Shape, square

Description automatically generated

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CNTT&TT**

-----🙞🙜🕮🙞🙜-----

**A picture containing text, sign

Description automatically generated**

**BÁO CÁO ASSIGNMENT 2**

**Tìm hiểu về 4 thư viện mã nguồn mở về ETL trên Github/Gitlab**

***Giảng viên hướng dẫn: TS. Đỗ Bá Lâm***

***Sinh viên thực hiện***

***Bùi Đức Đăng MSSV : 20200147***

Contents

[I.Thư viện BeautifulSoup 3](#_Toc164815702)

[1. Giới thiệu 3](#_Toc164815703)

[2. Cài đặt 3](#_Toc164815704)

[3. Cách sử dụng 3](#_Toc164815705)

[3.1. Import thư viện 3](#_Toc164815706)

[3.2. Tạo một đối tượng BeautifulSoup 3](#_Toc164815707)

[3.3. Phân tích HTML 4](#_Toc164815708)

[3.4. Các thuộc tính và phương thức phổ biến 4](#_Toc164815709)

[4. Kết luận 6](#_Toc164815710)

[II. Thư viện Pandas trong Python 6](#_Toc164815711)

[1.Giới thiệu về Pandas 6](#_Toc164815712)

[2.Các tính năng chính của Pandas 6](#_Toc164815713)

[3.Cách sử dụng Pandas 7](#_Toc164815714)

[3.1. Import thư viện 7](#_Toc164815715)

[3.2. Tạo và xử lý dữ liệu 7](#_Toc164815716)

[4. Các thuộc tính và phương thức phổ biến của pandas dùng trong ETL 10](#_Toc164815717)

[4.1. Các thuộc tính phổ biến 10](#_Toc164815718)

[4.2. Các phương thức phổ biến 10](#_Toc164815719)

[5.Kết luận 11](#_Toc164815720)

[Ⅲ.Spark 11](#_Toc164815721)

[1. Giới thiệu 11](#_Toc164815722)

[2. PySpark - Tính năng và ưu điểm 11](#_Toc164815723)

[3.Cách cài đặt và sử dụng 12](#_Toc164815724)

[3.1. Cài đặt Pyspark bằng pip 12](#_Toc164815725)

[3.2. Cách sử dụng 12](#_Toc164815726)

[4.Một số thuộc tính , phương thức phổ biến 14](#_Toc164815727)

[4.1. Một số thuộc tính 14](#_Toc164815728)

[4.2. Một số phương thức phổ biến 14](#_Toc164815729)

[5. Ứng dụng 15](#_Toc164815730)

[5.1. Phân Tích Dữ Liệu 15](#_Toc164815731)

[5.2. Machine Learning 16](#_Toc164815732)

[5.3. Xử Lý Dữ Liệu Lớn 16](#_Toc164815733)

[6. Nhược Điểm và Thách Thức 16](#_Toc164815734)

[6.1. Nhược Điểm 16](#_Toc164815735)

[6.2. Thách thức 16](#_Toc164815736)

[6.3. Giải pháp 16](#_Toc164815737)

[7.Kết luận 17](#_Toc164815738)

[Ⅳ.Airflow 17](#_Toc164815739)

[1.Giới thiệu về Airflow 17](#_Toc164815740)

[2.Kiến trúc 18](#_Toc164815741)

[3.Cài đặt Airflow bằng Docker 19](#_Toc164815742)

[3.1.Tải file docker-compose yaml của airflow 19](#_Toc164815743)

[3.2. Khởi tạo môi trường 19](#_Toc164815744)

[3.3. Chạy Airflow 20](#_Toc164815745)

[4.DAG trong Airflow 21](#_Toc164815746)

[4.1.Khái niệm 21](#_Toc164815747)

[4.2.Khai báo DAG 22](#_Toc164815748)

[4.3 Xác định thứ tự xử lý của các tasks 23](#_Toc164815749)

[5. Chạy file DAG 24](#_Toc164815750)

[6.Kết luận 24](#_Toc164815751)

I.Thư viện BeautifulSoup

1. Giới thiệu

Thư viện BeautifulSoup là một thư viện Python mạnh mẽ được sử dụng để phân tích và trích xuất dữ liệu từ các trang web. Nó cho phép các nhà phát triển dễ dàng điều hướng và tìm kiếm các thành phần HTML và XML của một trang web và trích xuất thông tin một cách linh hoạt.

## 2. Cài đặt

Để sử dụng BeautifulSoup, ta có thể cài đặt nó thông qua pip, một trình quản lý gói Python:

*pip install beautifulsoup4*

## 3. Cách sử dụng

### 3.1. Import thư viện

*from bs4 import BeautifulSoup*

### 3.2. Tạo một đối tượng BeautifulSoup

Ta có thể tạo một đối tượng BeautifulSoup từ một chuỗi HTML hoặc từ một tệp HTML.

**Tạo từ một chuỗi HTML:**

*html\_doc = "<html><head><title>Example</title></head><body><p>Hello, BeautifulSoup!</p></body></html>"*

*soup = BeautifulSoup(html\_doc, 'html.parser')*

**Tạo từ một tệp HTML:**

*with open("example.html") as file:*

*soup = BeautifulSoup(file, 'html.parser')*

### 3.3. Phân tích HTML

Sau khi có đối tượng BeautifulSoup, bạn có thể thực hiện các hoạt động phân tích trên nó, như tìm kiếm các thẻ, thuộc tính hoặc nội dung.

**Tìm kiếm các thẻ:**

*soup.find('p') # Tìm thẻ đầu tiên <p> trong tài liệu*

*soup.find\_all('a') # Tìm tất cả các thẻ <a> trong tài liệu*

**Trích xuất nội dung của thẻ:**

*paragraph = soup.find('p')*

*print(paragraph.text) # In ra nội dung của thẻ <p>*

### 3.4. Các thuộc tính và phương thức phổ biến

#### 3.4.1. Thuộc tính

**text:** Thuộc tính này được sử dụng để trả về nội dung văn bản của thẻ.

*VD :*

*paragraph = soup.find('p')*

*print(paragraph.text) # In ra nội dung của thẻ <p>*

**parent:** Thuộc tính này trả về thẻ cha của phần tử hiện tại.

*VD :*

*paragraph = soup.find('p')*

*parent\_tag = paragraph.parent # Trả về thẻ cha của thẻ <p>*

**parents:** Thuộc tính này trả về một generator cho tất cả các thẻ cha của phần tử hiện tại

*VD:*

*paragraph = soup.find('p')*

*for parent\_tag in paragraph.parents: # Duyệt qua tất cả các thẻ cha của thẻ <p>*

*print(parent\_tag.name)*

**string:** Thuộc tính này trả về nội dung văn bản của thẻ, không bao gồm các thẻ con.*VD:*

*paragraph = soup.find('p')*

*paragraph\_text = paragraph.string # Trả về nội dung văn bản của thẻ <p>*

#### 3.4.2. Phương thức

**find(tag):** Phương thức này được sử dụng để tìm và trả về thẻ đầu tiên có tên là tag. Nếu không tìm thấy, nó sẽ trả về None.

*VD: paragraph = soup.find('p') # Tìm thẻ đầu tiên <p> trong tài liệu*

**find\_all(tag):** Phương thức này được sử dụng để tìm và trả về tất cả các thẻ có tên là tag dưới dạng danh sách. Nếu không tìm thấy, nó sẽ trả về danh sách rỗng.

*VD : paragraphs = soup.find\_all('p') # Tìm tất cả các thẻ <p> trong tài liệu*

**select(selector):** Phương thức này được sử dụng để chọn các phần tử dựa trên CSS selector và trả về danh sách các phần tử phù hợp.

*VD : links = soup.select('a.link') # Chọn tất cả các thẻ <a> có class="link"*

**find\_parent(name, attrs):** Phương thức này tìm và trả về thẻ cha đầu tiên của phần tử hiện tại mà có tên là name và các thuộc tính được chỉ định trong attrs.

*VD:*

*paragraph = soup.find('p')*

*parent\_div = paragraph.find\_parent('div', class\_='content') # Tìm thẻ cha đầu tiên <div class="content">*

**find\_next\_sibling(name, attrs):** Phương thức này tìm và trả về thẻ kế tiếp của phần tử hiện tại mà có tên là name và các thuộc tính được chỉ định trong attrs

*VD :*

*paragraph = soup.find('p')*

*next\_paragraph = paragraph.find\_next\_sibling('p') # Tìm thẻ p kế tiếp*

**find\_previous\_sibling(name, attrs):** Phương thức này tìm và trả về thẻ trước đó của phần tử hiện tại mà có tên là name và các thuộc tính được chỉ định trong attrs.

VD :

*paragraph = soup.find('p')*

*next\_paragraph = paragraph.find\_previous\_sibling('p') # Tìm thẻ p trước đó*

**prettify():** Phương thức này trả về một chuỗi được hiển thị dễ đọc hơn với các thẻ và văn bản được căn chỉnh đúng cách.

*VD : print(soup.prettify()) # In ra tài liệu được căn chỉnh đẹp hơn*

## 4. Kết luận

Thư viện BeautifulSoup là một công cụ hữu ích để phân tích và trích xuất dữ liệu từ các trang web. Nó cung cấp một cách linh hoạt và dễ sử dụng để tìm kiếm và xử lý các thành phần HTML và XML. Sử dụng BeautifulSoup, bạn có thể dễ dàng xây dựng các ứng dụng web scraper và khai thác dữ liệu.

Báo cáo trên cung cấp một cái nhìn tổng quan về thư viện BeautifulSoup trong Python, bao gồm cách cài đặt, sử dụng và các phương thức phổ biến. Bạn có thể bổ sung thêm thông tin hoặc đi sâu vào các tính năng cụ thể của thư viện tùy thuộc vào yêu cầu của bạn.

# II. Thư viện Pandas trong Python

## 1.Giới thiệu về Pandas

Pandas là một thư viện mã nguồn mở trong ngôn ngữ lập trình Python được sử dụng rộng rãi cho xử lý và phân tích dữ liệu. Nó cung cấp các cấu trúc dữ liệu và công cụ xử lý dữ liệu mạnh mẽ, cho phép người dùng thực hiện các tác vụ như trích xuất dữ liệu, biến đổi dữ liệu, và phân tích dữ liệu một cách dễ dàng và hiệu quả.

## 2.Các tính năng chính của Pandas

1. Cấu trúc dữ liệu linh hoạt: Pandas cung cấp hai cấu trúc dữ liệu chính là Series và DataFrame, cho phép lưu trữ và xử lý dữ liệu một cách linh hoạt và dễ dàng.
2. Xử lý dữ liệu thiếu: Pandas cho phép xử lý các giá trị thiếu (NaN) trong dữ liệu một cách thuận tiện thông qua các phương thức như dropna() và fillna().
3. Đọc và ghi dữ liệu: Pandas hỗ trợ đọc và ghi dữ liệu từ nhiều nguồn khác nhau như tệp văn bản, tệp CSV, tệp Excel, cơ sở dữ liệu SQL, và nhiều định dạng dữ liệu khác.
4. Trích xuất và lựa chọn dữ liệu: Pandas cung cấp các phương thức mạnh mẽ để trích xuất và lựa chọn dữ liệu từ DataFrame bằng cách sử dụng các biểu thức điều kiện và chỉ mục.
5. Biến đổi dữ liệu: Pandas cho phép biến đổi dữ liệu bằng cách thêm, xóa, hoặc sửa đổi các cột, thực hiện các phép biến đổi dữ liệu, và áp dụng các hàm tự định nghĩa.
6. Phân tích và thống kê dữ liệu: Pandas cung cấp các công cụ cho phân tích và thống kê dữ liệu như tính toán các chỉ số thống kê mô tả, tính tổng, trung bình, và phương sai, và thực hiện các phân tích dữ liệu phức tạp.

## 3.Cách sử dụng Pandas

### 3.1. Import thư viện

Import thư viện: Để bắt đầu sử dụng Pandas, ta cần import thư viện vào dự án Python của mình bằng lệnh : *import pandas as pd.*

### 3.2. Tạo và xử lý dữ liệu

#### 3.2.1. Series cách tạo và sử dụng

Series là mảng một chiều giống như mảng Numpy, hay như một cột của một bảng, nhưng nó bao gồm thêm một bảng đánh label. Series có thể được khởi tạo thông qua NumPy, kiểu Dict hoặc các dữ liệu vô hướng bình thường

**Cách tạo Series:** Series có thể được tạo bằng cách sử dụng hàm tạo sau pandas.Series( data, index, dtype, copy)

|  |  |
| --- | --- |
|  | **Mô tả** |
| **data** | Có nhiều dạng khác nhau như ndarray (mảng n chiều), list, constants |
| **index** | Các giá trị index phải là duy nhất và có thể là hash có cùng độ dài với dữ liệu Mặc định np.arrange (n) nếu không có chỉ mục nào được thông qua |
| **dtype** | Loại dữ liệu. Nếu không có, kiểu dữ liệu sẽ được suy ra |
| **copy** | Sao chép dữ liệu, mặc định là false |

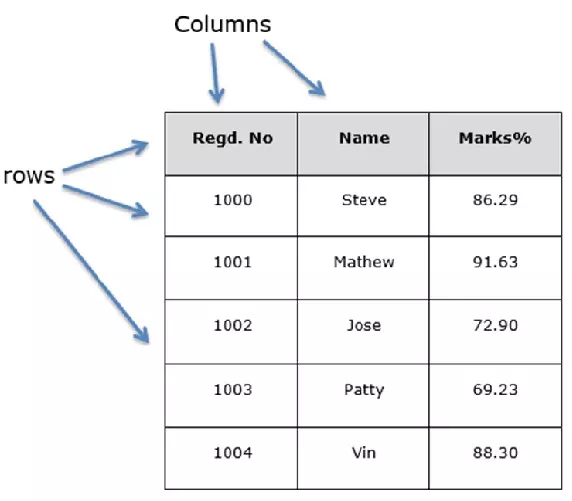
Một Series có thể được tạo bằng các đầu vào khác nhau như: **Array, Dict, Scalar value** hoặc **constant.**

|  |  |  |
| --- | --- | --- |
| **Ví dụ** | **Kết quả** | **Chú thích** |
| import pandas as pd  data = [1, 2, 3, 4, 5]  series = pd.Series(data)  print(series) | 0 1  1 2  2 3  3 4  4 5  dtype: int64 | Khi chúng ta không truyền index, thì mặc định nó sẽ đánh từ 0 đến len(data)-1 |
| import pandas as pd  import numpy as np  data = np.array(['a','b','c','d'])  s = pd.Series(data,index=[100,101,102,103])  print(s) | 100 a  101 b  102 c  103 d  dtype: object | Khi truyền vào index thì sẽ đánh theo index |
| import pandas as pd  import numpy as np  data = {'a' : 0., 'b' : 1., 'c' : 2.}  s = pd.Series(data,index=['b','c','d','a'])  print s | b 1.0  c 2.0  d NaN  a 0.0  dtype: float64 | Thứ tự của index được duy trì và phần tử bị thiếu được lấp đầy bằng NaN (không phải là số) |

#### 3.2.2. DataFrame cách tạo và sử dụng

Một Data frame là một cấu trúc dữ liệu 2 chiều, dữ liệu được sắp xếp theo kiểu bảng trong các hàng và cột có các đặc trưng sau:

* Các cột dữ liệu là các kiểu không đồng nhất: float64, int, bool, …
* Kích thước table có thể thay đổi: các cột có thể thêm hoặc xoá đi
* Các trục được dán nhãn (hàng và cột)
* Có thể thực hiện các phép toán số học trên các hàng và cột



**Cách tạo DataFrame** : Một DataFrame có thể được tạo bằng cách sử dụng hàm tạo sau pandas.DataFrame( data, index, columns, dtype, copy)

|  |  |
| --- | --- |
|  | **Mô tả** |
| **data** | Dữ liệu có nhiều dạng khác nhau như ndarray, series, map, lists, dict, constants hoặc một DataFrame khác |
| **index** | Đối với các row labels, **Index** được sử dụng cho resulting frame là Mặc định tùy chọn np.arrange (n) nếu không có **index** nào được thông qua |
| **columns** | Đối với các column labels. cú pháp mặc định tùy chọn là - np.arrange(n) ,Điều này chỉ đúng nếu không có **index** nào được thông qua |
| **dtype** | Kiểu dữ liệu của từng cột |
| **copy** | Sao chép dữ liệu, mặc định là false |

Một DataFrame có thể được tạo bằng các đầu vào khác nhau như: **Lists, Dict, Series, Numpy ndarrays, DataFrame** khác.

|  |  |  |
| --- | --- | --- |
| Ví dụ | Kết quả | Chú thích |
| import pandas as pd  data = [1,2,3,4,5]  df = pd.DataFrame(data)  print(df) | 0  0 1  1 2  2 3  3 4  4 5 | Tạo DataFrame bằng một list |
| import pandas as pd  data = [['Alex',10],['Bob',12],['Clarke',13]]  df = pd.DataFrame(data,columns=['Name','Age'])  print(df) | Name Age  0 Alex 10  1 Bob 12  2 Clarke 13 | Tạo từ một danh sach các lists |

Tạo một DataFrame từ Dict của ndarrays / Lists

Tất cả các ndarrays phải có cùng độ dài. Nếu index được thông qua, thì độ dài của index sẽ bằng với độ dài của mảng

Nếu không có index nào được truyền, thì theo mặc định, index sẽ là range (n), trong đó n là độ dài mảng.

## 4. Các thuộc tính và phương thức phổ biến của pandas dùng trong ETL

### 4.1. Các thuộc tính phổ biến

**dtypes**: Trả về kiểu dữ liệu của từng cột trong DataFrame.

**shape**: Trả về kích thước của DataFrame dưới dạng (số hàng, số cột).

**columns**: Trả về danh sách các tên cột trong DataFrame.

**index**: Trả về index của DataFrame.

### 4.2. Các phương thức phổ biến

**head(n):** Trả về n hàng đầu tiên của DataFrame.

*VD: df.head(5) # Trả về 5 hàng đầu tiên*

**info():** Hiển thị thông tin tổng quan về DataFrame, bao gồm tên cột, kiểu dữ liệu, và số lượng giá trị khác NULL.

*VD: df.info()*

**describe():** Trả về các thống kê mô tả cho các cột số trong DataFrame, bao gồm số lượng, mean, max, min, và quartiles.

*VD: df.describe()*

**drop(labels, axis):** Loại bỏ hàng hoặc cột từ DataFrame dựa trên nhãn (labels) và trục (axis).

*VD: df.drop(labels='column\_name', axis=1) # Loại bỏ cột*

**fillna(value):** Thay thế các giá trị NULL bằng giá trị được chỉ định

*VD: df.fillna(0) # Thay thế NULL bằng 0*

**drop\_duplicates():** Loại bỏ các hàng trùng lặp từ DataFrame

*VD: df.drop\_duplicates()*

**to\_csv(path\_or\_buf):** Xuất DataFrame thành tệp CSV.

*VD: df.to\_csv('file.csv') #Lưu thành file csv vào file có đường dẫn được truyền vào*

**to\_json(path\_or\_buf):** Xuất DataFrame thành tệp JSON.

*VD* : *df.to\_json('file.json') # #Lưu thành file json vào file có đường dẫn được truyền vào*

**to\_sql(name, con):** Ghi DataFrame vào cơ sở dữ liệu SQL.

*VD: df.to\_sql('table\_name', connection\_object)*

## 5.Kết luận

Pandas là một công cụ mạnh mẽ và linh hoạt cho xử lý và phân tích dữ liệu trong Python. Với các tính năng và công cụ đa dạng, nó đã trở thành một trong những thư viện phổ biến nhất cho xử lý dữ liệu và phân tích dữ liệu trong cộng đồng khoa học dữ liệu và phân tích dữ liệu.

# Ⅲ.Spark

## 1. Giới thiệu

Apache Spark đã trở thành một trong những công cụ quan trọng nhất trong việc xử lý dữ liệu lớn và phân tích dữ liệu trong thời đại Big Data ngày nay. Spark được phát triển bởi Apache Software Foundation và được thiết kế để xử lý công việc phân tán trên các cụm máy tính lớn. PySpark là giao diện Python cho Apache Spark, cho phép các nhà phân tích dữ liệu và nhà phát triển xây dựng các ứng dụng phức tạp trên nền tảng Spark bằng ngôn ngữ lập trình Python.

## 2. PySpark - Tính năng và ưu điểm

PySpark cung cấp một loạt các tính năng và lợi ích đáng chú ý:

* Tích Hợp Dễ Dàng: Với việc sử dụng Python, một trong những ngôn ngữ lập trình phổ biến nhất, PySpark dễ dàng tích hợp vào quy trình làm việc hiện có của các nhà phân tích dữ liệu và nhà phát triển.
* Hiệu Năng Cao: PySpark tận dụng sức mạnh của Apache Spark, cho phép xử lý dữ liệu phân tán nhanh chóng trên cụm máy tính lớn.
* API Phong Phú: PySpark cung cấp một loạt các API để làm việc với dữ liệu lớn, bao gồm DataFrame API (được lấy cảm hứng từ pandas) và RDD API (Resilient Distributed Dataset).
* Tương Thích Đa Dạng: PySpark tương thích với nhiều nguồn dữ liệu khác nhau như HDFS, Hive, JSON, CSV, và nhiều hơn nữa, giúp dễ dàng tiếp cận và xử lý dữ liệu từ nhiều nguồn khác nhau.
* Hỗ Trợ Machine Learning: PySpark cung cấp một thư viện machine learning mạnh mẽ được tích hợp sẵn, cho phép người dùng xây dựng và triển khai các mô hình machine learning trên dữ liệu lớn.

## 3.Cách cài đặt và sử dụng

### 3.1. Cài đặt Pyspark bằng pip

Để chạy pyspark trên một máy cục bộ, chúng ta cần Java và phần mềm khác. Vì vậy, thay vì thủ tục cài đặt nặng nề, ta có thể sử dụng Google Colaboratory có thông số kỹ thuật phần cứng tốt hơn và cũng đi kèm với một loạt các thư viện cho Khoa học dữ liệu và Máy học. Chúng ta cần phải cài đặt pyspark và Py4J gói. Các Py4J cho phép các chương trình Python chạy trong một thông dịch viên python để tự động truy cập các đối tượng Java trong một Java Virtual Machine. Lệnh để cài đặt các gói nói trên như sau :

*!pip install pyspark*

### 3.2. Cách sử dụng

3.2.1. Import các gói cần thiết  
Để bắt đầu sử dụng pyspark trên Google Colab ta cần import các gói cần thiết :

*VD:*

*from pyspark.sql import SparkSession*

#### 3.2.2. Khởi tạo một phiên Spark trên Google Colab

Để tạo một SparkSession, bạn cần sử dụng phương pháp mẫu trình tạo **builder() :**

*VD:*

*# Khởi tạo một phiên Spark trên Google Colab*

*spark = SparkSession.builder.master("local[\*]")\.appName(‘PySpark on Google Colab’).getOrCreate()*

* getOrCreate()- phương thức trả về một SparkSession đã tồn tại; nếu không tồn tại, nó sẽ tạo một SparkSession mới.
* master()- Nếu bạn đang chạy nó trên cụm, bạn cần sử dụng tên chính của mình làm đối số. thông thường, nó sẽ làyarnhoặc mesosphụ thuộc vào thiết lập cụm của bạn và cũng sử dụng local[X]khi chạy ở chế độ Độc lập. Xphải là một giá trị số nguyên và phải lớn hơn 0, đại diện cho số lượng phân vùng mà nó sẽ tạo khi sử dụng RDD, DataFrame và Dataset. Tốt nhất, giá trị Xphải là số lõi CPU.
* appName() phương pháp được sử dụng để đặt tên ứng dụng của bạn.
* getOrCreate() phương thức trả về một SparkSession hiện có nếu nó tồn tại nếu không nó sẽ tạo ra một SparkSession mới.

#### 3.2.3.Tạo DataFrame và thao tác với dữ liệu

Ta có thể tiếp tục sử dụng PySpark giống như làm trên một môi trường Python thông thường. Pyspark có thể đọc dữ liệu từ nhiều định dạng tệp khác nhau như Giá trị được phân tách bằng dấu phẩy (CSV), Ký hiệu đối tượng JavaScript (JSON), Parquet, v.v.

a.Tạo DataFrame

Ta có thể tạo DataFrame bằng cách đọc vào một file csv, json hoặc Parquet bằng cách sử dụng phương thức **read()** như sau :  
*VD:**# Reading CSV file*

*csv\_file = 'data/stocks\_price\_final.csv'*

*df = spark.read.csv(csv\_file)*

*# Reading JSON file*

*json\_file = 'data/stocks\_price\_final.json'*

*data = spark.read.json(json\_file)*

*# Reading parquet file*

*parquet\_file = 'data/stocks\_price\_final.parquet'*

*data1 = spark.read.parquet(parquet\_file)*

##### b. Cấu trúc dữ liệu bằng lược đồ Spark

Cấu trúc dữ liệu bằng lược đồ (Schema) trong Spark được sử dụng để định nghĩa cấu trúc của dữ liệu trong DataFrame. Điều này rất hữu ích trong việc xác định kiểu dữ liệu của từng cột và đảm bảo tính nhất quán và đúng đắn của dữ liệu trong quá trình xử lý.

* Xác định kiểu dữ liệu
* Kiểm soát tính nhất quán của dữ liệu
* Tối ưu hóa hiệu suất
* Tạo động cơ tương thích với các công cụ khác: như Hive, Parquet, Avro, và JDBC
* Tích hợp với Spark SQL

*VD:*

*df = spark.read.csv("Players.csv", header=True, inferSchema=True)  
# Định nghĩa lược đồ*

*schema = StructType([*

*StructField("surname", StringType(), True),*

*StructField("team", StringType(), True),*

*StructField("position", StringType(), True),*

*StructField("minutes", IntegerType(), True),*

*StructField("shots", IntegerType(), True),*

*StructField("passes", IntegerType(), True),*

*StructField("tackles", IntegerType(), True),*

*StructField("saves", IntegerType(), True)*

*])*

##### c.Thực hiện biến đổi trên dữ liệu

Ta có thể tiếp tục sử dụng PySpark giống như làm việc trên một môi trường Python thông thường.

##### d.Kết thúc phiên làm việc

Khi đã hoàn thành công việc của mình, hãy đảm bảo rằng ta đã đóng phiên làm việc PySpark của mình bằng cách gọi phương thức stop() của đối tượng SparkSession:

*# Kết thúc phiên làm việc PySpark*

*spark.stop()*

## 4.Một số thuộc tính , phương thức phổ biến

### 4.1. Một số thuộc tính

**sparkContext**: Thuộc tính này cung cấp quyền truy cập đến SparkContext, là cơ sở của mọi hoạt động Spark.

**catalog**: Thuộc tính này cung cấp quyền truy cập đến dữ liệu đã được đăng ký (registered) trong Catalog của Spark.

**conf**: Thuộc tính này cung cấp quyền truy cập đến cấu hình (configuration) của ứng dụng Spark.

### 4.2. Một số phương thức phổ biến

**createDataFrame(data, schema=None):** Phương thức này tạo một DataFrame từ dữ liệu và lược đồ chỉ định.

*df = spark.createDataFrame(data, schema)*

**read**: Phương thức này được sử dụng để đọc dữ liệu từ các nguồn khác nhau như CSV, JSON, JDBC, Parquet, và Hive.

*df = spark.read.csv("data.csv", header=True, inferSchema=True)*

**head (n)** : Phương thức này trả về n hàng dưới dạng danh sách.

**show ()** : Phương thức này hiển thị 20 hàng đầu tiên theo mặc định và nó cũng lấy một số làm tham số để hiển thị số hàng của dữ liệu.

**description ()** : Phương thức này tính toán thống kê tóm tắt của các cột với kiểu dữ liệu số.

**columns :** Phương thức này trả về một danh sách có chứa tên cột của dữ liệu.

**count () :** Phương thức này trả về số lượng hàng trong dữ liệu.

**printSchema () :** Phương thức này hiển thị lược đồ của dữ liệu

**sql(query):** Phương thức này được sử dụng để thực thi các truy vấn SQL trên DataFrame.

*result\_df = spark.sql("SELECT \* FROM people WHERE Age >= 30")*

**filter() :** Lọc dữ liệu dựa trên điều kiện đã cho, bạn cũng có thể đưa ra nhiều điều kiện bằng cách sử dụng toán tử AND (&), OR (|) và NOT (~).

**between()** : Phương thức này trả về True hoặc False nếu các giá trị được truyền vào giữa phương thức.

**when()** : Phương thức này trả về 0 hoặc 1 tùy thuộc vào điều kiện đã cho

**rlike()** : Phương thức này tương tự như toán tử like trong SQL

**groupBy()** : Bản thân cái tên giải thích rằng nó nhóm dữ liệu theo tên cột nhất định và nó có thể thực hiện các phép toán khác nhau như tổng, trung bình, tối thiểu, tối đa, v.v.

**stop():** Phương thức này được sử dụng để dừng phiên làm việc Spark.

*spark.stop()*

## 5. Ứng dụng

PySpark được sử dụng trong nhiều tình huống thực tế, bao gồm phân tích dữ liệu, machine learning và xử lý dữ liệu lớn. Dưới đây là một số ví dụ cụ thể về việc áp dụng PySpark trong các dự án công nghiệp và nghiên cứu:

### 5.1. Phân Tích Dữ Liệu

Phân tích dữ liệu của doanh nghiệp: Sử dụng PySpark để phân tích dữ liệu từ các nguồn khác nhau như cơ sở dữ liệu, tệp CSV, hoặc dữ liệu streaming để đưa ra thông tin hữu ích cho các quyết định kinh doanh.

Phân tích dữ liệu trực Ttuyến (OLAP): Xây dựng các hệ thống phân tích dữ liệu trực tuyến với PySpark, cho phép truy vấn và phân tích dữ liệu trực tiếp từ cơ sở dữ liệu lớn mà không cần di chuyển dữ liệu.

### 5.2. Machine Learning

Xây dựng mô hình dự đoán: Sử dụng PySpark để xây dựng các mô hình dự đoán với dữ liệu lớn, bao gồm việc tiền xử lý dữ liệu, lựa chọn tính năng, và huấn luyện mô hình.

Phân loại và gom cụm: Áp dụng các thuật toán phân loại và gom cụm trên dữ liệu lớn để tạo ra các nhóm hoặc phân loại dựa trên các đặc điểm của dữ liệu.

### 5.3. Xử Lý Dữ Liệu Lớn

Xử lý dữ liệu dòng: Xử lý dữ liệu dòng liên tục từ các nguồn như log hoặc dữ liệu IoT để phát hiện sự cố, giám sát hoặc phân tích xu hướng.

Xử lý dữ liệu vận tải và Log: Áp dụng PySpark để xử lý dữ liệu vận tải hoặc log từ các hệ thống phân tán, giúp tối ưu hóa tuyến đường, dự báo thời gian đến, hoặc phát hiện gian lận.

## 6. Nhược Điểm và Thách Thức

PySpark cũng đối diện với một số nhược điểm và thách thức khi được sử dụng trong môi trường công nghiệp và nghiên cứu.

### 6.1. Nhược Điểm

**Tốc độ**: So với các công cụ xử lý dữ liệu truyền thống như Hadoop, PySpark có thể chậm hơn trong một số trường hợp do yêu cầu nhiều tài nguyên và quản lý bộ nhớ phức tạp.

**Khó Khăn trong Debugging**: Vì tính phân tán của PySpark, việc debug có thể khó khăn hơn so với việc debug trong môi trường đơn giản.

### 6.2. Thách thức

**Quản lý tài nguyên**: PySpark đòi hỏi một hệ thống tài nguyên phân tán hiệu quả để hoạt động tốt, bao gồm quản lý bộ nhớ, tính toán và lưu trữ.

**Xử lý dữ liệu không đồng nhất**: Khi làm việc với dữ liệu có định dạng và cấu trúc không đồng nhất, việc tiền xử lý dữ liệu trở nên phức tạp và tốn nhiều thời gian.

### 6.3. Giải pháp

**Tối ưu hóa hiệu suất**: Sử dụng các kỹ thuật tối ưu hóa như lập lịch thực thi, lưu trữ bộ nhớ đệm, và tinh chỉnh cấu hình để cải thiện hiệu suất của PySpark.

**Tiêu chuẩn hóa dữ liệu**: Tiêu chuẩn hóa và làm sạch dữ liệu trước khi đưa vào PySpark giúp giảm thiểu thời gian tiền xử lý và tối ưu hóa hiệu suất.

**Quản lý tài nguyên tốt**: Sử dụng các công cụ quản lý tài nguyên như YARN hoặc Kubernetes để quản lý tài nguyên hiệu quả và tối ưu hóa việc triển khai PySpark trên các cụm máy tính.

## 7.Kết luận

Kết luận chung, PySpark là một công cụ mạnh mẽ và linh hoạt cho việc xử lý dữ liệu lớn và phân tích dữ liệu trong môi trường phân tán. Bằng cách kết hợp sức mạnh của Apache Spark với linh hoạt và dễ sử dụng của Python, PySpark cho phép các nhà phân tích dữ liệu và nhà phát triển xây dựng các ứng dụng phức tạp và mô hình machine learning trên dữ liệu lớn một cách hiệu quả.

Tuy nhiên, việc sử dụng PySpark cũng đối diện với một số thách thức như quản lý tài nguyên, hiệu suất và khó khăn trong việc debug. Tuy nhiên, bằng cách tối ưu hóa hiệu suất, tiêu chuẩn hóa dữ liệu và quản lý tài nguyên một cách thông minh, các nhà phân tích dữ liệu có thể vượt qua những thách thức này và tận dụng tối đa sức mạnh của PySpark trong các dự án công nghiệp và nghiên cứu.

# Ⅳ.Airflow

## 1.Giới thiệu về Airflow

Apache Airflow là một công cụ mã nguồn mở được sử dụng để lập lịch, giám sát và tự động hóa các quy trình làm việc phức tạp. Nó cho phép xác định, lên lịch và thực thi các luồng công việc (workflows) một cách linh hoạt và dễ dàng.

Airflow có thể tự động hóa quy trình công việc của bạn bằng DAGs ( Directed Acyclic Graphs ).

Airflow sẽ thực thi các phần công việc của bạn thông qua các worker mà vẫn đảm bảo các dependencies.

Nguyên tắc của Airflow:

* Tính năng động ( Dynamic ) : Airflow pipeline được config bằng code Python, cho phép bạn thay đổi code dễ dàng để tùy biến luồng làm việc của bạn.
* Tính tăng trưởng ( Scalable ) : Ví dụ đơn giản là bạn có thể mở rộng các task về xử lý dữ liệu để tiết kiệm thời gian
* Tính gọn gàng ( Elegant ) : code gọn gàng, ngăn nắp, rõ ràng giúp bạn đọc hiểu code nhanh chóng.
* Tính mở rộng ( Extensible ) : Bạn có thể thêm thắt thư viện, modules, packages, ... phù hợp với môi trường của bạn

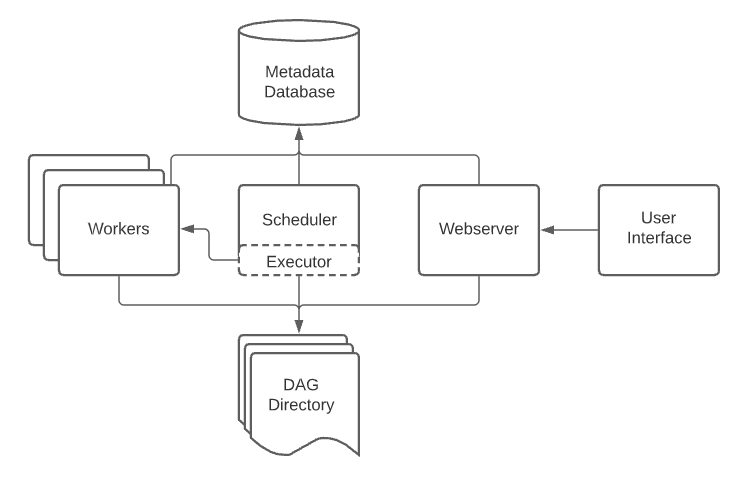
A diagram of a graph

Description automatically generated

## 2.Kiến trúc

Apache Airflow có kiến trúc mô-đun, bao gồm các thành phần chính sau:

* **Scheduler**: Quản lý việc lập lịch và thực thi các công việc theo lịch trình đã xác định.
* **Executor**: Thực hiện các công việc được lên lịch bởi Scheduler trên các nguồn tài nguyên khác nhau như máy cục bộ hoặc các cụm tính toán phân tán.
* **Metadata Database**: Lưu trữ thông tin về các luồng công việc, lịch trình và lịch sử thực thi.
* **Web Interface**: Giao diện người dùng cho phép người dùng xác định, quản lý và giám sát các luồng công việc.
* **CLI (Command Line Interface**): Cung cấp giao diện dòng lệnh để tương tác với Airflow.



## 3.Cài đặt Airflow bằng Docker

### 3.1.Tải file docker-compose yaml của airflow

Yêu cầu cần cài đặt docker compose trước đó.

Sau khi cài đặt xong docker compose, cần tải về file docker-compose yaml của airflow.

Truy cập tới đường link : [*https://airflow.apache.org/docs/apache-airflow/2.3.3/docker-compose.yaml*](https://airflow.apache.org/docs/apache-airflow/2.3.3/docker-compose.yaml) *.* Sau đó copy và lưu lại thành file docker-compose.yml.

Trong file yaml này có chứa các services:

* airflow-scheduler: giám sát các tasks cũng như chạy tasks khi đã có đủ dependencies
* airflow-webserver: webserver có local domain http://localhost:8080
* airflow-worker: các worker chạy các tasks theo lệnh của scheduler
* airflow-init: dịch vụ khởi tạo ban đầu ( tạo account, migrate database, ... )
* postgres: cơ sở dữ liệu
* redis: cầu nối truyền dẫn các lệnh từ scheduler tới worker.

### 3.2. Khởi tạo môi trường

Trước khi chạy Airflow lần đầu, bạn cần phải chuẩn bị môi trường chút: tạo files, folders cần thiết, **khởi tạo cơ sở dữ liệu** :

*mkdir -p ./dags ./logs ./plugins*

*echo -e "AIRFLOW\_UID=$(id -u)" > .env*

* ./dags : nơi chứa files DAG
* ./logs : nơi chứa log của executor và scheduler
* ./plugins : nơi chứa các plugin tùy chỉnh

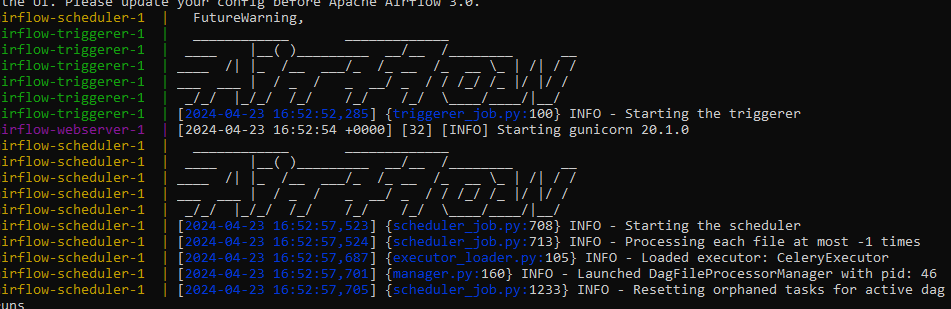
\*Note : trong một số trường hợp có thể tạo file .env tại thư mục chứ file docker-compose.yml và set nội dung trong .env là AIRFLOW\_UID=50000 .

**Khởi tạo cơ sở dữ liệu và tài khoản:***docker-compose up airflow-init*

### 3.3. Chạy Airflow

Sau khi khởi tạo cơ sở dữ liệu cũng như tài khoản airflow, giờ là lúc tôi chạy framework này. Trong folder chứa file docker-compose.yaml của airflow, chạy lệnh

*docker-compose up*

****

Airflow container đã chạy sau đó ta có thể sử dụng các dịch vụ của framework này bằng 3 cách:

* CLI
* Web interface
* RestAPI

**Web Interface**

Sau khi chạy Airflow container sẽ cung cấp một giao diện web. Well, trên trình duyệt truy cập tới http://localhost:8080.

Tài khoản mặc định: airflow.

Mật khẩu mặc định: airflow

Sau khi đăng nhập tài khoản airflow trên web http://localhost:8080 sẽ hiện ra một danh sách các DAG.

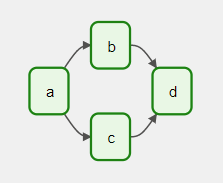
A screenshot of a computer

Description automatically generated

## 4.DAG trong Airflow

### 4.1.Khái niệm

Trong Apache Airflow, một DAG (Directed Acyclic Graph) là một tập hợp các công việc (Tasks) và các quy tắc luồng dữ liệu giữa chúng, mô tả một quy trình làm việc hoặc một luồng công việc. Mỗi nút trong DAG đại diện cho một Task và các cạnh đại diện cho luồng dữ liệu giữa các Task.



Hình bên trên bao gồm 4 task: A, B, C, D có mối liên kết với nhau và chạy theo thứ tự mà người dùng quy định: a -> b, c -> d. Nếu task a chết thì sẽ dẫn đến task b, c, d chết theo. Ta cũng có thể tùy chỉnh thời gian để các tasks này chạy vào thời điểm mong muốn.

### 4.2.Khai báo DAG

Có 3 cách để khai báo DAG trong code:

* Context manager
* Standard constructor
* Decorator

#### 4.2.1. Context manager

Với cách này ta sẽ bọc code của mình trong DAG bằng with

*with DAG(*

*"my\_dag\_name", start\_date=pendulum.datetime(2021, 1, 1, tz="UTC"),*

*schedule\_interval="@daily", catchup=False*

*) as dag:*

*op = EmptyOperator(task\_id="task")*

#### 4.2.2.Standard constructor

Với cách này, ta sẽ khai báo một constructer gọi tới class DAG

*my\_dag = DAG("my\_dag\_name", start\_date=pendulum.datetime(2021, 1, 1, tz="UTC"),*

*schedule\_interval="@daily", catchup=False)*

*op = EmptyOperator(task\_id="task", dag=my\_dag)*

#### 4.2.3.Decorator

*@dag(start\_date=pendulum.datetime(2021, 1, 1, tz="UTC"),*

*schedule\_interval="@daily", catchup=False)*

*def generate\_dag():*

*op = EmptyOperator(task\_id="task")*

*dag = generate\_dag()*

### 4.3 Xác định thứ tự xử lý của các tasks

Xác định thứ tự xử lý của các tasks ( Task Dependencies ): 1 task không phải lúc nào cũng hoạt động 1 mình, sẽ có các tasks phụ thuộc vào nó hoặc nó phụ thuộc vào. Vì vậy, một nhiệm vụ quan trọng của DAG là tạo mối quan hệ của các tasks.

**Cách 1: dùng toán tử << và >>**

*first\_task >> [second\_task, third\_task]*

*third\_task << fourth\_task*

**Cách 2: dùng hàm set\_downstream và set\_upstream**

*first\_task.set\_downstream(second\_task, third\_task)*

*third\_task.set\_upstream(fourth\_task)*

**Trường hợp các tasks giao nhau**

*VD:*

*from airflow.models.baseoperator import cross\_downstream*

*# Replaces*

*# [op1, op2] >> op3*

*# [op1, op2] >> op4*

*cross\_downstream([op1, op2], [op3, op4])*

**Trường hợp các tasks tạo thành dây chuyền**

*VD:*

*from airflow.models.baseoperator import chain*

*#1*

*# op1 >> op2 >> op3 >> op4*

*chain(op1, op2, op3, op4)*

*#2*

*# op1 >> op2 >> op3 >> op4 >> op5 >> op6*

*chain(\*[EmptyOperator(task\_id='op' + i) for i in range(1, 6)])*

*#3*

*# op1 >> op2 >> op4 >> op6*

*# op1 >> op3 >> op5 >> op6*

*chain(op1, [op2, op3], [op4, op5], op6)*

## 5. Chạy file DAG

Trong phần khởi tạo môi trường thư mục dags được tạo để lưu những file python sẽ chạy.

Khi đặt file python bọc bởi DAG trong folder, file này sẽ được copy vào container ( /opt/airflow/dags/ ), Airflow sẽ kiểm tra và chạy các file này.

Tạo 1 file DAG mẫu

Chuyển file DAG này vào thư mục dags/ mà Airflow container đã mount.

Refresh lại WebUI để cập nhật trạng thái mới nhất. Nếu bạn kiểm tra airflow database sẽ thấy thông tin file DAG đã được lưu lại.

## 6.Kết luận

Airflow là một công cụ quản lý quy trình làm việc mạnh mẽ và linh hoạt. Với khả năng xác định và quản lý các phụ thuộc giữa các công việc trong các DAGs, Airflow giúp tự động hóa và quản lý các quy trình làm việc phức tạp một cách dễ dàng. Giao diện người dùng web hiện đại và tính năng an toàn như Xác thực và Phân quyền cung cấp một môi trường quản lý lịch trình công việc an toàn và tiện lợi. Airflow hỗ trợ nhiều loại công việc và kết nối dữ liệu, giúp tích hợp và mở rộng dễ dàng. Với cộng đồng lớn và tính linh hoạt, Airflow là một giải pháp phù hợp cho việc tự động hóa và quản lý quy trình làm việc trong môi trường sản xuất.