bottleneck
fine wine spirit

Optimisez la gestion des données de la boutique avec Python

Elodie Mendes
Data Analyst

Analyses Exploratoires des Données: Données ERP de l'entreprise

```
df_erp.info()
```

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

Traitement du type de colonne pour les prix de vente et prix d'achat

```
#Conversion des colonnes 'price' et 'purchase_price' en type float (Elodie)
df_erp['price'] = df_erp['price'].str.replace(',','.').astype(float)
df_erp['purchase_price'] = df_erp['purchase_price'].str.replace(',','.').astype(float)
```

Vérification des doublons de la colonne clé 'product_id'

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

Traitement de l'incohérence des données sur la colonne 'stock_status'

```
df_erp.loc[df_erp['stock_status'] != df_erp["stock_status_2"],:]
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	stock_status_2
4	4039	1	46.0	3	outofstock	23.77	instock
398	4885	1	18.7	0	instock	9.66	outofstock

```
#Corriger la ou les données incohérentes
df_erp['stock_status'] = df_erp['stock_quantity'].apply(lambda x: 'outofstock' if x <= 0 else 'instock')

#Verification en utilisant le même code que plus haut pour afficher les problemes
print(df_erp.loc[df_erp['stock_status'] != df_erp["stock_status_2"],:])
```

```
#si la valeur de la colonne "stock_quantity" est nulle renseigner "outofstock" sinon mettre "instock"
df_erp['stock_status_2'] = np.where(df_erp['stock_quantity'] <= 0, 'outofstock', 'instock')
df_erp
```

	product_id	onsale_web	price	stock_quantity	stock_status	purchase_price	stock_status_2
0	3847	1	24.2	16	instock	12.88	instock

Analyses Exploratoires des Données: Données ERP de l'entreprise

Traitement des prix de vente négatifs

```
#Afficher les prix inférieurs à 0 (qu'est ce qu'i
df_erp.loc[df_erp['price'] < 0,:]
```

```
Nombres d'article avec un prix non renseignés: 0
Le prix minimum est de -20.0€
Le prix maximum est de 225.0€
```

	product_id	onsale_web	price	stock_quantity	s
151	4233	0	-20.0	0	
469	5017	0	-8.0	0	
739	6594	0	-9.1	19	

Traitement des stocks négatifs

```
#Afficher la quantité minimum de la colonne "stock_quantity"
print("La quantité minimum est {}pcs".format(df_erp['stock_quantity'].min()))
#Afficher la quantité maximum de la colonne "stock_quantity"
print("La quantité maximum est {}pcs".format(df_erp['stock_quantity'].max()))
#Afficher les stocks inférieurs à 0 (qu'est ce qu'il faut en faire ?)
df_erp.loc[df_erp['stock_quantity'] < 0,:]
```

```
La quantité minimum est -10pcs
La quantité maximum est 145pcs
```

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

Je garde les stocks négatifs. C'est un indicateur pour effectuer un réapprovisionnement sur le produit en vente sur le site.

Dataset après traitement

```
df_erp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   product_id      825 non-null    int64
 1   onsale_web      825 non-null    int64
 2   price           825 non-null    float64
 3   stock_quantity  825 non-null    int64
 4   purchase_price  825 non-null    float64
dtypes: float64(2), int64(3)
memory usage: 32.4 KB
```

Analyses Exploratoires des Données: Site E-commerce

```
df_web.info()
```

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

Dataset après traitement

```
df_web.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 714 entries, 0 to 1391
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   sku               714 non-null    object
 1   total_sales       714 non-null    float64
 2   tax_status        361 non-null    object
 3   post_date_gmt     714 non-null    object
 4   product_type      714 non-null    object
 5   post_title        714 non-null    object
 6   post_status       714 non-null    object
 7   post_modified_gmt  714 non-null   object
dtypes: float64(1), object(7)
```

Traitement des colonnes sans aucune données

```
 7   tax_class              0 non-null      float64
 11  post_content           0 non-null      float64
 22  post_content_filtered  0 non-null      float64
```

Traitement des colonnes non conforme au RGPD

```
 18  post_password          0 non-null      float64
```

Suppression des colonnes non utile à l'analyse

```
#Si vous avez défini des colonnes à supprimer, effectuer l'opération
df_web.drop(columns=(['virtual','downloadable','rating_count','average_rating','tax_class','post_date','post_modified','post_author',
                      'post_content','post_name','post_password','post_excerpt','post_content_filtered','post_parent','guid','post_type','menu_order',
                     ['post_mime_type','ping_status','comment_status','comment_count']),inplace=True)
```

Traitement des données de la colonne 'SKU'

```
#Si vous avez identifié des codes articles ne re
df_web['sku'].str.isdigit().value_counts()
```

```
sku
True     1424
False       4
Name: count, dtype: int64
```

Il y aurait 4 codes SKU qui ne sont pas de type numériqu

```
#Pour les codes articles identifiés, réalisé une analyse et définissez l'action à entreprendre
df_faux_sku = df_web[df_web['sku'].apply(lambda x: not str(x).isdigit())]
print(df_faux_sku['sku'].unique())
```

```
[nan '13127-1' 'bon-cadeau-25-euros']
```

Je garde les codes articles qui ne respectent pas la règle de codification.

Traitement des doublons

```
#La clé pour chaque ligne est-elle uniques? ou autrement dit, y a-t-il des doublons?
print('Il y a {} clé en doublons'.format(df_web['sku'].duplicated().sum()))
```

```
Il y a 798 clé en doublons
```

```
#Suppression des doublons
df_web.drop_duplicates(subset=['sku'],keep='first',inplace=True)
```

Analyses Exploratoires des Données: Fichier de Liaison

```
df_liaison.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id_web      734 non-null    object
 1   product_id  825 non-null    int64 
dtypes: int64(1), object(1)
```

Vérifications effectuées

```
#Les valeurs de la colonne "product_id" sont elles toutes uniques?
print ('Il y a {} doublons dans la colonne "product id"'.format(df_liaison['product_id'].duplicated().sum()))
```

```
Il y a 0 doublons dans la colonne "product id"
```

```
#Les valeurs de la colonne "id_web" sont-elles toutes uniques?
df_liaison[df_liaison["id_web"].notnull()]["id_web"].unique().size
```

```
734
```

Il y a 734 valeurs uniques pour la colonne 'id_web' dans le fichier liaison.

```
#Avons-nous des articles sans correspondances?
df_liaison['id_web'].isnull().sum()
```

```
91
```

Il y a 91 articles sans correspondances dans le fichier liaison.

Fusion ou consolidations des données

Jonction du fichier df erp et df liaison

```
#Fusion des fichiers df_erp et df_liaison
df_merge= pd.merge(df_liaison,df_erp, on='product_id', how='outer')
df_merge
```

	id_web	product_id	onsale_web	price	stock_quantity	purchase_price
0	15298	3847	1	24.2	16	12.88
1	15296	3849	1	34.3	10	17.54
2	15300	3850	1	20.8	0	10.64
3	19814	4032	1	14.1	26	6.92
4	19815	4039	1	46.0	3	23.77

```
df_merge.isnull().sum()
```

```
id_web             91
product_id          0
onsale_web          0
price               0
stock_quantity      0
purchase_price      0
dtype: int64
```

```
indicator=True)
```

_merge

both

left_only

left_only

Jonction du fichier df merge et df web

```
#Fusionnez les datasets df_merge et df_web
df_merge= pd.merge(df_merge, df_web, left_on='id_web', right_on= 'sku', how='outer')
```

	id_web	product_id	onsale_web	price	stock_quantity	purchase_price	sku	total_sales	tax_status	post_date_gmt	product_type	post_title
0	10014	5913	1	36.0	7	17.16	10014	10.0	taxable	2019-04-04 13:45:23	Gin	Darnley's London Dry Gin Original
1	10459	4617	1	67.5	6	35.57	10459	4.0	taxable	2018-04-13 13:58:19	Vin	Alphonse Mellot Sancerre Rouge Génération XIX ...

```
#Avons-nous des lignes sans correspondances
df_merge.isnull().sum()
```

```
id_web              91
product_id           0
onsale_web           0
price                0
stock_quantity       0
purchase_price       0
sku                111
total_sales        111
tax_status         464
post_date_gmt      111
product_type       111
post_title         111
post_status        111
post_modified_gmt  111
dtype: int64
```

Il y a des lignes sans correspondances.

Analyses univariées du prix

Exploration par la méthode statistique Z- Score

```
#Calculer la moyenne du prix
df_merge['price'].mean()
print("La moyenne des prix de vente est de {}€".format(round(df_merge['price'].mean(),2)))
#Calculer l'écart-type du prix
df_merge['price'].std()
print("L'écart-type des prix de vente est de {}".format(round(df_merge['price'].std(),2)))
#Calculer le Z-score
df_merge['Z_score'] = round((df_merge['price'] - df_merge['price'].mean(skipna=True))/df_merge['price'].std(),2)
```

```
La moyenne des prix de vente est de 32.19€
L'écart-type des prix de vente est de 26.71
```

```
#Quel est le seuil prix dont z-score est supérieur à 3?
df_merge.loc[df_merge['Z_score'] > 3,]['price'].min()
print("Pour un Z-score supérieur à 3,Le seuil minimun du prix est de {}€".
      format(df_merge.loc[df_merge['Z_score'] > 3,]['price'].min()))
```

```
Pour un Z-score supérieur à 3,Le seuil minimun du prix est de 114.0€
```

Boxplot des prix de vente au mois d'octobre

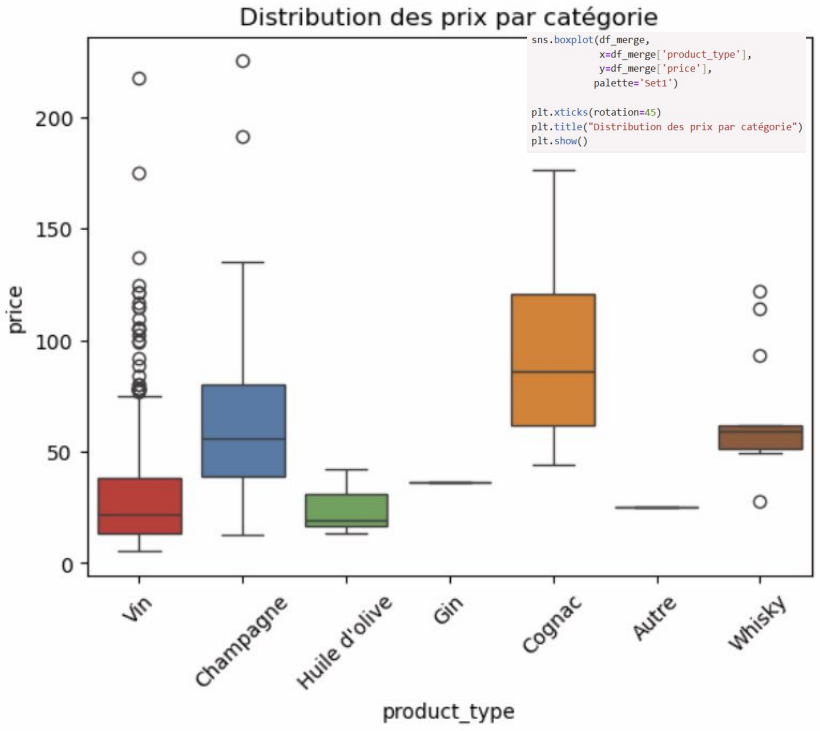
```
#Autre méthode avec plotly express
px.box(df_merge,
       x='price',
       labels={'price':'Prix de Vente en Euros'},
       title= "Boxplot des prix de vente au mois d'octobre")
```

Exploration par la méthode statistique : Intervalle interquartile

```
#Définissez un seuil pour les articles "outliers" en prix

#Calcul du 1er quantile (Elodie)
Q1 = df_merge['price'].quantile(0.25)
print("25% des prix de vente sont en dessous de {}€".format(round(Q1,2)))

#Calcul du 3ème quantile (Elodie)
Q3 = df_merge['price'].quantile(0.75)
print("75% des prix de vente sont en dessous de {}€".format(round(Q3,2)))

#Calcul de l'intervalle interquartile (Elodie)
IQR = Q3 - Q1
print("L'intervalle interquartile est de {}".format(round(IQR,2)))

#Seuil du prix inférieur
Seuil_prix_inf = Q1 -1.5 * IQR
print("Le seuil du prix inférieur est de {}€".format(round(Seuil_prix_inf,2)))

#Seuil du prix supérieur
Seuil_prix_sup = Q3 +1.5 * IQR
print("Le seuil du prix supérieur est de {}€".format(round(Seuil_prix_sup,2)))
```

```
25% des prix de vente sont en dessous de 14.5€
75% des prix de vente sont en dessous de 42.0€
L'intervalle interquartile est de 27.5
Le seuil du prix inférieur est de -26.75€
Le seuil du prix supérieur est de 83.25€
```

Analyses univariées du prix




```
sns.boxplot(df_merge,
            x=df_merge['product_type'],
            y=df_merge['price'],
            palette='Set1')

plt.xticks(rotation=45)
plt.title("Distribution des prix par catégorie")
plt.show()
```

Dataset des articles avec des prix "extrêmes"

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

#Dataframe des prix outliers inférieurs et supérieurs (Elodie)
outliers = df_merge[(df_merge['price'] < Seuil_prix_inf) | (df_merge['price'] > Seuil_prix_sup)]

#Afficher le nombre d'articles du catalogue "outliers" (Elodie)

print("Il y a {} articles 'outliers'".format(outliers['product_id'].shape[0]))

#la proportion de l'ensemble du catalogue "outliers"
print("La proportion des articles 'outliers' est de {}% sur le total des articles".
      format(round(outliers['product_id'].shape[0]/df_merge['product_id'].shape[0]*100,2)))
```

```
Il y a 36 articles 'outliers'
La proportion des articles 'outliers' est de 4.36% sur le total des articles
```

```
#Analyse des ventes pour les spiritieux dont les prix sont des "valeurs extrêmes"
outliers.groupby(['product_type','price'])[['total_sales']].sum()
```

		total_sales
product_type	price	
Champagne	85.6	7.0
	86.8	9.0
	112.0	6.0
	135.0	5.0
	191.3	6.0
	225.0	11.0

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations

Analyse des ventes en Chiffre d'Affaires

```
#Créez une colonne calculant le CA par article
df_merge['CA']= df_merge['price'] * df_merge['total_sales']
#Calculez la somme de la colonne "ca_par_article"
print("Le CA du mois d'octobre est de {}€".format(round(df_merge['CA'].sum(),2)))
```

```
Le CA du mois d'octobre est de 153748.1€
```

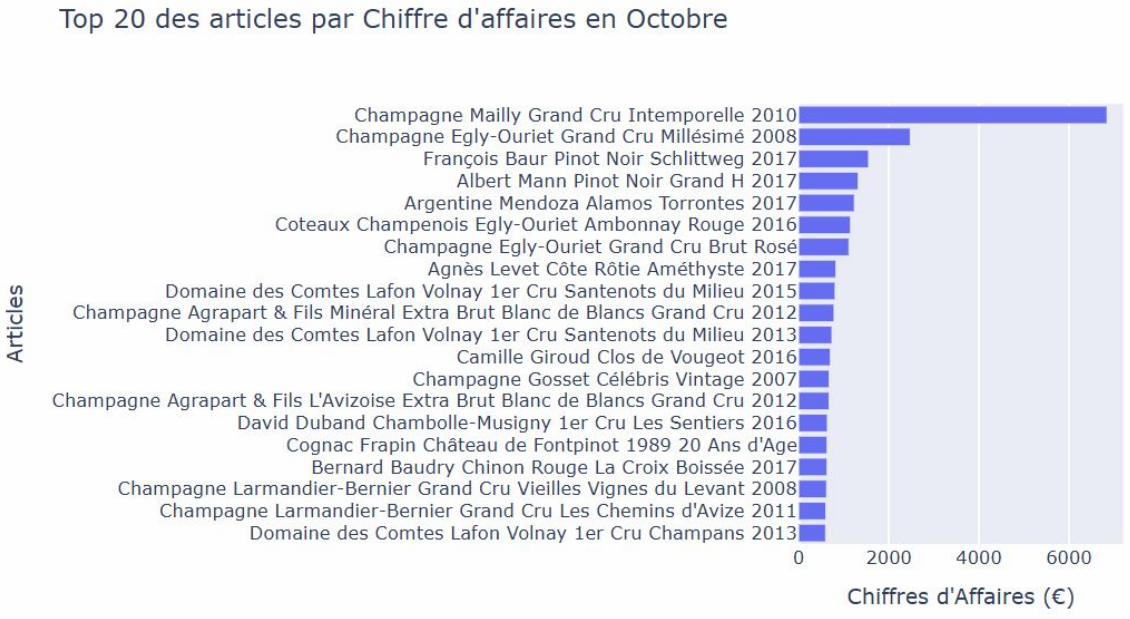
```
#Calcul du chiffre d'affaires par catégorie de produit
CA_par_catégorie = df_merge.groupby(['product_type']) [['CA']].sum(
CA_par_catégorie
```

	product_type	CA
0	Autre	175.0
1	Champagne	19418.6
2	Cognac	3170.2
3	Gin	504.0
4	Huile d'olive	497.7
5	Vin	127096.0
6	Whisky	2886.6

```
#Graphique représentant le CA par catégorie de produit
px.bar(CA_par_catégorie.sort_values(by='CA',ascending= False),
       x= 'product_type',
       y='CA',
       title="Chiffre d'affaires par catégorie de produit au mois d'octobre")
```

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations

Analyse des ventes en Chiffre d'Affaires




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

#Effectuer le tri dans l'ordre décroissant du CA du dataset df_merge
df_merge.sort_values(by=['CA'], ascending=False, inplace=True)

#Réinitialiser l'index du dataset par un reset_index
df_merge.reset_index(drop=True, inplace=True)
#Afficher les 20 premier articles en CA
Top_20_CA = df_merge.head(20)
#Graphique en barre des 20 premiers articles avec plotly express
px.bar(df_merge[:20][::-1] ,
       y= 'post_title' ,
       x= "CA",
       orientation = 'h',
       title= "Top 20 des articles par Chiffre d'affaires en Octobre",
       labels={'post_title':'Articles','CA':"Chiffres d'Affaires (€)"}, height=470
```

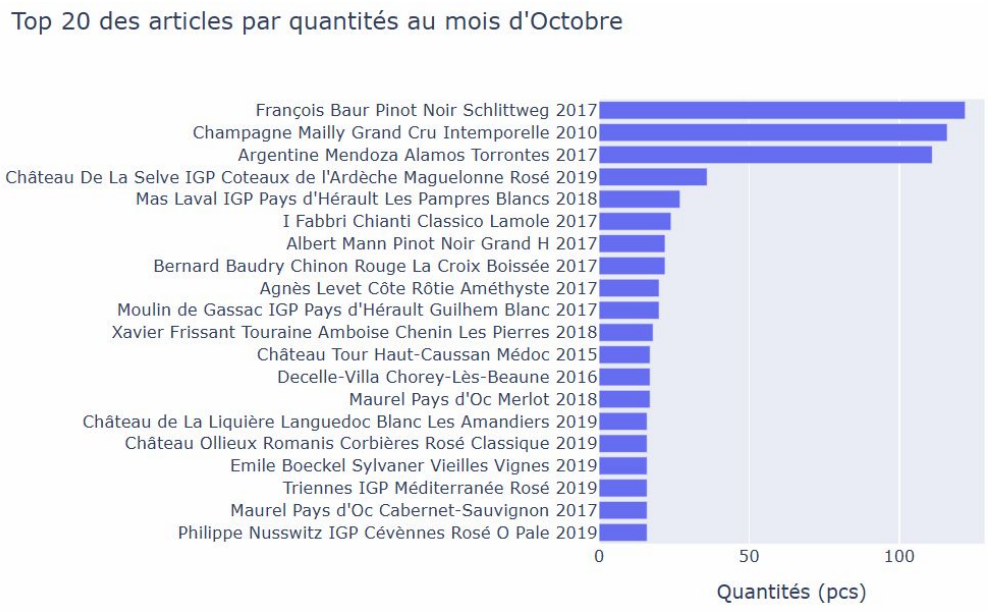
```
############################
# Calculer le 20 / 80 en CA #
############################

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

#Créer une colonne réalisant la somme cumulative de la colonne précedemment créée
df_merge['CA_cumulé'] = df_merge['CA'].cumsum()
df_merge['Proportion_CA_cumulé'] = df_merge['CA_cumulé'] / df_merge['CA'].sum()

#Grâce au deux colonnes créées précedemment, calculer le nombre d'articles représentant 80% du CA
Nb_articles_80_CA = df_merge.loc[df_merge['Proportion_CA_cumulé']<= 0.8,:].shape[0]
print("{} articles représentent 80% du CA".format(Nb_articles_80_CA))

Total_articles = df_merge.shape[0]  #Nb total d'articles

#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier du site web
print("Ces articles représentent {}% de l'offre ".format(
    round((Nb_articles_80_CA/Total_articles)*100,2)))
```

```
420 articles représentent 80% du CA
Ces articles représentent 50.91% de l'offre
```

	post_title	price	total_sales
0	Champagne Mailly Grand Cru Intemporelle 2010	59.0	116.0
1	Champagne Egly-Ouriet Grand Cru Millésimé 2008	225.0	11.0

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations

Analyse des ventes en Quantités




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

#Réinitialiser l'index du dataset par un reset_index
df_merge.reset_index(drop=True, inplace=True)
#Afficher les 20 premier articles en quantité
Top_20_quantite = df_merge.head(20)
#Graphique en barre des 20 premiers articles avec plotly express
px.bar(df_merge[:20][::-1],
       y='post_title',
       x= 'total_sales',
       title= "Top 20 des articles par quantités au mois d'Octobre",
       labels={'post_title':'Articles','total_sales':'Quantités (pcs)'}, height=500)
```

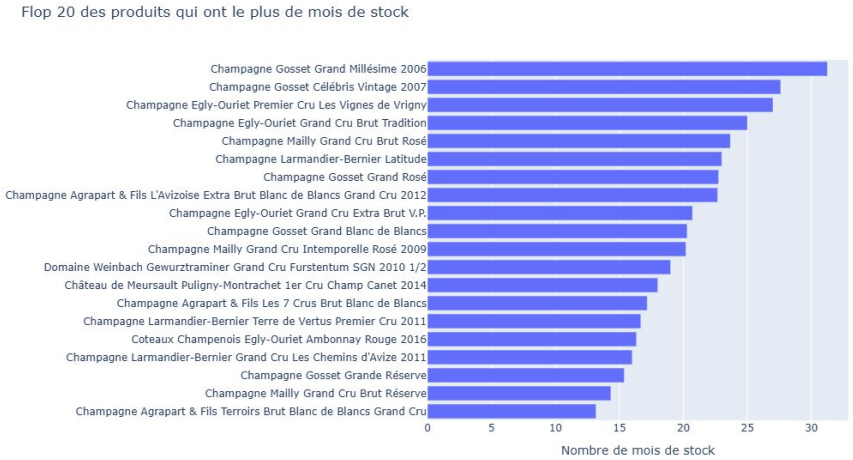
```python
#Créer une colonne calculant la part en quantité de la ligne dans le dataset
#Créer une colonne réalisant la somme cumulative de la colonne précedemment créée
df_merge['total_sales_cumulé'] = df_merge['total_sales'].cumsum()
df_merge['total_sales_cumulé_%'] = df_merge['total_sales_cumulé'] / df_merge['total_sales'].sum()

#Grâce au deux colonnes créées précedemment, calculer le nombre d'articles représentant 80% des ventes en quantité
df_merge.loc[df_merge['total_sales_cumulé_%'] <= 0.8,:].shape[0]
print("{} articles représentent 80% des ventes en quantités".format(df_merge.loc[df_merge['total_sales_cumulé_%'] <= 0.8,:].shape[0]))

#Afficher la proportion que représentent ce groupe d'articles dans le catalogue entier du site web
print("{}% des articles du catalogue représentent 80% des ventes en quantité".format
      (round((df_merge.loc[df_merge['total_sales_cumulé_%'] <= 0.8,:].shape[0] /df_merge.shape[0])*100,2)))
```

```
424 articles représentent 80% des ventes en quantités
51.39% des articles du catalogue représentent 80% des ventes en quantité
```

	post_title	price	total_sales
0	François Baur Pinot Noir Schlittweg 2017	12.7	122.0
1	Champagne Mailly Grand Cru Intemporelle 2010	59.0	116.0
2	Argentine Mendoza Alamos Torrontes 2017	11.1	111.0

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations

Analyse des Stocks




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

#Création de la colonne Rotation de stock
df_merge['rotation_stock'] = df_merge['stock_quantity']/df_merge['total_sales']

#Remplacement des "inf" par 0
df_merge['rotation_stock'] = round(df_merge['rotation_stock'].replace([np.inf, -np.inf, np.nan], 0),2)

#Effectuer le tri dans l'ordre décroissant du nombre de mois de stock dans le dataset df_merge
flop_20_stock_mois = df_merge.sort_values(by= ['rotation_stock'],ascending=False).head(20)
#Graphique en barre du flop 20 des produits qui ont le plus de mois de stock
px.bar(flop_20_stock_mois[::-1],
       x= 'rotation_stock',
       y= 'post_title',
       width=1100,
       height= 600,
       title= "Flop 20 des produits qui ont le plus de mois de stock",
       labels={'post_title':'Articles','rotation_stock':'Nombre de mois de stock'})
```

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations

Analyse des Stocks

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

#Calculer la somme de la colonne stock quantity
print("Il y a {} produits en stock".format(df_merge['stock_quantity'].sum()))
```

```
Il y a 17811 produits en stock
```

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

#Création de la colonne Valorisation des stocks en euros
df_merge['valorisation_stock'] = round(df_merge['stock_quantity'] * df_merge['price'],2)

#Calculer la somme de la colonne "Valorisation_stock_euros"
print("La valeur du stock est de {}€".format(df_merge['valorisation_stock'].sum()))
```

```
La valeur du stock est de 531628.8€
```

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations

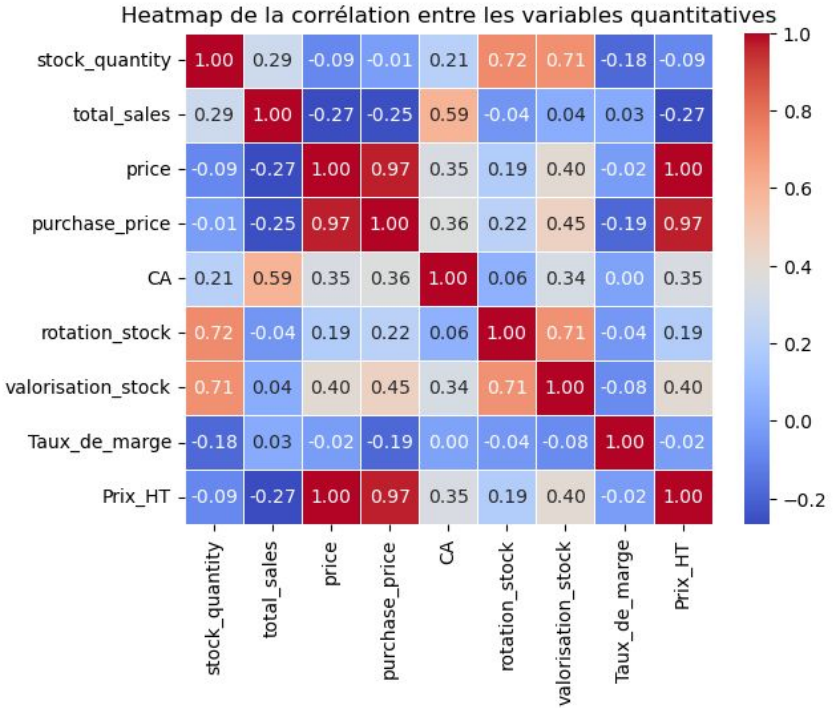
Analyse du Taux de marge moyen par type de produit


```python
#création d'un dataframe avec le taux de marge moyen par type de produit
df_merge_tx_marge_positif.groupby(['product_type'])[['Taux_de_marge']].mean().reset_index()

#Affichage dans un graphique du taux de marge par type de produit
px.bar(df_merge_tx_marge_positif.groupby(['product_type'])[['Taux_de_marge']].mean()
       .reset_index().sort_values(by='Taux_de_marge',ascending=False),
       x= 'product_type',
       y= 'Taux_de_marge',
       title = "Taux de marge moyen par type de produit",
       labels= {'product_type':'Catégorie de produit','Taux_de_marge':'Taux de marge moyen'})
```

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

#Création de la colonne prix HT
df_merge['Prix_HT'] = round(df_merge['price']/ 1.20,2)
#Création de la colonne Taux de marge
df_merge['Taux_de_marge'] = round((df_merge['Prix_HT'] - df_merge['purchase_price'])/df_merge['Prix_HT'],2)

#Afficher le prix minimum de la colonne "taux_marge"
print("Le taux de marge minimum est de {}".format(df_merge['Taux_de_marge'].min()))
#Afficher le prix maximum de la colonne "taux_marge"
print("Le taux de marge maximum est de {}".format(df_merge['Taux_de_marge'].max()))
```

```
Le taux de marge minimum est de -6.35
Le taux de marge maximum est de 1.65
```

Analyses complémentaires
CA, quantités, stocks, taux de marge et corrélations



Analyse des corrélations entre les variables quantitatives

```
#Calcul la matrice de corrélation
df_merge_correlation = df_merge[['stock_quantity','total_sales','price',
                                 'purchase_price','CA','rotation_stock',
                                 'valorisation_stock','Taux_de_marge', 'Prix_HT']].corr()

plt.title("Heatmap de la corrélation entre les variables quantitatives")
sns.heatmap(df_merge_correlation,
            annot=True,
            fmt=".2f",
            cmap = 'coolwarm',
            linewidths=0.5)

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

```
#Calcul la matrice de corrélation
df_merge_correlation = df_merge[['stock_quantity','total_sales','price',
                                 'purchase_price','CA','rotation_stock',
                                 'valorisation_stock','Taux_de_marge', 'Prix_HT']].corr()
```

Actions pour la suite

→ Optimisation de la gestion des données : centraliser les flux de données entre l'ERP de l'entreprise et le site web.

→ Automatiser le contrôle de la qualité des données pour toujours avoir des données à jour et fiable entre l'ERP et le site web.

→ Créer un tableau de bord pour analyser de façon simple les KPIs et mieux piloter la performance commerciale de l'entreprise.

Point sur les compétences apprises

→ L’exploration des données et le traitement des données

→ Le plus difficile a été de se décider quelles variables et valeurs nettoyer sur les données collectées sur le site web.

→ La consolidation des données et la méthodologie de nettoyage des erreurs rencontrées.