15818', '15179', '15185', '15183', '15254', '15178', '15184',
'15180', '15264', '14338', '15561', '16213', '14692', '13291',
'13895', '15688', '14461', '14689', '11277', '15399', '13572',
'14955', '13567', '15471', '15080', '14429', '15238', '15237',
'14600', '15241', '11933', '15240', '15325', '15328', '15329',
'15775', '15774', '14983', '13910', '16539', '15910', '12339',
'12869', '14095', '14099', '15856', '12881', '15857', '12882',
'15227', '10014', '14265', '14774', '14775', '14773', '15343',
'15351', '16323', '523', '15432', '16472', '14379', '15609',
'14377', '15895', '13577', '15577', '15766', '15892', '16326',
'15574', '13662', '11669', '13215', '13211', '15342', '15318',
'13073', '16159', '16264', '14899', '15134', '16133', '16028',
'15951', '15487', '15486', '15489', '15529', '14089', '14100',
'14092', '14090', '14106', '14101', '14797', '15201', '14923',
'14573', '14569', '14570', '15834', '14596', '15126', '14604',
'16565', '16580', '16077', '13996', '15072', '11601', '12790',
'15070', '16096', '7032', '15324', '15162', '15161', '15163',
'16273', '16247', '15654', '15710', '15745', '15678', '15810',
'15779', '15707', '15705', '15706', '15704', '15473', '15479',
'15647', '15769', '15434', '15764', '16071', '15781', '16031',
'15539', '16046', '15204', '15205', '15790', '15791', '15792',
'15793', '15795', '15794', '15763', '16152', '15661', '16068',
'16067', '8193', '16144', '15256', '15735', '14897', '15736',
'15740', '15845', '15741', '16135', '15891', '15887', '13127-1',
'14680-1', '16230'], dtype=object)
```

In [ ]:
```
# on recherche la valeur 'bon-cadeau-25-euros'
liaison.loc[liaison['sku'] == 'bon-cadeau-25-euros']
```

Out[ ]:

	product_id	sku
443	4954	bon-cadeau-25-euros

In [ ]:
```
liaison.loc[liaison['sku'] == 'bon-cadeau-25-euros']
```

Out[ ]:

	product_id	sku
443	4954	bon-cadeau-25-euros

In [ ]:
```
# on affiche les valeurs uniques prises par la variable
liaison['product_id'].unique()
```

Out[ ]:
```
array([3847, 3849, 3850, 4032, 4039, 4040, 4041, 4042, 4043, 4045, 4046,
       4047, 4048, 4049, 4050, 4051, 4052, 4053, 4054, 4055, 4056, 4057,
       4058, 4059, 4060, 4062, 4063, 4064, 4065, 4066, 4067, 4068, 4069,
       4070, 4071, 4072, 4073, 4074, 4075, 4076, 4077, 4078, 4079, 4081,
       4083, 4084, 4085, 4086, 4087, 4090, 4092, 4094, 4095, 4096, 4097,
       4098, 4099, 4100, 4101, 4102, 4103, 4104, 4105, 4106, 4107, 4108,
       4115, 4130, 4132, 4137, 4138, 4139, 4141, 4142, 4144, 4146, 4147,
       4148, 4149, 4150, 4151, 4152, 4153, 4154, 4155, 4156, 4157, 4158,
       4159, 4160, 4161, 4162, 4163, 4164, 4165, 4166, 4167, 4168, 4170,
       4171, 4172, 4173, 4174, 4176, 4177, 4178, 4179, 4180, 4181, 4182,
       4183, 4186, 4187, 4188, 4190, 4191, 4192, 4193, 4194, 4195, 4196,
       4197, 4198, 4200, 4201, 4202, 4203, 4204, 4205, 4207, 4208, 4209,
       4210, 4211, 4212, 4213, 4215, 4216, 4217, 4219, 4220, 4221, 4222,
       4223, 4224, 4225, 4227, 4228, 4229, 4231, 4232, 4233, 4235, 4239,
       4240, 4241, 4242, 4244, 4245, 4246, 4248, 4250, 4251, 4253, 4254,
       4256, 4257, 4258, 4260, 4261, 4262, 4263, 4264, 4265, 4267, 4268,
       4269, 4270, 4271, 4272, 4274, 4275, 4276, 4277, 4278, 4279, 4280,
       4281, 4283, 4285, 4286, 4287, 4288, 4289, 4297, 4298, 4299, 4300,
       4301, 4303, 4304, 4306, 4307, 4334, 4336, 4337, 4348, 4350, 4352,
       4353, 4355, 4356, 4357, 4358, 4359, 4364, 4391, 4392, 4393, 4394,
       4395, 4396, 4397, 4398, 4399, 4400, 4401, 4402, 4404, 4405, 4406,
       4407, 4558, 4564, 4565, 4566, 4568, 4573, 4577, 4578, 4582, 4584,
       4594, 4596, 4597, 4598, 4599, 4600, 4601, 4602, 4603, 4604, 4605,
       4606, 4607, 4609, 4610, 4611, 4612, 4613, 4614, 4615, 4616, 4617,
       4618, 4619, 4620, 4621, 4625, 4626, 4627, 4628, 4629, 4630, 4631,
       4632, 4633, 4634, 4635, 4636, 4646, 4647, 4648, 4649, 4650, 4651,
       4653, 4654, 4655, 4656, 4657, 4658, 4659, 4662, 4664, 4665, 4666,
       4668, 4669, 4670, 4671, 4672, 4673, 4674, 4675, 4676, 4677, 4678,
       4679, 4680, 4681, 4682, 4683, 4684, 4686, 4687, 4689, 4690, 4692,
       4693, 4697, 4698, 4702, 4703, 4704, 4705, 4706, 4707, 4708, 4709,
       4711, 4712, 4713, 4714, 4715, 4716, 4717, 4718, 4719, 4720, 4721,
       4722, 4723, 4725, 4726, 4727, 4728, 4729, 4730, 4731, 4733, 4734,
       4738, 4739, 4740, 4741, 4744, 4748, 4749, 4750, 4752, 4753, 4755,
       4757, 4758, 4759, 4776, 4778, 4779, 4780, 4782, 4783, 4784, 4785,
       4786, 4788, 4789, 4790, 4791, 4792, 4793, 4794, 4795, 4797, 4798,
       4799, 4858, 4860, 4861, 4862, 4863, 4864, 4865, 4867, 4869, 4870,
       4874, 4876, 4885, 4886, 4888, 4889, 4890, 4891, 4892, 4893, 4899,
       4900, 4901, 4902, 4903, 4904, 4907, 4908, 4909, 4910, 4911, 4912,
       4913, 4914, 4915, 4918, 4919, 4920, 4921, 4922, 4923, 4924, 4925,
       4926, 4927, 4928, 4929, 4930, 4931, 4932, 4933, 4934, 4936, 4937,
       4938, 4939, 4940, 4954, 4962, 4963, 4964, 4965, 4970, 4973, 4974,
       4975, 4976, 4977, 4978, 4980, 4994, 4995, 4996, 5000, 5001, 5002,
       5003, 5004, 5006, 5007, 5008, 5010, 5016, 5017, 5018, 5019, 5020,
       5021, 5024, 5025, 5026, 5027, 5047, 5056, 5061, 5062, 5063, 5067,
       5068, 5069, 5070, 5075, 5375, 5377, 5379, 5380, 5382, 5383, 5384,
       5389, 5391, 5393, 5394, 5395, 5396, 5397, 5398, 5439, 5443, 5444,
       5445, 5446, 5448, 5465, 5474, 5477, 5479, 5480, 5481, 5483, 5484,
       5485, 5486, 5487, 5488, 5491, 5504, 5505, 5506, 5519, 5520, 5522,
       5523, 5524, 5525, 5544, 5545, 5546, 5547, 5548, 5550, 5551, 5552,
       5554, 5559, 5560, 5561, 5563, 5564, 5565, 5566, 5569, 5570, 5573,
       5574, 5580, 5608, 5609, 5610, 5611, 5612, 5613, 5614, 5615, 5616,
       5617, 5618, 5619, 5628, 5629, 5630, 5690, 5693, 5694, 5695, 5696,
       5697, 5700, 5703, 5704, 5705, 5706, 5707, 5709, 5711, 5712, 5715,
       5722, 5736, 5737, 5738, 5739, 5741, 5742, 5743, 5747, 5753, 5756,
       5760, 5761, 5764, 5766, 5767, 5768, 5769, 5770, 5771, 5772, 5773,
       5777, 5778, 5779, 5794, 5795, 5796, 5797, 5799, 5800, 5801, 5802,
```

```
 5803, 5804, 5805, 5806, 5807, 5808, 5809, 5810, 5815, 5816, 5817,
 5818, 5819, 5820, 5826, 5827, 5829, 5890, 5891, 5892, 5893, 5894,
 5896, 5899, 5900, 5902, 5903, 5904, 5905, 5906, 5907, 5912, 5913,
 5914, 5916, 5917, 5918, 5922, 5925, 5930, 5932, 5950, 5951, 5952,
 5953, 5954, 5955, 5956, 5957, 5958, 5959, 5960, 5962, 5963, 5964,
 5967, 5968, 5969, 6035, 6038, 6041, 6042, 6047, 6049, 6050, 6070,
 6072, 6073, 6093, 6094, 6095, 6100, 6101, 6103, 6104, 6105, 6106,
 6107, 6108, 6109, 6125, 6126, 6127, 6128, 6129, 6137, 6201, 6202,
 6204, 6205, 6206, 6207, 6212, 6213, 6214, 6215, 6216, 6221, 6222,
 6223, 6225, 6226, 6227, 6278, 6279, 6280, 6299, 6301, 6324, 6325,
 6327, 6328, 6567, 6568, 6569, 6570, 6572, 6573, 6575, 6578, 6584,
 6585, 6592, 6594, 6615, 6616, 6617, 6618, 6620, 6621, 6622, 6626,
 6627, 6628, 6629, 6631, 6632, 6635, 6663, 6664, 6665, 6666, 6738,
 6751, 6753, 6821, 6824, 6825, 6826, 6864, 6866, 6869, 6875, 6884,
 6886, 6887, 6898, 6899, 6900, 6901, 6902, 6903, 6904, 6905, 6906,
 6907, 6908, 6909, 6920, 6926, 6928, 6930, 7008, 7009, 7010, 7015,
 7023, 7025, 7081, 7084, 7085, 7086, 7087, 7088, 7131, 7132, 7133,
 7136, 7137, 7159, 7161, 7162, 7163, 7164, 7168, 7169, 7170, 7192,
 7193, 7194, 7195, 7196, 7200, 7201, 7203, 7204, 7247, 7329, 7338],
dtype=int64)
```

In [ ]:
```
# on affiche les valeurs uniques prises par la variable
liaison['sku'].unique()
```

Out[ ]:
```
array(['15298', '15296', '15300', '19814', '19815', '15303', '14975',
       '16042', '14980', '16041', '15269', '14977', '16044', '16043',
       '16449', '16045', '16030', '13127', '19816', 0, '16029', '16039',
       '16318', '16275', '16498', '16320', '16319', '15966', '15022',
       '15967', '15490', '16416', '11862', '15444', '15953', '12045',
       '13074', '15941', '16069', '13072', '15440', '13435', '13078',
       '13117', '16296', '16014', '16462', '16013', '16180', '15676',
       '16120', '15564', '15675', '15378', '15813', '13416', '14905',
       '15767', '16505', '15683', '16504', '15787', '14800', '15353',
       '15382', '15339', '11668', '13209', '15341', '13217', '304',
       '11641', '1662', '1360', '15648', '1364', '7086', '1366', '15140',
       '16238', '16237', '15141', '14944', '14941', '14751', '16093',
       '15668', '15373', '15375', '14474', '15482', '13453', '15075',
       '16124', '15785', '15784', '15786', '14332', '16210', '16211',
       '16209', '15629', '15583', '16160', '16166', '15783', '16560',
       '15747', '15746', '16190', '16189', '16265', '16191', '16263',
       '15605', '16529', '15441', '13032', '16256', '16322', '16295',
       '15656', '15655', '15415', '15414', '15413', '16023', '16024',
       '15720', '15714', '15717', '15718', '15480', '15213', '15672',
       '12599', '15758', '15829', '15759', '16585', '15306', '16497',
       '15261', '12657', '15403', '15461', '16269', '13905', '16567',
       '15436', '14725', '15310', '15770', '16097', '15428', '15033',
       '16317', '15032', '6616', '12203', '14253', '12476', '14485',
       '14945', '15662', '15663', '15664', '15665', '15136', '16537',
       '16307', '16244', '15839', '13460', '13089', '12942', '14864',
       '14527', '14865', '15690', '16330', '16154', '16153', '16066',
       '16065', '15292', '13771', '16246', '16501', '16578', '15567',
       '16553', '13172', '15120', '15949', '15946', '7818', '13599',
       '4679', '12586', '12588', '15940', '12587', '12589', '12585',
       '9562', '13854', '13853', '11585', '11467', '11586', '13765',
       '13766', '11587', '9636', '12639', '12641', '12640', '14768',
       '3506', '3510', '3507', '13230', '7819', '3509', '15426', '15621',
       '15457', '15065', '13604', '12857', '14785', '15476', '14000',
       '15478', '15475', '16151', '15659', '15147', '15660', '15148',
       '15149', '15146', '15145', '15801', '15452', '15038', '15030',
       '15875', '16186', '14371', '10459', '14372', '11049', '15850',
       '15849', '812', '807', '805', '802', '2534', '793', '791', '2179',
       '804', '41', '798', '2361', '15848', '16525', '16262', '16261',
       '15206', '11849', '13515', '13514', '13516', '10814', '11847',
       '13517', '16081', '15402', '15404', '13647', '14657', '16053',
       '15525', '15527', '14676', '16057', '16056', '13762', '15280',
       '15282', '15281', '15283', '15934', '15933', '15575', '16239',
       '14451', '16324', '15582', '13736', '13659', '15465', '15004',
       '14699', '15349', '15466', '14700', '10775', '16119', '15667',
       '14746', '15361', '15196', '15657', '15658', '15670', '16527',
       '16513', '15880', '15879', '16010', '14950', '16540', '15729',
       '38', '5646', '8344', '15576', '16138', '14366', '13412', '12601',
       '14632', '15315', '13627', '14184', '15429', '16132', '14680',
       '15859', '16229', '14302', '16072', '14300', '13096', '16564',
       '13754', '15734', '15448', '15881', '15731', '15316', '15732',
       '14599', '15733', '15730', '12771', '3568', '14506', '15811',
       '16342', '16292', '15307', '16047', '16255', '15154', '16274',
       '16148', '14360', '16149', '16289', '14981', '15773', '15776',
       '16037', '16038', '15807', '15952', '15808', '16062', '16063',
       '14802', '13052', '14805', '14220', '14374', '14395', '15614',
       '13809', '15612', '13814', '15613', '15615', '15533', '15531',
```

```
'15530', '15608', '15586', '15928', '16276', '16277', '15456',
'15425', '15047', '15927', '16155', '16280', '9937', '16281',
'15554', '15106', '16283', '13379', '15338', '15337',
'bon-cadeau-25-euros', '15737', '15958', '16515', '16586', '11225',
'16004', '14756', '16005', '14930', '13313', '15229', '14507',
'14509', '14508', '15868', '14581', '14580', '15869', '15871',
'15870', '12791', '11602', '15073', '14839', '15272', '14696',
'15630', '11996', '13914', '13913', '11997', '531', '13531',
'15711', '15713', '15715', '15346', '15345', '15344', '15755',
'15677', '14561', '16022', '16011', '3383', '14149', '13904',
'14141', '12494', '15462', '15095', '14626', '12496', '12315',
'15649', '14809', '15155', '12194', '16328', '14469', '16034',
'14679', '15526', '16305', '16306', '15138', '15753', '15756',
'16131', '16130', '16129', '14712', '15481', '16146', '14648',
'14192', '15860', '15863', '15861', '15862', '15864', '14819',
'14828', '14827', '15202', '13959', '13965', '13958', '13957',
'13520', '13969', '14715', '19820', '19821', '15748', '19822',
'16192', '14730', '14729', '8463', '13982', '15944', '15930',
'14912', '15945', '14915', '14855', '14856', '15923', '14845',
'14844', '15921', '15922', '12366', '8365', '12365', '14647',
'15812', '14661', '16304', '15797', '16094', '14736', '11736',
'15036', '15360', '15674', '13557', '15035', '16121', '14241',
'14982', '15026', '15116', '15369', '15566', '16003', '15127',
'15125', '14323', '15631', '16147', '7033', '11258', '13849',
'15818', '15179', '15185', '15183', '15254', '15178', '15184',
'15180', '15264', '14338', '15561', '16213', '14692', '13291',
'13895', '15688', '14461', '14689', '11277', '15399', '13572',
'14955', '13567', '15471', '15080', '14429', '15238', '15237',
'14600', '15241', '11933', '15240', '15325', '15328', '15329',
'15775', '15774', '14983', '13910', '16539', '15910', '12339',
'12869', '14095', '14099', '15856', '12881', '15857', '12882',
'15227', '10014', '14265', '14774', '14775', '14773', '15343',
'15351', '16323', '523', '15432', '16472', '14379', '15609',
'14377', '15895', '13577', '15577', '15766', '15892', '16326',
'15574', '13662', '11669', '13215', '13211', '15342', '15318',
'13073', '16159', '16264', '14899', '15134', '16133', '16028',
'15951', '15487', '15486', '15489', '15529', '14089', '14100',
'14092', '14090', '14106', '14101', '14797', '15201', '14923',
'14573', '14569', '14570', '15834', '14596', '15126', '14604',
'16565', '16580', '16077', '13996', '15072', '11601', '12790',
'15070', '16096', '7032', '15324', '15162', '15161', '15163',
'16273', '16247', '15654', '15710', '15745', '15678', '15810',
'15779', '15707', '15705', '15706', '15704', '15473', '15479',
'15647', '15769', '15434', '15764', '16071', '15781', '16031',
'15539', '16046', '15204', '15205', '15790', '15791', '15792',
'15793', '15795', '15794', '15763', '16152', '15661', '16068',
'16067', '8193', '16144', '15256', '15735', '14897', '15736',
'15740', '15845', '15741', '16135', '15891', '15887', '13127-1',
'14680-1', '16230'], dtype=object)
```

C.I) Analyse du Fichier WEB

In [ ]:
```
# Remplacer les valeurs NaN par une valeur par défaut (par exemple, 0) dans le fich
# produits = produits.fillna(0)
```

```
# Sélectionne les lignes duplicatées basées sur la colonne 'product_id' dans le Dat
liaison_duplications = liaison.loc[liaison['product_id'].duplicated(keep=False), :]

# Affiche les premières lignes du DataFrame 'produits'.
produits.head()
produits.shape
```

Out[ ]: (1513, 28)

A.II) Merge fichier liaison et erp

In [ ]:
```
# Fusionner les DataFrames 'stock' et 'liaison' sur la colonne 'product_id'
liaison_stock = pd.merge(stock, liaison, on = 'product_id')
liaison_stock = liaison_stock.drop('price', axis=1)

# Afficher les premières lignes du DataFrame
print(liaison_stock.shape)

liaison_stock.head()
```

(825, 6)

Out[ ]:

	product_id	onsale_web	stock_quantity	stock_status	price_numeric	sku
0	3847	1	0	outofstock	24.2	15298
1	3849	1	0	outofstock	34.3	15296
2	3850	1	0	outofstock	20.8	15300
3	4032	1	0	outofstock	14.1	19814
4	4039	1	0	outofstock	46.0	19815

B.II) Merge fichier liaison et stock

In [ ]:
```
# Supprime les lignes du DataFrame où la référence SKU ('sku') est manquante.
produits_no_nan_sku = produits.dropna(subset=['sku'])
produits_no_nan_sku.shape
```

Out[ ]: (1428, 28)

In [ ]:
```
# Fusionner les DataFrames après la conversion en minuscules
resultat_final = pd.merge(liaison_stock, produits_no_nan_sku, on='sku')

print(resultat_final.shape)
resultat_final.head()
resultat_final.duplicated(subset = 'product_id')
```

(1428, 33)

Out[ ]:
```
0       False
1        True
2       False
3        True
4       False
        ...
1423     True
1424    False
1425     True
1426    False
1427     True
Length: 1428, dtype: bool
```

In [ ]:
```
resultat_final.duplicated(subset = 'product_id')
```

Out[ ]:
```
0       False
1        True
2       False
3        True
4       False
        ...
1423     True
1424    False
1425     True
1426    False
1427     True
Length: 1428, dtype: bool
```

In [ ]:
```
print(resultat_final['post_type'].value_counts())
```

```
product       714
attachment    714
Name: post_type, dtype: int64
```

In [ ]:
```
print(resultat_final.shape)
resultat_final.head()
```

```
(1428, 33)
```

Out[ ]:

	product_id	onsale_web	stock_quantity	stock_status	price_numeric	sku	virtual	dov
0	3847	1	0	outofstock	24.2	15298	0	
1	3847	1	0	outofstock	24.2	15298	0	
2	3849	1	0	outofstock	34.3	15296	0	
3	3849	1	0	outofstock	34.3	15296	0	
4	3850	1	0	outofstock	20.8	15300	0	

5 rows × 33 columns

Filtrage sur la DataFrame resultat_final, en ne conservant que les lignes où la colonne 'post_type' a la valeur 'product'.

Par ailleurs, dans le traitement de nos fichiers et l'élaboration des calcul, j'ai procédé à un flitre sur la DataFrame resultat_final, en ne conservant que les lignes où la colonne 'post_type' a la valeur 'product'. Je me suis rendu compte que si je n'effectuer pas ce filtre, j'avais des valeurs négatives dans le CA, dans le stocks ainsi que dans les prix. Ce filtrage spécifique permet de me concentrer uniquement par produits et permet donc d'éliminer des données non pertinentes et assuré la cohérence des calculs. Les avantages sont de conserver uniquement les produits néanmoins le risque et la perte de quelques élements de valeurs.

Si on utliser pas ce filtre on aurait par exemple des pertes d'informations importantes notamment dans le calcul du chiffre d'affaire (exemple : resultat_final = resultat_final.dropna(axis=1))

In [ ]:
```
# Supprimer la colonne "product_id_y" de la DataFrame "resultat_final"
resultat_final = resultat_final.dropna(axis=1)

# Filtrer et ne garder que les post_type product
resultat_final = resultat_final.loc[resultat_final['post_type'] == 'product']
```

```
resultat_final.head()
```

Out[ ]:

	product_id	onsale_web	stock_quantity	stock_status	price_numeric	sku	virtual	dov
0	3847	1	0	outofstock	24.2	15298	0	
3	3849	1	0	outofstock	34.3	15296	0	
5	3850	1	0	outofstock	20.8	15300	0	
6	4032	1	0	outofstock	14.1	19814	0	
8	4039	1	0	outofstock	46.0	19815	0	

5 rows × 26 columns

In [ ]:
```
resultat_final.shape
```

Out[ ]: (714, 26)

III]Calculs

A.III) Calcul du chiffre d'affaires

In [ ]:
```
# on créé une colonne 'ca_web_par_produit' qui est égale au nb de ventes x le prix

resultat_final['ca'] = resultat_final['total_sales'] * resultat_final['price_numeri

# on affiche les premières lignes
resultat_final.head()
```

Out[ ]:

	product_id	onsale_web	stock_quantity	stock_status	price_numeric	sku	virtual	dov
0	3847	1	0	outofstock	24.2	15298	0	
3	3849	1	0	outofstock	34.3	15296	0	
5	3850	1	0	outofstock	20.8	15300	0	
6	4032	1	0	outofstock	14.1	19814	0	
8	4039	1	0	outofstock	46.0	19815	0	

5 rows × 27 columns

In [ ]:
```
#Vérification
resultat_final.shape
```

Out[ ]: (714, 27)

In [ ]:
```
ca_produits_sum = round(resultat_final['ca'].sum(), 2)
print(f"Le chiffre d'affaires réalisé en ligne est de {ca_produits_sum} €")

# Triez le DataFrame par chiffre d'affaires dans l'ordre décroissant
resultat_final = resultat_final.sort_values(by='ca', ascending=False)

# Sélectionnez les 10 premières lignes (top 10)
top_10 = resultat_final.head(10)

# Arrondissez le chiffre d'affaires dans le top 10 à deux décimales
top_10['ca'] = top_10['ca'].round(2)

# Affichez les informations sur le top 10
display(top_10[['post_name', 'product_id', 'ca']])
```

```
Le chiffre d'affaires réalisé en ligne est de 70568.6 €
```

```
C:\Users\Rorsharks\AppData\Local\Temp\ipykernel_10664\1469444159.py:11: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  top_10['ca'] = top_10['ca'].round(2)
```

	post_name	product_id	ca
388	champagne-gosset-grand-blanc-de-blanc	4334	4704.0
143	champagne-gosset-grand-rose	4144	4263.0
436	cognac-frapin-vip-xo	4402	2288.0
141	champagne-gosset-grand-millesime-2006	4142	1590.0
138	gosset-champagne-grande-reserve	4141	1560.0
402	champagne-egly-ouriet-grand-cru-brut-blanc-de-...	4355	1391.5
399	champagne-egly-ouriet-grand-cru-millesime-2008	4352	1125.0
158	elian-daros-cotes-du-marmandais-clos-baquey-2015	4153	1044.0
1303	domaine-giudicelli-patrimonio-blanc-2019	6206	1033.2
61	gilles-robin-crozes-hermitage-papillon-2019	4068	1029.2

In [ ]:
```python
# Triez le DataFrame par chiffre d'affaires dans l'ordre décroissant
resultat_final = resultat_final.sort_values(by='ca', ascending=False)

# Sélectionnez les 5 premières lignes (top 5)
top_5 = resultat_final.head(5)

# Arrondissez le chiffre d'affaires dans le top 5 à deux décimales
top_5['ca'] = top_5['ca'].round(2)

# Affichez les informations sur le top 5
display(top_5[['post_name', 'product_id', 'ca']])
```

```
C:\Users\Rorsharks\AppData\Local\Temp\ipykernel_10664\4079872414.py:8: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  top_5['ca'] = top_5['ca'].round(2)
```

	post_name	product_id	ca
388	champagne-gosset-grand-blanc-de-blanc	4334	4704.0
143	champagne-gosset-grand-rose	4144	4263.0
436	cognac-frapin-vip-xo	4402	2288.0
141	champagne-gosset-grand-millesime-2006	4142	1590.0
138	gosset-champagne-grande-reserve	4141	1560.0

Observations sur les Produits :

- Les 10 premiers et 5 premiers produits semblent être exclusivement des champagnes, cependant, il est important de noter que le prix élevé des champagnes peut ne pas être un indicateur fiable des meilleures ventes réelles.

In [ ]:
```
# Filtrer les valeurs supérieures à 0
valeurs_positives = resultat_final[resultat_final['ca'] > 0]

# Sélectionner les 20 premières lignes avec les valeurs les plus faibles
top_20_faibles = valeurs_positives.nsmallest(20, 'ca')

# Afficher toutes les informations triées par chiffre d'affaires croissant
display(top_20_faibles[['post_name', 'product_id', 'ca']])
```

	post_name	product_id	ca
718	chateau-turcaud-rose-2019	4858	6.5
882	domaine-de-montgilet-anjou-rouge-2016-2	5056	7.5
192	ollieux-romanis-monsieur-pinot-2017	4171	7.8
305	parce-freres-igp-pays-oc-zoe-viognier-2019	4245	8.9
298	parce-freres-hommage-fernand-blanc-2019	4241	8.9
590	chateau-plaisance-fronton-rose-2019	4683	9.1
346	domaine-serol-cote-roannaise-cabochard-2016	4272	9.2
1256	borie-la-vitarele-pays-herault-cigales-2019	6070	9.3
579	domaine-rotier-gaillac-rouge-les-gravels-2016	4677	9.5
730	chateau-de-la-selve-igp-coteaux-de-lardeche-ma...	4867	9.9
327	chermette-domaine-du-vissoux-beaujolais-griott...	4261	9.9
694	maurice-schoech-pinot-auxerrois-2018	4785	10.1
946	domaine-bulliat-chiroubles-2019	5480	10.4
384	domaine-de-lidylle-savoie-roussette-2018	4306	10.7
1085	triennes-igp-mediterranee-rouge-merlot-2016	5739	10.7
289	catena-zapata-mendoza-alamos-malbec-2018	4231	11.1
584	domaine-de-joy-cotes-de-gascogne-envie-de-joy-...	4680	12.6
1262	chateau-de-villeneuve-saumur-champigny-2017	6093	12.6
598	domaine-plageoles-cotes-du-tarn-blanc-sec-ondenc	4689	12.8
103	paul-ginglinger-pinot-gris-prelats-2018	4097	12.8

In [ ]:
```
#Vérification
resultat_final.head()
```

Out[ ]:

	product_id	onsale_web	stock_quantity	stock_status	price_numeric	sku	virtual	d
388	4334	1	0	outofstock	49.0	7818	0	
143	4144	1	11	instock	49.0	1662	0	
436	4402	1	8	instock	176.0	3510	0	
141	4142	1	8	instock	53.0	11641	0	
138	4141	1	1	instock	39.0	304	0	

5 rows × 27 columns

In [ ]:
```
# Calcule et affiche l'asymétrie arrondie à deux décimales de la distribution des p
print(round(resultat_final['price_numeric'].skew(), 2))
```

```
2.58
```

Observation sur la Variabilité des Prix :

- Une variabilité des prix en fonction des produits est observée, ce qui est normal étant donné que le coût financier de l'achat peut varier significativement d'un produit à l'autre.

Résumé de l'Asymétrie des Prix :

- Cela suggère que la majorité des produits ont des prix concentrés du côté des valeurs plus basses, avec quelques produits ayant des prix significativement plus élevés, étirant la distribution vers la droite.

B.III.) Analyse des produits les plus fréquemment rencontrés

Résumé de la Vérification des Occurrences :

- La vérification des occurrences dans les fichiers a été effectuée pour identifier d'éventuels problèmes.
- Une mise en relation a été établie entre les occurrences, les prix des produits, le chiffre d'affaires, et les niveaux de stock.
- Cette analyse vise à détecter des anomalies, à valider l'intégrité des données, et à fournir des insights pour guider les décisions analytiques.

- Les résultats obtenus peuvent orienter des actions correctives ou des ajustements dans le traitement des données en fonction des relations observées.

In [ ]:
```
# Compter les occurrences des produits
occurrences_produits = resultat_final['product_id'].value_counts()

# Afficher les 10 premières occurrences avec 'post_name', 'product_id', et 'ca'
top_occurrences_info = resultat_final[resultat_final['product_id'].isin(occurrences

# Afficher le résultat
print("Occurrences des produits avec 'post_name', 'product_id', et 'ca' :")
display(top_occurrences_info)
```

```
Occurrences des produits avec 'post_name', 'product_id', et 'ca' :
```

	post_name	product_id	ca
388	champagne-gosset-grand-blanc-de-blanc	4334	4704.0
759	domaine-des-croix-savigny-les-beaune-1er-cru-l...	4902	0.0
830	champagne-larmandier-bernier-terre-de-vertus-p...	4970	0.0
683	clos-du-mont-olivet-cotes-du-rhone-rose-farel-...	4778	0.0
676	domaine-saint-nicolas-fiefs-vendeens-rouge-cuv...	4758	0.0
566	pelle-menetou-salon-rouge-les-cris-2015	4671	0.0
569	pelle-menetou-salon-rouge-morogues-2017	4672	0.0
570	pelle-sancerre-rouge-la-croix-au-garde-2017	4673	0.0
575	domaine-rotier-gaillac-blanc-sec-renaissance-2015	4675	0.0
692	maurice-schoech-riesling-vendanges-tardives-2017	4784	0.0

In [ ]:
```
# Compter les occurrences des produits
occurrences_produits = resultat_final['product_id'].value_counts()

# Ajouter une colonne 'occurrences' au DataFrame résultant
resultat_final_occurrences = resultat_final.merge(occurrences_produits.rename('occu

# Afficher le DataFrame résultant avec 'post_name', 'product_id', 'ca' et 'occurren
print("Occurrences des produits avec 'post_name', 'product_id', 'ca' et 'occurrence
display(resultat_final_occurrences[['post_name', 'product_id', 'ca', 'occurrences']
```

```
Occurrences des produits avec 'post_name', 'product_id', 'ca' et 'occurrences' :
```

	post_name	product_id	ca	occurrences
388	champagne-gosset-grand-blanc-de-blanc	4334	4704.0	1
143	champagne-gosset-grand-rose	4144	4263.0	1
436	cognac-frapin-vip-xo	4402	2288.0	1
141	champagne-gosset-grand-millesime-2006	4142	1590.0	1
138	gosset-champagne-grande-reserve	4141	1560.0	1
...	...	...	...	...
1216	cognac-normandin-mercier-vfc	5932	0.0	1
1383	weingut-besson-strasser-zurich-blauer-zweigelt...	6628	0.0	1
1221	domaine-de-vaccelli-aop-ajaccio-rouge-granit-1...	5951	0.0	1
1223	clos-du-prieur-terrasses-du-larzac-2018	5956	0.0	1
1426	domaine-saint-nicolas-fiefs-vendeens-blanc-les...	7338	0.0	1

714 rows × 4 columns

In [ ]:
```
# Filtrer les produits avec une occurrence différente de 0
occurrences_non_nulles = resultat_final_occurrences[resultat_final_occurrences['occ

# Créer un top 10 des produits avec occurrences non nulles
top_10_occurrences_non_nulles = occurrences_non_nulles.sort_values(by=['occurrences

# Afficher le top 10 avec 'post_name', 'product_id', 'ca', 'occurrences', 'stock_qu
print("Top 10 des occurrences non nulles avec 'post_name', 'product_id', 'ca', 'occ
display(top_10_occurrences_non_nulles[['post_name', 'product_id', 'ca', 'occurrence
```

```
Top 10 des occurrences non nulles avec 'post_name', 'product_id', 'ca', 'occurrences
', 'stock_quantity', et 'price_numeric' :
```

	post_name	product_id	ca	occurrences	stock_quantity	price_numeric
553	pierre-martin-sancerre-les-monts-damnes-2018	4662	0.0	1	24	20.8
236	moulin-gassac-igp-pays-herault-guilhem-rouge-2019	4198	0.0	1	105	5.8
150	champagne-mailly-grand-cru-intemporelle-rose-2009	4149	0.0	1	34	69.0
167	planeta-sicilia-alastro-2017	4157	0.0	1	10	12.0
168	8planeta-sicilia-etna-rosso-2018	4158	0.0	1	0	18.5
175	planeta-sicilia-plumbago-2017	4161	0.0	1	0	11.6
202	hortus-la-bergerie-blanc-2018	4177	0.0	1	109	13.5
208	hortus-la-grande-cuvee-blanc-2018	4180	0.0	1	47	24.0
215	borie-la-vitarele-saint-chinian-les-cres-2016	4183	0.0	1	48	21.4
242	mas-de-daumas-gassac-igp-saint-guilhem-le-dese...	4202	0.0	1	46	38.0

Résumé de l'Analyse des Données :

- La colonne `post_name` présente une diversité de noms de produits.
- Certains produits ont généré du chiffre d'affaires (`ca`), tandis que d'autres affichent un chiffre d'affaires nul.
- La majorité des produits ont une seule occurrence (`1`), bien que certains aient `0.0` occurrences.
- Cette analyse préliminaire vise à identifier des anomalies, à valider l'intégrité des données, et à fournir des insights pour guider les décisions analytiques.
- Les résultats obtenus peuvent orienter des actions correctives ou des ajustements dans le traitement des données en fonction des relations observées.

In [ ]:
```
# Filtrer les produits avec une occurrence différente de 0 et une colonne 'ca' diff
occurrences_non_nulles_et_ca_non_nul = resultat_final_occurrences[(resultat_final_o
```

```
# Créer un top 20 des produits avec occurrences non nulles et 'ca' non nul
top_20_occurrences_non_nulles_et_ca_non_nul = occurrences_non_nulles_et_ca_non_nul.

# Afficher le top 20 avec 'post_name', 'product_id', 'ca', 'occurrences', 'stock_qu
print("Top 20 des occurrences non nulles et 'ca' non nul trié par 'ca' ascendant av
display(top_20_occurrences_non_nulles_et_ca_non_nul[['post_name', 'product_id', 'ca
```

```
Top 20 des occurrences non nulles et 'ca' non nul trié par 'ca' ascendant avec 'post
_name', 'product_id', 'ca', 'occurrences', 'stock_quantity', et 'price_numeric' :
```

	post_name	product_id	ca	occurrences	stock_quantity	price_numeric
718	chateau-turcaud-rose-2019	4858	6.5	1	257	6.5
882	domaine-de-montgilet-anjou-rouge-2016-2	5056	7.5	1	9	7.5
192	ollieux-romanis-monsieur-pinot-2017	4171	7.8	1	65	7.8
305	parce-freres-igp-pays-oc-zoe-viognier-2019	4245	8.9	1	141	8.9
298	parce-freres-hommage-fernand-blanc-2019	4241	8.9	1	7	8.9
590	chateau-plaisance-fronton-rose-2019	4683	9.1	1	42	9.1
346	domaine-serol-cote-roannaise-cabochard-2016	4272	9.2	1	25	9.2
1256	borie-la-vitarele-pays-herault-cigales-2019	6070	9.3	1	124	9.3
579	domaine-rotier-gaillac-rouge-les-gravels-2016	4677	9.5	1	161	9.5
730	chateau-de-la-selve-igp-coteaux-de-lardeche-ma...	4867	9.9	1	0	9.9
327	chermette-domaine-du-vissoux-beaujolais-griott...	4261	9.9	1	89	9.9
694	maurice-schoech-pinot-auxerrois-2018	4785	10.1	1	18	10.1
946	domaine-bulliat-chiroubles-2019	5480	10.4	1	37	10.4
384	domaine-de-lidylle-savoie-roussette-2018	4306	10.7	1	41	10.7
1085	triennes-igp-mediterranee-rouge-merlot-2016	5739	10.7	1	39	10.7
289	catena-zapata-mendoza-alamos-malbec-2018	4231	11.1	1	0	11.1

	post_name	product_id	ca	occurrences	stock_quantity	price_numeric
584	domaine-de-joy-cotes-de-gascogne-envie-de-joy-...	4680	12.6	1	34	6.3
1262	chateau-de-villeneuve-saumur-champigny-2017	6093	12.6	1	0	12.6
598	domaine-plageoles-cotes-du-tarn-blanc-sec-ondenc	4689	12.8	1	6	12.8
103	paul-ginglinger-pinot-gris-prelats-2018	4097	12.8	1	26	12.8

In [ ]:
```
# Filtrer les produits avec une occurrence strictement supérieure à 1
occurrences_sup_1 = resultat_final_occurrences[resultat_final_occurrences['occurren

# Créer un top 10 des produits avec occurrences supérieures à 1
top_10_occurrences_sup_1 = occurrences_sup_1.sort_values(by=['occurrences', 'ca'],

# Afficher le top 10 avec 'post_name', 'product_id', 'ca', 'occurrences', 'stock_qu
print("Top 10 des occurrences supérieures à 1 avec 'post_name', 'product_id', 'ca',
display(top_10_occurrences_sup_1[['post_name', 'product_id', 'ca', 'occurrences', '
```

```
Top 10 des occurrences supérieures à 1 avec 'post_name', 'product_id', 'ca', 'occurr
ences', 'stock_quantity', et 'price_numeric' :
```

post_name	product_id	ca	occurrences	stock_quantity	price_numeric

In [ ]:
```
# Filtrer les lignes avec des valeurs négatives dans la colonne 'ca'
valeurs_negatives = resultat_final_occurrences[resultat_final_occurrences['ca'] < 0

# Afficher les lignes avec des valeurs négatives dans la colonne 'ca' avec 'post_na
print("Lignes avec des valeurs négatives dans la colonne 'ca' :")
display(valeurs_negatives[['post_name', 'product_id', 'ca', 'occurrences', 'stock_q
```

```
Lignes avec des valeurs négatives dans la colonne 'ca' :
```

post_name	product_id	ca	occurrences	stock_quantity	price_numeric

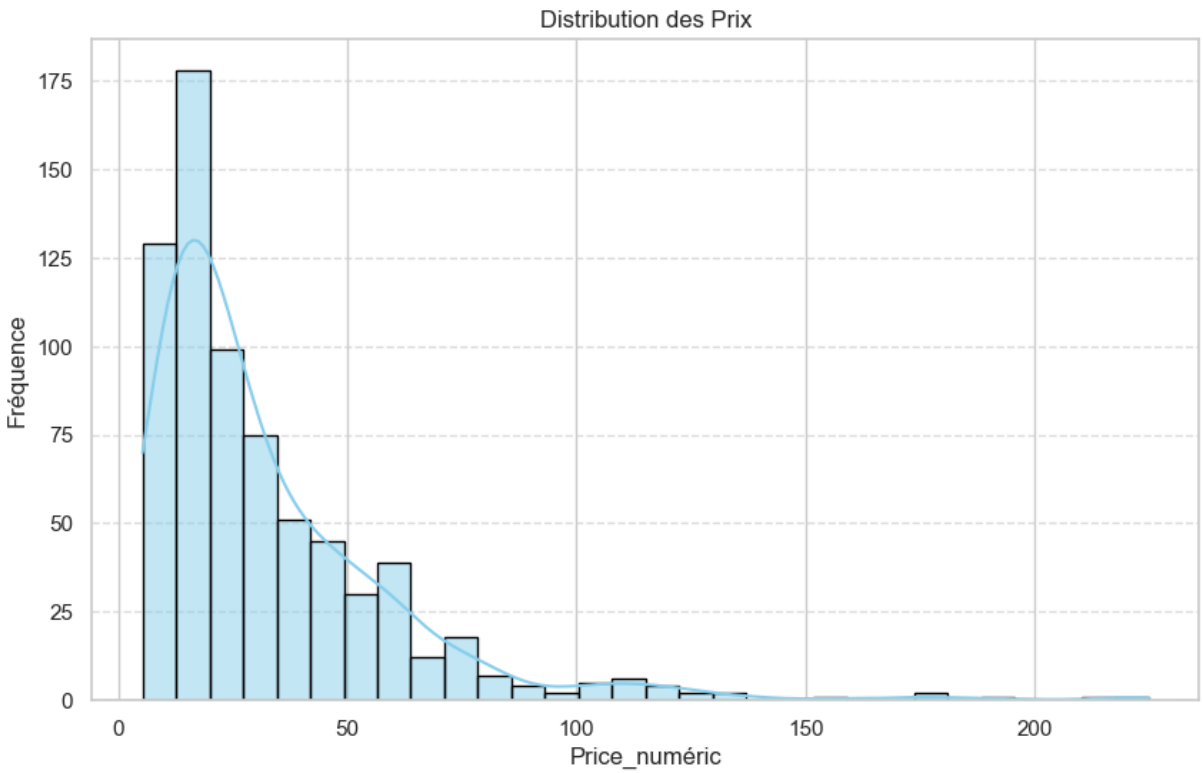
Observations sur les Clés et les Valeurs :

- Les 'product_id' sont des clés uniques, assurant une identification distincte pour chaque produit.

C.III.) Analyse des prix

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

# Configure le style de seaborn pour une meilleure apparence des graphiques
```

```
sns.set(style="whitegrid")

# Crée un histogramme des prix dans le DataFrame 'resultat_final' avec seaborn
plt.figure(figsize=(10, 6))
sns.histplot(resultat_final['price_numeric'], bins=30, kde=True, color='skyblue', e

# Ajoute des labels et un titre au graphique
plt.xlabel('Price_numéric')
plt.ylabel('Fréquence')
plt.title('Distribution des Prix')

# Ajoute une grille pour une meilleure lisibilité
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Affiche le graphique
plt.show()


# Affiche les statistiques descriptives sur la colonne des prix
price_stats = resultat_final['price_numeric'].describe().reset_index()

# Affiche le tableau de résultats
print(price_stats)
```




```
   index  price_numeric
0  count     714.000000
1   mean      32.493137
2    std      27.810525
3    min       5.200000
4    25%      14.100000
5    50%      23.550000
6    75%      42.175000
7    max     225.000000
```

Résumé des Statistiques des Prix :

- Nombre d'observations : 714
- Moyenne des prix : 32.49
- Écart type : 27.81
- Prix minimum : 5.20
- Premier quartile (Q1) : 14.10
- Médiane (50%) : 23.55
- Troisième quartile (Q3) : 42.18
- Prix maximum : 225.00

Ces statistiques fournissent une vue détaillée de la distribution des prix, allant de la centralité avec la moyenne et la médiane, à la dispersion avec l'écart type, ainsi que les valeurs minimales et maximales.

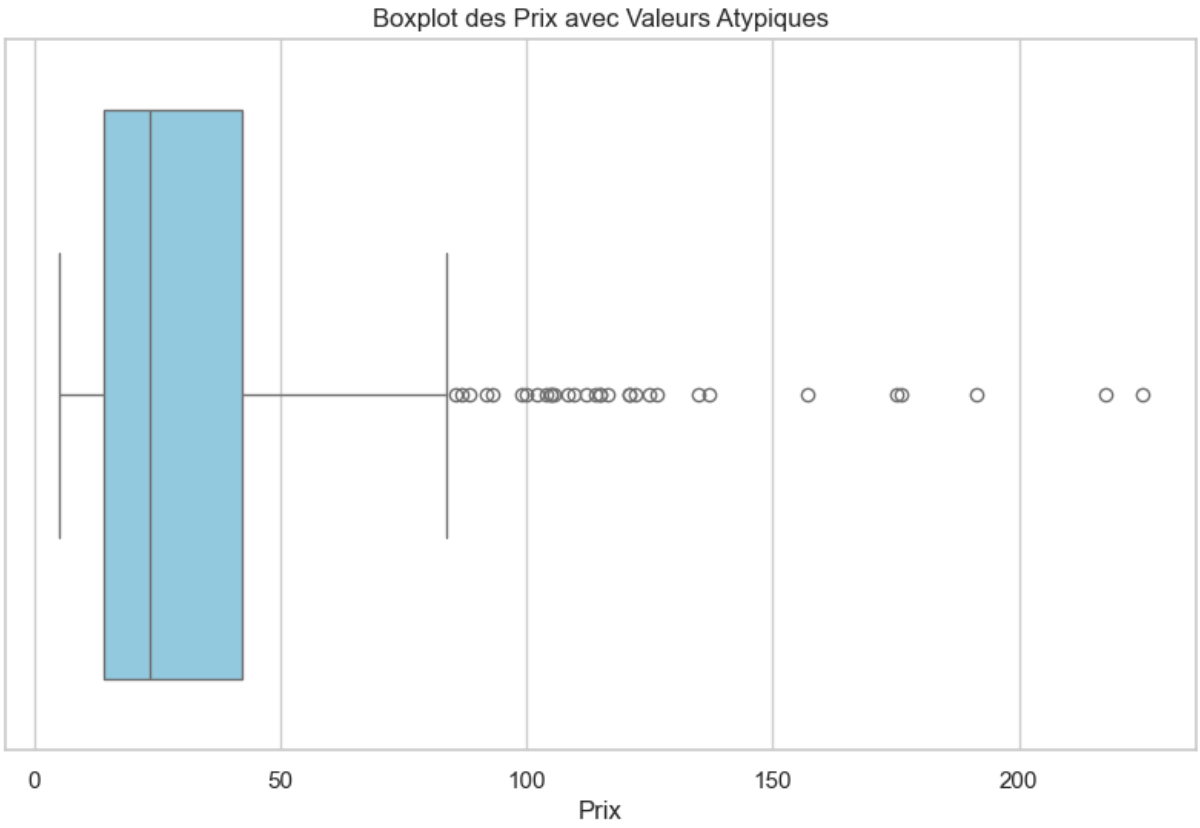
• Observations sur la Distribution des Prix :

La variation importante des prix suggère une diversité de produits dans l'ensemble de données.

La majorité des produits présente des prix concentrés dans une fourchette basse.

Quelques produits ont des prix significativement plus élevés.

In [ ]:
```
# Crée un boxplot pour visualiser les valeurs atypiques des prix
plt.figure(figsize=(10, 6))
sns.boxplot(x=resultat_final['price_numeric'], color='skyblue')

# Ajoute des labels et un titre au graphique
plt.xlabel('Prix')
plt.title('Boxplot des Prix avec Valeurs Atypiques')

# Affiche le graphique
plt.show()

# Analyse des Valeurs Atypiques des Prix
# Utilisation d'un boxplot pour identifier les valeurs atypiques dans la distributi

# Résultats de l'analyse
boxplot_results = resultat_final['price_numeric'].describe()
print(boxplot_results)
```




```
count    714.000000
mean      32.493137
std       27.810525
min        5.200000
25%       14.100000
50%       23.550000
75%       42.175000
max      225.000000
Name: price_numeric, dtype: float64
```

Résumé de l'Analyse des Statistiques des Prix :

- Nombre d'observations : 714
- Moyenne des prix : 32.49
- Écart type : 27.81
- Prix minimum : 5.20
- Premier quartile (Q1) : 14.10
- Médiane (50%) : 23.55
- Troisième quartile (Q3) : 42.18
- Prix maximum : 225.00

Cette analyse détaillée offre une vue complète de la distribution des prix, mettant en lumière la centralité, la dispersion, ainsi que les valeurs minimales et maximales. Ces données peuvent orienter des décisions commerciales en fournissant des insights sur la variabilité des prix dans votre ensemble de données.

In [ ]:
```
# Calcul des quartiles
```

```
Q1, Q3 = np.percentile(resultat_final['price_numeric'], [25, 75])

# Calcul de l'IQR
IQR = Q3 - Q1

# Définition des bornes pour les valeurs atypiques
borne_inf = Q1 - 1.5 * IQR
borne_sup = Q3 + 1.5 * IQR

# Définition des bornes minimale et maximale
borne_min = resultat_final['price_numeric'].min()
borne_max = resultat_final['price_numeric'].max()

# Sélection des valeurs atypiques
valeurs_atypiques = resultat_final[(resultat_final['price_numeric'] < borne_inf) |

# Affichage des résultats
print(f"Borne minimale : {borne_min}")
print(f"Borne pour les valeurs atypiques : [{borne_inf}, {borne_sup}]")
print(f"Borne maximale : {borne_max}")

print(f"Valeurs atypiques :")
print(valeurs_atypiques)

# Affichage du total des valeurs atypiques
total_atypiques = len(valeurs_atypiques)
print(f"Total des valeurs atypiques : {total_atypiques}")
```

```
Borne minimale : 5.2
Borne pour les valeurs atypiques : [-28.012500000000003, 84.28750000000001]
Borne maximale : 225.0
Valeurs atypiques :
      product_id  onsale_web  stock_quantity stock_status  price_numeric  \
436         4402           1               8      instock          176.0
402         4355           1               2      instock          126.5
399         4352           1               0   outofstock          225.0
763         4904           1              13      instock          137.0
1174        5892           1              10      instock          191.3
1284        6126           1              10      instock          135.0
1306        6212           1               2      instock          115.0
438         4404           1               2      instock          108.5
445         4407           1               6      instock          104.0
411         4359           1               0   outofstock           85.6
876         5026           1               2      instock           86.8
875         5025           1               0   outofstock          112.0
862         5007           1              17      instock          105.0
865         5008           1              10      instock          105.0
126         4115           1              11      instock          100.0
131         4132           1               5      instock           88.4
853         5001           1              20      instock          217.5
760         4903           1              20      instock          102.3
1005        5565           1               0   outofstock           92.0
1296        6202           1              14      instock          116.4
1309        6213           1               7      instock          121.0
1310        6214           1               7      instock           99.0
1312        6215           1               4      instock          115.0
1315        6216           1               6      instock          121.0
1106        5767           1              12      instock          175.0
1022        5612           1              12      instock          124.8
1294        6201           1               7      instock          105.6
454         4582           1               7      instock          109.6
443         4406           1               3      instock          157.0
1205        5916           1               3      instock           93.0
1206        5917           1               4      instock          122.0
1209        5918           1               8      instock          114.0

        sku  virtual  downloadable  rating_count  average_rating  ...  \
436    3510        0             0             0             0.0  ...
402   12589        0             0             0             0.0  ...
399   15940        0             0             0             0.0  ...
763   14220        0             0             0             0.0  ...
1174  14983        0             0             0             0.0  ...
1284  14923        0             0             0             0.0  ...
1306  13996        0             0             0             0.0  ...
438    3507        0             0             0             0.0  ...
445    3509        0             0             0             0.0  ...
411   13853        0             0             0             0.0  ...
876   13913        0             0             0             0.0  ...
875   13914        0             0             0             0.0  ...
862   12791        0             0             0             0.0  ...
865   11602        0             0             0             0.0  ...
126   15382        0             0             0             0.0  ...
131   11668        0             0             0             0.0  ...
853   14581        0             0             0             0.0  ...
```

```
760   14805        0             0             0             0.0  ...
1005  19822        0             0             0             0.0  ...
1296  15126        0             0             0             0.0  ...
1309  15072        0             0             0             0.0  ...
1310  11601        0             0             0             0.0  ...
1312  12790        0             0             0             0.0  ...
1315  15070        0             0             0             0.0  ...
1106  15185        0             0             0             0.0  ...
1022  14915        0             0             0             0.0  ...
1294  14596        0             0             0             0.0  ...
454   12857        0             0             0             0.0  ...
443    7819        0             0             0             0.0  ...
1205  14774        0             0             0             0.0  ...
1206  14775        0             0             0             0.0  ...
1209  14773        0             0             0             0.0  ...

      ping_status                                          post_name  \
436        closed                                 cognac-frapin-vip-xo
402        closed  champagne-egly-ouriet-grand-cru-brut-blanc-de-...
399        closed     champagne-egly-ouriet-grand-cru-millesime-2008
763        closed  domaine-des-croix-corton-charlemagne-grand-cru...
1174       closed  coteaux-champenois-egly-ouriet-ambonnay-rouge-...
1284       closed             champagne-gosset-celebris-vintage-2007
1306       closed  domaine-des-comtes-lafon-volnay-1er-cru-santen...
438        closed                          cognac-frapin-fontpinot-xo
445        closed                           cognac-frapin-cigar-blend
411        closed  champagne-larmandier-bernier-grand-cru-vieille...
876        closed  champagne-agrapart-fils-mineral-extra-brut-bla...
875        closed  champagne-agrapart-fils-lavizoise-grand-cru-20...
862        closed  domaine-des-comtes-lafon-volnay-1er-cru-santen...
865        closed  domaine-des-comtes-lafon-volnay-1er-cru-santen...
126        closed  zind-humbrecht-riesling-gc-rangen-thann-clos-s...
131        closed  zind-humbrecht-pinot-gris-grand-cru-rangen-de-...
853        closed     david-duband-charmes-chambertin-grand-cru-2014
760        closed  domaine-des-croix-corton-grand-cru-les-greves-...
1005       closed                       tempier-bandol-cabassaou-2017
1296       closed          domaine-clerget-echezeaux-en-orveaux-2015
1309       closed  domaine-des-comtes-lafon-volnay-1er-cru-santen...
1310       closed  domaine-des-comtes-lafon-volnay-1er-cru-champa...
1312       closed  domaine-des-comtes-lafon-volnay-1er-cru-champa...
1315       closed  domaine-des-comtes-lafon-volnay-1er-cru-champa...
1106       closed                camille-giroud-clos-de-vougeot-2016
1022       closed  domaine-weinbach-gewurztraminer-gc-furstentum-...
1294       closed  david-duband-chambolle-musigny-1er-cru-les-sen...
454        closed  chateau-de-puligny-montrachet-1cru-champ-canet...
443        closed     cognac-frapin-chateau-de-fontpinot-1989-20-ans
1205       closed       wemyss-malts-single-cask-chocolate-moka-cake
1206       closed  wemyss-malts-single-cask-scotch-whisky-choc-n-...
1209       closed  wemyss-malts-single-cask-scotch-whisky-chai-ca...

           post_modified    post_modified_gmt post_parent  \
436   2020-08-22 11:35:03  2020-08-22 09:35:03         0.0
402   2020-08-13 10:15:02  2020-08-13 08:15:02         0.0
399   2020-03-07 11:18:45  2020-03-07 10:18:45         0.0
763   2020-05-19 17:15:02  2020-05-19 15:15:02         0.0
1174  2020-04-01 09:30:09  2020-04-01 07:30:09         0.0
```

1284	2020-08-27 11:45:02	2020-08-27 09:45:02	0.0
1306	2020-06-16 09:30:16	2020-06-16 07:30:16	0.0
438	2020-08-12 09:30:16	2020-08-12 07:30:16	0.0
445	2020-07-04 09:45:03	2020-07-04 07:45:03	0.0
411	2019-12-23 09:30:11	2019-12-23 08:30:11	0.0
876	2020-05-11 14:35:02	2020-05-11 12:35:02	0.0
875	2020-07-09 17:05:02	2020-07-09 15:05:02	0.0
862	2020-07-02 09:30:03	2020-07-02 07:30:03	0.0
865	2020-06-23 15:35:02	2020-06-23 13:35:02	0.0
126	2020-02-08 11:45:02	2020-02-08 10:45:02	0.0
131	2020-02-20 09:55:02	2020-02-20 08:55:02	0.0
853	2020-05-16 09:00:05	2020-05-16 07:00:05	0.0
760	2020-06-27 09:00:07	2020-06-27 07:00:07	0.0
1005	2020-01-04 13:57:04	2020-01-04 12:57:04	0.0
1296	2020-06-06 15:45:01	2020-06-06 13:45:01	0.0
1309	2020-06-25 09:30:06	2020-06-25 07:30:06	0.0
1310	2020-07-04 11:35:02	2020-07-04 09:35:02	0.0
1312	2019-11-04 09:30:25	2019-11-04 08:30:25	0.0
1315	2020-07-30 09:30:08	2020-07-30 07:30:08	0.0
1106	2020-06-11 15:25:04	2020-06-11 13:25:04	0.0
1022	2019-01-23 09:33:57	2019-01-23 08:33:57	0.0
1294	2020-02-29 15:25:02	2020-02-29 14:25:02	0.0
454	2020-02-06 16:35:02	2020-02-06 15:35:02	0.0
443	2020-03-14 16:05:04	2020-03-14 15:05:04	0.0
1205	2019-12-23 09:30:21	2019-12-23 08:30:21	0.0
1206	2020-03-11 09:30:09	2020-03-11 08:30:09	0.0
1209	2020-07-31 18:25:03	2020-07-31 16:25:03	0.0

	guid	menu_order	post_type
436	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
402	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
399	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
763	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1174	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1284	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1306	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
438	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
445	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
411	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
876	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
875	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
862	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
865	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
126	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
131	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
853	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
760	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1005	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1296	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1309	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1310	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1312	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1315	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1106	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1022	https://www.bottle-neck.fr/?post_type=product&...	0.0	product
1294	https://www.bottle-neck.fr/?post_type=product&...	0.0	product

```
454   https://www.bottle-neck.fr/?post_type=product&...        0.0   product
443   https://www.bottle-neck.fr/?post_type=product&...        0.0   product
1205  https://www.bottle-neck.fr/?post_type=product&...        0.0   product
1206  https://www.bottle-neck.fr/?post_type=product&...        0.0   product
1209  https://www.bottle-neck.fr/?post_type=product&...        0.0   product

     comment_count      ca
436            0.0  2288.0
402            0.0  1391.5
399            0.0  1125.0
763            0.0   685.0
1174           0.0   573.9
1284           0.0   270.0
1306           0.0   230.0
438            0.0   217.0
445            0.0   104.0
411            0.0    85.6
876            0.0     0.0
875            0.0     0.0
862            0.0     0.0
865            0.0     0.0
126            0.0     0.0
131            0.0     0.0
853            0.0     0.0
760            0.0     0.0
1005           0.0     0.0
1296           0.0     0.0
1309           0.0     0.0
1310           0.0     0.0
1312           0.0     0.0
1315           0.0     0.0
1106           0.0     0.0
1022           0.0     0.0
1294           0.0     0.0
454            0.0     0.0
443            0.0     0.0
1205           0.0     0.0
1206           0.0     0.0
1209           0.0     0.0

[32 rows x 27 columns]
Total des valeurs atypiques : 32
```

Résumé de l'Analyse des Statistiques :

- Borne minimale : 5.2
- Borne pour les valeurs atypiques : [-28.0125, 84.2875]
- Borne maximale : 225.0

Ces résultats suggèrent une distribution de données, avec la majorité des valeurs se situant entre 5.2 et 225.0. Des valeurs atypiques, définies entre -28.0125 et 84.2875, ont également été identifiées. Cette analyse pourrait être utile pour comprendre la variabilité et la présence éventuelle d'outliers dans l'ensemble de données.

Analyse des Valeurs Atypiques :

- Nombre total de valeurs atypiques : 32

Les valeurs atypiques dans votre ensemble de données concernent principalement la colonne "price_numeric". Voici quelques exemples de produits avec des prix considérés comme atypiques :

1. Product ID 4402 - Prix : 176.0
2. Product ID 4355 - Prix : 126.5
3. Product ID 4352 - Prix : 225.0 (out of stock)
4. Product ID 4904 - Prix : 137.0
5. Product ID 5892 - Prix : 191.3

Il est important de noter que certaines de ces valeurs atypiques coïncident avec des produits qui sont en rupture de stock. Cela pourrait être un aspect intéressant à explorer davantage, car cela peut avoir un impact sur la perception des clients et les décisions d'achat. Une gestion appropriée des prix et des stocks pour ces produits pourrait être envisagée pour optimiser la performance globale de la boutique en ligne.

In [ ]:
```
import numpy as np

# Supposons que vous ayez déjà importé votre ensemble de données dans la variable '

# Filtrer les valeurs supérieures à 0
resultat_final_positives = resultat_final[resultat_final['price_numeric'] > 0]

# Calcul des quartiles pour les valeurs supérieures à 0
Q1, Q3 = np.percentile(resultat_final_positives['price_numeric'], [25, 75])

# Calcul de l'IQR pour les valeurs supérieures à 0
IQR = Q3 - Q1

# Calcul des bornes pour les valeurs atypiques
borne_inf = Q1 - 1.5 * IQR
borne_sup = Q3 + 1.5 * IQR

# Calcul de la médiane pour les valeurs supérieures à 0
median_positif = np.median(resultat_final_positives['price_numeric'])

# Affichage des résultats
print(f"Borne inférieure : {borne_inf:.2f}")
print(f"Borne supérieure : {borne_sup:.2f}")
print(f"Médiane : {median_positif:.2f}")
```

```
Borne inférieure : -28.01
Borne supérieure : 84.29
Médiane : 23.55
```

Résumé de l'Analyse :

- Borne inférieure : -28.01

- Borne supérieure : 84.29
- Médiane : 23.55

Ces résultats délimitent les valeurs potentiellement atypiques des prix dans notre ensemble de données, avec une borne inférieure de -28.01, une borne supérieure de 84.29, et une médiane de 23.55. Nous utilisons ces seuils pour identifier et examiner les prix qui pourraient nécessiter une attention particulière.

In [ ]:
```
# Filtrer les valeurs supérieures à 0
resultat_final_positives = resultat_final[resultat_final['price_numeric'] > 0]

# Affichage des résultats
print(resultat_final_positives[['product_id', 'stock_status', 'price_numeric']])
```

```
     product_id stock_status  price_numeric
388        4334   outofstock           49.0
143        4144      instock           49.0
436        4402      instock          176.0
141        4142      instock           53.0
138        4141      instock           39.0
...         ...          ...            ...
1216       5932      instock           59.9
1383       6628      instock           32.2
1221       5951      instock           74.5
1223       5956      instock           17.2
1426       7338      instock           16.3

[714 rows x 3 columns]
```

In [ ]:
```
# Filtrer les valeurs inférieures à 0
resultat_final_negatives = resultat_final[resultat_final['price_numeric'] < 0]

# Vérifier si le DataFrame filtré n'est pas vide
if not resultat_final_negatives.empty:
    # Calcul des quartiles pour les valeurs inférieures à 0
    Q1, Q3 = np.percentile(resultat_final_negatives['price_numeric'], [25, 75])

    # Calcul de l'IQR pour les valeurs inférieures à 0
    IQR = Q3 - Q1

    # Calcul des bornes pour les valeurs atypiques
    borne_inf = Q1 - 1.5 * IQR
    borne_sup = Q3 + 1.5 * IQR

    # Calcul de la médiane pour les valeurs inférieures à 0
    median_negatif = np.median(resultat_final_negatives['price_numeric'])

    # Affichage des résultats
    print(f"Borne inférieure : {borne_inf:.2f}")
    print(f"Borne supérieure : {borne_sup:.2f}")
    print(f"Médiane : {median_negatif:.2f}")
else:
    print("Aucune valeur négative dans 'price_numeric'.")
```

```
Aucune valeur négative dans 'price_numeric'.
```

In [ ]:
```
# Filtrer les valeurs inférieures à 0
resultat_final_negatives = resultat_final[resultat_final['price_numeric'] < 0]

# Affichage des résultats
print(resultat_final_negatives[['product_id', 'stock_status', 'price_numeric']])
```

```
Empty DataFrame
Columns: [product_id, stock_status, price_numeric]
Index: []
```

In [ ]:
```
# Filtrer les valeurs supérieures à 0
ventes_sup_0 = resultat_final[resultat_final['ca'] > 0]

# Statistiques descriptives des ventes supérieures à 0
ventes_stats_sup_0 = ventes_sup_0['ca'].describe()

# Affichage des statistiques descriptives pour les ventes supérieures à 0
print(ventes_stats_sup_0)
```

```
count     385.000000
mean      183.295065
std       400.324073
min         6.500000
25%        38.600000
50%        81.600000
75%       164.400000
max      4704.000000
Name: ca, dtype: float64
```

Résumé des Statistiques du Chiffre d'Affaires :

- Nombre d'observations : 385
- Moyenne du chiffre d'affaires : 183.30
- Écart type : 400.32
- Chiffre d'affaires minimum : 6.50
- Premier quartile (Q1) : 38.60
- Médiane (50%) : 81.60
- Troisième quartile (Q3) : 164.40
- Chiffre d'affaires maximum : 4704.00

Ces statistiques offrent un aperçu détaillé de la distribution du chiffre d'affaires, mettant en évidence la centralité, la dispersion, ainsi que les valeurs minimales et maximales. La moyenne élevée et l'écart type important indiquent une variabilité significative dans les performances financières, avec des observations allant de valeurs basses à très élevées.

IV] Analyse Exploratoire des Données (AED) pour le Chiffre d'Affaires des Produits

In [ ]:
```
# Définir les points de découpe pour la segmentation
ca_bins = [0, 100, 500, float('inf')]
ca_labels = ['Bas', 'Moyen', 'Haut']

# Ajouter une colonne 'categorie_ca' basée sur la segmentation par chiffre d'affair
resultat_final['categorie_ca'] = pd.cut(resultat_final['ca'], bins=ca_bins, labels=

# Analyse des ventes et des prix par segment de chiffre d'affaires
segmentation_ca_stats = resultat_final.groupby('categorie_ca').agg({'ca': 'mean', '

# Affichage des statistiques de segmentation par chiffre d'affaires
print(segmentation_ca_stats)
```

```
  categorie_ca           ca  price_numeric
0          Bas    45.171889      21.321198
1        Moyen   193.386331      27.611151
2         Haut  1168.468966      61.006897
```

Résumé des Statistiques par Catégorie :

Catégorie "Bas" :

- CA moyen : 45.17
- Prix moyen : 21.32

Catégorie "Moyen" :

- CA moyen : 193.39
- Prix moyen : 27.61

Catégorie "Haut" :

- CA moyen : 1168.47
- Prix moyen : 61.01

Des différences significatives entre les catégories sont observées, avec des chiffres d'affaires et des prix moyens variant considérablement. La catégorie "Haut" se distingue par des performances plus élevées, tandis que la catégorie "Bas" affiche des valeurs plus modestes.

In [ ]:
```
#Analyse des produits les plus vendus
# Top 10 des produits avec le chiffre d'affaires le plus élevé
top_products = resultat_final.nlargest(10, 'ca')[['product_id', 'ca', 'price_numeri

# Affichage des résultats
print("Top 10 des Produits les Plus Vendus :")
print(top_products)
```

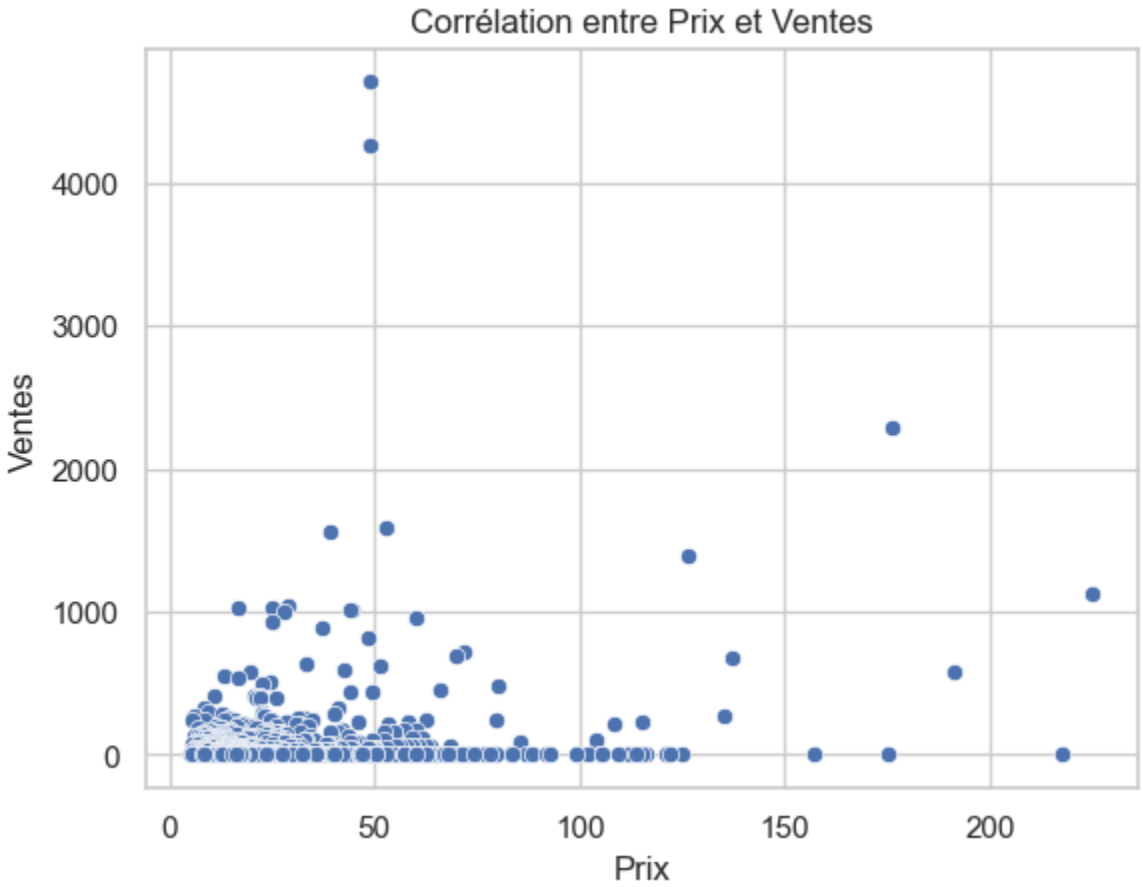
```
Top 10 des Produits les Plus Vendus :
      product_id      ca  price_numeric  stock_quantity
388         4334  4704.0           49.0               0
143         4144  4263.0           49.0              11
436         4402  2288.0          176.0               8
141         4142  1590.0           53.0               8
138         4141  1560.0           39.0               1
402         4355  1391.5          126.5               2
399         4352  1125.0          225.0               0
158         4153  1044.0           29.0               0
1303        6206  1033.2           25.2             120
61          4068  1029.2           16.6             157
```

Analyse Synthétique des Top 10 Produits les Plus Vendus :

- Diversité de Performances :Les produits varient considérablement en termes de chiffre d'affaires, de prix et de disponibilité en stock.
- Épuisement Rapide : Certains produits génèrent un CA élevé malgré des stocks épuisés, suggérant une forte demande initiale.
- Prix Moyens Élevés : Plusieurs produits affichent des prix moyens élevés, contribuant significativement au chiffre d'affaires.
- Stock Limité : Certains produits ont des stocks limités, ce qui peut influencer la stratégie de gestion des stocks.
- Dynamique de Prix : Des variations importantes dans les prix suggèrent des stratégies de tarification différentes pour optimiser les ventes.

Cette analyse fournit un aperçu rapide des performances des produits les plus vendus, offrant des indications sur la demande, la tarification et la gestion des stocks.

In [ ]:
```
# Analyse de la corrélation entre les ventes et les prix
sns.scatterplot(x='price_numeric', y='ca', data=resultat_final)
plt.title('Corrélation entre Prix et Ventes')
plt.xlabel('Prix')
plt.ylabel('Ventes')
plt.show()

# Exemple de comparaison de performances entre différentes catégories
performances_categories = resultat_final.groupby('product_id').agg({'ca': 'sum', 'p

# Affichage des performances par catégorie
print(performances_categories)
```




```
     product_id     ca  price_numeric
0          3847  145.2           24.2
1          3849    0.0           34.3
2          3850    0.0           20.8
3          4032   42.3           14.1
4          4039    0.0           46.0
..          ...    ...            ...
709        6930   42.0            8.4
710        7023    0.0           27.5
711        7025    0.0           69.0
712        7247    0.0           54.8
713        7338    0.0           16.3

[714 rows x 3 columns]
```

Les résultats indiquent des informations sur le chiffre d'affaires (CA) et les prix pour

différents produits identifiés par leur "product_id". Voici un résumé synthétique :

- Diversité des Performances :Certains produits génèrent du chiffre d'affaires, tandis que d'autres n'enregistrent aucune vente (CA de 0.0).
- Variabilité des Prix : Les prix varient significativement d'un produit à l'autre, allant de 8.4 à 69.0.
- Potentiel d'Optimisation : Les produits avec un CA nul pourraient nécessiter une attention particulière pour comprendre les raisons de leur faible performance.
- Segmentation de Prix :La diversité des prix suggère une segmentation du marché, avec des produits adaptés à différentes gammes de prix.

Cette analyse offre un aperçu de la variabilité des performances des produits, soulignant la nécessité de stratégies différenciées en fonction de la demande et des prix.

On remarque plus de vente pour les produits ayant un prix situé entre 50 et 100.

Conclusion de l'analyse des données

L'examen approfondi de l'ensemble de données dévoile une diversité marquée dans les performances des produits de la boutique en ligne. Les aspects tels que la variabilité des prix, la présence de valeurs atypiques, et la catégorisation par niveau de chiffre d'affaires et de prix fournissent des perspectives clés. Une vigilance particulière est recommandée pour les produits atypiques en rupture de stock, ayant généré un chiffre d'affaires significatif. Cette approche ciblée peut optimiser la gestion des prix, des stocks, et ajuster la stratégie commerciale, favorisant ainsi une amélioration globale des performances de la boutique en ligne.

Il est également à noter la détection d'anomalies, telles que la présence de bons cadeaux dans la colonne "sku". Ces bons devraient être réaffectés à une modalité de paiement spécifique pour assurer une structuration plus cohérente des données.

En ce qui concerne le système post_parent, je ne dispose pas d'informations sur son utilisation au sein de l'entreprise. En cas de non-utilisation, il pourrait être intéressant d'explorer cet outil pour une meilleure gestion des catégories de produits. De plus, son utilisation pourrait être bénéfique pour le rating, permettant ainsi de filtrer par commentaire ou d'observer la moyenne des commentaires, par exemple, pour les vins. Il serait également possible d'effectuer une analyse de texte, comme un nuage de mots, afin de déterminer la qualité perçue des vins. Cela pourrait aider à identifier des problèmes potentiels de logistique, tels que des retards mentionnés dans les commentaires. En résumé, l'utilisation du système post_parent pourrait permettre d'établir des corrélations entre les notations des produits et les commentaires associés.

On pourrait enrichir notre ensemble de données en ajoutant des colonnes telles que "moyen de paiement", "statut de la commande" et "date de vente". Ces nouvelles colonnes amélioreraient la lisibilité des différentes méthodes de paiement et préviendraient les potentielles erreurs, par exemple, liées aux bons cadeaux. En ce qui concerne le statut de la commande, l'ajout de cette colonne permettrait d'obtenir des retours sur les délais de livraison et l'état des produits. Enfin, l'introduction de la colonne "date de vente" faciliterait une segmentation temporelle plus fine, favorisant ainsi une meilleure analyse des données dans le temps.

Par aillieurs, il y deux colonne price, il serait judicieux d'en concerver qu'une seule pour une millieure lecture des donnnées.

En conclusion, l'incorporation de ces éléments permettrait une meilleure traçabilité de la satisfaction client et une gestion temporelle plus précise.

In [ ]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\rorsharks\appdata\local\program
s\python\python310\lib\site-packages (7.12.0)Note: you may need to restart the kerne
l to use updated packages.

Requirement already satisfied: beautifulsoup4 in c:\users\rorsharks\appdata\local\pr
ograms\python\python310\lib\site-packages (from nbconvert) (4.12.2)
Requirement already satisfied: bleach!=5.0.0 in c:\users\rorsharks\appdata\local\pro
grams\python\python310\lib\site-packages (from nbconvert) (6.1.0)
Requirement already satisfied: defusedxml in c:\users\rorsharks\appdata\local\progra
ms\python\python310\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\users\rorsharks\appdata\local\progr
ams\python\python310\lib\site-packages (from nbconvert) (3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\rorsharks\appdata\roami
ng\python\python310\site-packages (from nbconvert) (5.5.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\rorsharks\appdata\loc
al\programs\python\python310\lib\site-packages (from nbconvert) (0.3.0)
Requirement already satisfied: markupsafe>=2.0 in c:\users\rorsharks\appdata\local\p
rograms\python\python310\lib\site-packages (from nbconvert) (2.1.3)
Requirement already satisfied: mistune<4,>=2.0.3 in c:\users\rorsharks\appdata\loca
l\programs\python\python310\lib\site-packages (from nbconvert) (3.0.2)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\rorsharks\appdata\local\p
rograms\python\python310\lib\site-packages (from nbconvert) (0.9.0)
Requirement already satisfied: nbformat>=5.7 in c:\users\rorsharks\appdata\local\pro
grams\python\python310\lib\site-packages (from nbconvert) (5.9.2)
Requirement already satisfied: packaging in c:\users\rorsharks\appdata\local\program
s\python\python310\lib\site-packages (from nbconvert) (21.3)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\rorsharks\appdata\lo
cal\programs\python\python310\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in c:\users\rorsharks\appdata\roamin
g\python\python310\site-packages (from nbconvert) (2.16.1)
Requirement already satisfied: tinycss2 in c:\users\rorsharks\appdata\local\program
s\python\python310\lib\site-packages (from nbconvert) (1.2.1)
Requirement already satisfied: traitlets>=5.1 in c:\users\rorsharks\appdata\roaming\
python\python310\site-packages (from nbconvert) (5.13.0)
Requirement already satisfied: six>=1.9.0 in c:\users\rorsharks\appdata\local\progra
ms\python\python310\lib\site-packages (from bleach!=5.0.0->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in c:\users\rorsharks\appdata\local\prog
rams\python\python310\lib\site-packages (from bleach!=5.0.0->nbconvert) (0.5.1)
Requirement already satisfied: platformdirs>=2.5 in c:\users\rorsharks\appdata\roami
ng\python\python310\site-packages (from jupyter-core>=4.7->nbconvert) (3.11.0)
Requirement already satisfied: pywin32>=300 in c:\users\rorsharks\appdata\roaming\py
thon\python310\site-packages (from jupyter-core>=4.7->nbconvert) (306)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\rorsharks\appdata\
roaming\python\python310\site-packages (from nbclient>=0.5.0->nbconvert) (8.5.0)
Requirement already satisfied: fastjsonschema in c:\users\rorsharks\appdata\local\pr
ograms\python\python310\lib\site-packages (from nbformat>=5.7->nbconvert) (2.19.0)
Requirement already satisfied: jsonschema>=2.6 in c:\users\rorsharks\appdata\local\p
rograms\python\python310\lib\site-packages (from nbformat>=5.7->nbconvert) (4.20.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\rorsharks\appdata\local\pro
grams\python\python310\lib\site-packages (from beautifulsoup4->nbconvert) (2.5)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\rorsharks\appdat
a\local\programs\python\python310\lib\site-packages (from packaging->nbconvert) (3.
0.9)
Requirement already satisfied: attrs>=22.2.0 in c:\users\rorsharks\appdata\local\pro
grams\python\python310\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbcon
vert) (23.1.0)
```

```
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\rors
harks\appdata\local\programs\python\python310\lib\site-packages (from jsonschema>=2.
6->nbformat>=5.7->nbconvert) (2023.11.2)
Requirement already satisfied: referencing>=0.28.4 in c:\users\rorsharks\appdata\loc
al\programs\python\python310\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7-
>nbconvert) (0.32.0)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\rorsharks\appdata\local\pr
ograms\python\python310\lib\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbco
nvert) (0.14.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\rorsharks\appdata\
local\programs\python\python310\lib\site-packages (from jupyter-client>=6.1.12->nbcl
ient>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\rorsharks\appdata\roaming\pyt
hon\python310\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconver
t) (25.1.1)
Requirement already satisfied: tornado>=6.2 in c:\users\rorsharks\appdata\roaming\py
thon\python310\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconver
t) (6.3.3)

[notice] A new release of pip is available: 23.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```