UNIVERSITE ABDELMALEK ESSAADI

FACULTE DES SCIENCES ET TECHNIQUES DE TANGER

PROJET DE FIN D'Module

Master Mobiquité & Big Data

« Rapport de projet de fin de module NoSQL (Cryptocurrency Data Visualisation / Analysis) »

Réalisé PAR :

Benamar Zaid

Sous l'encadrement de :

Pr. M. EL AACHAK Lotfi

Fais-le : 01/06/2022

Année Universitaire : 2021/2022

TABLE DES MATIÈRES :

INTRODUCTION :

L'analyse des données est la méthode permettant d'extraire des solutions aux problèmes par l'interrogation et l'interprétation des données. Le processus d'analyse comprend la découverte des problèmes, la résolution de l'accessibilité des données appropriées, la détermination de la méthode qui peut aider à trouver la solution au problème intéressant et la transmission du résultat. Pour les besoins de l'analyse, les données doivent être séparées en plusieurs étapes telles que la spécification, l'assemblage, l'organisation, le nettoyage, la ré-analyse, l'application de modèles et d'algorithmes et le résultat final. Un nombre considérable d'individus ont utilisé ces stratégies dans la recherche et les affaires pour créer de la substance ou offrir des critiques pour étendre l'exactitude de la publicité commerciale qui permet aux individus de fournir des ressources dans l'avancement et le développement de l'entreprise.

PARTIE I

« Contexte générale du projet »

I. Objectif du projet

L'objectif principal du projet est la réalisation d'une tableau de bord de type single page application afin de visualisé les analyses des données qui sont stocké dans une base de données NoSQL orienté document, Il s'agit des données historiques sur 4 **crypto-monnaies (Cryptocurrencies)** extracté à partir le Scrapping qui les collecte d'après le site officiel https://coinmarketcap.com/.

II. Technologies et outils utilisé

Afin de réaliser ce Project, on a utilisé plusieurs technologies lesquelles :

II.1 Front-end :

Angular 8 : Angular est une plate-forme et un framework permettant de créer des applications clientes de type single-page application à l'aide de HTML et de TypeScript. Angular est utilisé pour créer des applications Web dynamiques.

Plotly.js : Plotly.js est une bibliothèque de graphiques JavaScript qui comprend plus de 40 types de graphiques, des graphiques en 3D, des graphiques statistiques et des cartes SVG.

II.2 **Back-end :**

Django : Django est un framework web python open-source utilisé pour le développement rapide, pragmatique, maintenable, propre, et sécurise les sites web. Participe au développement Web permet aux utilisateurs de se concentrer sur le développement des composantes nécessaires à leur application.

Djongo : Djongo est une approche unifiée de l'interfaçage des bases de données. Il s'agit d'un prolongement du Django ORM framework mais mappe les objets python aux documents MongoDB.

Django REST Framework : Django REST framework (DRF) est une bibliothèque Python/Django open source, mature et bien prise en charge qui vise à créer des API Web sophistiquées. Il s'agit d'une boîte à outils flexible et complète avec une architecture modulaire et personnalisable qui permet le développement de points de terminaison d'API simples et clés en main et de constructions REST complexes**.**

Elasticsearch : Elasticsearch est un moteur de recherche et d'analyse distribué gratuit et ouvert pour tout type de données, la vitesse et la scalabilité d'Elasticsearch, ainsi que sa capacité à indexer de nombreux types de contenus signifient qu'il peut être employé dans différents cas d'utilisation.

II.3 **Analyse des données :**

Jupyter Notebook : Jupyter Notebook est une application Web open source que vous pouvez utiliser pour créer et partager des documents contenant du code en direct, des équations, des visualisations et du texte.

Pandas : Pandas est un paquetage Python open source très largement utilisé pour la science des données, l'analyse des données et les tâches d'apprentissage automatique. Il vise à être le composant fondamental de haut niveau pour faire une analyse de données pratique et réelle en Python.

Seaborn : Seaborn est une bibliothèque de visualisation de données pour le traçage de graphiques statistiques en Python. Il fournit de beaux styles par défaut et des palettes de

couleurs pour rendre les tracés statistiques plus attrayants. J'ai utilisé bibliothéque pour visualisé les données lors de l'analyse dans jupyter notebook.

Matplotlib & NumPy : Matplotlib est une bibliothèque de traçage pour Python. Il est utilisé avec NumPy pour fournir un environnement qui est une alternative open source efficace.

II.4 Web Scrapping :

Beautiful Soup : Beautiful Soup est un package Python pour l'analyse de documents HTML et XML. Il crée un arbre d'analyse pour les pages analysées qui peuvent être utilisées pour extraire des données de HTML, ce qui est utile pour le grattage Web.

II.5 Stockage des données :

MongoDB : MongoDB est une base de données orientée documents. En clair, vous bénéficiez de la scalabilité et de la flexibilité que vous voulez, avec les fonctions d'interrogation et d'indexation qu'il vous faut.

PARTIE II

« Préparation et analyse des données »

I. Collection des données

Afin de récupérer les données depuis **CoinMarket** sur lesquelles on va travailler. On a utilisé l'outil **Beautiful Soup** qui offre plusieurs mécanismes d'extraire les données depuis les pages Web, comme **find** qui permet de sélectionner les bloc de code HTML on spécifiant le nom de la balise.

Par exemple, afin de récupérer les données historiques de Bitcoin depuis **CoinMarket.**

Historical Data for Bitcoin

Date Range

Date	Open*	High	Low	Close**	Volume	Market Cap
May 31, 2022	$31,723.87	$32,249.86	$31,286.15	$31,792.31	$33,538,210,634	$605,797,887,876
May 30, 2022	$29,443.37	$31,949.63	$29,303.57	$31,726.39	$39,277,993,274	$604,513,442,526
May 29, 2022	$29,019.87	$29,498.01	$28,841.11	$29,445.96	$18,093,886,409	$561,034,743,433
May 28, 2022	$28,842.10	$29,135.92	$28,554.57	$29,023.49	$19,252,320,708	$552,958,193,308

On peut procéder comme suit :

Entrée [4]:
```
URL = "https://coinmarketcap.com/historical/20220417/"
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 

crypto_page = requests.get(URL,headers=headers)
soup1 = bs(crypto_page.content,'html.parser')
soup2 = bs(soup1.prettify(), "html.parser")
```

Entrée [5]:
```
all_coins = soup2.find_all('tr', attrs={'class' : 'cmc-table-row'})
```

```
soup3 = bs(bitcoin,'html.parser')
rows = soup3.find('table', class_='h7vnx2-2 hLKazY cmc-table').find('tbody').find_all('tr')
l = list()
for row in rows :

    values = row.find_all('td')
    date_val = values[0].get_text()
    open_val = values[1].get_text()
    high_val = values[2].get_text()
    low_val = values[3].get_text()
    close_val = values[4].get_text()
    volume_val = values[5].get_text()
    market_cap_val = values[6].get_text()

    data = {'Date':date_val, 'Open':open_val, 'High':high_val, 'Low': low_val, 'Close': close_val, 'Volume': volume_va
    l.append(data)

    df = pd.DataFrame(l)
    df
    df.to_csv(name + '2.csv',sep=";")
```

Le processus derniér serve a extracté les données **d'une durée de 1 an** de chaque crypto et les stocké dans une Dataframe afin de l'exporté dans un fichier csv pour l'analyse.

- **Résultat :**

	A	B	C	D	E	F	G	H	I
1	id	Date	Open	High	Low	Close	Volume	Market Cap	
2	0	May 03, 2022	$38,528.11	$38,629.99	$37,585.62	$37,750.45	$27,326,943,2	$718,385,701,943	
3	1	May 02, 2022	$38,472.19	$39,074.97	$38,156.56	$38,529.33	$32,922,642,4	$733,170,725,791	
4	2	May 01, 2022	$37,713.27	$38,627.86	$37,585.79	$38,469.09	$27,002,760,1	$731,986,764,312	
5	3	Apr 30, 2022	$38,605.86	$38,771.21	$37,697.94	$37,714.88	$23,895,713,7	$717,596,901,509	
6	4	Apr 29, 2022	$39,768.62	$39,887.27	$38,235.54	$38,609.82	$30,882,994,6	$734,589,762,490	
7	5	Apr 28, 2022	$39,241.43	$40,269.47	$38,941.42	$39,773.83	$33,903,704,9	$756,698,317,682	
8	6	Apr 27, 2022	$38,120.30	$39,397.92	$37,997.31	$39,241.12	$30,981,015,1	$746,531,438,329	
9	7	Apr 26, 2022	$40,448.42	$40,713.89	$37,884.99	$38,117.46	$34,569,088,4	$725,113,482,674	
10	8	Apr 25, 2022	$39,472.61	$40,491.75	$38,338.38	$40,458.31	$35,445,730,5	$769,607,056,243	
11	9	Apr 24, 2022	$39,478.37	$39,845.92	$39,233.54	$39,469.29	$17,964,398,1	$750,755,023,764	
12	10	Apr 23, 2022	$39,738.72	$39,935.86	$39,352.20	$39,486.73	$16,138,021,2	$751,045,477,435	
13	11	Apr 22, 2022	$40,525.86	$40,777.76	$39,315.42	$39,740.32	$28,011,716,7	$755,827,581,233	
14	12	Apr 21, 2022	$41,371.52	$42,893.58	$40,063.83	$40,527.36	$35,372,786,3	$770,762,297,843	

Date : date de l'observation

Open : Le prix d'ouverture

High : Le prix le plus élevé

Low : Prix le plus bas

Close : Le prix de clôture

Volume : Total des actions négociées

Market Cap : La valeur totale de toutes les pièces qui ont été frappées

II. Prétraitement des données

La premiére des choses avant de procédé au processus de detection des missing values et gérer les valeurs aberrantes (outliers), on va tout d'abord préparer le dataframe pour cela.

```
Entrée [351]: btc_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          365 non-null    int64 
 1   Date        365 non-null    object
 2   Open        365 non-null    object
 3   High        365 non-null    object
 4   Low         365 non-null    object
 5   Close       365 non-null    object
 6   Volume      365 non-null    object
 7   Market Cap  365 non-null    object
dtypes: int64(1), object(7)
memory usage: 22.9+ KB
```

Tout les champ sont de type object (String), ce qu'est pas compatible avec l'analyse. Concernant le champ Date on va le formatter pour qu'il devient de type date, et les autres champs en les convertissant en nombres entiers.

Formatting date

```
Entrée [3]: btc_df['Date'] = btc_df['Date'].apply(lambda x : datetime.datetime.strptime(f'{x}',"%b %d, %Y") )
            eth_df['Date'] = eth_df['Date'].apply(lambda x : datetime.datetime.strptime(f'{x}',"%b %d, %Y") )
            bnb_df['Date'] = bnb_df['Date'].apply(lambda x : datetime.datetime.strptime(f'{x}',"%b %d, %Y") )
            xrp_df['Date'] = xrp_df['Date'].apply(lambda x : datetime.datetime.strptime(f'{x}',"%b %d, %Y") )
```

Cleaning Rows

Entrée [9]:
```
def crypto_row_cleaning(row):

    def clean_data(val) :
        new_val = float(val.replace("$","").replace(',','_'))
        return new_val

    def clean_rows(row) :
        #this is a pandas Serie
        row['Open'] = clean_data(row['Open'])
        row['High'] = clean_data(row['High'])
        row['Low'] = clean_data(row['Low'])
        row['Close'] = clean_data(row['Close'])
        row['Volume'] = clean_data(row['Volume'])
        row['Market Cap'] = clean_data(row['Market Cap'])
        return row

    return clean_rows(row)


btc_df_cleaned = btc_df.apply(crypto_row_cleaning,axis=1)
```

- **Résultat :**

Entrée [364]:
```
btc_df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 365 entries, (Timestamp('2022-05-03 00:00:00'), 0) to (Timestamp('2021-05-04 00:00:00'), 364)
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Open        365 non-null    float64
 1   High        365 non-null    float64
 2   Low         365 non-null    float64
 3   Close       365 non-null    float64
 4   Volume      365 non-null    float64
 5   Market Cap  365 non-null    float64
dtypes: float64(6)
memory usage: 40.5 KB
```

Entrée [366]:
```
btc_df_cleaned.head()
```

Out[366]:

Date	id	Open	High	Low	Close	Volume	Market Cap
2022-05-03	0	38528.11	38629.99	37585.62	37750.45	2.732694e+10	7.183857e+11
2022-05-02	1	38472.19	39074.97	38156.56	38529.33	3.292264e+10	7.331707e+11
2022-05-01	2	37713.27	38627.86	37585.79	38469.09	2.700276e+10	7.319868e+11
2022-04-30	3	38605.86	38771.21	37697.94	37714.88	2.389571e+10	7.175969e+11
2022-04-29	4	39768.62	39887.27	38235.54	38609.82	3.088299e+10	7.345898e+11

Jusqu'à moment nous disposont de 4 dataframes, chaqune contient des données sur un crypto monnaies spécifique, et pour chaqune on a effectuer le processus de formatage précédant.

On veut d'abord combiner ces 4 dataframe pour qu'ils devient une seul, cela nécéssite la création d'une MultiIndex Dataframe, donc le processus suivante vise a atteint ce objectif.

Creating MultiIndex DataFrame

```
Entrée [13]: features = list(btc_df_cleaned.columns)
             features

Out[13]: ['Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap']

Entrée [14]: #merged_df = pd.DataFrame(data=btc_df_cleaned, columns=pd.MultiIndex.from_arrays(zip(['id', 'Open', 'High', 'Low', 'Close', 'Volu
             col2 = ['BTC-USD','ETH-USD','BNB-USD','XRP-USD']
             my_list = list(itertools.product(features[0:], col2))

Entrée [15]: columns = pd.MultiIndex.from_tuples(my_list)

Entrée [16]: len(list(btc_df.index)) # 99
             tuples = tuple(btc_df.index)
             index = pd.MultiIndex.from_tuples(tuples, names=["Date", "id"])
```

La méthode **filling_df** vise à remplire la nouvelle dataframe avec les données des 4 autre dataframes.

```
Entrée [20]: def filling_df(data_frame,name) :

                 for column in range(0,len(features)) :
                     merged_df.loc[:, (features[column], f'{name}-USD')] = data_frame[features[column]]

             #s.loc[:, ('Open', 'BTC-USD')] = btc_df_cleaned["Open"]
             filling_df(btc_df_cleaned,'BTC')
             filling_df(eth_df_cleaned,'ETH')
             filling_df(bnb_df_cleaned,'BNB')
             filling_df(xrp_df_cleaned,'XRP')
```

- **Résultat :**

```
Entrée [21]: merged_df.head()
```

Out[21]:

		Open				High				Low		...	Close		Volume		
		BTC-USD	ETH-USD	BNB-USD	XRP-USD	BTC-USD	ETH-USD	BNB-USD	XRP-USD	BTC-USD	ETH-USD	...	BNB-USD	XRP-USD	BTC-USD	ETH-USD	BNB-USD
Date	id																
2022-05-03	0	38528.11	2857.15	389.63	0.6142	38629.99	2859.19	391.75	0.6251	37585.62	2762.12	...	383.48	0.6048	2.732694e+10	1.302609e+10	1.295878e+0
2022-05-02	1	38472.19	2827.61	390.26	0.6078	39074.97	2874.15	392.22	0.6293	38156.56	2785.52	...	389.67	0.6143	3.292264e+10	1.860974e+10	1.388078e+0
2022-05-01	2	37713.27	2729.99	377.77	0.5871	38627.86	2838.70	391.08	0.6102	37585.79	2728.08	...	390.28	0.6078	2.700276e+10	1.533273e+10	1.553847e+0
2022-04-30	3	38605.86	2815.53	393.00	0.6114	38771.21	2836.83	399.59	0.6271	37697.94	2727.41	...	377.77	0.5871	2.389571e+10	1.352094e+10	1.467606e+0
2022-04-29	4	39768.62	2936.78	406.64	0.6443	39887.27	2943.45	410.18	0.6456	38235.54	2782.44	...	393.06	0.6114	3.088299e+10	1.877104e+10	1.928210e+0

Avec la nouvelle dataframe qu'on a crée on peut mieux visualisé les données des 4 crypto monnaies par rapport a l'un des features comme suit :

```
Entrée [367]: merged_df['Open']
```

Out[367]:

Date	BTC-USD	ETH-USD	BNB-USD	XRP-USD
2022-05-03	38528.11	2857.15	389.63	0.6142
2022-05-02	38472.19	2827.61	390.26	0.6078
2022-05-01	37713.27	2729.99	377.77	0.5871
2022-04-30	38605.86	2815.53	393.00	0.6114
2022-04-29	39768.62	2936.78	406.64	0.6443
...	...	...	...	...
2021-05-08	57352.77	3481.99	624.39	1.5800
2021-05-07	56413.95	3490.11	634.01	1.6000
2021-05-06	57441.31	3524.93	651.05	1.6200
2021-05-05	53252.16	3240.55	609.33	1.3900
2021-05-04	57214.18	3431.13	676.32	1.5600

365 rows × 4 columns

Il nous reste d'abord que la detection des missing values et gérer les valeurs aberrantes (outliers).

Cocernant les valeurs manquants, notre MultiIndex dataframe contient aucun.

```
Entrée [25]: merged_df.isna().sum()
```

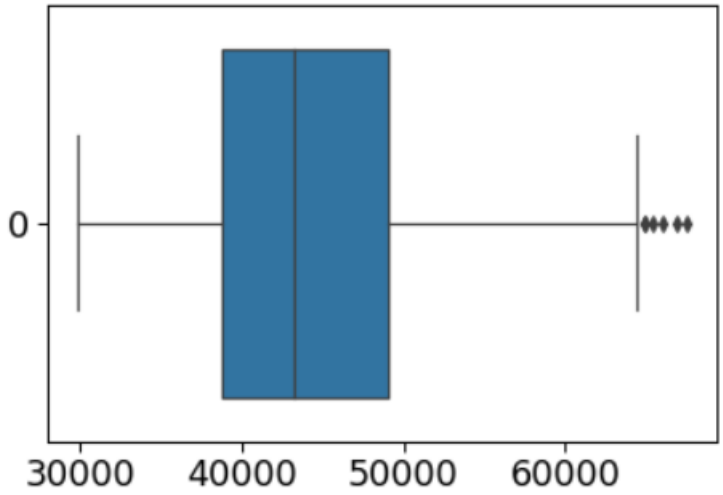
Out[25]:

```
Open       BTC-USD    0
           ETH-USD    0
           BNB-USD    0
           XRP-USD    0
High       BTC-USD    0
           ETH-USD    0
           BNB-USD    0
           XRP-USD    0
Low        BTC-USD    0
           ETH-USD    0
           BNB-USD    0
           XRP-USD    0
Close      BTC-USD    0
           ETH-USD    0
           BNB-USD    0
           XRP-USD    0
Volume     BTC-USD    0
           ETH-USD    0
           BNB-USD    0
           XRP-USD    0
Market Cap BTC-USD    0
           ETH-USD    0
           BNB-USD    0
           XRP-USD    0
dtype: int64
```

D’abord on va voir si notre dataframe contient des outliers à l’aide d’un boxplot, et puisque nous sommes intéressés par le prix de clôture (Closing Price) de chaque crypto monnaies, donc on va traité les outliers just pour ces colonnes.

Detecting outliers

```
Entrée [338]: ax = sns.boxplot(data=adj_close['BTC-USD'], orient="h")
```



On remarque par exemple que la colonne « BTC-USD » qui référence au Bitcoin ayant des outliers, pour les enlevé on va faire le jeux d’opérations suivant :

```
Entrée [174]: def handling_outliers2(name) :
                  Q1 = adj_close[name].quantile(0.25)
                  Q3 = adj_close[name].quantile(0.75)
                  IQR = Q3 - Q1
                  lower_lim = Q1 - 1.5 * IQR
                  upper_lim = Q3 + 1.5 * IQR
                  outliers_15_low = (adj_close[name] < lower_lim)
                  outliers_15_up  = (adj_close[name] > upper_lim)
                  adj_close[name] = adj_close[name][~(outliers_15_low | outliers_15_up)]
```

```
Entrée [175]: handling_outliers2('BTC-USD')
              handling_outliers2('BNB-USD')
              handling_outliers2('ETH-USD')
              handling_outliers2('XRP-USD')
```

Après ce traitement on remarque que le dataframe contient des missing values, donc j’ai essayé de les enlevé apartir cette méthode :

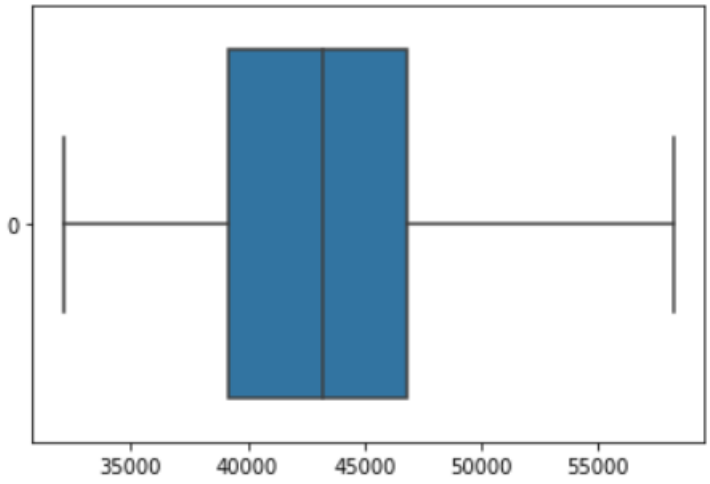
```
Entrée [176]: adj_close.isna().sum()
```

```
Out[176]: BTC-USD    6
          ETH-USD    0
          BNB-USD    0
          XRP-USD    4
          dtype: int64
```

```
Entrée [177]: imputer = SimpleImputer(missing_values= np.NAN, strategy= 'mean', fill_value=None, verbose=0, copy=True)
              X = imputer.fit_transform(adj_close)
              adj_close = pd.DataFrame(X, columns=adj_close.columns)
```

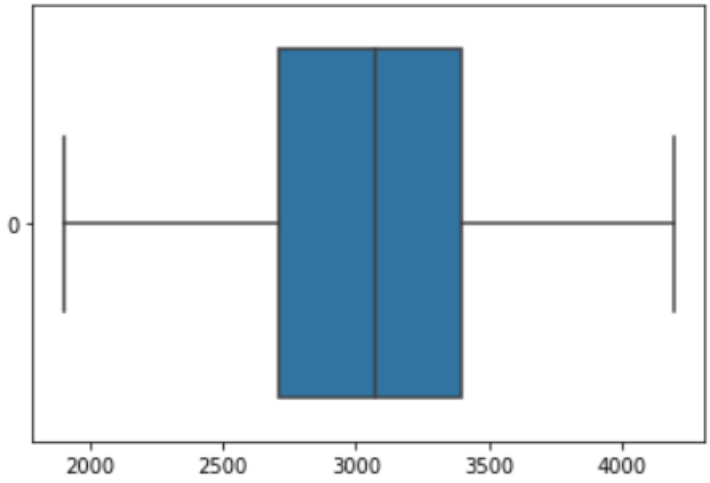
- **Résultat de Bitcoin :**

```
ax = sns.boxplot(data=adj_close['BTC-USD'], orient="h")
```



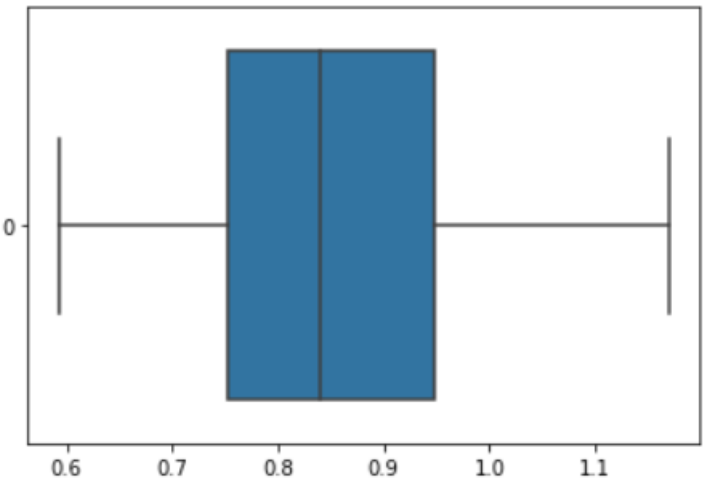
- **Résultat de Ethereum :**

```
ax = sns.boxplot(data=adj_close['ETH-USD'], orient="h")
```



- **Résultat de XRP :**

```
ax = sns.boxplot(data=adj_close['XRP-USD'], orient="h")
```



III. Chargement des données dans MongoDB

Après avoir finir l'étape de pre-processing, on a besoins de stocker ces données dans une bases de données pour les utilisées.

La méthode to_dict() de Pandas convertie une dataframe au dictionnaire, ce qui est équivalent au format des documents JSON au base de données MongoDB.

```
Entrée [48]: xrp_df_cleaned.to_dict("records")

    Out[48]: [{'Open': 0.6142,
               'High': 0.6251,
               'Low': 0.5982,
               'Close': 0.6048,
               'Volume': 1460418466.0,
               'Market Cap': 29130203226.0},
              {'Open': 0.6078,
               'High': 0.6293,
               'Low': 0.6018,
               'Close': 0.6143,
               'Volume': 1687184077.0,
               'Market Cap': 29549544283.0},
              {'Open': 0.5871,
               'High': 0.6102,
               'Low': 0.581,
               'Close': 0.6078,
               'Volume': 1657999135.0,
               'Market Cap': 29240494806.0},
              {'Open': 0.6114,
               'High': 0.6271,
               'Low': 0.5816,
               'Close': 0.5871,
               'Volume': 1649221930.0,
               'Market Cap': 28242161687.0},
```

Il faut aussi réinitialiser l'index qui est la Date, sinon il ne va pas être stocké lors de chargement de dataframe.

```
Entrée [49]: btc_df_cleaned.reset_index(inplace = True,drop = False)
             xrp_df_cleaned.reset_index(inplace = True,drop = False)
```

```
Entrée [50]: xrp_df_cleaned.head()
```

Out[50]:

	Date	id	Open	High	Low	Close	Volume	Market Cap
0	2022-05-03	0	0.6142	0.6251	0.5982	0.6048	1.460418e+09	2.913020e+10
1	2022-05-02	1	0.6078	0.6293	0.6018	0.6143	1.687184e+09	2.954954e+10
2	2022-05-01	2	0.5871	0.6102	0.5810	0.6078	1.657999e+09	2.924049e+10
3	2022-04-30	3	0.6114	0.6271	0.5816	0.5871	1.649222e+09	2.824216e+10
4	2022-04-29	4	0.6443	0.6456	0.6039	0.6114	1.962198e+09	2.941186e+10

Aussi j'ai fait un étape très important qu'est de renommer les colonnes de dataframe pour qu'il devient adapté au modèle créer pour celle-là dans Django.

```
Entrée [51]: new_cols = [x.lower() for x in list(btc_df_cleaned.columns)]
             mapped_columns = dict(zip(xrp_df_cleaned.columns,new_cols))
             mapped_columns
```

```
Out[51]: {'Date': 'date',
          'id': 'id',
          'Open': 'open',
          'High': 'high',
          'Low': 'low',
          'Close': 'close',
          'Volume': 'volume',
          'Market Cap': 'market cap'}
```

```
Entrée [52]: btc_df_cleaned.rename(columns=mapped_columns,inplace = True)
             eth_df_cleaned.rename(columns=mapped_columns,inplace = True)
             bnb_df_cleaned.rename(columns=mapped_columns,inplace = True)
             xrp_df_cleaned.rename(columns=mapped_columns,inplace = True)
```

Les noms des colonnes définie dans Django pour tout les crypto monnaies sont les suivants :

```
class Bitcoin(models.Model) :
    date       = models.DateTimeField(default=datetime.now, blank = False)
    open       = models.DecimalField(max_digits=10, decimal_places=3,null=False)
    high       = models.DecimalField(max_digits=10, decimal_places=3,null=False)
    low        = models.DecimalField(max_digits=10, decimal_places=3,null=False)
    close      = models.DecimalField(max_digits=10, decimal_places=3,null=False)
    volume     = models.DecimalField(max_digits=15, decimal_places=3,null=False)
    market_cap = models.DecimalField(max_digits=15, decimal_places=3,null=False,db_column='market cap')

    class Meta :
        ordering = ['-date']
```

- **Résultat :**

```
Entrée [53]: xrp_df_cleaned.head()
```

Out[53]:

	date	id	open	high	low	close	volume	market cap
0	2022-05-03	0	0.6142	0.6251	0.5982	0.6048	1.460418e+09	2.913020e+10
1	2022-05-02	1	0.6078	0.6293	0.6018	0.6143	1.687184e+09	2.954954e+10
2	2022-05-01	2	0.5871	0.6102	0.5810	0.6078	1.657999e+09	2.924049e+10
3	2022-04-30	3	0.6114	0.6271	0.5816	0.5871	1.649222e+09	2.824216e+10
4	2022-04-29	4	0.6443	0.6456	0.6039	0.6114	1.962198e+09	2.941186e+10

Le dernier étape maintenant et de chargé ce dataframe dans MongoDB, cela nécessite l'utilisation de bibliothéque **PyMongo** pour se connecte et interagit avec les bases de données.

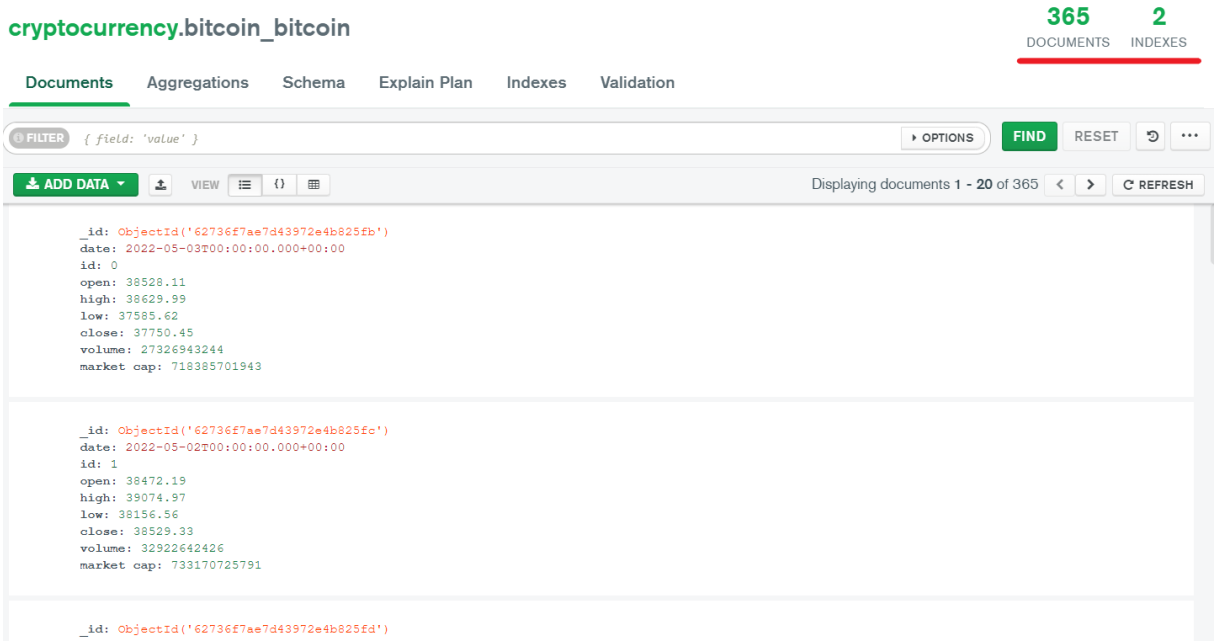
Tout d'abord il faut établir une connection vers la base de données, on spécifiant à la fois le nom de la BDD et la collection où on souhaiter de stocker les donneés.

```
Entrée [54]: # Making a Connection with MongoClient
client = MongoClient("mongodb://localhost:27017/")
# database
db = client["cryptocurrency"]
# collection
company= db["bitcoin_bitcoin"]
```

On peut d'abord chargé le dataframe dans notre collection comme suite :

```
Entrée [55]: btc_dict = btc_df_cleaned.to_dict("records")
bnb_dict = bnb_df_cleaned.to_dict("records")
eth_dict = eth_df_cleaned.to_dict("records")
xrp_dict = xrp_df_cleaned.to_dict("records")
company.insert_many(btc_dict)
```

- **Résultat :**


cryptocurrency.bitcoin_bitcoin
365
DOCUMENTS
2
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
{ field: 'value' }
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 20 of 365
REFRESH
_id: ObjectId('62736f7ae7d43972e4b825fb')
date: 2022-05-03T00:00:00.000+00:00
id: 0
open: 38528.11
high: 38629.99
low: 37585.62
close: 37750.45
volume: 27326943244
market cap: 718385701943
_id: ObjectId('62736f7ae7d43972e4b825fc')
date: 2022-05-02T00:00:00.000+00:00
id: 1
open: 38472.19
high: 39074.97
low: 38156.56
close: 38529.33
volume: 32922642426
market cap: 733170725791
_id: ObjectId('62736f7ae7d43972e4b825fd')

IV. **Analyse des données**

Nous sommes intéressés par le Adjusted closing price. Par conséquent, nous allons sélectionner le prix de clôture ajusté des crypto-monnaies.

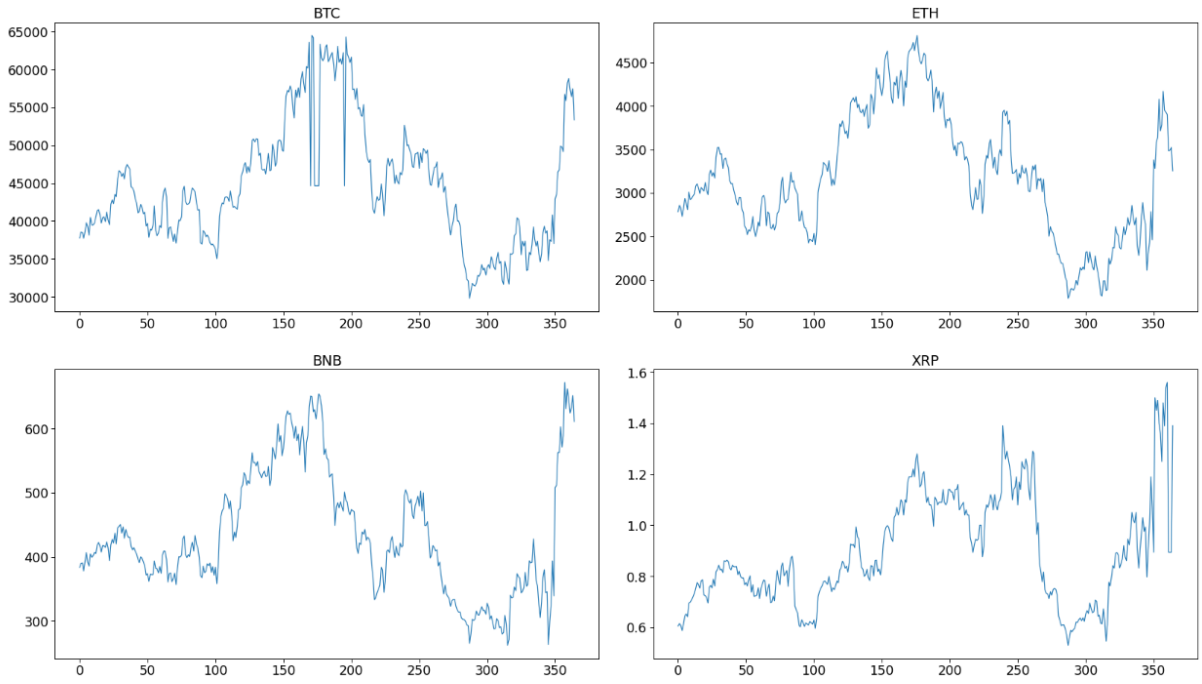
```
Entrée [185]: adj_close=merged_df["Close"]
```

```
Entrée [186]: adj_close.tail()
              # We can use either merged_df
              #print(merged_df.loc[:,('Close','BTC-USD')].head())
```

Out[186]:

Date	BTC-USD	ETH-USD	BNB-USD	XRP-USD
2021-05-08	58803.78	3902.65	646.10	1.56
2021-05-07	57356.40	3484.73	624.56	1.58
2021-05-06	56396.51	3490.88	633.28	1.60
2021-05-05	57424.01	3522.78	651.66	1.61
2021-05-04	53333.54	3253.63	611.20	1.39

Nous allons tracer le Adjusted closing price. Nous utilisons des subplots puisque les crypto-monnaies sont sur des échelles différentes.



Les graphiques sont des graphiques de séries chronologiques (time series plots), indiquant l'évolution du cours de l'action dans le temps. Comme les échelles sont différentes, nous ne pouvons pas comparer les graphiques, mais nous utiliserons plutôt le graphique des rendements cumulés.

IV.1 **Série de routeur**

Un rendement (return) est une variation du prix d'un actif dans le temps.

Les rendements peuvent être positifs, représentant un profit, ou négatifs, indiquant une perte.

Nous allons utiliser la fonction pandas pct_change() pour calculer les rendements.

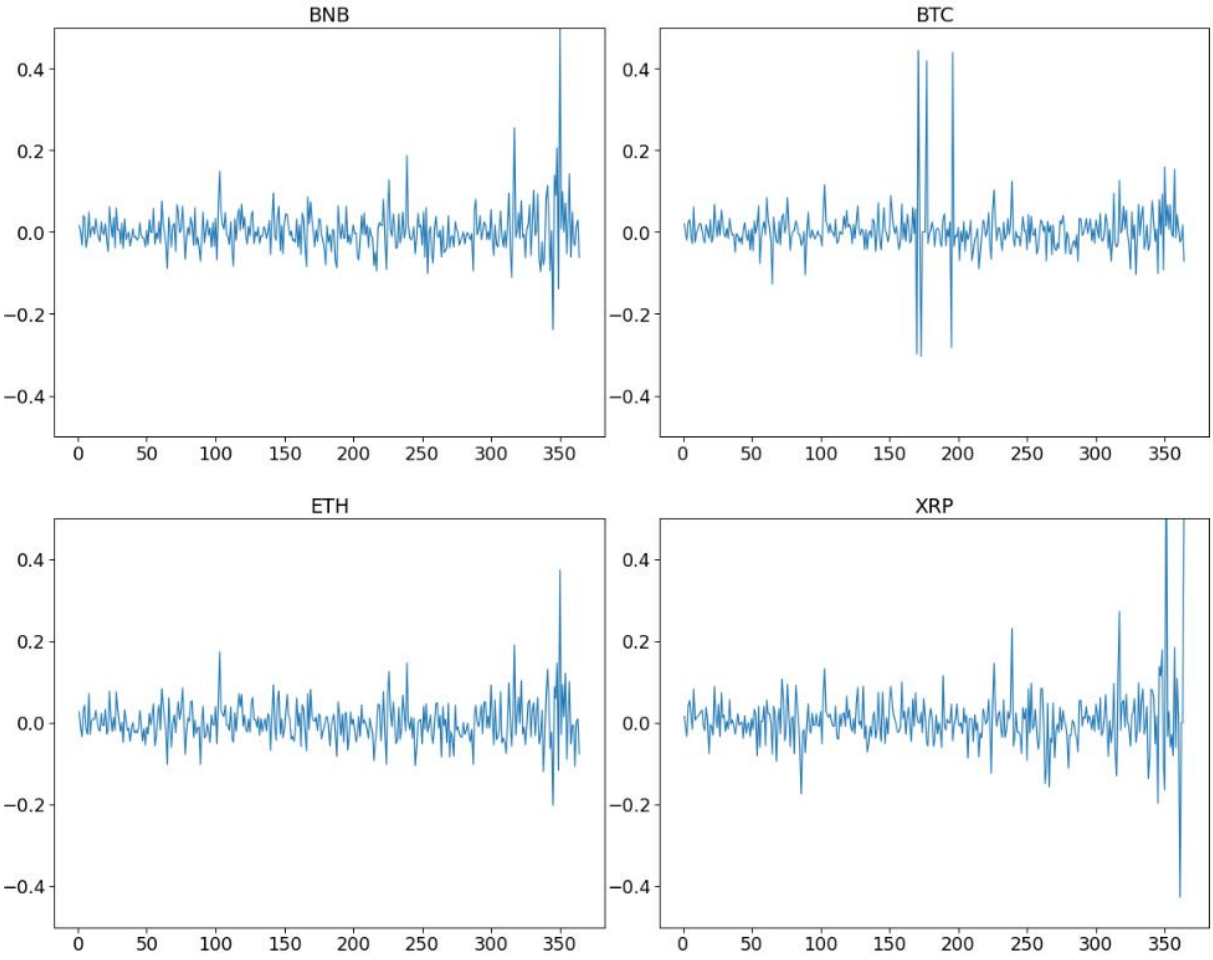
```
Entrée [156]: returns = adj_close.pct_change().dropna(axis=0)
```

```
Entrée [157]: returns.head()
```

Out[157]:

	BTC-USD	ETH-USD	BNB-USD	XRP-USD
1	0.020632	0.026560	0.016142	0.015708
2	-0.001563	-0.010377	0.001565	-0.010581
3	-0.019606	-0.034504	-0.032054	-0.034057
4	0.023729	0.031284	0.040474	0.041390
5	0.030148	0.043096	0.034753	0.053974

Voici les changements de prix relatifs des crypto-monnaies.



XRP est la plus volatile, suivie de Bitcoin, tandis que Ethereum est la moins volatile.

IV.2 **Volatilité**

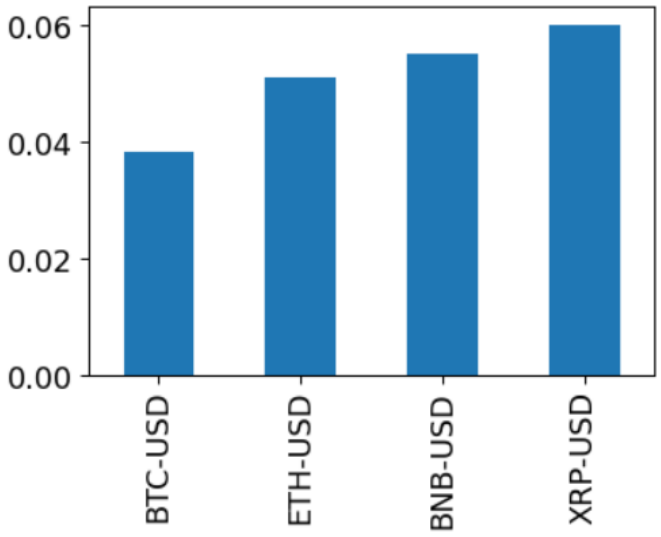
La volatilité est une mesure de la variation du prix d'un actif dans le temps. Plus l'écart-type est élevé, plus un actif est volatil.

```
Entrée [197]: returns.std()

Out[197]: BTC-USD    0.038221
          ETH-USD    0.051093
          BNB-USD    0.055182
          XRP-USD    0.060105
          dtype: float64
```

```
Entrée [198]: returns.std().plot(kind='bar')
```

```
Out[198]: <AxesSubplot:>
```



XRP (Ripple) est le plus volatil des quatre actifs, tandis que le Bitcoin est le moins volatil.

IV.3 **Rendements cumulatifs**

Le rendement cumulé exprime la variation totale du prix d'un actif dans le temps.

Nous utilisons la fonction pandas cumprod() pour calculer les rendements simples cumulés quotidiens.

```
Entrée [168]: # Cumulative return series
cum_returns = ((1 + returns).cumprod() - 1) *100
cum_returns.head()
```

Out[168]:

	BTC-USD	ETH-USD	BNB-USD	XRP-USD
1	2.063234	2.656028	1.614165	1.570767
2	1.903659	1.590814	1.773235	0.496032
3	-0.094224	-1.914510	-1.488996	-2.926587
4	2.276450	1.153951	2.498175	1.091270
5	5.359883	5.513242	6.060290	6.547619

```
Entrée [188]: cum_returns.plot(figsize=(20,6))
              plt.title('Cumulative Returns')
              plt.Text(0.5, 1.0, 'Cumulative Returns')

Out[188]: Text(0.5, 1.0, 'Cumulative Returns')
```

XRP (Ripple) surpasse l'Ethereum, le Bitcoin et le Binance à partir de juin 2021, tandis que le Bitcoin surpasse l'Ethereum et le XRP.

IV.4 **Corrélation**

J'ai calculé la corrélation sur le rendement car la corrélation sur les données de prix brutes peut donner des résultats biaisés.

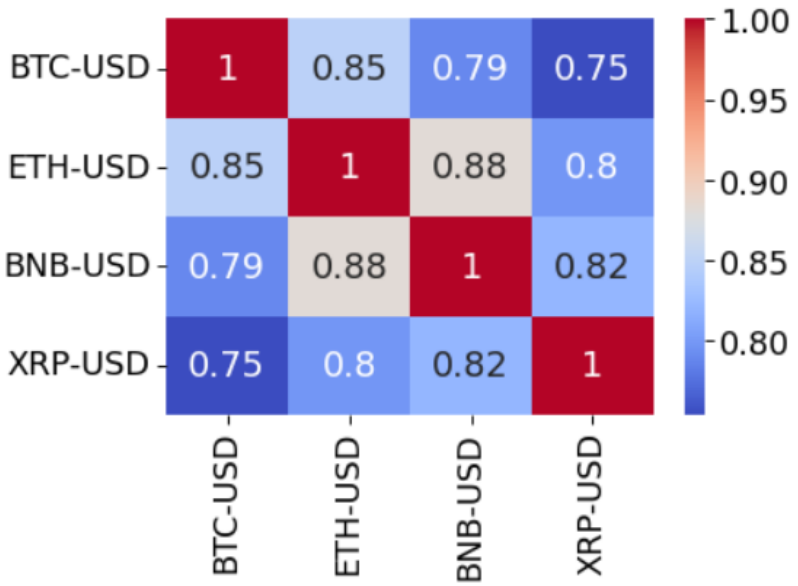
Les coefficients de corrélation proches de 1 indiquent une forte association positive, -1 indique une forte association négative et les coefficients proches de zéro n'indiquent aucune association.

```
Entrée [201]: #compute the correlations
              returns.corr()
```

Out[201]:

	BTC-USD	ETH-USD	BNB-USD	XRP-USD
BTC-USD	1.000000	0.848711	0.789368	0.754065
ETH-USD	0.848711	1.000000	0.881362	0.798101
BNB-USD	0.789368	0.881362	1.000000	0.821310
XRP-USD	0.754065	0.798101	0.821310	1.000000

Entrée [202]:
```
#plot the correlations
sns.heatmap(returns.corr(), annot=True, cmap='coolwarm')
plt.show()
```



Nous pouvons voir que le Binance et l'Ethereum sont fortement corrélés. Cela signifie que lorsque le Binance augmente, l'Ethereum augmente également, et lorsque le Binance diminue, l'Ethereum diminue aussi.

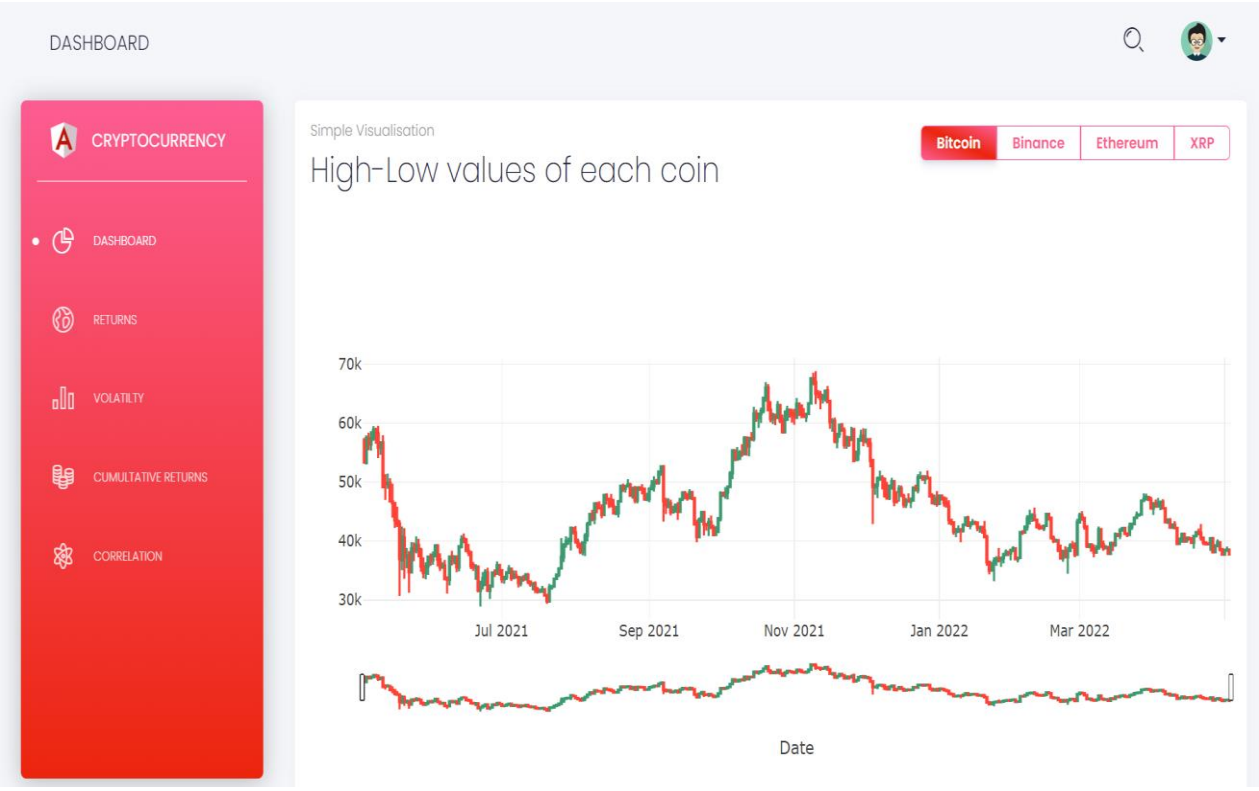
PARTIE III

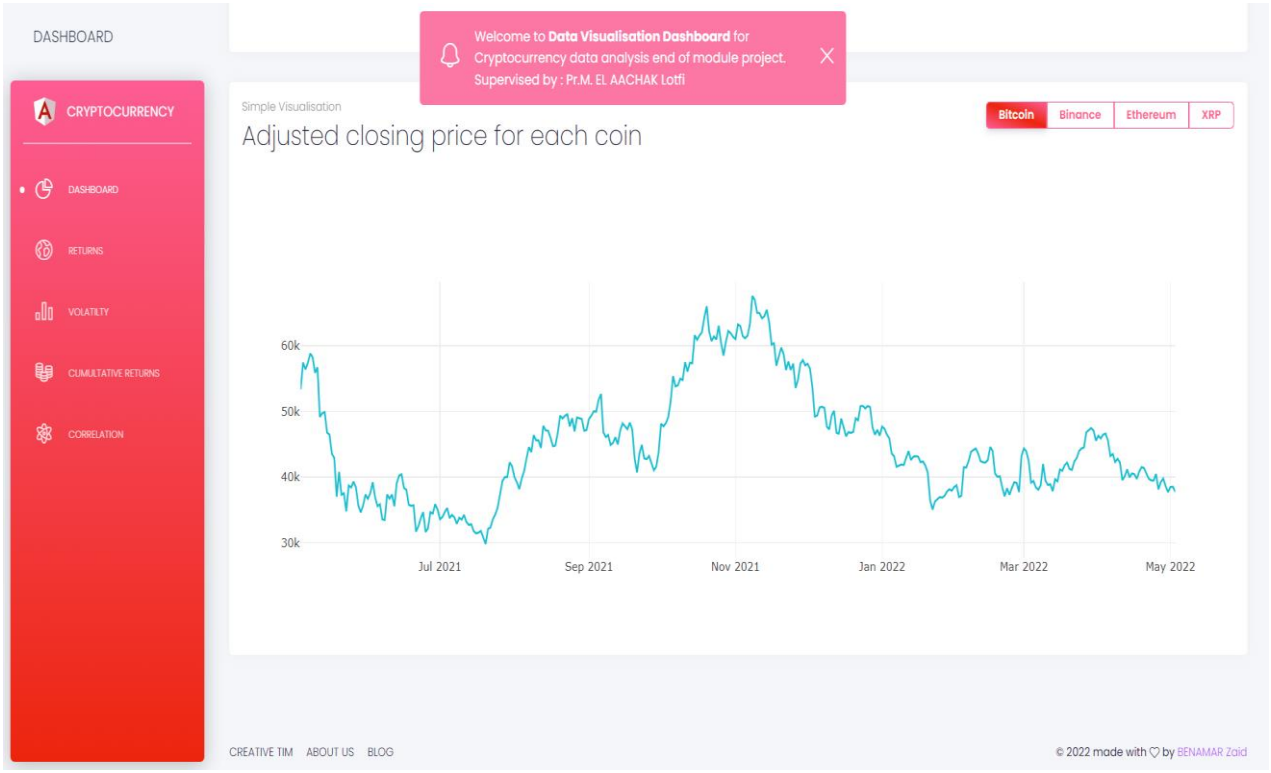
« Réalisation »

I. Présentation des interfaces de travail réalisé

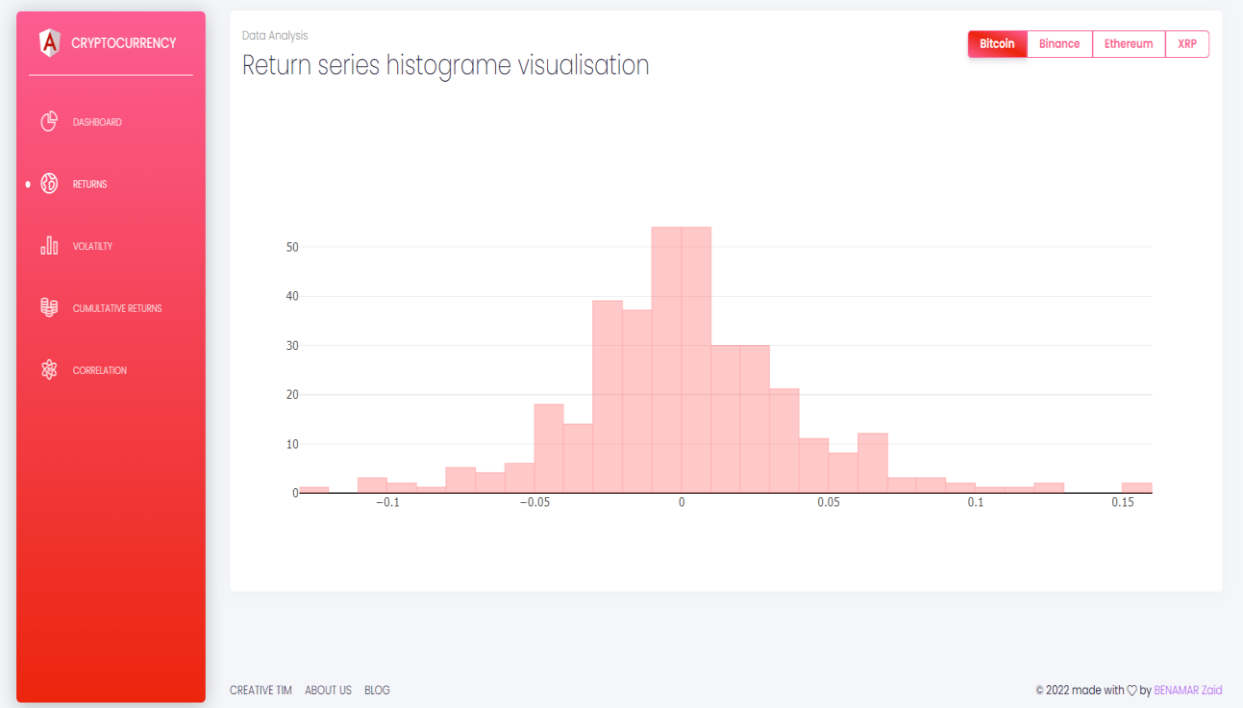
Dans cette partie on va réalisé une tableau de bord **(Dashboard)** de type single page application afin de visualisé les analyses des données qu'on a effectué dans la partie précédente.

Le tableau de bord developpé par Angular 13 qui contient des graph affiché par la bibliothéque Plotly.js.

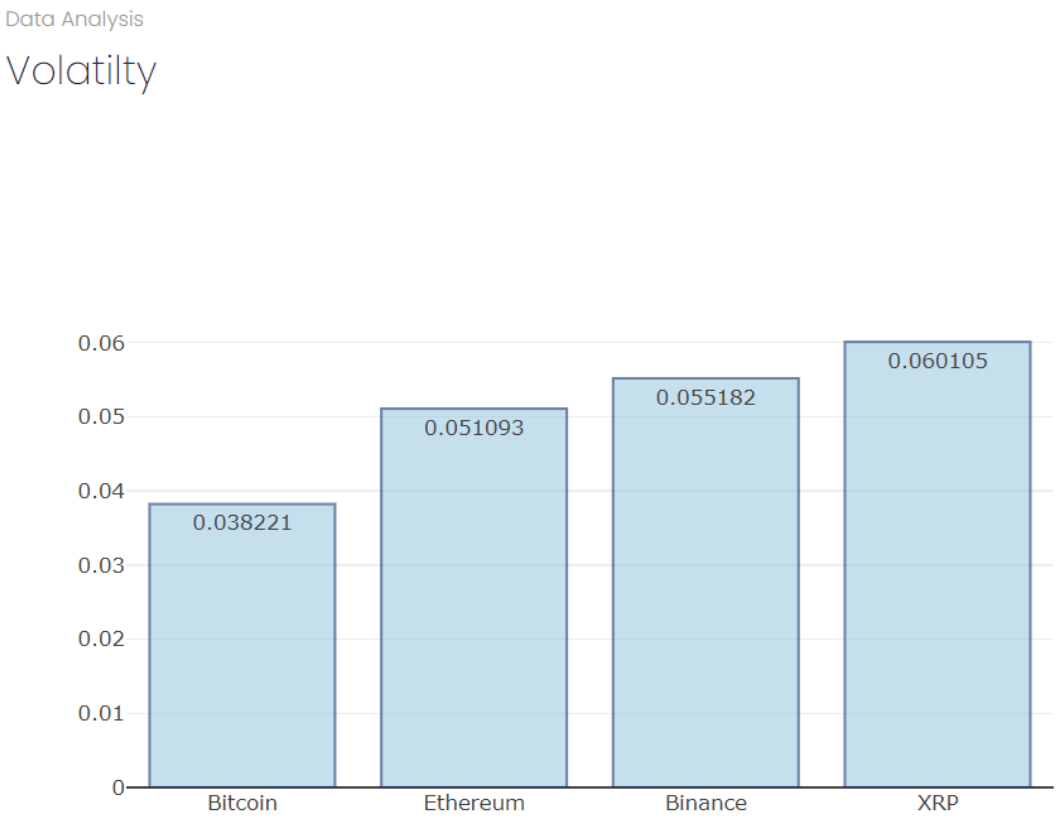




V.1 **Série de routeur**



V.2 **Volatilité**



V.3 **Rendements cumulatifs**

V.4 **Corrélation**

CONCLUSION :

Ce travail a pour objectif de concevoir et faire face aux différentes techniques de la préparation et l'analyse des données, qui permet d'évaluer les données à l'aide d'outils analytiques et statistiques.

Pour pouvoir compléter mon projet, j'avais détaillé les différentes étapes collection, de prétraitement et l'analyse de cette Data frame.

Durant ces parties, j'avais présenté la conception détaillée, à la fois, de mon projet, d'autre part j'avais aussi présenté les tâches qui a été accompli pour chaque partie, ensuite, j'ai présenté quelques captures d'écran montrant le bon fonctionnement du tableau de bord.