University Abdelmalek Essaadi
Faculty of Sciences and Techniques Tangier
Software & Intelligent Systems

Projet BI & BIG DATA

Réalisé par :

ZBAKH Manal

BARROU Mouad

Encadré par :

Mr. FENNAN Abdelhadi

Table des matières

Introduction

La **Business Intelligence** (BI) et le **Big Data** sont des concepts étroitement liés qui sont souvent utilisés ensemble dans le contexte de l'analyse des données et de la prise de décision.

Business intelligence (BI) est l'utilisation de données, de technologies et d'outils pour collecter, stocker et analyser des données afin de gagner des insights et informer les décisions d'affaires. La BI implique souvent l'utilisation de tableaux de bord, de rapports et d'autres visualisations pour présenter les données de manière facilement compréhensible, et peut inclure des activités telles que la recherche de marché, l'exploration de données et l'analyse prédictive.

Le **big data** fait référence aux grandes volumes de données structurées et non structurées que les organisations génèrent et collectent à partir de différentes sources, y compris les réseaux sociaux, les capteurs, les transactions, etc. Le volume, la variété et la vitesse du big data peuvent rendre difficile le traitement et l'analyse à l'aide de méthodes traditionnelles, mais les nouvelles technologies telles que le calcul distribué et l'apprentissage automatique ont rendu possible l'extraction de valeur du big data de manières qui n'étaient pas possibles auparavant.

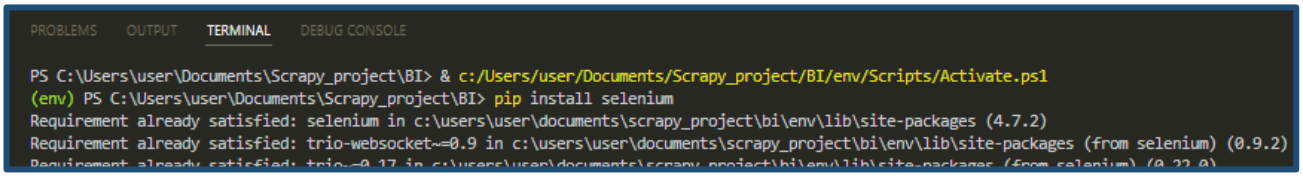
La **BI** et le **big data** sont souvent utilisées ensemble pour analyser de grandes quantités de données et extraire des insights qui peuvent informer les décisions d'affaires. Les outils et techniques de BI peuvent être utilisés pour visualiser et présenter les données de manière facilement compréhensible, tandis que les technologies de big data peuvent être utilisées pour stocker et traiter rapidement et efficacement de grandes quantités de données. Ensemble, la BI et le big data peuvent aider les organisations à prendre des décisions mieux informées et à stimuler la croissance de l'entreprise.

Outils et installation

Sélénium:

Sélénium est un outil open source disponible pour automatiser les tests requis à effectuer sur les navigateurs Web existants. (En termes plus simples, il teste les applications Web en utilisant n'importe quel navigateur Web comme Google Chrome, Mozilla Firefox, Opera Browser, Internet Explorer, MS Edge, etc.)

Installation de Sélénium:



Flask:

Flask est un micro framework open-source de développement web en Python. Il est classé comme microframework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. Il n'intègre pas de système d'authentification, pas de couche d'abstraction de base de données, ni d'outil de validation de formulaires. Cependant, de nombreuses extensions permettent d'ajouter facilement des fonctionnalités.

PySpark:

PySpark est un outil d'analyse de données créé par Apache Spark Community pour utiliser Python avec Spark. Cela nous permet de travailler avec RDD (Resilient Distributed Dataset) et DataFrames en Python. PySpark possède de nombreuses fonctionnalités qui en font un cadre aussi étonnant et lorsqu'il s'agit de traiter l'énorme quantité de données, PySpark nous fournit un traitement rapide et en temps réel, une flexibilité, un calcul en mémoire et diverses autres fonctionnalités. Il s'agit d'une bibliothèque Python pour utiliser Spark qui combine la simplicité du langage Python avec l'efficacité de Spark.

MongoDB :

MongoDB est une base de données NoSQL orientée document. Elle se distingue des bases de données relationnelles par sa flexibilité et ses performances. Découvrez tout ce que vous devez savoir sur cet outil incontournable pour l'ingénierie des données.

Angular :

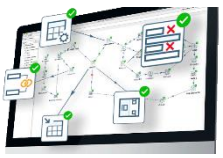
Angular est un framework open source JavaScript développé par Google. Ce framework est utilisé pour développer des applications web et mobile. Avec cette technologie, on réalise des interfaces de type monopage ou “one page” qui fonctionnent sans rechargement de la page web.

Chart.js :

Chart.js est une bibliothèque JavaScript open source gratuite pour la visualisation de données. Créée par le développeur Web Nick Downie en 2013, la bibliothèque est maintenant maintenue par la communauté et est la deuxième bibliothèque de graphiques JS la plus populaire sur GitHub par le nombre d'étoiles après D3.js.

Pentaho Data Integration:

PDI (Pentaho Data Integration), qui était auparavant connu sous le nom de Kettle, est un logiciel d'ETL (Extract, Transform, Load) Open Source qui permet la conception ainsi que l'exécution des opérations de manipulation et de transformation de données très complexes.



Installation de Kettle:

On installe Pentaho Data Intégration a partir du site suivant : https://sourceforge.net/ Ensuite il faut accéder au dossier data intégration et ouvrir spoon.bat , d’où Spoon est l'interface graphique utilisée pour créer des transformations et des travaux .

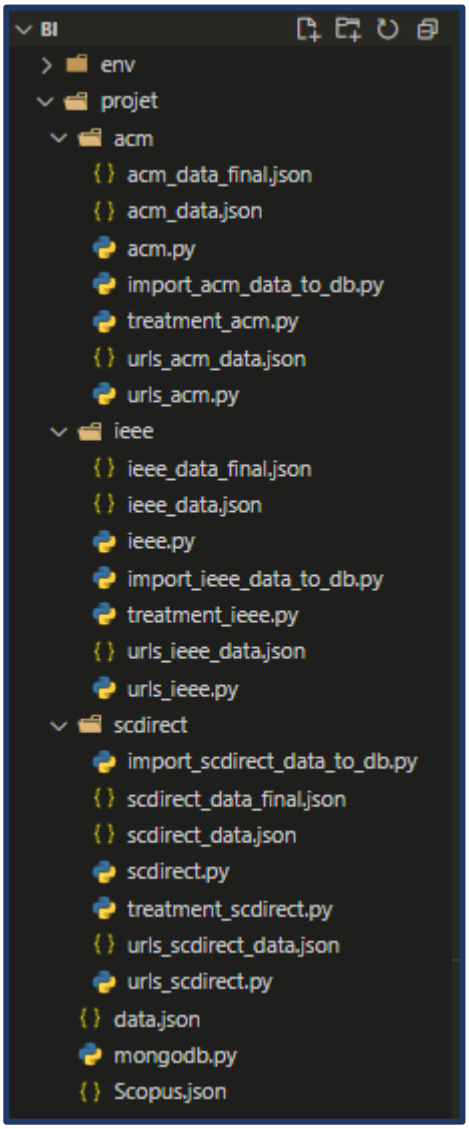
Power BI:

Microsoft Power BI est une solution d'analyse de données de Microsoft. Il permet de créer des visualisations de données personnalisées et interactives avec une interface suffisamment simple pour que les utilisateurs finaux créent leurs propres rapports et tableaux de bord.

Extraction des données

Dans cette étape on a commencé le grattage des données à partir du 3 sites sont: ACM, scienceDirect et IEEE, afin d'enregistrer les articles publiés dans une base de données no relationnel qui est MongoDB.

L’architecture du projet



ACM

La bibliothèque numérique ACM (DL) est la base de données la plus complète au monde d'articles en texte intégral et de littérature bibliographique couvrant l'informatique et les technologies de l'information.

Pour collecter les donnes à partir de ce site on a commencé à rassembler les urls des journaux en utilisant le script suivant (existe dans le fichier de paramétrage “**urls_acm.py**”):

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from lib2to3.pgen2 import driver
from time import sleep
from selenium.webdriver.common.by import By
import json
from selenium.webdriver import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import StaleElementReferenceException
from selenium.webdriver.support import expected_conditions


DRIVER_PATH = 'C:\chromedriver.exe'
current_page = 0
turn_it = True
all_journals = list()


while (turn_it):
    dico_journal = {}
    driver = webdriver.Chrome(executable_path=DRIVER_PATH)
    web_site =
"https://dl.acm.org/action/doSearch?AllField=Blockchain&expand=all&ConceptID=118230
&startPage=" + \
        str(current_page)+"&pageSize=50"
    driver.get(web_site)
    sleep(4)

    journals = driver.find_elements(By.CLASS_NAME, 'issue-item__title')

    if journals == []:
        turn_it = False

    enf_of_recherche = False

    for index, val in enumerate(journals):
        journals = driver.find_elements(By.CLASS_NAME, 'issue-item__title')
        Title = journals[index].find_element(
            By.CSS_SELECTOR, 'span.hlFld-Title')
        url = Title.find_element(By.XPATH, './a[1]').get_attribute('href')

        dico_journal = url

        all_journals.append(dico_journal)
```

```
    print(current_page)

    driver.close
    # if(current_page == 0):
    #     turn_it = False
    current_page += 1

    if (dico_journal == {}):
        turn_it = False


with open("urls_acm_data.json", "w") as write_file:
    json.dump(all_journals, write_file, indent=4)

driver.close()
```

Ensuite, basant sur ce dernier on collecte les donnée en utilisant le script suivant (existe dans le fichier de paramétrage “**acm.py**”):

```
import json
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from lib2to3.pgen2 import driver
from time import sleep
from selenium.webdriver.common.by import By
import json
from selenium.webdriver import ActionChains
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import WebDriverException


DRIVER_PATH = 'C:\chromedriver.exe'
current_link = 1
turn_it = True
all_journals = list()

chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')


while (turn_it):
    dico_journal = {}

    with open('urls_acm.json') as data_file:
        data = json.load(data_file)
    for v in data:
```

```
        driver = webdriver.Chrome(
            executable_path=DRIVER_PATH, chrome_options=chrome_options)
        try:
            driver.get(v)
        except WebDriverException:
            print("page down")
        sleep(4)
        driver.find_element(
            By.CSS_SELECTOR, 'a.btn.blue.cookiePolicy-popup__close.pull-
right').click()

        authors = []
        locations = []

        try:
            fullpage = driver.find_element(By.XPATH, '//main')
        except:
            pass

        try:
            sectionOne = fullpage.find_element(
                By.XPATH, '//article/div[1]/div[2]/div[1]/div[1]')
        except:
            pass

        try:
            TypeAccessSection = sectionOne.find_element(By.XPATH, './div[1]')
        except:
            pass

        try:
            type = sectionOne.find_element(By.CLASS_NAME, 'issue-heading').text
        except:
            type = ""

        try:
            access = sectionOne.find_element(
                By.CLASS_NAME, 'article__access').text
        except:
            access = ""

        try:
            TitleSection = sectionOne.find_element(By.XPATH, './div[2]')
        except:
            pass

        try:
            title = TitleSection.find_element(By.XPATH, './h1').text
        except:
            title = ""
```

```
        try:
            AuthorSection = sectionOne.find_element(
                By.XPATH, './div[3]/div[1]/ul[1]')
        except:
            pass

        try:
            listauthors = AuthorSection.find_elements(
                By.CLASS_NAME, 'loa__item')
        except:
            pass

        try:
            for auth in listauthors:
                author = auth.find_element(
                    By.XPATH, './a[1]').get_attribute('title')
                authors.append(author)
        except:
            pass

        try:
            for auth in WebDriverWait(AuthorSection,
8).until(EC.visibility_of_all_elements_located((By.CLASS_NAME, "loa__item"))):
                listauthors = AuthorSection.find_elements(
                    By.CLASS_NAME, 'loa__item')
                WebDriverWait(auth, 8).until(
                    EC.element_to_be_clickable((By.XPATH, './a[1]'))).click()
                sleep(4)
                location = auth.find_element(By.XPATH, './div[1]/div[2]').text
                if (location != "Search about this author"):
                    location = location.replace(
                        '\nSearch about this author', '')
                    location = location.replace('\nView Profile', '')
                    locations.append(location)
                TitleSection.find_element(By.XPATH, './h1').click()
        except:
            pass

        try:
            ResourcesDOiSection = sectionOne.find_element(By.XPATH, './div[4]')
        except:
            pass

        try:
            publisher = ResourcesDOiSection.find_element(
                By.XPATH, './div[1]/a[1]/span[1]').text
        except:
            publisher = ""
```

```
        try:
            doi = str(ResourcesDOiSection.find_element(
                By.CLASS_NAME, 'issue-item__doi').get_attribute('href'))
        except:
            doi = ""

        try:
            DateSection = sectionOne.find_element(By.XPATH, './div[5]')
        except:
            pass

        try:
            date = DateSection.find_element(By.XPATH, './span[2]').text
        except:
            date = ""

        try:
            sectionTwoAbstract = fullpage.find_element(
                By.XPATH,
'//article/div[2]/div[2]/div[2]/div[1]/div[1]/div[2]/div[1]')
        except:
            pass

        try:
            abstract = sectionTwoAbstract.find_element(By.XPATH, './p[1]').text
        except:
            abstract = ""

        try:
            sectionTreekey = fullpage.find_element(
                By.XPATH, '//article/div[2]/div[2]/div[2]/div[5]/ol[1]/li[1]')
        except:
            pass

        try:
            Keywords = sectionTreekey.find_element(By.XPATH, './h6').text
        except:
            Keywords = ""

        dico_journal = {"Title": title, "Abstract": abstract, "Type": type,
"Access": access,
                        "Date": date, "DOI": doi, "Publisher": publisher,
"Authors": authors, "Locations": locations}

        # stocker data au nivaeu de cette liste
        all_journals.append(dico_journal)

        print(current_link)
        current_link += 1
```

```
    driver.close

    turn_it = False


with open("acm_data.json", "w") as write_file:
    # convertir la liste sous une format json
    json.dump(all_journals, write_file, indent=4)

driver.close()
```

ScienceDirect

ScienceDirect est un site web géré par l'éditeur Elsevier. Lancée en mars 19971, la plateforme permet d'accéder à plus de 3 800 revues académiques qui forment plus de 14 millions de publications scientifiques revues par des pairs2.

Pour collecter les donnes à partir de ce site on a commencé à rassembler les urls des journaux en utilisant le script suivant (existe dans le fichier de paramétrage "**urls_scdirect.py**"):

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from lib2to3.pgen2 import driver
from time import sleep
from selenium.webdriver.common.by import By
import json
from selenium.webdriver import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
from selenium.common.exceptions import NoSuchElementException
from selenium.common.exceptions import StaleElementReferenceException
from selenium.webdriver.support import expected_conditions


DRIVER_PATH = 'C:\chromedriver.exe'
current_page = 1
turn_it = True
all_journals = list()


while (turn_it):
    dico_journal = {}
    driver = webdriver.Chrome(executable_path=DRIVER_PATH)
    web_site =
"https://www.sciencedirect.com/search?qs=blockchain&show=100&offset=" + \
        str(current_page)+"00"
    driver.get(web_site)
    sleep(4)
```

```
    journals = driver.find_elements(
        By.CLASS_NAME, 'ResultItem.col-xs-24.push-m')

    if journals == []:
        turn_it = False

    enf_of_recherche = False

    for index, val in enumerate(journals):
        journals = driver.find_elements(
            By.CLASS_NAME, 'ResultItem.col-xs-24.push-m')
        url = journals[index].find_element(
            By.XPATH, './div[1]/div[2]/h2[1]/span[1]/a[1]').get_attribute('href')

        dico_journal = url

        all_journals.append(dico_journal)

    print(current_page)

    driver.close
    # if (current_page == 50):
    #     turn_it = False
    current_page += 1

    if (dico_journal == {}):
        turn_it = False


with open("urls_scdirect_data.json", "w") as write_file:
    json.dump(all_journals, write_file, indent=4)

driver.close()
```

Ensuite, basant sur ce dernier on collecte les donnée en utilisant le script suivant (existe dans le fichier de paramétrage “**scdirect.py**”):

```
import json
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from lib2to3.pgen2 import driver
from time import sleep
from selenium.webdriver.common.by import By
import json
from selenium.webdriver import ActionChains
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from selenium.common.exceptions import WebDriverException
```

```
DRIVER_PATH = 'C:\chromedriver.exe'
current_link = 1
turn_it = True
all_journals = list()

while (turn_it):
    dico_journal = {}

    with open('urls_scdirect_data.json') as data_file:
        data = json.load(data_file)
    for v in data:
        driver = webdriver.Chrome(
            executable_path=DRIVER_PATH)
        try:
            driver.get(v)
        except WebDriverException:
            print("page down")
        sleep(4)

        authors = []
        locations = []

        try:
            fullpage = driver.find_element(
                By.CLASS_NAME, 'article-wrapper.u-padding-s-top.grid.row')
        except:
            pass

        try:
            underFullPage = fullpage.find_element(
                By.XPATH, '//article')
        except:
            pass

        try:
            JournalDateSection = underFullPage.find_element(
                By.XPATH, './div[1]')
        except:
            pass

        try:
            TypeTitleSection = underFullPage.find_element(By.XPATH, './h1[1]')
        except:
            pass

        try:
            AuthorsSection = underFullPage.find_element(By.XPATH, './div[2]')
        except:
            pass
```

```
        try:
            DOISection = underFullPage.find_element(By.XPATH, './div[3]')
        except:
            pass

        try:
            AccessSection = underFullPage.find_element(
                By.CLASS_NAME, 'LicenseInfo')
        except:
            pass

        try:
            AbstractSection = underFullPage.find_element(
                By.CLASS_NAME, 'Abstracts')
        except:
            pass

        try:
            Publisher = JournalDateSection.find_element(
                By.XPATH, './div[2]/h2[1]/a[1]').text
        except:
            Publisher = ""

        try:
            Date = JournalDateSection.find_element(
                By.XPATH, './div[2]/div[1]').text
        except:
            Date = ""

        try:
            Title = TypeTitleSection.find_element(By.XPATH, './span[1]').text
        except:
            Title = ""

        try:
            Type = TypeTitleSection.find_element(By.XPATH, './div[1]').text
        except:
            Type = ""

        Authors = []
        Locations = []

        try:
            AuthorPart = AuthorsSection.find_element(
                By.XPATH, './div[1]/div[1]/div[1]')
        except:
            pass

        try:
```

```
            listauthors = AuthorPart.find_elements(By.XPATH, "./button")
        except:
            listauthors = []

        try:
            for auth in listauthors:
                author = auth.find_element(
                    By.XPATH, './span[1]').text
                Authors.append(author)
        except:
            pass

        try:
            DOI = DOISection.find_element(
                By.XPATH, "./a[1]").get_attribute('href')
        except:
            DOI = ""

        try:
            Access = AccessSection.find_element(By.XPATH, "./div[2]").text
        except:
            Access = ""

        try:
            Abstract = AbstractSection.find_element(
                By.XPATH, "./div[1]/div[1]/p[1]").text
        except:
            Abstract = ""

        try:
            for auth in listauthors:
                auth.click()
                sleep(4)
                Location = driver.find_element(
                    By.CLASS_NAME, "affiliation").text
                Locations.append(Location)
                TypeTitleSection.find_element(By.XPATH, './span[1]').click()
        except:
            pass

        dico_journal = {"Title": Title, "Abstract": Abstract, "Type": Type,
"Access": Access,
                        "Date": Date, "DOI": DOI, "Publisher": Publisher,
"Authors": Authors, "Locations": Locations}

        # stocker data au nivaeu de cette liste
        all_journals.append(dico_journal)
        with
open("C:/Users/user/Documents/Scrapy_project/BI/projet/scdirect1/scdirect_data.json
", "w") as write_file:
```

```
            json.dump(all_journals, write_file, indent=4)

        print(current_link)
        current_link += 1

    driver.close

    turn_it = False


with open("scdirect_data.json", "w") as write_file:
    # convertir la liste sous une format json
    json.dump(all_journals, write_file, indent=4)

driver.close()
```

IEEE

IEEE publie les principaux journaux, transactions, lettres et magazines en génie électrique, informatique, biotechnologie, télécommunications, électricité et énergie, et des dizaines d'autres technologies

Au niveau de ce site on fait la même chose que les sites précèdent pour la collection des donnes.

Pour accéder aux scripts, vous pouvez consulter le lien suivant : **https://github.com/manal-zbakh/BigData-Bi.git**

Aussi on fait comme pour les autre site un traitement pour garder les même index pour les 3 sites, on utilisant par exemple pour ieee le script suivant :

```
import json
import datetime
from dateutil.parser import parse


with open('ieee_data.json') as data_file:
    data = json.load(data_file)

dico_journal = {}
all_journals = []
# date_set = set()

for v in data:
    try:
        date1 = v["Day"]+' ' + v["MonthName"]+' ' + v["Year"]
    except:
        date1 = ""
    Access = ""
```

```
    LocationsPro = []
    Universities = []
    Countries = []
    university_set = set()
    Country_set = set()
    Author_set = set()

    for author in v["Authors"]:

        Author_set.add(author)
        Author_s = ';'.join(author for author in Author_set)

    for location in v["Locations"]:
        location = location.split('\n', 1)[0]
        location = location.replace('View Profile', '')
        university_set.update(location.split(',')[0:2])
        University = ';'.join(university for university in university_set)

        Country = Country_set.add((location.split(',')[-1]))
        Country = ';'.join(Country for Country in Country_set)

        if (Country == University):
            Country = ""
        LocationsPro.append(location)

    dico_journal = {"Title": v["Title"], "Abstract": v["Abstract"], "Type":
v["Type"], "Access": Access, "Date": date1, "Year": v["Year"], "Month": v["Month"],
"MonthName": v["MonthName"],
                    "Day": v["Day"], "DOI": v["DOI"], "Publisher": v["Publisher"],
"Authors": Author_s, "Locations": v["Locations"], "Locations": LocationsPro,
"Universities": University, "Countries": Country}

    all_journals.append(dico_journal)

with open("ieee_data_final.json", "w") as write_file:
    json.dump(all_journals, write_file, indent=4)
```

La structure suivie est la suivante:


```
{
    "Title": "Incentivizing Data Quality in Blockchain-Based Systems\u2014The Case of the Digital Cardossier",
    "Abstract": "Inspired by an industry initiative to address the celebrated market for lemons (poor-quality used cars), we investigate how incentives for a permissioned
    "Type": "RESEARCH-ARTICLE",
    "Access": "OPEN ACCESS",
    "Date": "09 September 2022",
    "Year": "2022",
    "Month": "9",
    "MonthName": "September",
    "Day": "9",
    "DOI": "https://doi.org/10.1145/3538228",
    "Publisher": "Distributed Ledger Technologies: Research and Practice",
    "Authors": "Liudmila Zavolokina;Florian Spychiger;Gerhard Schwabe;Claudio J. Tessone",
    "Locations": [
        "Institute of Organizational Viability, School of Management and Law University of Applied Sciences, Blockchain and Distributed Ledger Group and UZH Blockchain Cen
        "Blockchain and Distributed Ledger Group and UZH Blockchain Center, Faculty of Business, Economics and Informatics, University of Zurich, Zurich, Switzerland",
        "Digital Society Initiative and UZH Blockchain Center, Faculty of Business, Economics and Informatics, University of Zurich, Zurich, Switzerland",
        "Information Management Research Group and UZH Blockchain Center, Faculty of Business, Economics and Informatics, University of Zurich, Zurich, Switzerland"
    ],
    "Universities": "Information Management Research Group and UZH Blockchain Center;Blockchain and Distributed Ledger Group and UZH Blockchain Center; Faculty of Business
    "Countries": " Switzerland"
},
```

Enregistrements des données

Pour stocker les données dans Mongodb on est besoin d'ouvrir une session dans MongoDB et d'ajouter la configuration ci-dessous dans le fichier de paramétrage "**mongodb.py**" :

```
import json
from pymongo import MongoClient

client = MongoClient('localhost', 27017)

# le nom de bdd
db = client['Projet']

# le nom de collection
collection = db['data']
# collection = db['scopus']

# le chemin de data sous form json
with open('C:/Users/user/Documents/Scrapy_project/BI/projet/data.json') as file:
    # with open('C:/Users/user/Documents/Scrapy_project/BI/projet/Scopus.json') as
file:
    file_data = json.load(file)

collection.insert_many(file_data)

client.close()
```

Résultats sur MongoDb


```
_id: ObjectId('63adcb95e6bf26d2f46a06c5')
Title: "Blockchain for Genomics: A Systematic Literature Review"
Abstract: "Human genomic data carry unique information about an individual and of…"
Type: "REVIEW-ARTICLE"
Access: "OPEN ACCESS"
Date: "01 December 2022"
Year: "2022"
Month: "12"
MonthName: "December"
Day: "1"
DOI: "https://doi.org/10.1145/3563044"
Publisher: "Distributed Ledger Technologies: Research and Practice"
Authors: "Fatih Turkmen;Dimka Karastoyanova;Mohammed Alghazwi;Joeri Van Der Veld…"
› Locations: Array
Universities: " Groningen;University Medical Center Groningen;University of Groningen"
Countries: " The Netherlands"
```

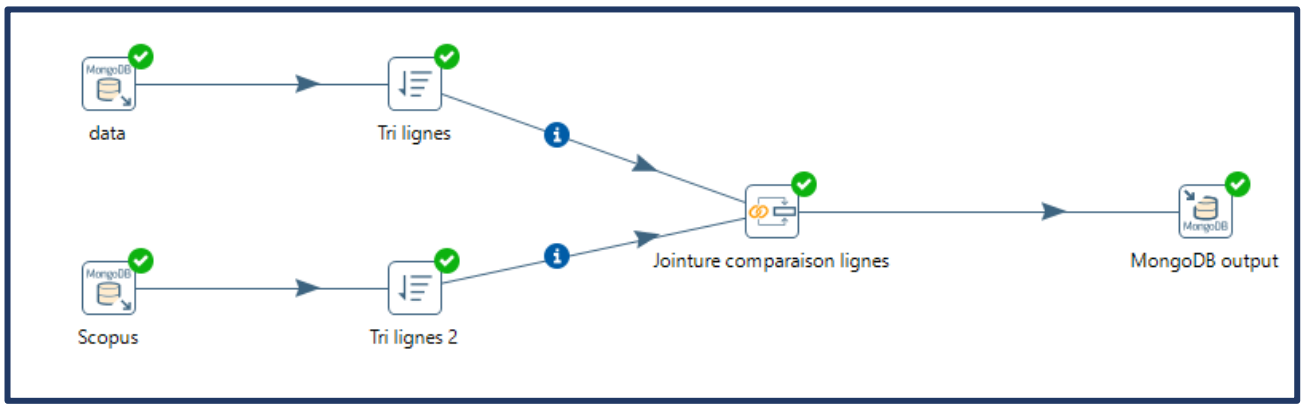
Analyse Bi

Cette partie est dédié pour l'analyse BI en misant en place un entrepôt de données dans SGBDR MySQL , en utilisant PDI (Pentaho Data Integration), ou Kettle comme un outil ETL, et pour faire l'analyse et reporting on utilise l'outil Microstrategy …

Pour faire une analyse BI on doit tout d'abord passer par les ETLs et construire une schéma en étoile.

Les jointures :

On va essayer de faire les jointures entre notre base de donnée et la table des scopus qui contient les quartiles (Q1, Q2 …) de chaque journal pour faire la correspondance entre chaque article et son quartile.



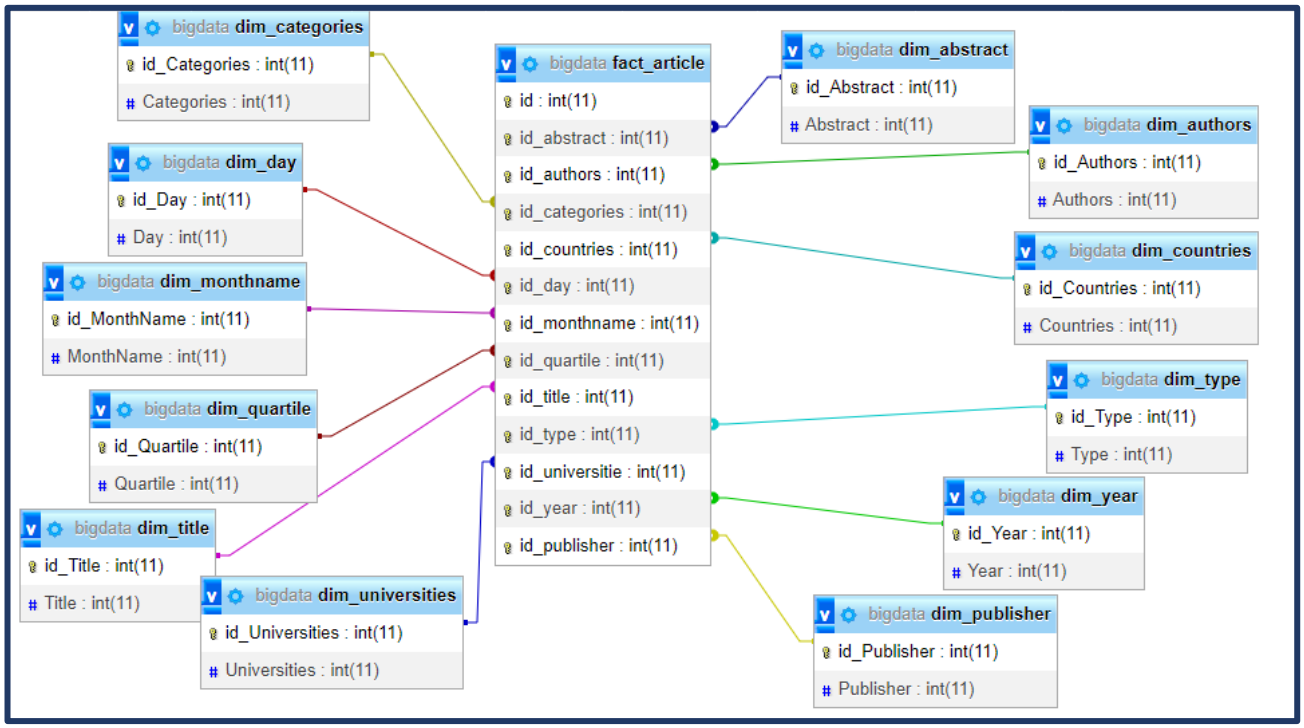
Et voilà le résultat :


```
_id: ObjectId('63adea689de841044c18f25d')
Month: "9"
DOI: "https://doi.org/10.1145/2946802"
Title: "Performance and Security Improvements for Tor: A Survey"
Publisher: "ACM Computing Surveys"
Date: "21 September 2016"
Year: "2016"
Day: "21"
MonthName: "September"
Abstract: "Tor [Dingledine et al. 2004] is the most widely used anonymity network…"
Universities: " ON;University of Waterloo; Doha-Qatar;Qatar University and Qatar Comp…"
Type: "SURVEY"
Countries: " Canada; Doha-Qatar"
Locations: "[Qatar University and Qatar Computing Research Institute, Doha-Qatar, …"
Authors: "Mashael Alsabah;Ian Goldberg"
Title_1: "ACM Computing Surveys"
Year_1: "2016"
SJR Best Quartile: "Q1"
Categories: "Computer Science (miscellaneous) (Q1); Theoretical Computer Science (Q…"
```

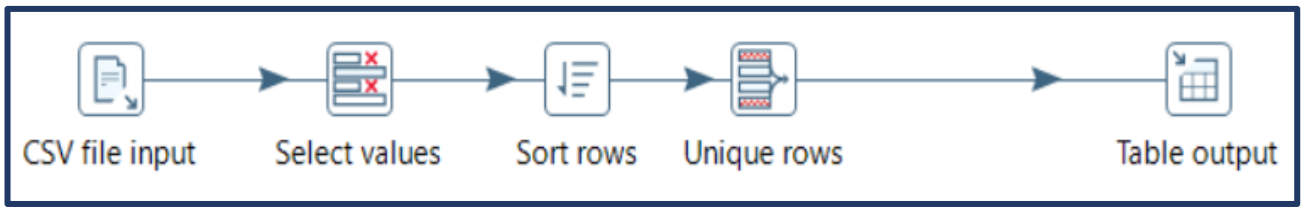
Génération de l'entrepôt de données :

Avant de commencer l'analyse, il faut faire la modélisation en faisant le schéma en étoile qui est le schéma d'entrepôt de données le plus simple, il contient la table de faits et les tables de dimensions associées.

Notre table de faits et les table de dimensions sont schématiser comme la suivante :



Pour la génération des dimensions on va utiliser le logiciel pentaho data integration, et pour générer les dimensions on doit faire des transformations, comme la suivante :



- Explications des étapes :

Etape 1 : Dans cette étape on importe le fichier csv (exporté à partir la base de données mongodb) .

Etape 2 : on fait la sélection de notre colonne.

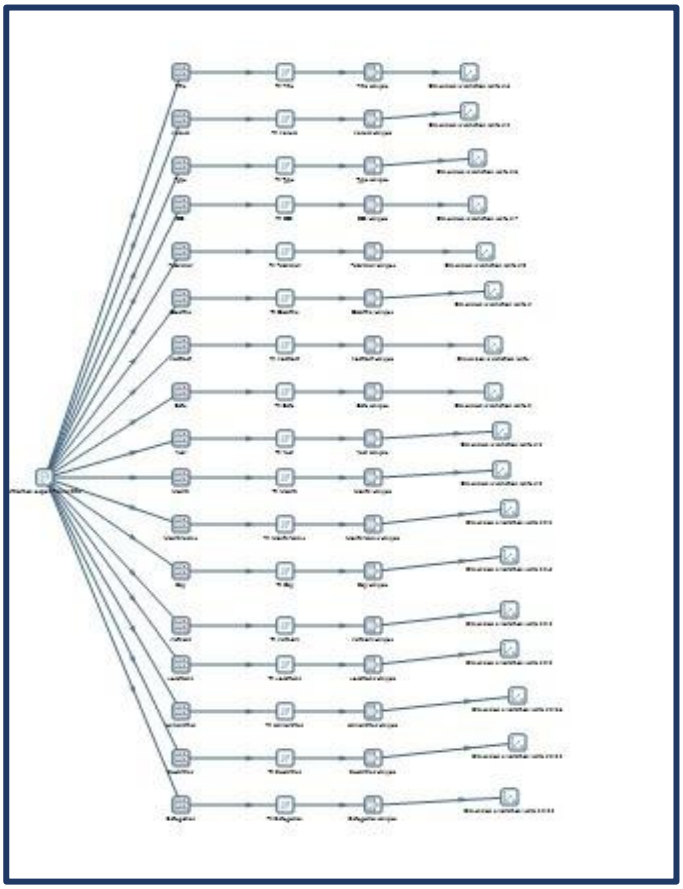
Etape 3 : on fait le tri des données pour faciliter la manipulation après.

Etape 4 : on élimine les lignes doublant.

Etape 5 : On doit premièrement configurer la connexion à la base de données.

L'étape finale dans laquelle on va stocker la dimension dans une table output.

On répète la même chose pour toutes les dimensions.

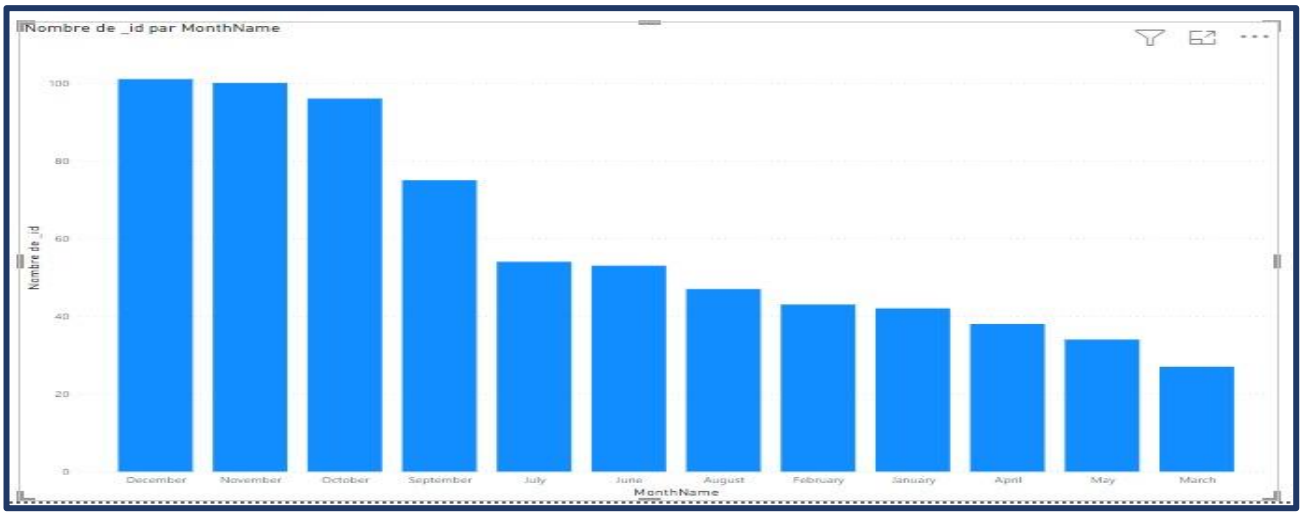


La génération de la table des faits

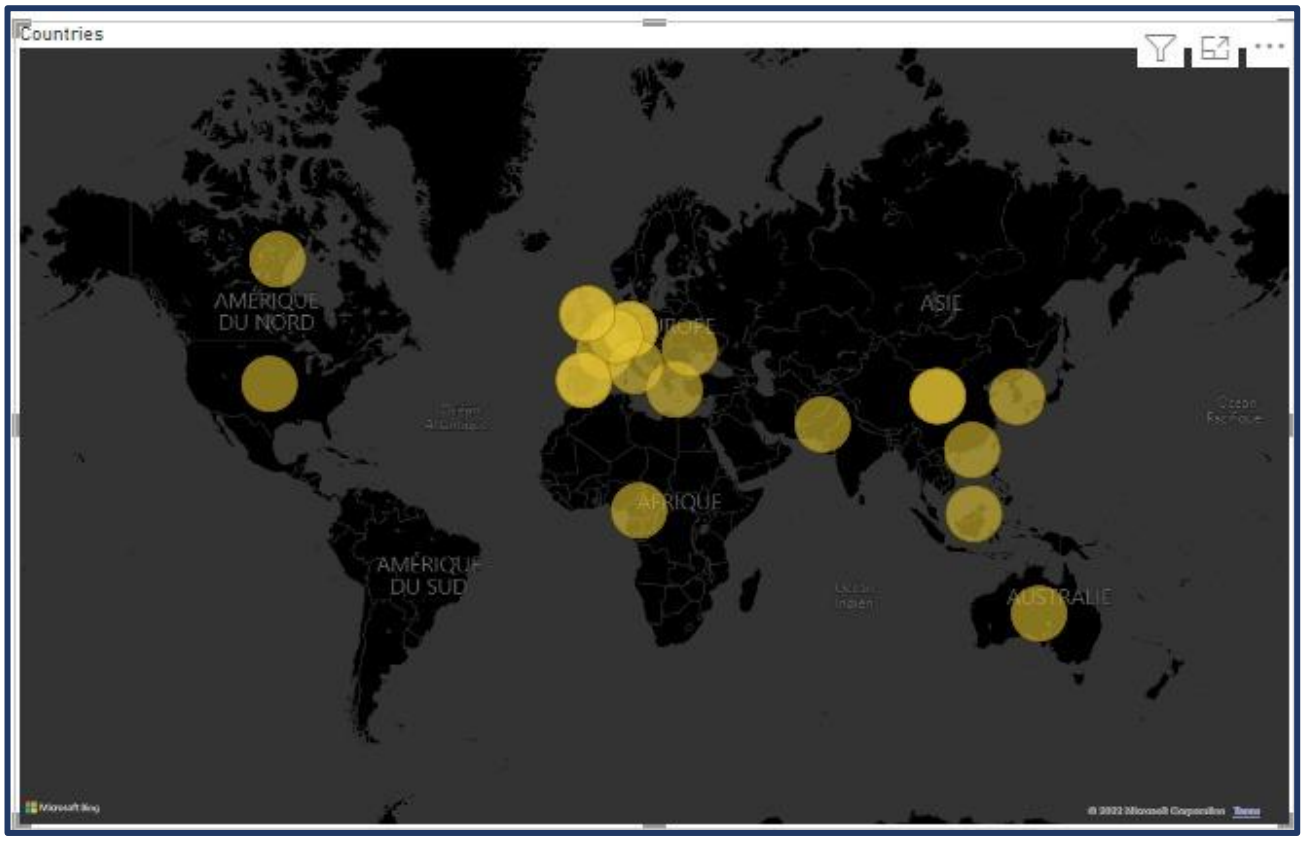
on doit faire appel à toutes les autres dimensions en faisant un lookup.

Analyse BI et Reporting avec Power BI :

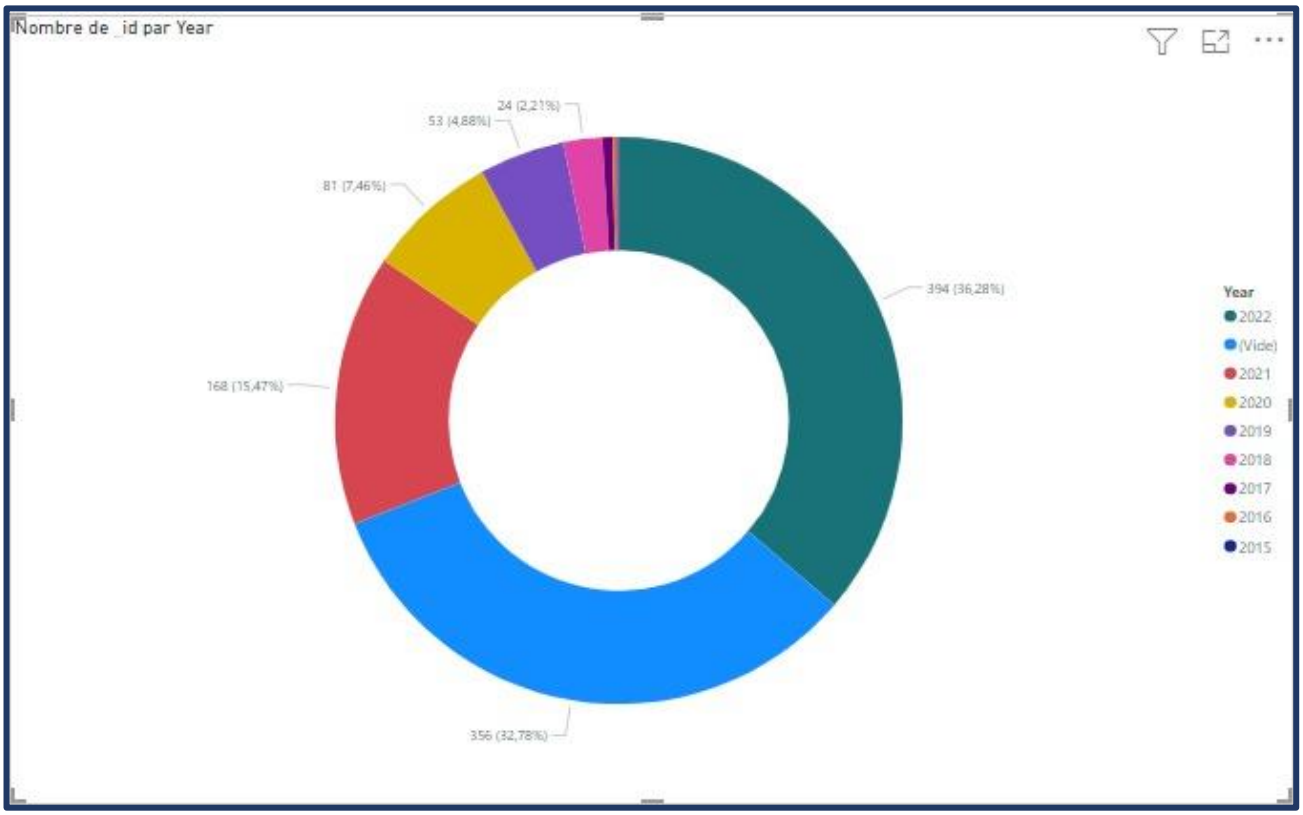
- Le nombre d'article par mois



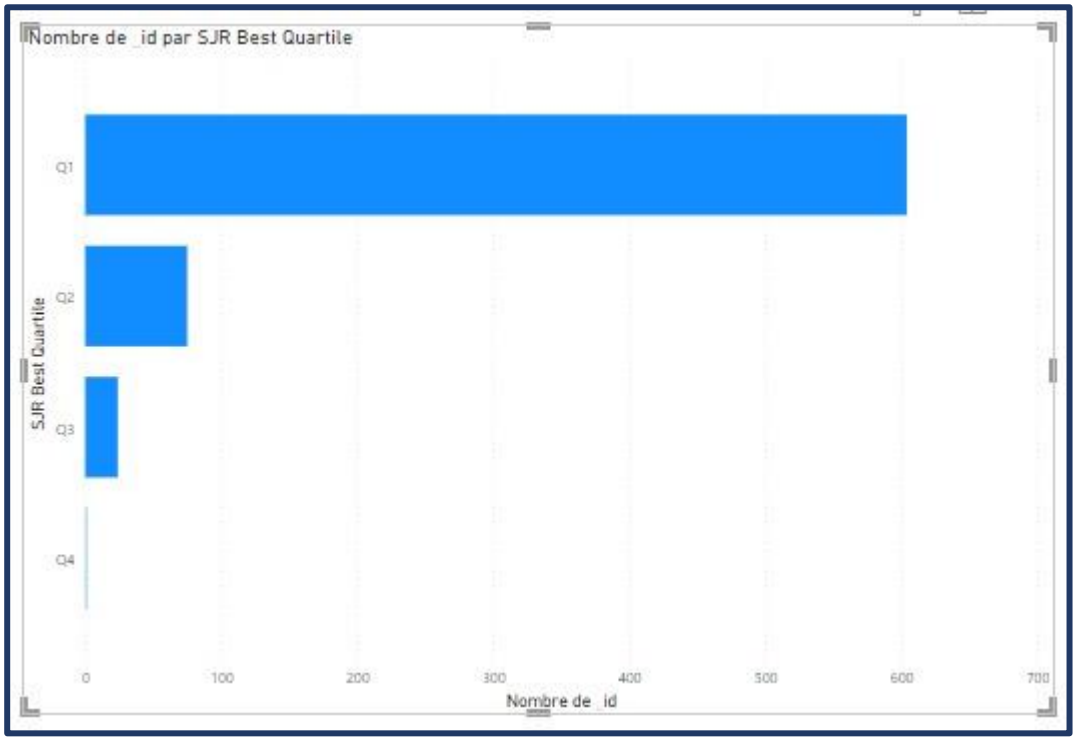
- Les pays où se localise la production scientifique



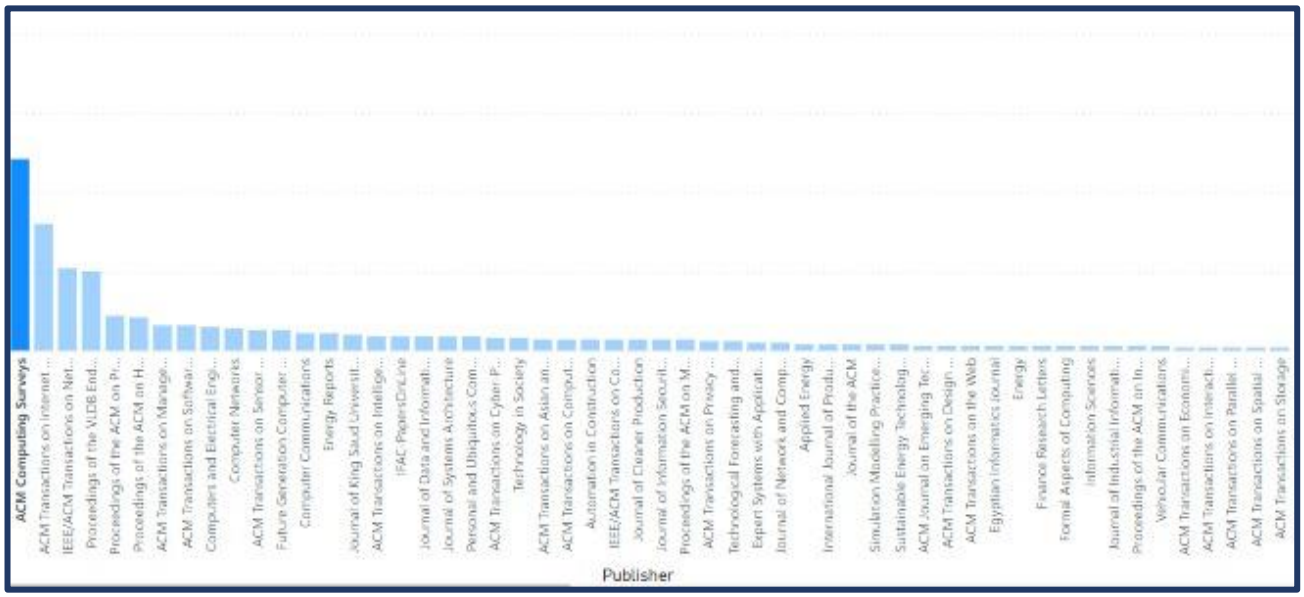
- Le nombre des articles par année

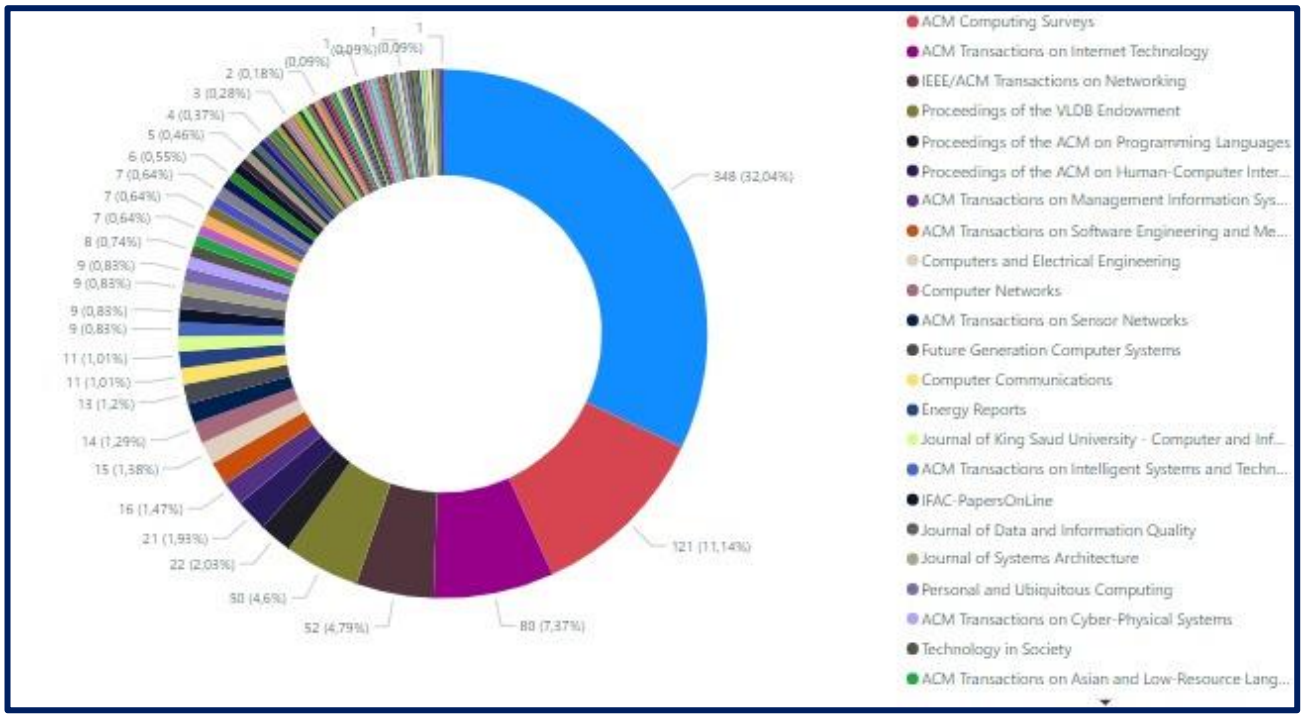


- Le nombre des articles par quartile



- Le nombre de publication par journaux

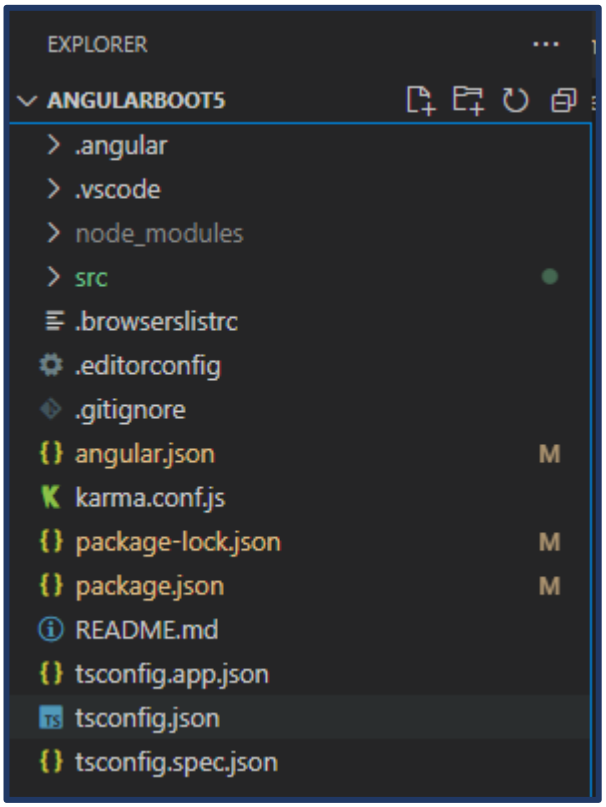


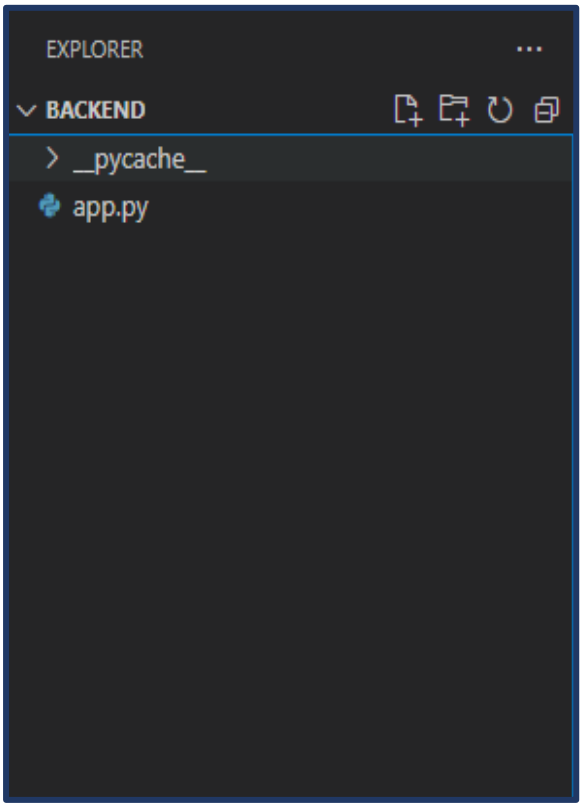


Analyse Big Data

Dans cette partie on va effectuer une analye big data sur l'ensemble des données collectées et stockées dans la base de données MongoDB.

L'architecture du projet





Connexion entre flask et mongoDB :

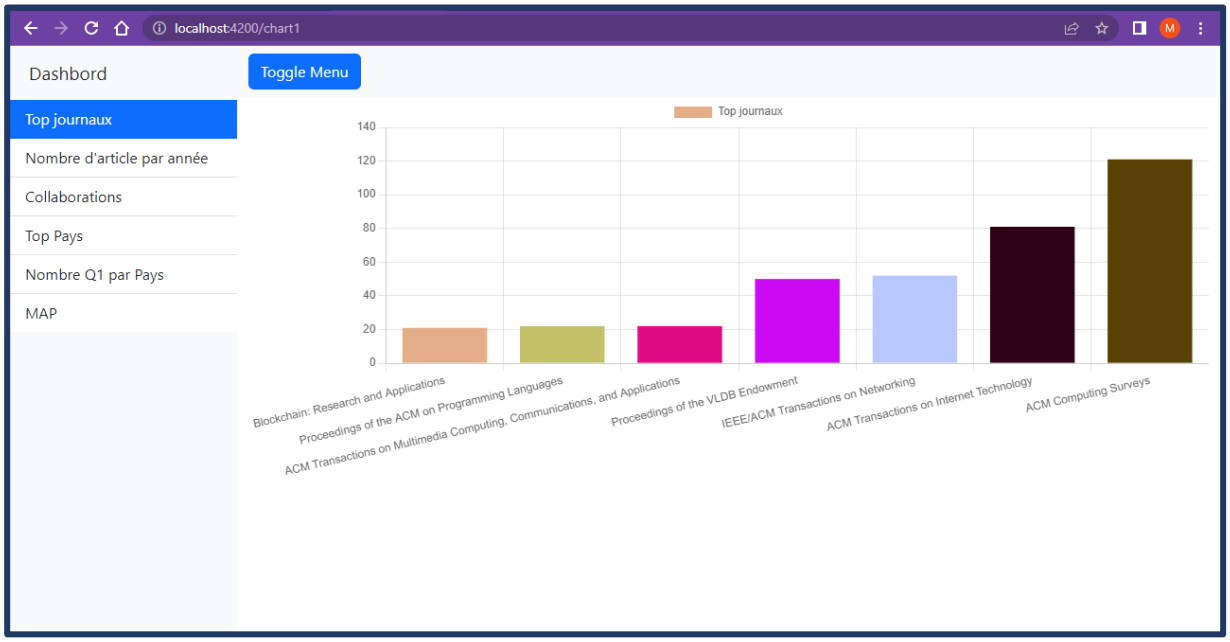
Pour connecter pyspark avec mongodb et manipuler les données stockées dans les collections de notre base des données , on a besoin d'ouvrir une session spark .

```
spark = SparkSession.builder.appName("myApp") \
.config("spark.mongodb.input.uri",
"mongodb://127.0.0.1/bigdata.items") \
.config("spark.mongodb.output.uri",
"mongodb://127.0.0.1/bigdata.items") \
.config('spark.jars.packages',
'org.mongodb.spark:mongo-spark-connector_2.12:2.4.2') \
.getOrCreate()
```

Avec **Spark.mongodb.input.uri** spécifie l'adresse du serveur MongoDB (127.0.0.1), la base de données à connecter (Bigdata) et la collection (items) à partir de laquelle lire les données.

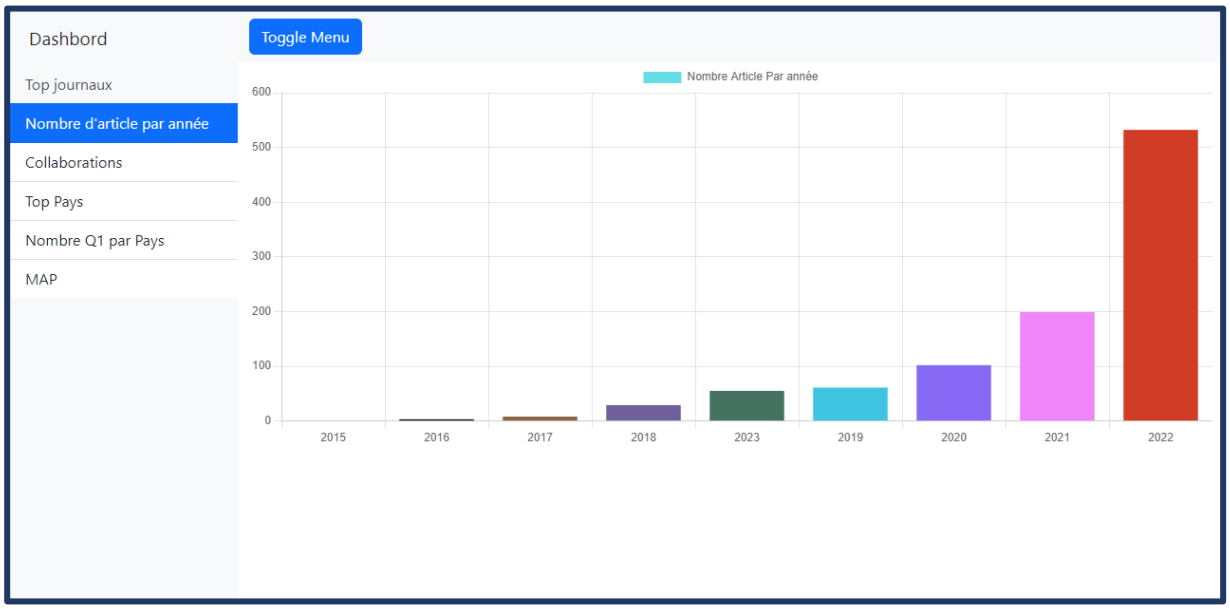
Analyse des données :

- Le nombre de publication par journaux



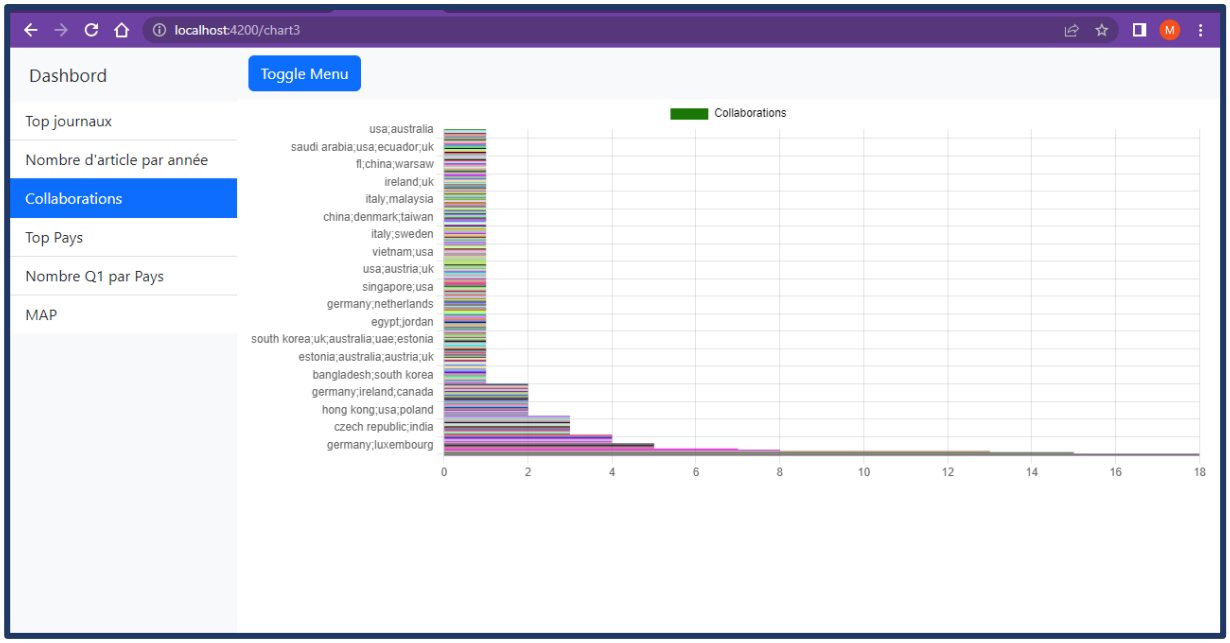
On peut voir que des journaux comme ACM Computing Surveys et ACM Transactions on Internet Technology ont beaucoup de contributions par rapport aux autres.

- Le nombre d'article par année



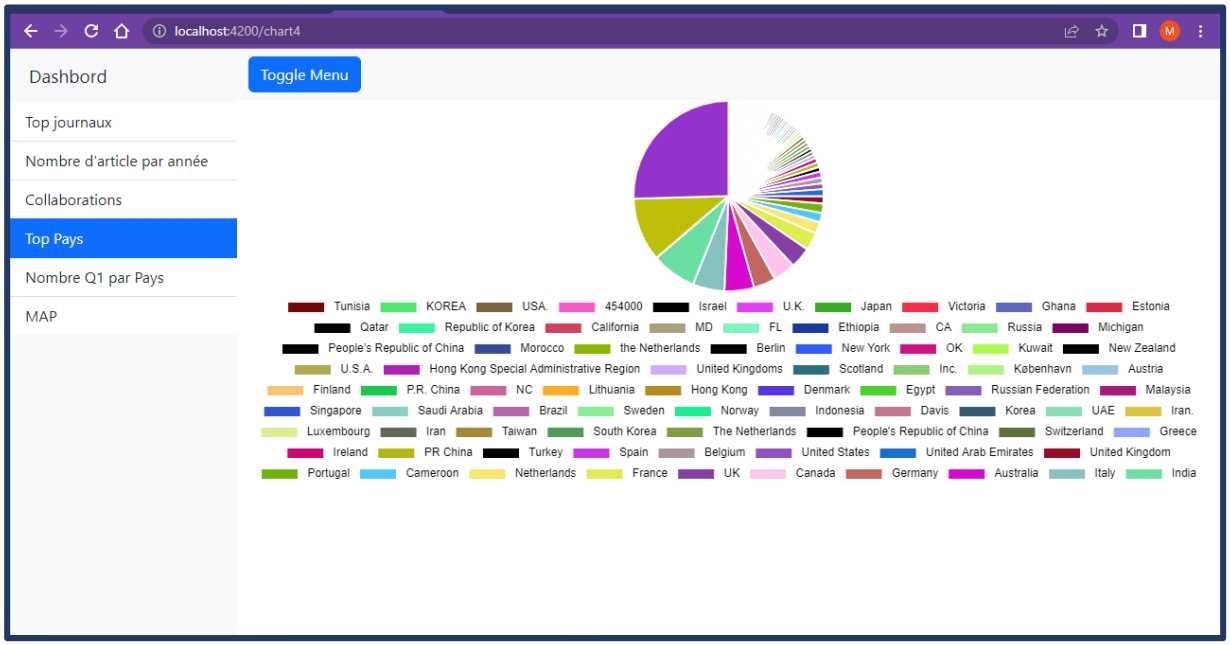
On peut avoir que le nombre de publications dans le domaine de Blochain augmente avec le temps ce qui montre que c'est un sujet d'actualité.

- Les collaborations entre les pays



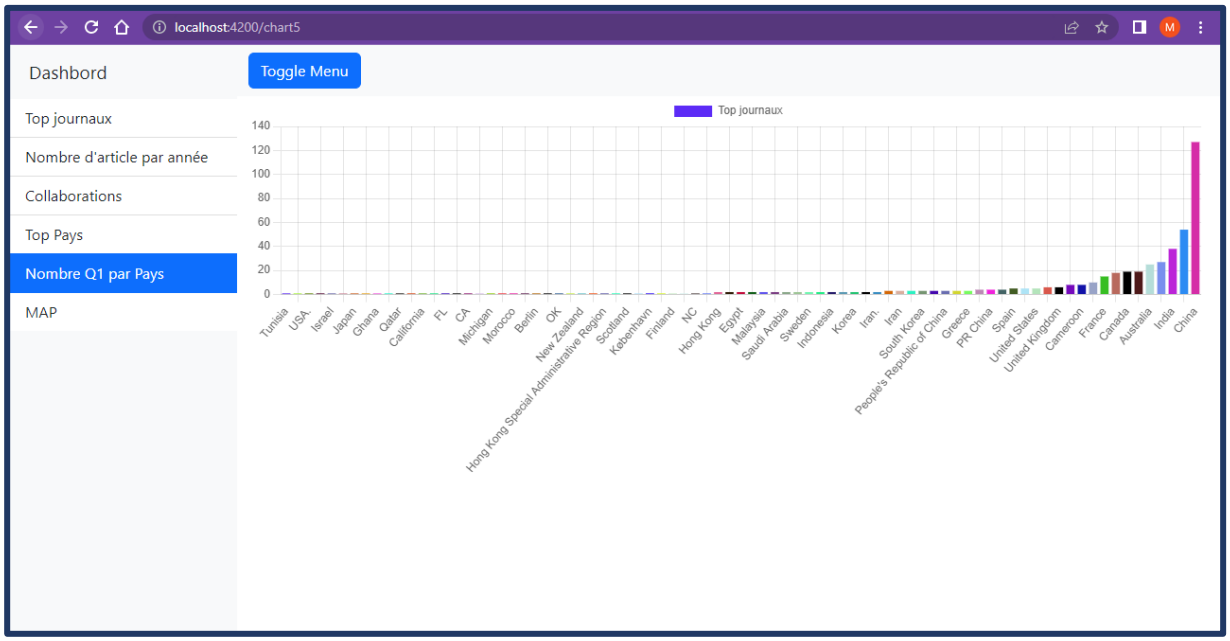
On trouve d'après ce graphe que on a des collaborations très fréquents comme la chine avec USA aussi l'australie avec united kingdom.

- Le nombre de publication par pays



Dans ce graphe on peut voir clairement une différence majeure entre les grands pays comme la chine,USA ,india et l'allemagne dans le domaine de la recherche cela revient aux grands investissements dans la recherche scientifique.

- Analyse des Q1 par pays



Dans ce graphe on peut voir clairement que les grands pays comme la chine,USA ,india et l'allemagne ont des bonnes qualités dans la recherche scientifique.

Les liens vers le projet

Partie de scrapping :

https://github.com/manal-zbakh/BigData-Bi.git

Partie d'analyse Bigdata :

- FrontEnd

https://github.com/barrou-mouad/bi-frontend

- BackEnd

https://github.com/barrou-mouad/bi-backend

CONCLUSION

Le rôle de la Business Intelligence (BI) et du Big Data dans la recherche scientifique consiste à fournir aux chercheurs des outils et des techniques pour collecter, stocker, traiter et analyser des données afin de mieux comprendre les phénomènes étudiés. La BI et le Big Data peuvent être utilisés dans de nombreux domaines de la recherche scientifique, tels que la biologie, la physique, l'astronomie, l'économie et la psychologie.

La BI et le Big Data peuvent aider les chercheurs à mieux comprendre les données qu'ils collectent et à en tirer des conclusions plus précises. Par exemple, en utilisant des outils de BI, les chercheurs peuvent visualiser et analyser des données complexes pour en dégager des tendances et des modèles. De même, en utilisant des outils de Big Data, les chercheurs peuvent traiter de grandes quantités de données rapidement et de manière efficace, ce qui leur permet d'obtenir des résultats plus précis et de mieux comprendre les phénomènes qu'ils étudient.

En résumé, la BI et le Big Data jouent un rôle crucial dans l'aide à la recherche scientifique en fournissant aux chercheurs des outils et des techniques pour collecter, stocker, traiter et analyser des données de manière efficace et précise, ce qui leur permet de mieux comprendre les phénomènes qu'ils étudient.

Références

https://stackoverflow.com/questions/67801494/scrapy-is-not-recognized-as-an-internal-or-external-command-after-installation

https://www.simplified.guide/scrapy/spider-create

https://docs.scrapy.org/en/latest/intro/tutorial.html

https://dl.acm.org/action/doSearch?AllField=cloud+computing&startPage=0&pageSize=100

https://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=Blockchain&highlight=true&returnFacets=ALL&returnType=SEARCH&matchPubs=true&pageNumber=11

https://www.sciencedirect.com/search?qs=Blockchain&show=100&offset=100

https://www.magnitude.com/blog/connect-mongodb-tableau#download

https://www.youtube.com/watch?v=ujyEgHJ2blc