Rapport des TPS

visualisation des données

Réalisée par : BELFADLA FATIMA EZZAHRA

I. DATASET

1. Information :

Liste des boissons qui contiennent généralement de la caféine. Certains cas ne sont pas exactement des boissons. Le café moulu ou les feuilles de thé produiraient cette quantité de volume (ml) s'ils étaient préparés selon le fournisseur. Ils n'ont pas de calories puisque vous pouvez contrôler le niveau de sucre.

2. Attributs:

drink: Drink's name.
Volume (ml): Volume quantity.
Calories: Calories quantity.
Caffeine (mg): Caffeine quantity.
type: Drink's type. (Coffe, Energy Drinks, Energy Shots, Soft Drinks, Tea, Water)

3. Source :

https://www.caffeineinformer.com/the-caffeine-database

II. INTERPRETATION

1. TP1 :

- **Charger la dataset "Caffeine",la stocker dans la variable data et afficher les premiers enregistrements**

```
In [19]: data = pd.read_csv("caffeine.csv",encoding='utf-8')
         data.head()
```

Out[19]:

	drink	Volume	Calories	Caffeine	type
0	Costa Coffee	256.993715	0	277	Coffee
1	Coffee Friend Brewed Coffee	250.191810	0	145	Coffee
2	Hell Energy Coffee	250.191810	150	100	Coffee
3	Killer Coffee (AU)	250.191810	0	430	Coffee
4	Nescafe Gold	250.191810	0	66	Coffee

- **Afficher le type de la variable data**

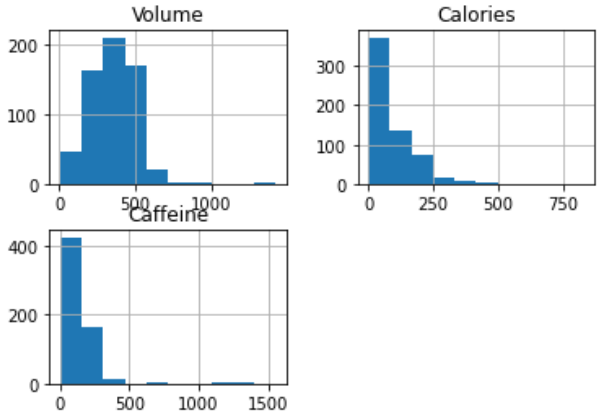
```
In [5]: print(type(data))
        <class 'pandas.core.frame.DataFrame'>
```

- **Pour visualiser la distribution des variables univariées numériques continues nous allons tracer un histogramme.**

  Tous les 3 variables sont univariés - nous sommes intéressés par les distributions individuelles (pas leur relation les unes avec les autres)

```
In [6]: data.hist()
```

```
Out[6]: array([[<AxesSubplot:title={'center':'Volume'}>,
                <AxesSubplot:title={'center':'Calories'}>],
               [<AxesSubplot:title={'center':'Caffeine'}>, <AxesSubplot:>]],
              dtype=object)
```

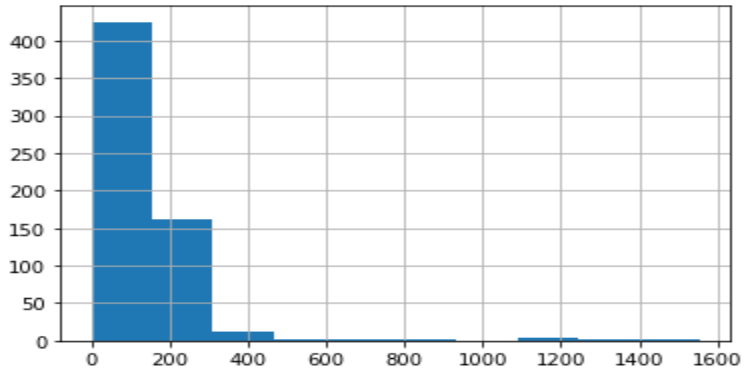


- **Je suis intéressé seulement par la distribution de caffeine pas leur relation avec les autres variables donc je vais me concentrer uniquement sur elle.**

  Le résultat montre que la plupart des valeurs se trouve dans l'intervalle [0 :200]. D'une autre façon la majorité des boissons contient une dose de caféine compris entre 0 et 200. Par contre les boissons contenant 1400 sont peu

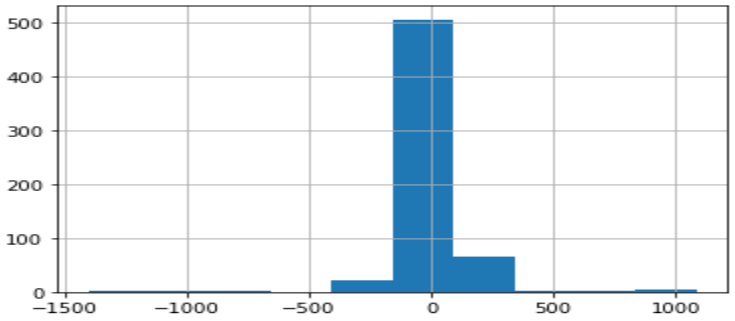
```
[7]: data1 = data['Caffeine']
     data1.hist()
```

```
Out[7]: <AxesSubplot:>
```



- **Si nous voulons produire l'histogramme le plus centré pour une colonne donnée contenant des données numériques continues, c'est ainsi que nous pouvons le faire.**

```
]:  plt.figure();
    data1.diff().hist()
ut[8]: <AxesSubplot:>
```



- **Diviser les données en intervalles appelé bins pour une meilleure visualisation.**

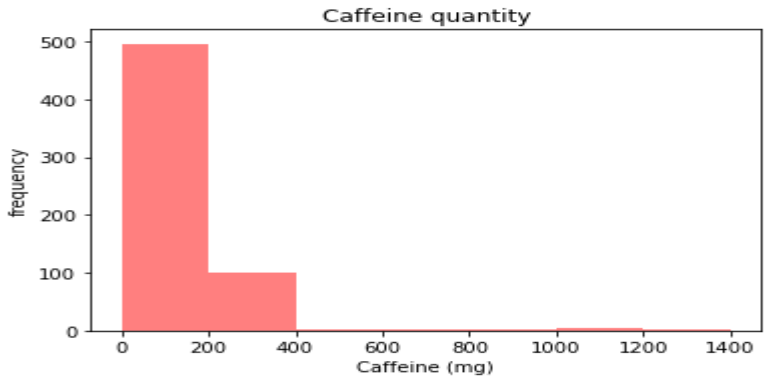
  J'ai fixe mon bin a 200. Les bins se sont les valeurs stockes dans la variable b

Coffee Energy Drinks Energy Shots Soft Drinks Tea Water

```
In [42]: #create bins
         b = np.arange(data1.min(), data1.max(), 200)

In [43]: print(b)
[   0  200  400  600  800 1000 1200 1400]
```

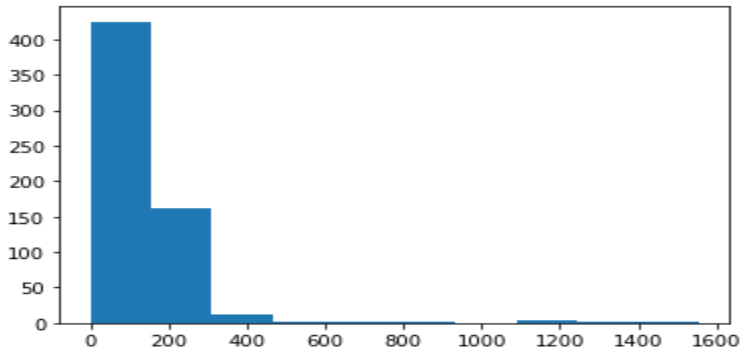
- **Maintenant nous traçons nos données.**

```
:  plt.hist(df1, bins =b, alpha=0.5, color='red', label='Caffeine (mg)')
   plt.title('Caffeine quantity')
   plt.xlabel('Caffeine (mg)')
   plt.ylabel('frequency')
   plt.show()
```



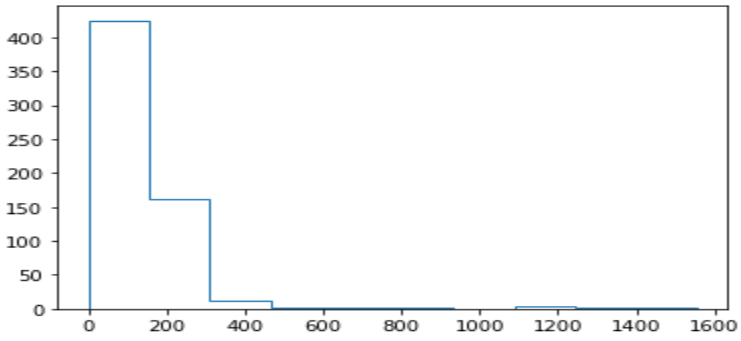
- **Ici j'ai fixe le bin a 10 est voilà le résultat:**

```
[55]: plt.hist(df1, bins =10)

Out[55]: (array([425., 162.,  12.,   1.,   2.,   1.,   0.,   4.,   2.,   1.]),
          array([   0. ,  155.5,  311. ,  466.5,  622. ,  777.5,  933. , 1088.5,
                 1244. , 1399.5, 1555. ]),
          <BarContainer object of 10 artists>)
```



- **Cette figure montre qu'on peut changer le type d'histogramme de field hist en step hist**

```
plt.hist(df1, bins =10, histtype = 'step')

6]: (array([425., 162.,  12.,   1.,   2.,   1.,   0.,   4.,   2.,   1.]),
     array([   0. ,  155.5,  311. ,  466.5,  622. ,  777.5,  933. , 1088.
            1244. , 1399.5, 1555. ]),
     [<matplotlib.patches.Polygon at 0x1db5f067340>])
```



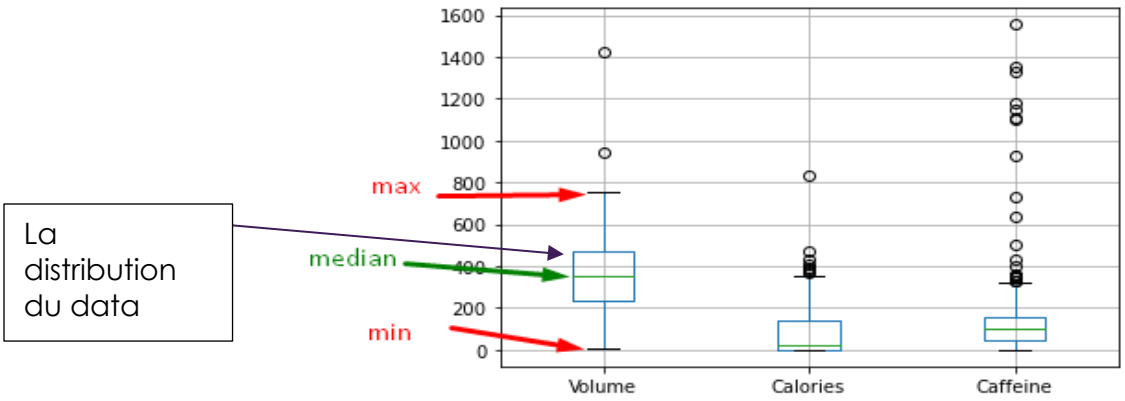
2. TP2 :

boxplot est une autre façon pour visualiser la distribution des variables univariées numériques continues

- **La figure ci-dessus montre comment on construit un boxplot**

```
[4]: data.boxplot()
```

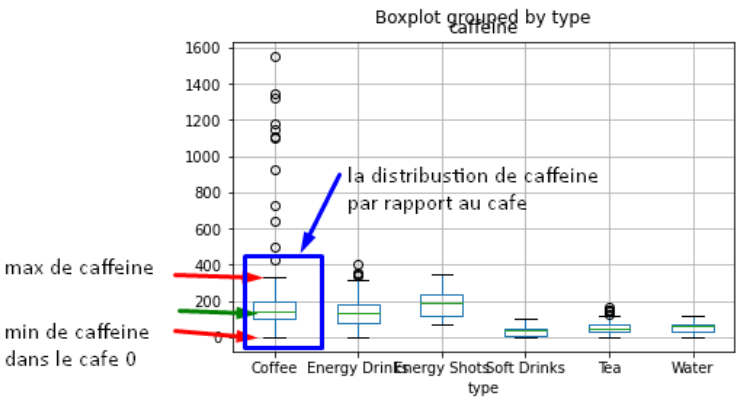
```
Out[4]: <AxesSubplot:>
```



- **Cette figure comment on affiche la distribution de caféine par rapport au type des boissons**

  en traçant notre boxplot, nous pouvons voir comment il s'agit de valeurs continues à travers les différentes boissons


```
In [5]: data.boxplot(column = "Caffeine",by = "type")
        plt.title('caffeine')
        plt.show()
```



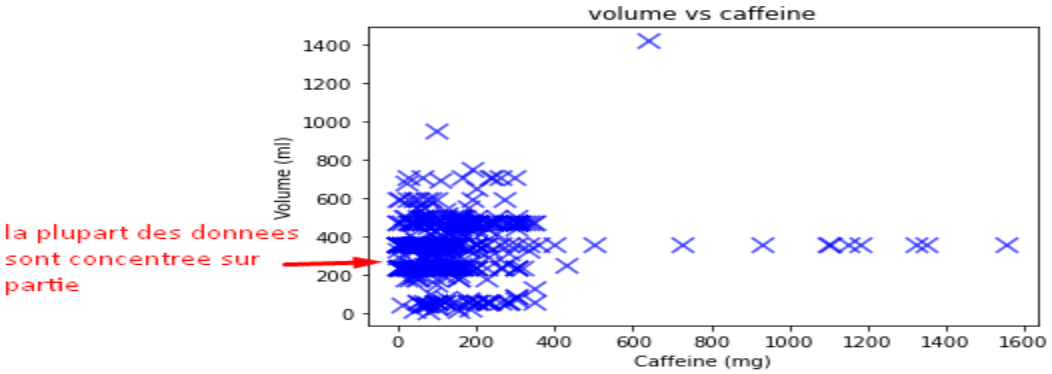
3. TP3 :

Scatterplot explore la relation entre 2 ou plus des variables quantitatives pour voir si les variables augmente ensemble ou non ainsi de suite

- **On va explorer la relation entre caffeine et volume comme la flèche montre que la plupart des valeurs se trouve la même zone mais sont**

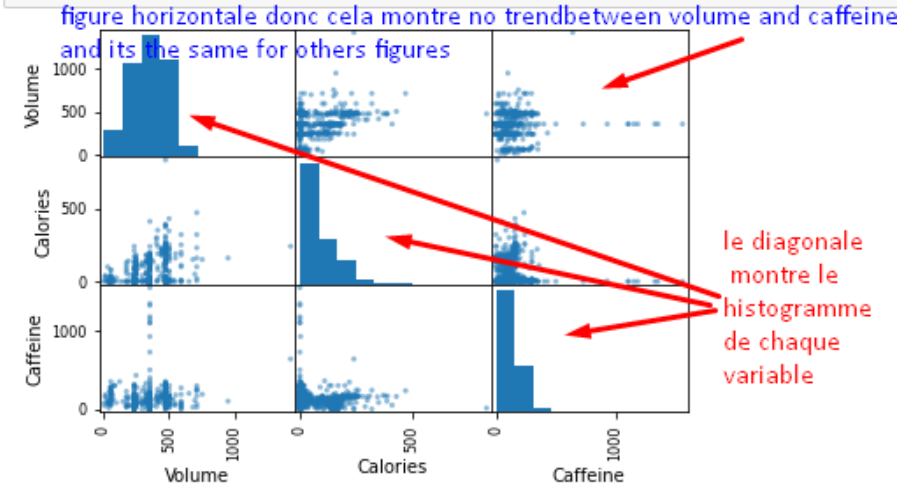
Distribuées horizontalement cela indique qu'il n'y a pas de trend increasing and decreasing value entre volume et caffeine

```
In [28]:
#Caffeine (mg)VS Volume (ml)
plt.scatter( df['Caffeine'], df['Volume'], marker='x',
             color='b', alpha=0.7, s = 124)
plt.title("volume vs caffeine")
plt.xlabel("Caffeine (mg)")
plt.ylabel("Volume (ml)")
plt.show()
```



- **Bien sûr si on a plusieurs variables on fait appel à la fonction scatter_matrix**

```
plt.show()
```

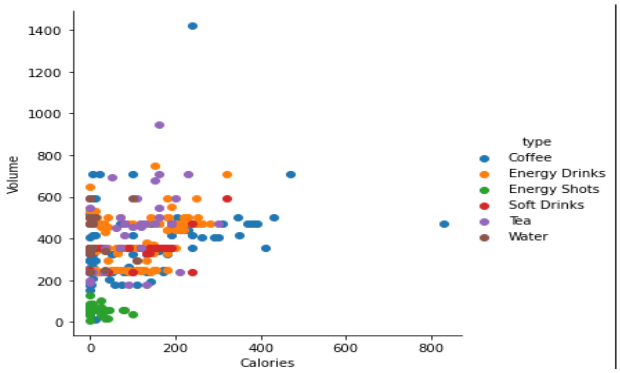


- **Scatter plot des variables quantitatives en relation avec des catégories.**

```
sns.FacetGrid(data, hue="type",size=5).map(plt.scatter,"Calories", "Volume").add_legend()
```

```
C:\ProgramData\anaconda3\lib\site-packages\seaborn\axisgrid.py:316: UserWarning: The `size
eight`; please update your code.
  warnings.warn(msg, UserWarning)
```

```
<seaborn.axisgrid.FacetGrid at 0x1d33ffce9d0>
```



4. TP4 :

Barplot

Un bar chart or bar plot affiche des barres rectangulaires avec des longueurs proportionnelles à la (discrète)valeurs qu'ils représentent.
Les valeurs discrètes peuvent être des valeurs ordinales, des catégories numériques, des scores ou une valeur agrégée (comme avg) dérivée de la valeur numérique continue.

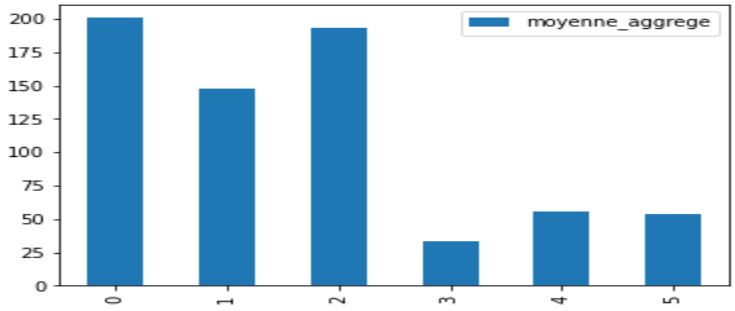
- **Dans cette partie j'ai calculé la moyenne de caféine (valeur discrète) de chaque catégorie des boissons et j'ai la stocker dans une nouvelle dataset .**

```
data6 = pd.read_csv("grouped_drinks.csv")
data6.sort_values(by='moyenne_aggrege', ascending=False)
```

	type	moyenne_aggrege
0	Coffee	200.589595
2	Energy Shots	193.416667
1	Energy Drinks	147.867580
4	Tea	55.863636
5	Water	53.730769
3	Soft Drinks	33.677778

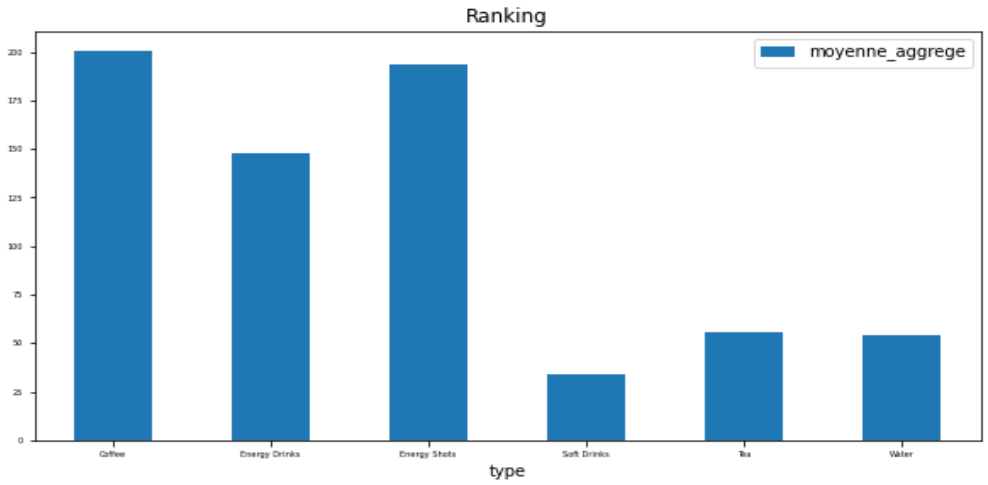
- **Apres j'ai trace barplot montre que la moyenne de caféine de coffee c'est la plus élevé suivi par Energy shots et les numéraux de 0 a 5 représentent les categories**

```
[39]: data6.plot.bar()
      plt.show()
```



- **Par la suite j'ai affecté à chaque bar son nom de catégorie au lieu de ID et les valeurs [0:200] représentent la moyenne agrégée**

```
data6.set_index('type').plot.bar(rot=0, title='Ranking', figsize=(10,5), fontsize=5)
]: <AxesSubplot:title={'center':'Ranking'}, xlabel='type'>
```



- **De même j'ai fait la moyenne agrégée des autres caractéristiques des boissons**

```
In [145]: df=y.sort_values(by='Caffeine', ascending=False)
In [147]: df
```

Out[147]:

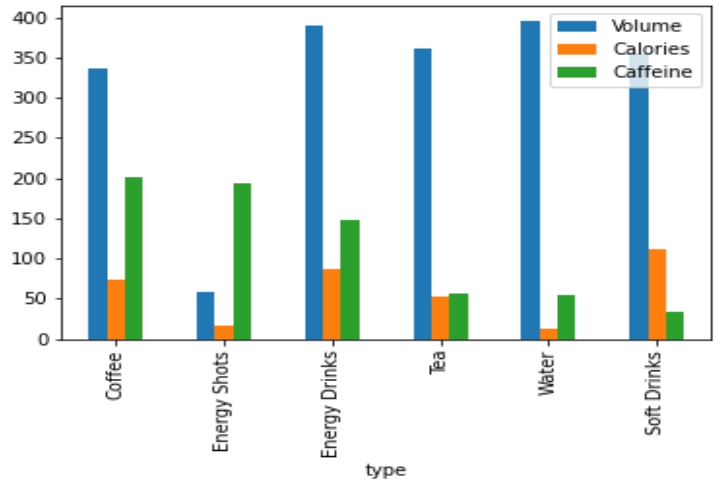
type	Volume	Calories	Caffeine
Coffee	335.870855	73.497110	200.589595
Energy Shots	57.742259	16.500000	193.416667
Energy Drinks	388.971198	86.671233	147.867580
Tea	360.474080	52.757576	55.863636
Water	394.590111	11.538462	53.730769
Soft Drinks	355.243454	111.111111	33.677778

- **Dans ce plot j'ai remarqué que coffee a plus de caféine que les autres**

Et soft drinks contiennent beaucoup plus de calories que les autres

Si on prend water il a le grand volume par rapport aux autres mais le calories et le caféines sont plus petit par rapport au autres

```
[146]: df.plot.bar()
       plt.show()
```



5. TP5 :

Pie Chart : un type de graphique dans lequel un cercle est divisé en secteurs qui représentent chacun une proportion de l'ensemble.

- **J'ai fait appel aux valeurs agrégées que j'ai calculé dans le TP précédent pour que je puisse créer un pie chart**

```
]: data = pd.read_csv("grouped_d.csv",encoding='utf-8')
   data.head()
```

ıt[7]:

	type	Volume_agg	Calories_agg	Caffeine_agg
0	Coffee	335.870855	73.497110	200.589595
1	Energy Shots	57.742259	16.500000	193.416667
2	Energy Drinks	388.971198	86.671233	147.867580
3	Tea	360.474080	52.757576	55.863636
4	Water	394.590111	11.538462	53.730769

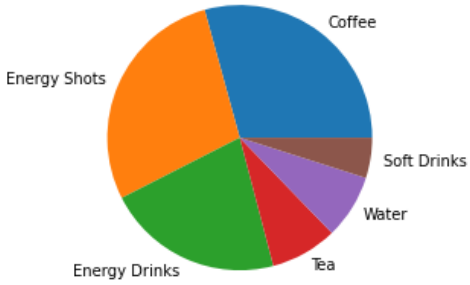
- **Pour créer un pie chart j'ai exécuté la fonction plt.pie() avec:**

Premier argument représente la valeur discrète de caféines et le 2eme les catégories.

La charte montre que le coffee occupe la plus grande partie de la charte suivie par Energy drinks.

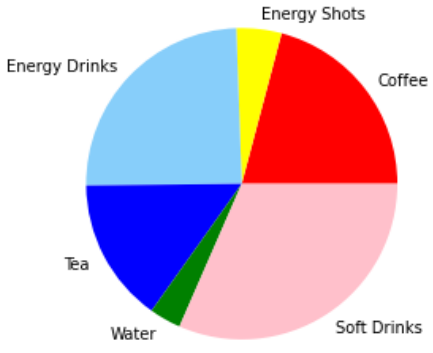
```
In [9]: #create a pie chart
        plt.pie(data['Caffeine_agg'], labels = data['type'])
```

```
Out[9]: ([<matplotlib.patches.Wedge at 0x17317867610>,
          <matplotlib.patches.Wedge at 0x17317867af0>,
          <matplotlib.patches.Wedge at 0x17317867f70>,
          <matplotlib.patches.Wedge at 0x17317875460>,
          <matplotlib.patches.Wedge at 0x173178758e0>,
          <matplotlib.patches.Wedge at 0x17317875d60>],
         [Text(0.6666111651618843, 0.8750026025569954, 'Coffee'),
          Text(-1.0065396346467175, 0.4437093236402101, 'Energy Shots'),
          Text(-0.44964976814280677, -1.0038999382454012, 'Energy Drinks'),
          Text(0.5399279569936064, -0.958372475218644, 'Tea'),
          Text(0.9347649164054173, -0.579840108182914, 'Water'),
          Text(1.0869104564341117, -0.16919119272051816, 'Soft Drinks')])
```



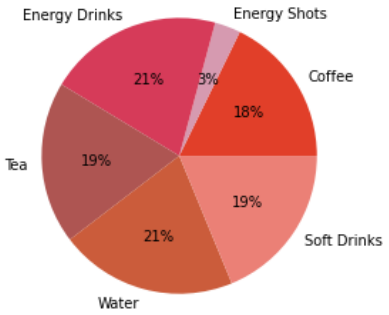
- **De même pour les calories Soft drinks contiennent beaucoup plus de calories**

```
colors = ["red", "yellow", "lightskyblue","blue", "green", "pink", "purple"]
plt.pie(data['Calories_agg'], labels = data['type'], colors = colors)
plt.axis('equal')
plt.show()
```



- **La même chose ici Energy drinks et water ont le plus grand volume**

```
colors = ["#E13F29", "#D69AB0", "#D63B59","#AE5552", "#CB5C3B", "#EB8076", "#96624E"]
plt.pie(data['Volume_agg'], labels = data['type'], colors = colors, autopct= '%1.0f%%')
plt.axis('equal')
plt.show()
```

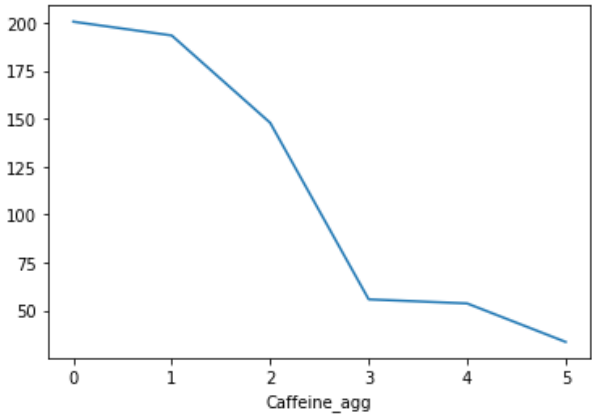


6. TP6 :

GRAPHIQUE EN LIGNE : Les graphiques linéaires sont utilisés pour suivre les changements sur des périodes courtes et longues en traçant des points de données et en les reliant par une ligne

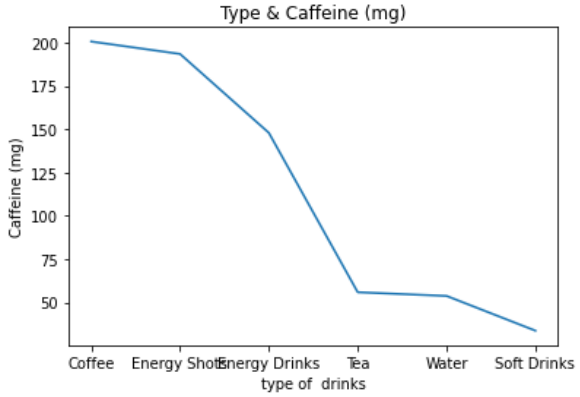
- **Tracer les valeurs numériques sous forme de graphique linéaire**

```
]: plt.plot(data1["Caffeine_agg"])
   plt.xlabel("Caffeine_agg")
   plt.show()
```

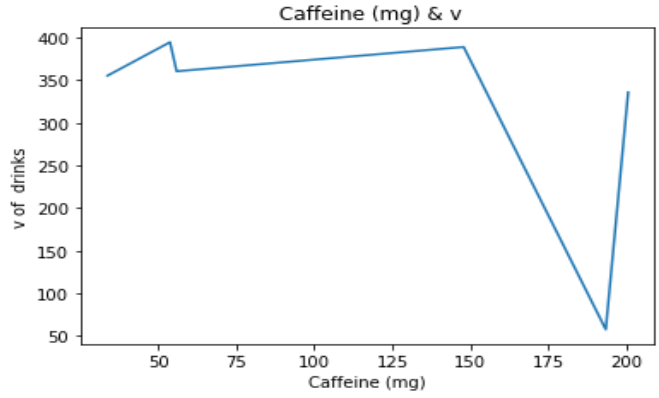


- **Nous pouvons traces aussi la relation entre caféine et type de boissons et aussi la relation entre volume et caféine**

```
plt.plot(data1["type"],data1["Caffeine_agg"])
plt.title("Type & Caffeine (mg)")
plt.xlabel("type of  drinks")
plt.ylabel("Caffeine (mg)")
plt.show()
```




```
plt.plot(data["Caffeine_agg"],data["Volume_agg"])
plt.title(" Caffeine (mg) & v")
plt.xlabel("Caffeine (mg)")
plt.ylabel("v of  drinks")
plt.show()
```



7. TP7 :

Multiple lines c'est la suite de tp 6 pour voir comment les variables interagir entre eux en un seul plot

- **J'ai travaillé sur la même dataset**

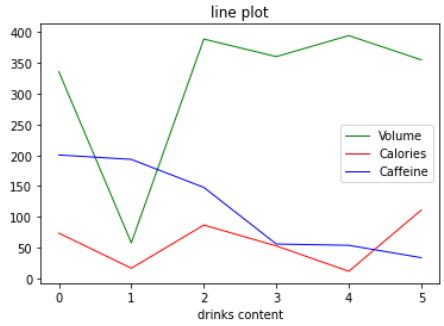
```
data = pd.read_csv("grouped_d.csv")
data.head()
```

2]:

	type	Volume_agg	Calories_agg	Caffeine_agg
0	Coffee	335.870855	73.497110	200.589595
1	Energy Shots	57.742259	16.500000	193.416667
2	Energy Drinks	388.971198	86.671233	147.867580
3	Tea	360.474080	52.757576	55.863636
4	Water	394.590111	11.538462	53.730769

- **La figure montre comment les 3 variables interagir entre eux .on peut dire il y a pas d'interactions entre ces 3 variables sauf pour la catégorie water entre le caféine et les calories**

```
plt.subplot()
data.Volume_agg.plot(kind='line',color='g',label='Volume',linewidth=1,linestyle='-')
data.Calories_agg.plot(color='red',label='Calories',linewidth=1,linestyle='-')
data.Caffeine_agg.plot(color='blue',label='Caffeine',linewidth=1,linestyle='-')
plt.legend(loc='best')
plt.xlabel('drinks content')
plt.title('line plot')
plt.show()
```



8. TP8-9 :Dans ce tp j'ai changer la dataset au Netflix Movies et tv show et l'inpretation je l'ai fait comme des commentaire sur les screens

]:
```
#importation des librairies
```

]:
```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import plotly.express as px
sns.set()
```

```
##charger dataset et la stocker dans la variable data
```

```
netflix_df = pd.read_csv("netflix_titles.csv",encoding='utf-8',parse_dates=True)
netflix_df.head()
```

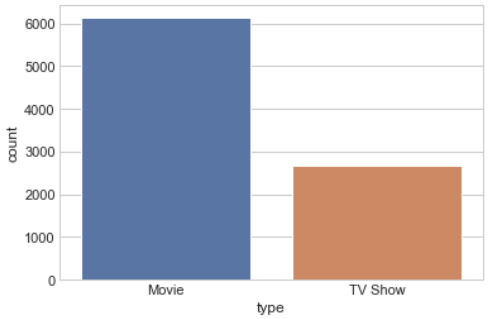
.3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	des
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As h nears th his life
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After paths at Cape

```
#checking number of movie vs tv shows /total number of movies its much higher than tv shows
plt.style.use('seaborn-whitegrid')
sns.countplot(netflix_df["type"])
```

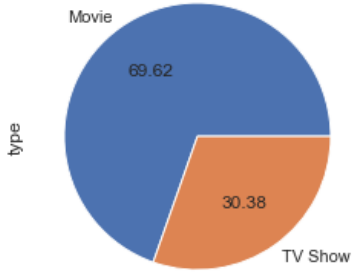
```
C:\ProgramData\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
d arg: x. From version 0.12, the only valid positional argument will be `data`, and passing c
cit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
: <AxesSubplot:xlabel='type', ylabel='count'>
```

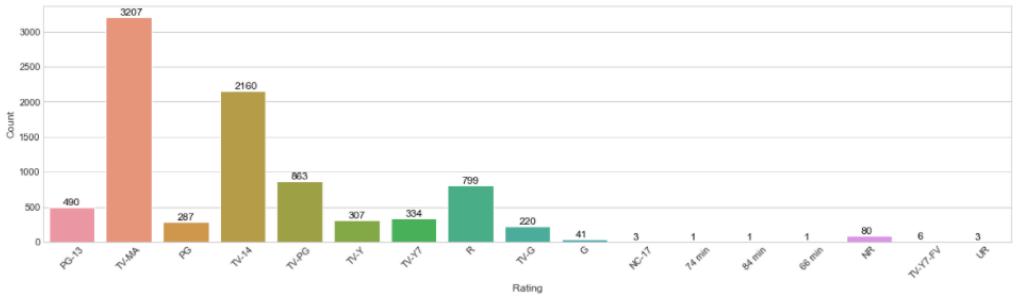



```
netflix_df["type"].value_counts().plot(kind = "pie", autopct = "%.2f")
```

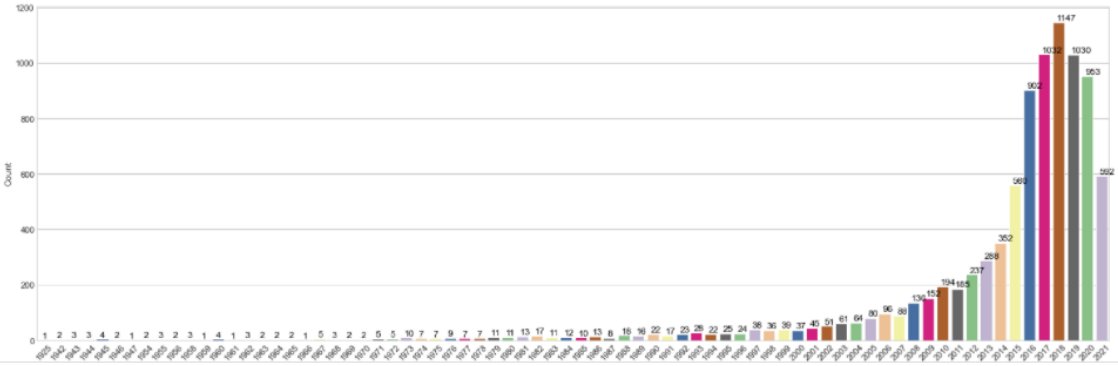
```
23]: <AxesSubplot:ylabel='type'>
```




```
#checking number of rating in each category for movies and tv shows
plt.figure(figsize = (20, 5))
plt.style.use('seaborn-whitegrid')
ax = sns.countplot(x = 'rating', data = netflix_df)
plt.xlabel('Rating')
plt.ylabel('Count')
plt.xticks(rotation = 45)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va = 'bottom', color = 'black')
```

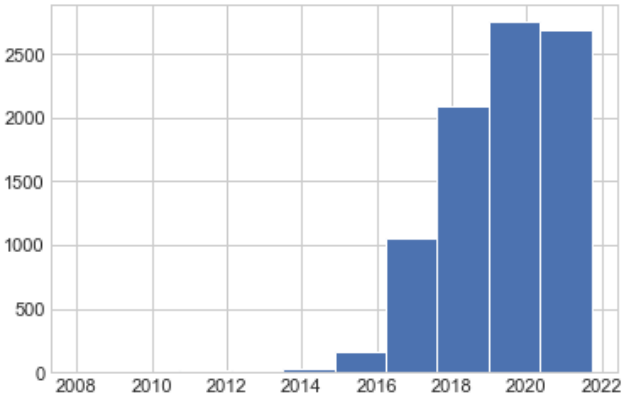


```
#checking how many Movies/tv shows release each year
plt.figure(figsize = (25, 8))
plt.style.use('seaborn-whitegrid')
ax = sns.countplot(x = 'release_year', data = netflix_df, palette = "Accent")
plt.xlabel('Year of Release')
plt.ylabel('Count')
plt.xticks(rotation = 45)
for p in ax.patches:ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va = 'bottom', color = 'black')
```

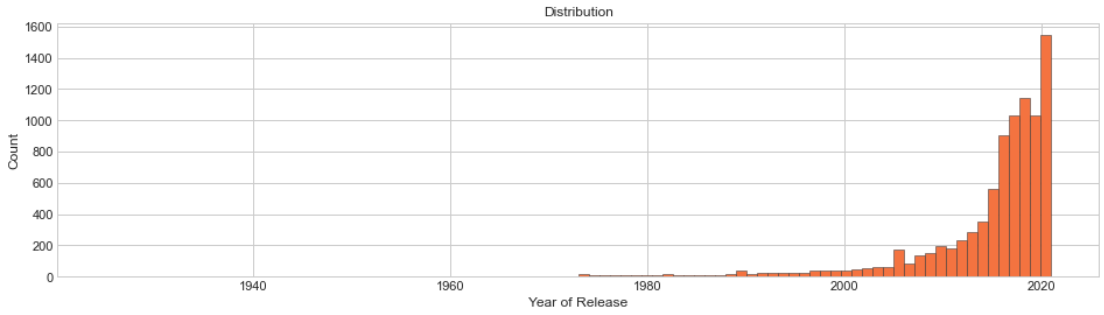


```
#plotting histogram of how many Movies/Tv Shows added in Netflix each year
netflix_df["date_added"].hist()
```

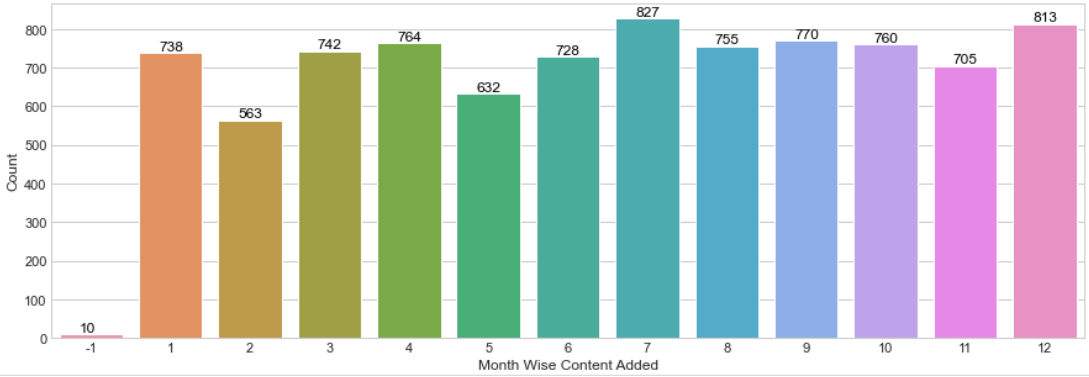
```
]: <AxesSubplot:>
```




```
#plotting histogram of how many Movies/Tv Shows added in Netflix each year - matplotlib
plt.figure(figsize=(16, 4))
plt.style.use('seaborn-whitegrid')
plt.hist(netflix_df["release_year"], bins = 90, facecolor = '#F47340', edgecolor='#323232', linewidth=0.5)
plt.title('Distribution')
plt.xlabel('Year of Release')
plt.ylabel('Count')
plt.show()
```

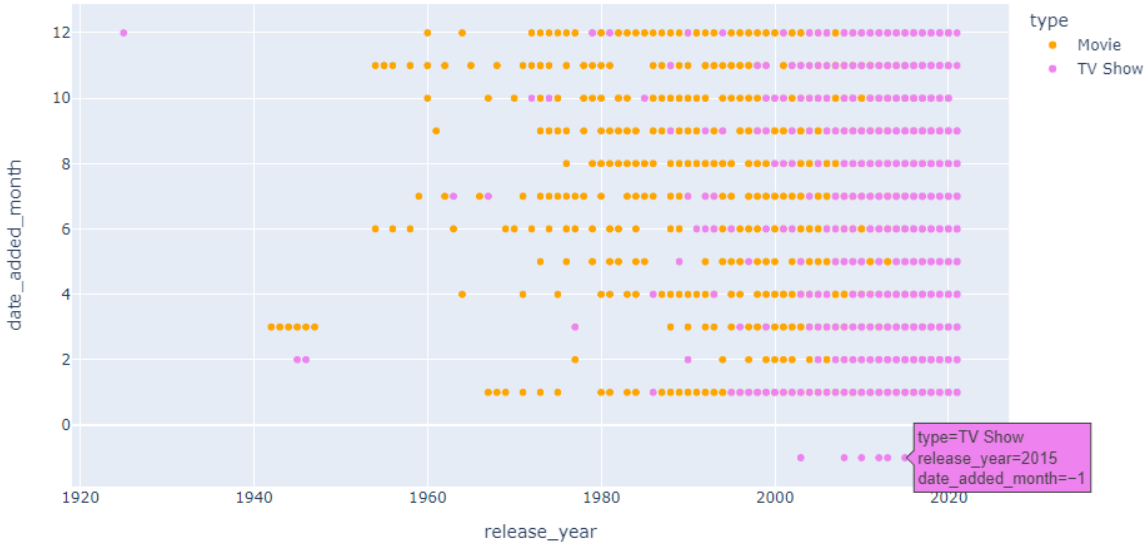


```
#checking the number of content added Month-wise
plt.figure(figsize = (15, 5))
ax = sns.countplot(x = 'date_added_month', data = netflix_df)
plt.xlabel('Month Wise Content Added')
plt.ylabel('Count')
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, p.get_height()+1), va = 'bottom', color = 'black')
```



Month Wise Content Added

```
#scatter Plot for Content released per year vs Month wise Content Added
df = px.data.tips()
fig = px.scatter(netflix_df, x = "release_year", y = "date_added_month", color = "type", color_discrete_sequence = ["orange",
fig.show()
```




```
#Top 30 countries where most of movies /tv shows was produced
plt.figure(figsize = (15, 7))
country = country_count["country"].value_counts()[:30]
sns.barplot(x = country, y = country.index, palette = "Accent")
plt.xlabel("Count")
```

```
Text(0.5, 0, 'Count')
```