
KỸ THUẬT XỬ LÝ DỮ LIỆU LỚN

Phân tích dữ liệu sân bay



NGÀNH KHOA HỌC DỮ LIỆU
KHOA TOÁN - TIN HỌC
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

Giảng viên hướng dẫn: PGS. TS. Nguyễn Thanh Bình

Nhóm học viên:

MSHV: 23C01036 - Nguyễn Lê Thành Phước

MSHV: 23C01041 - Lê Thị Mai Thảo

MSHV: 23C01042 - Vũ Thi Thi

Ngày 18 tháng 11 năm 2024

Mục lục

1	Giới thiệu	1
1.1	Đặt vấn đề	1
1.2	Mô tả Dataset	2
2	Cơ sở lý thuyết	3
2.1	Crawling	3
2.1.1	Các thành phần chính của quá trình Crawl Data	3
2.1.2	Phương pháp Crawl Data	5
2.1.3	Công cụ và công nghệ sử dụng trong Crawl Data	8
2.2	SQL Server	12
2.2.1	Cơ sở dữ liệu quan hệ (Relational Database)	12
2.2.2	Mối quan hệ giữa các bảng trong cơ sở dữ liệu	14
2.2.3	Ngôn ngữ SQL	14
2.2.4	Cấu trúc của SQL Server	15
2.3	Task Scheduler	17
2.3.1	Tự động hóa tác vụ với Task Scheduler	17
2.3.2	Ghi lại sự kiện trong Task Scheduler	19
2.3.3	Tùy chỉnh điều kiện thực hiện trong Task Scheduler	20
2.4	Tableau	21
2.4.1	Các thành phần chính của Tableau:	22
2.4.2	Nguyên lý hoạt động của Tableau	22
2.4.3	Khả năng trực quan hóa của Tableau	23
2.4.4	Các khái niệm quan trọng trong Tableau	23
2.5	Streamlit	24
2.5.1	Giới thiệu lý thuyết về Streamlit	24
2.5.2	Cách sử dụng Streamlit	24
2.5.3	Ứng dụng của Streamlit	25
3	Qui trình	27
3.1	Thu thập dữ liệu	27
3.2	Tự động hoá qui trình với Task Scheduler	31
3.3	Trực quan hóa dữ liệu cho dữ liệu 50 sân bay bận rộn nhất	34

3.4	Dự đoán số lượng khách ở tất cả sân bay của Việt Nam	41
3.4.1	Thu thập dữ liệu	41
3.4.2	Dự đoán	47
3.5	Ứng dụng Web-App: Top 50 Busiest Airports	50
3.5.1	Cấu trúc ứng dụng	50
3.5.2	Tính năng chính	50
3.5.3	Mã nguồn	51
3.5.4	Kết quả nhận được	54
3.5.5	Nhận xét	57

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Trong bối cảnh kinh tế và du lịch toàn cầu ngày càng phát triển, việc nâng cao năng lực và sức hút của các sân bay quốc gia trở thành yếu tố quan trọng trong chiến lược phát triển hạ tầng giao thông và dịch vụ hàng không của Việt Nam. Trong năm 2020, 2022, 2023, Việt Nam liên tục nằm trong top 50 sân bay có số lượng khách đông nhất thế giới, dù Việt Nam vẫn có một khoảng cách đáng kể về lưu lượng hành khách, quy mô, và chất lượng dịch vụ. Do vậy, để hiểu rõ vị thế của Việt Nam trên bản đồ hàng không quốc tế và định hướng xây dựng chiến lược phát triển - nâng cao năng lực cạnh tranh của các sân bay trong nước; **báo cáo này sẽ tiến hành xây dựng hệ thống tự động cập nhật và phân tích dữ liệu top 50 sân bay đông đúc nhất năm 2023 được tổng hợp từ Hội đồng Sân bay Quốc tế (ACI).**

Bên cạnh đó, dự đoán xu hướng tăng giảm lượng khách tại các sân bay nội địa trong tương lai cũng là một mục tiêu quan trọng. Bằng cách phân tích và dự báo lưu lượng hành khách, chúng ta có thể xác định các nhu cầu, cơ hội phát triển, và tối ưu hóa kế hoạch đầu tư hạ tầng, nguồn nhân lực ở sân bay. Do vậy, bên cạnh phân tích dữ liệu top 50 sân bay đông đúc nhất năm 2023, **báo cáo này cũng sẽ dự đoán xu hướng và số lượng hành khách của tất cả sân bay ở Việt Nam vào năm 2024.**

1.2 Mô tả Dataset

Dataset được sử dụng là dữ liệu về 50 sân bay bận rộn nhất thế giới dựa trên số liệu từ báo cáo năm 2023 của Cảng vụ New York và New Jersey:

- **Tên Dataset:** Báo cáo số lượng hành khách tại các sân bay bận rộn nhất thế giới năm 2023.
- **Nguồn Dữ liệu:** Cảng vụ New York và New Jersey.
- **Số lượng quan sát:** 50 sân bay.
- **Các biến trong Dataset:**
 - **Rank:** Xếp hạng của sân bay dựa trên tổng số hành khách.
 - **Airport:** Tên đầy đủ của sân bay.
 - **Location:** Vị trí địa lý (thành phố, tiểu bang).
 - **Country:** Quốc gia nơi sân bay tọa lạc.
 - **Code (IATA/ICAO):** Mã sân bay theo tiêu chuẩn IATA/ICAO.
 - **Total passengers:** Tổng số hành khách đã sử dụng sân bay trong năm 2023.
 - **Rank change:** Thay đổi xếp hạng so với năm trước (tăng/giảm/ổn định).
 - **% change:** Tỷ lệ phần trăm thay đổi tổng số hành khách so với năm trước.
- **Mô tả dữ liệu:**
 - Dữ liệu cung cấp thông tin chi tiết về số lượng hành khách tại 50 sân bay bận rộn nhất trên thế giới, cho phép phân tích và so sánh hiệu suất của các sân bay theo từng quốc gia và khu vực.
 - Thông qua tỷ lệ phần trăm thay đổi và thay đổi xếp hạng, để đánh giá xu hướng tăng trưởng hoặc suy giảm của các sân bay trong bối cảnh toàn cầu.
 - Dữ liệu cũng có thể được sử dụng để nghiên cứu tác động của các yếu tố kinh tế, chính trị và xã hội đến ngành hàng không.

Chương 2

Cơ sở lý thuyết

2.1 Crawling

Crawl data, hay còn gọi là thu thập dữ liệu tự động, là quá trình sử dụng các chương trình hoặc công cụ tự động (gọi là web crawlers hoặc bots) để truy cập và thu thập dữ liệu từ các trang web. Các công cụ này có thể duyệt qua các trang web, lấy thông tin cần thiết và lưu trữ lại để phục vụ cho các mục đích phân tích, tìm kiếm, hoặc xây dựng cơ sở dữ liệu.

2.1.1 Các thành phần chính của quá trình Crawl Data

Web Crawler (Spider/Bot):

Web Crawler, còn gọi là spider hoặc bot, là một chương trình tự động có nhiệm vụ duyệt qua các trang web để thu thập dữ liệu. Nó hoạt động bằng cách bắt đầu từ một hoặc nhiều URL khởi điểm (seed URLs), sau đó truy cập vào các liên kết có trên trang đó. Khi một trang được truy cập, crawler tiếp tục duyệt qua các liên kết nội bộ (liên kết trong cùng một miền) và có thể cả liên kết bên ngoài (liên kết đến các miền khác).

Crawler làm việc tuần tự hoặc song song, truy cập từng URL, tải nội dung trang về, và lưu trữ dữ liệu hoặc phân tích trực tiếp. Một web crawler thường được lập trình để tuân thủ các nguyên tắc truy cập đã được đặt ra bởi trang web, chẳng hạn như tệp robots.txt – một tệp hướng dẫn cho biết những phần nào của trang web được phép hoặc không được phép crawl. Điều này giúp tránh việc gây quá tải hoặc vi phạm quyền riêng tư của các trang web.

URL Frontier (Hàng đợi URL):

URL Frontier là nơi lưu trữ danh sách các URL mà crawler cần duyệt qua trong suốt quá trình hoạt động. Nó hoạt động như một hàng đợi (queue), trong đó các URL sẽ lần lượt được lấy ra để crawler truy cập và thu thập dữ liệu. Khi crawler truy cập một trang web, nó sẽ phân tích nội dung trang đó để tìm các liên kết mới (có thể là liên kết nội bộ hoặc bên ngoài) và đưa chúng vào hàng đợi để xử lý sau.

URL Frontier có vai trò quan trọng trong việc quản lý hiệu quả của việc crawl, vì lượng URL có thể rất lớn. Việc sắp xếp và ưu tiên các URL trong hàng đợi (theo chiều sâu, chiều rộng, hoặc theo tiêu chí quan trọng khác như độ tin cậy, tần suất cập nhật) có thể ảnh hưởng đến hiệu suất của quá trình thu thập dữ liệu. Ngoài ra, Frontier cần có cơ chế tránh lặp lại (duplicate avoidance) để không crawl nhiều lần cùng một URL, cũng như lưu trữ thông tin về lịch sử truy cập và trạng thái của từng URL.

Parsing và Extracting Data (Phân tích cú pháp và trích xuất dữ liệu):

Sau khi crawler tải trang web về, bước tiếp theo là phân tích cú pháp (parsing) HTML của trang để trích xuất các thông tin cần thiết. Nội dung của một trang web thường được viết dưới dạng HTML, CSS, JavaScript, và có thể chứa văn bản, hình ảnh, bảng biểu, hoặc dữ liệu động. Parser có nhiệm vụ chuyển đổi cấu trúc HTML này thành dữ liệu có cấu trúc mà hệ thống có thể hiểu và xử lý.

Các bước phân tích cú pháp có thể bao gồm:

- **Xử lý HTML:** Parser duyệt qua các thẻ HTML để tìm các thành phần như tiêu đề, đoạn văn, hình ảnh, hoặc siêu dữ liệu (*metadata*) trong thẻ `<meta>`.
- **Xử lý JavaScript:** Nhiều trang web hiện đại sử dụng JavaScript để tải nội dung động. Trong trường hợp này, parser có thể sử dụng các công cụ như Selenium hoặc Puppeteer để tương tác với trang và lấy nội dung đã được tải qua JavaScript.
- **Trích xuất dữ liệu:** Sau khi cú pháp HTML hoặc JavaScript đã được xử lý, hệ thống sẽ trích xuất thông tin quan trọng như văn bản, giá sản phẩm, hình ảnh, liên kết, hoặc bất kỳ thông tin nào mà crawler đang tìm kiếm.

Một parser hiệu quả cần có khả năng xử lý được nhiều định dạng và cấu trúc trang web khác nhau, cũng như có khả năng đối phó với các trường hợp ngoại lệ như lỗi cú pháp hoặc trang bị bảo vệ bằng mã hóa.

Data Storage (Lưu trữ dữ liệu):

Dữ liệu thu thập được từ các trang web cần được lưu trữ để có thể sử dụng cho các mục đích phân tích hoặc xử lý sau này. Tùy vào mục tiêu của quá trình crawl, dữ liệu có thể được lưu trữ dưới nhiều dạng khác nhau:

- **Cơ sở dữ liệu quan hệ (SQL):** Nếu dữ liệu cần có cấu trúc rõ ràng và có mối liên hệ giữa các thực thể, cơ sở dữ liệu quan hệ như SQL Server, MySQL, PostgreSQL. Dữ liệu có thể bao gồm nhiều bảng chứa các thông tin khác nhau.
- **Cơ sở dữ liệu NoSQL:** Trong trường hợp dữ liệu phi cấu trúc hoặc bán cấu trúc (như dữ liệu từ mạng xã hội hoặc tệp JSON), các hệ thống NoSQL như MongoDB hoặc Elasticsearch.
- **File lưu trữ:** Dữ liệu có thể được lưu trữ dưới dạng các file đơn lẻ như JSON, CSV, hoặc XML để dễ dàng phân tích hoặc chia sẻ với các hệ thống khác. Cách tiếp cận này phù hợp khi dữ liệu không quá phức tạp và không cần tính năng truy vấn nâng cao.
- **Kho lưu trữ lớn (Big Data):** Đối với các dự án thu thập dữ liệu ở quy mô rất lớn, các hệ thống Big Data như Hadoop hoặc Apache Spark có thể được sử dụng để lưu trữ và xử lý dữ liệu phân tán.

2.1.2 Phương pháp Crawl Data

Crawl Data có thể được thực hiện theo nhiều cách khác nhau tùy thuộc vào mục tiêu và cách tổ chức của hệ thống web mà crawler phải duyệt qua. Các phương pháp crawl phổ biến bao gồm:

Crawl theo chiều sâu (Depth-first crawl)

Phương pháp crawl theo chiều sâu (Depth-first search) tập trung vào việc truy cập và thu thập dữ liệu từ các trang con (subpages) của một trang cụ thể trước khi chuyển sang các trang web khác ở cùng cấp độ.

- **Quy trình:** Khi crawler gặp một trang web, nó sẽ bắt đầu từ trang gốc (root page), tìm và truy cập tất cả các liên kết trên trang này. Sau đó, nó sẽ đi sâu vào từng liên kết và tiếp tục truy cập tất cả các trang con trước khi quay lại và chuyển sang các liên kết ở cấp trên.
- **Ưu điểm:** Phương pháp này giúp nhanh chóng thu thập toàn bộ thông tin từ một khu vực cụ thể của trang web. Điều này rất hữu ích khi hệ thống muốn

ưu tiên những trang con quan trọng hoặc khi chỉ một vài phần của trang web chứa nội dung có giá trị.

- **Nhược điểm:** Crawl theo chiều sâu có thể khiến hệ thống bỏ qua các trang web khác hoặc các trang ngang cấp quan trọng. Điều này dẫn đến việc mất cân đối trong việc thu thập dữ liệu từ nhiều phần khác nhau của trang web, làm cho crawler có thể mất nhiều thời gian để quay lại những khu vực khác của trang web nếu cấu trúc liên kết phức tạp.

Ví dụ, nếu một trang thương mại điện tử có hàng ngàn sản phẩm được phân loại thành nhiều mục và danh mục, crawl theo chiều sâu sẽ ưu tiên thu thập toàn bộ thông tin trong một danh mục trước khi chuyển sang các danh mục khác.

Crawl theo chiều rộng (Breadth-first crawl)

Crawl theo chiều rộng (Breadth-first search) là phương pháp mà crawler ưu tiên thu thập dữ liệu từ tất cả các trang cùng cấp độ (ngang hàng) trước khi đi sâu vào các trang con.

- **Quy trình:** Crawler bắt đầu từ trang gốc, tìm kiếm và truy cập tất cả các liên kết cấp cao (first-level links). Sau đó, nó sẽ quay lại và tiếp tục tìm kiếm các liên kết con của những trang này (second-level links), và cứ thế tiếp tục theo từng cấp độ.
- **Ưu điểm:** Phương pháp này giúp crawler có cái nhìn tổng quan về toàn bộ cấu trúc trang web trước khi đi sâu vào bất kỳ khu vực cụ thể nào. Điều này đặc biệt hữu ích khi hệ thống muốn thu thập nhanh chóng nhiều trang hoặc cần duy trì sự cân bằng trong quá trình thu thập dữ liệu từ nhiều phần khác nhau của một trang web lớn.
- **Nhược điểm:** Crawl theo chiều rộng có thể khiến crawler mất nhiều thời gian trước khi đi sâu vào những trang con có giá trị. Điều này không hiệu quả nếu các trang con mới là nơi chứa thông tin quan trọng. Hơn nữa, phương pháp này yêu cầu lưu trữ tạm thời nhiều URL trong hàng đợi, có thể làm tăng yêu cầu về bộ nhớ.

Ví dụ, nếu crawler đang duyệt qua một trang tin tức, crawl theo chiều rộng sẽ thu thập tất cả các bài viết chính trên trang chủ trước khi bắt đầu duyệt qua các bài viết chi tiết trong từng chuyên mục.

Focused Crawling (Crawl có trọng tâm)

Focused Crawling là phương pháp crawl chỉ tập trung vào những trang web có nội dung liên quan đến một chủ đề cụ thể mà crawler đang tìm kiếm, thay vì crawl tất cả các trang mà crawler có thể truy cập được. Đây là một cách tối ưu hóa crawl khi hệ thống chỉ quan tâm đến một loại nội dung nhất định và không muốn lãng phí tài nguyên vào các trang không liên quan.

- **Quy trình:** Crawler sẽ duyệt qua các trang web và phân tích nội dung của các trang được truy cập để xác định xem chúng có liên quan đến chủ đề mà nó đang tập trung hay không. Chỉ những trang có nội dung liên quan mới được crawler thu thập thông tin, còn những trang khác sẽ bị bỏ qua. Hệ thống có thể sử dụng các từ khóa, bộ lọc, hoặc các thuật toán học máy để xác định mức độ liên quan của trang web với chủ đề cần tìm kiếm.
- **Ưu điểm:** Focused Crawling giúp tiết kiệm tài nguyên, thời gian, và băng thông bằng cách chỉ thu thập những nội dung có giá trị. Điều này đặc biệt hữu ích khi thu thập dữ liệu cho các ứng dụng chuyên biệt, chẳng hạn như công cụ tìm kiếm chuyên về y tế, công nghệ, hoặc sản phẩm cụ thể.
- **Nhược điểm:** Do chỉ tập trung vào một phần nhỏ của web, crawler có thể bỏ qua những thông tin liên quan ở những trang không nằm trong phạm vi chủ đề hoặc bị che giấu trong các phần khác của trang web mà crawler không duyệt qua.

Ví dụ, nếu hệ thống đang xây dựng một công cụ tìm kiếm chuyên về y tế, Focused Crawling sẽ tập trung vào các trang web liên quan đến thông tin y khoa và bỏ qua những trang không có liên quan, chẳng hạn như blog cá nhân hoặc tin tức không thuộc lĩnh vực này.

Incremental Crawling (Crawl gia tăng)

Incremental Crawling là phương pháp crawl chỉ thu thập dữ liệu từ các trang web đã thay đổi kể từ lần crawl trước đó, thay vì crawl lại toàn bộ trang web. Điều này giúp tối ưu hóa quy trình crawl và tiết kiệm tài nguyên, đặc biệt đối với các trang web có nội dung được cập nhật liên tục.

- **Quy trình:** Hệ thống sẽ theo dõi thời gian thay đổi hoặc phiên bản mới của từng trang web và chỉ thu thập dữ liệu từ những trang đã thay đổi nội dung so với lần thu thập trước. Crawler sẽ so sánh phiên bản hiện tại của trang với phiên bản đã lưu trữ để xác định có cần thu thập lại hay không.

- **Ưu điểm:** Incremental Crawling giúp tiết kiệm băng thông, tài nguyên máy tính và giảm thiểu thời gian thu thập dữ liệu. Nó cũng giúp giảm tải cho các trang web vì không yêu cầu crawl toàn bộ nội dung nhiều lần.
- **Nhược điểm:** Một số thay đổi nhỏ hoặc không dễ thấy trên trang web có thể không được phát hiện nếu crawler chỉ dựa vào thời gian hoặc kích thước thay đổi, làm mất một số thông tin quan trọng.

Ví dụ, đối với các trang tin tức lớn như BBC hoặc CNN, nơi nội dung được cập nhật liên tục, Incremental Crawling sẽ chỉ thu thập dữ liệu từ các bài viết hoặc chuyên mục mới được cập nhật kể từ lần thu thập trước đó.

Distributed Crawling (Crawl phân tán)

Distributed Crawling là phương pháp crawl trong đó nhiệm vụ crawl được chia thành nhiều phần nhỏ và phân phối cho nhiều crawler hoạt động đồng thời trên nhiều máy chủ khác nhau. Điều này giúp tăng tốc độ thu thập dữ liệu và cho phép crawl một khối lượng lớn các trang web trong thời gian ngắn.

- **Quy trình:** Hệ thống quản lý sẽ chia các URL cần thu thập thành các phân đoạn khác nhau và phân phối chúng cho các crawler trên nhiều máy tính hoặc máy chủ. Các crawler hoạt động độc lập nhưng có sự phối hợp để tránh việc trùng lặp URL và đảm bảo tất cả các URL cần thiết đều được thu thập.
- **Ưu điểm:** Distributed Crawling cho phép hệ thống thu thập dữ liệu từ một số lượng lớn trang web với tốc độ cao và hiệu quả hơn. Nó giúp phân tải công việc, giảm nguy cơ hệ thống bị quá tải.
- **Nhược điểm:** Quá trình này cần sự đồng bộ và quản lý tốt để tránh trùng lặp URL hoặc bỏ sót dữ liệu. Việc triển khai cũng phức tạp hơn do yêu cầu về hạ tầng và quản lý nhiều máy chủ.

Distributed Crawling thường được sử dụng bởi các công cụ tìm kiếm lớn như Google hoặc Bing để duyệt và cập nhật dữ liệu từ hàng tỷ trang web trên Internet.

2.1.3 Công cụ và công nghệ sử dụng trong Crawl Data

Crawl Data là quá trình thu thập và trích xuất dữ liệu từ các trang web thông qua việc duyệt qua nhiều URL và phân tích cú pháp của các trang web đó. Để thực hiện crawl data một cách hiệu quả, các công cụ và thư viện khác nhau được sử dụng, tùy thuộc vào độ phức tạp của dự án và yêu cầu cụ thể về tốc độ, hiệu suất,

và khả năng xử lý dữ liệu động. Dưới đây là một số công cụ phổ biến và mạnh mẽ trong crawl data, được sử dụng rộng rãi trong cả các dự án cá nhân lẫn các hệ thống quy mô lớn.

– BeautifulSoup và Requests (Python)

BeautifulSoup và *Requests* là hai thư viện Python phổ biến, thường được sử dụng kết hợp với nhau để thực hiện việc thu thập dữ liệu từ các trang web.

BeautifulSoup: Là thư viện mạnh mẽ để phân tích cú pháp HTML và XML, giúp người dùng dễ dàng trích xuất thông tin cần thiết từ các trang web. BeautifulSoup hỗ trợ các truy vấn theo cây DOM (Document Object Model), giúp người dùng lấy các phần tử cụ thể như tiêu đề, hình ảnh, liên kết hoặc đoạn văn bản.

* Ưu điểm:

- Dễ học và sử dụng, phù hợp cho các dự án nhỏ hoặc các ứng dụng đơn giản.
- Có khả năng xử lý HTML "bẩn" hoặc không tuân thủ hoàn toàn quy tắc HTML.
- Tích hợp tốt với các công cụ khác như Requests để thu thập dữ liệu một cách nhanh chóng.

* **Nhược điểm:** BeautifulSoup không phải là lựa chọn tối ưu cho việc xử lý lượng lớn dữ liệu hoặc crawl toàn bộ trang web với hàng triệu URL, vì nó thiếu tính năng quản lý hàng đợi và tối ưu hóa tốc độ.

Requests: Thư viện Python phổ biến để thực hiện các yêu cầu HTTP. Requests cho phép người dùng dễ dàng gửi các yêu cầu GET, POST và lấy nội dung trang web về để phân tích cú pháp bằng BeautifulSoup.

* **Ưu điểm:** Requests rất dễ sử dụng và hiệu quả, với cú pháp rõ ràng cho phép truy cập các trang web một cách nhanh chóng.

* **Nhược điểm:** Requests không hỗ trợ việc xử lý JavaScript hoặc các thao tác yêu cầu tương tác phức tạp với trang web.

– Scrapy

Scrapy là một framework mã nguồn mở mạnh mẽ được viết bằng Python, được sử dụng để xây dựng các hệ thống crawl dữ liệu phức tạp. Scrapy hỗ trợ crawl và trích xuất dữ liệu ở quy mô lớn với hiệu suất cao.

Tính năng chính:

Tích hợp sẵn hàng đợi URL và quản lý luồng dữ liệu: Scrapy tự động quản lý hàng đợi URL và việc xử lý song song các yêu cầu HTTP, giúp tăng tốc quá trình thu thập dữ liệu.

Pipeline xử lý dữ liệu: Scrapy cung cấp các pipeline để xử lý và lưu trữ dữ liệu một cách linh hoạt, chẳng hạn như xuất dữ liệu ra tệp CSV, JSON hoặc lưu trực tiếp vào cơ sở dữ liệu.

Khả năng tái sử dụng và mở rộng: Scrapy có thể được mở rộng bằng cách thêm các bộ lọc hoặc bộ thu thập dữ liệu tùy chỉnh, giúp người dùng dễ dàng xây dựng các hệ thống crawl có khả năng tùy biến cao.

Hỗ trợ cho crawl dữ liệu động và bảo mật: Scrapy có thể tích hợp với các công cụ như Splash hoặc Selenium để xử lý JavaScript và các trang web yêu cầu tương tác.

*** Ưu điểm:**

- Hiệu suất cao: Scrapy được tối ưu hóa cho các dự án lớn với hàng triệu URL. Crawler của Scrapy có khả năng xử lý song song nhiều yêu cầu HTTP, giúp tăng tốc quá trình thu thập dữ liệu.
- Cộng đồng lớn và hỗ trợ mạnh mẽ: Scrapy có một cộng đồng lớn và cung cấp nhiều tài liệu hướng dẫn, giúp người dùng dễ dàng học hỏi và áp dụng vào các dự án của mình.
- Tích hợp tốt với các hệ thống lưu trữ dữ liệu: Scrapy hỗ trợ việc xuất dữ liệu ra nhiều định dạng và lưu trữ vào cơ sở dữ liệu như MongoDB, MySQL.

*** Nhược điểm:** Scrapy phức tạp hơn BeautifulSoup và có thể khó tiếp cận cho những người mới bắt đầu. Cần phải hiểu rõ cấu trúc framework để triển khai hiệu quả.

– Selenium

Selenium là một công cụ tự động hóa trình duyệt, thường được sử dụng để crawl dữ liệu từ các trang web yêu cầu tương tác phức tạp, chẳng hạn như các trang cần điền form, cuộn trang, hoặc xử lý nội dung được tạo động bằng JavaScript.

Tính năng chính:

Tự động hóa tương tác với trang web: Selenium cho phép người dùng tự động hóa các hành động như điền form, nhấp chuột, cuộn trang và tải lại trang.

Xử lý JavaScript: Selenium có khả năng xử lý các trang web phụ thuộc vào JavaScript, giúp thu thập dữ liệu từ các trang web hiện đại không thể được crawl bằng các công cụ không hỗ trợ JavaScript như BeautifulSoup.

Hỗ trợ nhiều trình duyệt: Selenium hỗ trợ tự động hóa trên nhiều trình duyệt khác nhau như Chrome, Firefox, Edge, Safari.

*** Ưu điểm:**

- Khả năng tương tác mạnh mẽ: Selenium cho phép crawl các trang web yêu cầu người dùng tương tác, chẳng hạn như trang web chứa nội dung AJAX hoặc JavaScript phức tạp.
- Độ tùy chỉnh cao: Người dùng có thể dễ dàng thiết lập các hành động tương tác cụ thể, giúp crawl dữ liệu chính xác hơn từ các trang web phức tạp.

*** Nhược điểm:**

- Hiệu suất chậm: Vì Selenium tương tác trực tiếp với trình duyệt web, tốc độ crawl dữ liệu sẽ chậm hơn so với các công cụ không sử dụng giao diện trình duyệt.
- Tốn tài nguyên: Selenium yêu cầu nhiều tài nguyên hệ thống hơn và không phù hợp để crawl dữ liệu quy mô lớn.

– Heritrix

Heritrix là một công cụ web crawler mã nguồn mở được phát triển bởi Internet Archive. Đây là một trong những công cụ mạnh mẽ nhất hiện nay để crawl dữ liệu ở quy mô lớn, với mục đích lưu trữ và bảo tồn nội dung web.

Tính năng chính:

Crawl quy mô lớn: Heritrix được thiết kế để thu thập dữ liệu từ hàng triệu trang web, với khả năng xử lý hàng loạt và lưu trữ dữ liệu trên các hệ thống phân tán.

Kiến trúc modular: Heritrix cung cấp kiến trúc linh hoạt, cho phép người dùng dễ dàng tùy chỉnh và thêm các module theo nhu cầu cụ thể của dự án.

Quản lý crawl hiệu quả: Heritrix sử dụng hàng đợi tiên tiến và các thuật toán crawl tối ưu hóa để đảm bảo việc thu thập dữ liệu được thực hiện một cách hiệu quả nhất.

*** Ưu điểm:**

- Hiệu suất cao và khả năng mở rộng: Heritrix là lựa chọn lý tưởng cho các tổ chức và doanh nghiệp muốn thu thập dữ liệu từ một lượng lớn các trang web với tốc độ cao và độ ổn định cao.
- Được sử dụng bởi các tổ chức lớn: Heritrix là công cụ tiêu chuẩn của Internet Archive để thu thập và lưu trữ nội dung web cho các thư viện và bảo tàng kỹ thuật số trên toàn thế giới.

*** Nhược điểm:**

- Phức tạp khi cài đặt và sử dụng: Heritrix yêu cầu kiến thức chuyên sâu về cấu hình hệ thống và không dễ tiếp cận đối với người mới bắt đầu.
- Không linh hoạt cho các dự án nhỏ: Với các dự án nhỏ hoặc các yêu cầu crawl đơn giản, Heritrix có thể quá phức tạp và không hiệu quả về chi phí và tài nguyên.

2.2 SQL Server

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS - Relational Database Management System) do Microsoft phát triển. Nó sử dụng ngôn ngữ SQL (Structured Query Language) để quản lý và truy vấn dữ liệu. SQL Server cung cấp nhiều tính năng cho việc lưu trữ, truy vấn, bảo mật, và phân tích dữ liệu.

2.2.1 Cơ sở dữ liệu quan hệ (Relational Database)

Cơ sở dữ liệu quan hệ (RDBMS - Relational Database Management System) là hệ thống lưu trữ và quản lý dữ liệu theo mô hình bảng (table). Mỗi bảng trong cơ sở dữ liệu được tổ chức thành các dòng (hay bản ghi) và cột (hay thuộc tính). Các bảng này có thể được liên kết với nhau thông qua các khóa như khóa chính (primary key) và khóa ngoại (foreign key), cho phép dữ liệu trong nhiều bảng có thể kết nối và liên hệ với nhau một cách dễ dàng.

SQL Server là một RDBMS mạnh mẽ, có nhiệm vụ quản lý, lưu trữ và truy xuất dữ liệu thông qua việc thực hiện các thao tác trên các bảng liên kết với nhau. Dưới đây là các khái niệm quan trọng trong cơ sở dữ liệu quan hệ:

Primary Key (Khóa chính)

Primary Key (Khóa chính) là một hoặc một nhóm các cột được sử dụng để xác định duy nhất một bản ghi (row) trong bảng. Các đặc điểm chính của khóa chính bao gồm:

- **Duy nhất:** Giá trị của khóa chính phải là duy nhất trong toàn bộ bảng, nghĩa là không có hai bản ghi nào có thể có cùng giá trị khóa chính.
- **Không null:** Giá trị của khóa chính không thể rỗng (NULL), vì cần phải có giá trị để xác định mỗi bản ghi.
- **Một bảng chỉ có một khóa chính:** Tức là, mỗi bảng chỉ có thể có một khóa chính duy nhất để phân biệt từng bản ghi.

Ví dụ, trong một bảng quản lý sinh viên, cột StudentID có thể được dùng làm khóa chính vì mỗi sinh viên sẽ có một mã số duy nhất.

Foreign Key (Khóa ngoại)

Foreign Key (Khóa ngoại) là một hoặc một nhóm cột trong một bảng, tham chiếu đến Primary Key của một bảng khác. Khóa ngoại tạo ra một mối quan hệ giữa hai bảng, cho phép liên kết dữ liệu giữa các bảng khác nhau. Điều này giúp duy trì tính toàn vẹn dữ liệu, đảm bảo rằng các giá trị trong khóa ngoại phải tồn tại trong bảng được tham chiếu. Ví dụ, trong cơ sở dữ liệu quản lý sinh viên và khóa học, bảng Enrollments lưu trữ thông tin đăng ký của sinh viên vào các khóa học, có thể có StudentID làm khóa ngoại, tham chiếu đến bảng Students.

Normalization (Chuẩn hóa dữ liệu)

Normalization (Chuẩn hóa dữ liệu) là quá trình tổ chức dữ liệu trong cơ sở dữ liệu theo cách giảm thiểu sự trùng lặp và phụ thuộc không cần thiết. Quá trình này giúp cho cơ sở dữ liệu trở nên linh hoạt hơn, dễ bảo trì, và tránh được lỗi liên quan đến việc cập nhật, chèn, hoặc xóa dữ liệu.

Normalization thường được chia thành nhiều cấp độ khác nhau, gọi là *normal forms (dạng chuẩn hóa)*, với ba dạng chính phổ biến là:

- **First Normal Form (1NF):** Yêu cầu dữ liệu trong bảng phải ở dạng nguyên tử (atomic), tức là mỗi cột chỉ chứa một giá trị đơn lẻ, không chứa các tập giá trị hoặc mảng trong một ô.
Ví dụ: Nếu một bảng chứa thông tin về sách, thì mỗi ô dữ liệu phải chứa một giá trị duy nhất, chẳng hạn như chỉ một tên tác giả hoặc một thể loại sách, thay vì liệt kê nhiều tên tác giả trong một ô.

- **Second Normal Form (2NF):** Đạt được khi bảng đã ở dạng 1NF và tất cả các thuộc tính không phải khóa trong bảng đều phụ thuộc hoàn toàn vào khóa chính, không có sự phụ thuộc từng phần.

Ví dụ: Nếu một bảng lưu trữ thông tin về sinh viên và khóa học, cần phải đảm bảo rằng các thuộc tính như tên khóa học hoặc điểm số chỉ phụ thuộc vào cả StudentID và CourseID (khóa chính phức hợp), không phải vào từng thuộc tính riêng lẻ.

- **Third Normal Form (3NF):** Đạt được khi bảng đã ở dạng 2NF và tất cả các thuộc tính không phải khóa chính đều độc lập với nhau, tức là không có sự phụ thuộc bắc cầu giữa các thuộc tính không phải khóa.

Ví dụ: Nếu bảng sinh viên lưu thông tin địa chỉ và thành phố, không nên lưu trữ tên thành phố ở bảng sinh viên mà nên tách thành một bảng riêng để tránh việc lặp lại tên thành phố cho mỗi sinh viên.

Các bước chuẩn hóa giúp giảm thiểu sự dư thừa dữ liệu và tăng tính toàn vẹn, đảm bảo rằng dữ liệu được lưu trữ một cách hợp lý, dễ bảo trì và giảm thiểu lỗi.

2.2.2 Môi quan hệ giữa các bảng trong cơ sở dữ liệu

Trong cơ sở dữ liệu quan hệ, có ba loại mối quan hệ chính giữa các bảng:

- **One-to-One (Một-một):** Mỗi bản ghi trong bảng A liên kết với một bản ghi duy nhất trong bảng B và ngược lại. Mối quan hệ này ít gặp trong thực tế.
- **One-to-Many (Một-nhiều):** Một bản ghi trong bảng A có thể liên kết với nhiều bản ghi trong bảng B, nhưng mỗi bản ghi trong bảng B chỉ liên kết với một bản ghi trong bảng A. Đây là mối quan hệ phổ biến nhất.
- **Many-to-Many (Nhiều-nhiều):** Một bản ghi trong bảng A có thể liên kết với nhiều bản ghi trong bảng B và ngược lại. Để biểu diễn mối quan hệ này, cần sử dụng một bảng trung gian (bridge table) chứa khóa ngoại của cả hai bảng A và B.

2.2.3 Ngôn ngữ SQL

SQL (Structured Query Language) là ngôn ngữ chính mà SQL Server sử dụng để truy vấn và quản lý dữ liệu. SQL có thể chia thành ba nhóm ngôn ngữ chính:

- **DDL (Data Definition Language):** Là ngôn ngữ định nghĩa dữ liệu, dùng để tạo, thay đổi, hoặc xóa cấu trúc cơ sở dữ liệu.
 - * Tạo bảng: **CREATE TABLE**
 - * Sửa đổi bảng: **ALTER TABLE**
 - * Xóa bảng: **DROP TABLE**
- **DML (Data Manipulation Language):** Là ngôn ngữ thao tác dữ liệu, dùng để truy vấn và thay đổi dữ liệu trong cơ sở dữ liệu.
 - * Truy vấn dữ liệu: **SELECT**
 - * Thêm dữ liệu: **INSERT**
 - * Cập nhật dữ liệu: **UPDATE**
 - * Xóa dữ liệu: **DELETE**
- **DCL (Data Control Language):** Là ngôn ngữ kiểm soát dữ liệu, dùng để quản lý quyền truy cập dữ liệu trong hệ thống.
 - * Cấp quyền: **GRANT**
 - * Thu hồi quyền: **REVOKE**

SQL Server còn hỗ trợ **T-SQL (Transact-SQL)**, là phần mở rộng của SQL, cung cấp thêm các tính năng như điều kiện rẽ nhánh, vòng lặp và quản lý lỗi.

2.2.4 Cấu trúc của SQL Server

SQL Server là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mạnh mẽ, có cấu trúc phân cấp từ các thành phần cấp cao nhất như instance (phiên bản) đến các thành phần cấp thấp hơn như bảng (table), chỉ mục (index), và các đối tượng khác bên trong cơ sở dữ liệu. Các thành phần chính của SQL Server bao gồm:

Instance (Phiên bản)

Một instance của SQL Server là một phiên bản hoạt động của phần mềm SQL Server trên một máy chủ cụ thể. Một máy chủ vật lý hoặc máy chủ ảo có thể chạy nhiều instance SQL Server đồng thời, giúp quản trị viên dễ dàng quản lý nhiều cơ sở dữ liệu khác nhau trên cùng một máy chủ.

Mỗi instance có thể là default instance (phiên bản mặc định) hoặc named instance (phiên bản đặt tên).

Mỗi instance hoạt động như một máy chủ riêng biệt với cấu hình và bộ tài nguyên riêng, bao gồm các cơ sở dữ liệu, bảng, chỉ mục, và các đối tượng khác.

Ví dụ, bạn có thể có một instance dành cho môi trường phát triển và một instance khác dành cho môi trường sản xuất trên cùng một máy chủ.

Database (Cơ sở dữ liệu)

Một cơ sở dữ liệu trong SQL Server là tập hợp các bảng, chỉ mục, view, stored procedure, và các đối tượng khác. Mỗi cơ sở dữ liệu có các thành phần quan trọng sau:

Data File (tệp dữ liệu): Mỗi cơ sở dữ liệu lưu trữ dữ liệu của nó trong các tệp vật lý (các tệp .mdf, .ndf).

Log File (tệp nhật ký): Cơ sở dữ liệu cũng có tệp nhật ký (.ldf), dùng để ghi lại các thay đổi đối với cơ sở dữ liệu và đảm bảo tính toàn vẹn của dữ liệu trong trường hợp khôi phục.

Mỗi cơ sở dữ liệu trong một instance là một đơn vị riêng biệt, có thể chứa nhiều bảng và đối tượng khác nhau.

Table (Bảng)

Bảng là thành phần cơ bản nhất của cơ sở dữ liệu, nơi lưu trữ dữ liệu dưới dạng các hàng (record/row) và cột (field). Mỗi bảng có cấu trúc cố định với các cột có kiểu dữ liệu nhất định (như INT, VARCHAR, DATETIME), và mỗi hàng trong bảng đại diện cho một bản ghi của dữ liệu.

Primary Key (Khóa chính): Mỗi bảng thường có một khóa chính (primary key) là cột (hoặc tập hợp cột) dùng để xác định duy nhất mỗi bản ghi.

Foreign Key (Khóa ngoại): Các bảng có thể liên kết với nhau thông qua các khóa ngoại (foreign key), tạo mối quan hệ giữa các bảng trong cơ sở dữ liệu.

Các bảng có thể áp dụng các constraints (ràng buộc) để đảm bảo tính toàn vẹn dữ liệu, như NOT NULL, UNIQUE, CHECK, và DEFAULT.

Index (Chỉ mục)

Chỉ mục là một cấu trúc dữ liệu đặc biệt được tạo ra trên một hoặc nhiều cột của bảng, nhằm tăng tốc độ truy vấn dữ liệu. Khi dữ liệu trong bảng được sắp xếp theo thứ tự trong chỉ mục, SQL Server có thể tìm kiếm dữ liệu nhanh hơn.

Clustered Index: Chỉ mục này sắp xếp dữ liệu vật lý trong bảng theo thứ tự của cột chỉ mục. Mỗi bảng chỉ có thể có một clustered index, vì nó xác định thứ tự lưu trữ vật lý của dữ liệu.

Non-Clustered Index: Chỉ mục này tạo một bản sao sắp xếp của dữ liệu dựa trên cột chỉ mục, trong khi dữ liệu vật lý trong bảng vẫn giữ nguyên. Một bảng có thể có nhiều non-clustered index.

Chỉ mục giúp tăng tốc độ truy vấn nhưng đồng thời có thể làm giảm hiệu suất của các thao tác cập nhật, chèn, và xóa dữ liệu vì cần cập nhật lại chỉ mục.

2.3 Task Scheduler

Task Scheduler là một công cụ quản lý trong hệ điều hành Windows cho phép người dùng tự động hóa các tác vụ bằng cách lập lịch và thực hiện chúng theo thời gian hoặc theo sự kiện cụ thể. Công cụ này giúp cải thiện hiệu suất làm việc, giảm thiểu công việc lặp đi lặp lại và đảm bảo rằng các tác vụ quan trọng được thực hiện đúng thời gian.

2.3.1 Tự động hóa tác vụ với Task Scheduler

Lập kế hoạch thực hiện

- **Tạo và quản lý tác vụ:** Người dùng có thể dễ dàng tạo tác vụ mới thông qua giao diện người dùng của Task Scheduler. Việc lập kế hoạch cho các tác vụ cụ thể giúp giảm thiểu công việc thủ công. Ví dụ, một người quản trị hệ thống có thể lập lịch cho các tác vụ sao lưu dữ liệu vào cuối mỗi ngày hoặc mỗi tuần.
- **Chọn tác vụ cụ thể:** Người dùng có thể chọn từ nhiều loại tác vụ khác nhau, chẳng hạn như chạy ứng dụng, thực hiện script Powershell, hoặc gửi email tự động. Các tác vụ này có thể được cấu hình để chạy với các tham số hoặc điều kiện nhất định, giúp tăng cường tính linh hoạt.
- **Xác định thời gian thực hiện:** Người dùng có thể chỉ định thời gian chính xác để thực hiện tác vụ, chẳng hạn như vào lúc 3 giờ chiều mỗi ngày hoặc vào lúc khởi động máy tính. Điều này đảm bảo rằng các tác vụ quan trọng không bị bỏ lỡ và được thực hiện đúng thời gian.

Tích hợp với các sự kiện

- **Kích hoạt dựa trên sự kiện hệ thống:** Task Scheduler có thể thiết lập để thực hiện các tác vụ khi một sự kiện cụ thể xảy ra trong hệ thống. Ví dụ, người dùng có thể thiết lập để chạy một script tự động mỗi khi máy tính khởi động, giúp đảm bảo rằng các quy trình cần thiết được thực hiện ngay từ đầu.
- **Theo dõi các sự kiện người dùng:** Người dùng có thể lập lịch cho các tác vụ khi họ đăng nhập vào hệ thống hoặc khi đăng xuất. Điều này rất hữu ích cho các tác vụ như chạy các chương trình cần thiết cho công việc của người dùng hoặc thực hiện các tác vụ dọn dẹp sau khi người dùng rời khỏi máy tính.
- **Kích hoạt theo các điều kiện tùy chỉnh:** Task Scheduler cũng cho phép người dùng thiết lập các điều kiện cụ thể hơn, chẳng hạn như chỉ thực hiện tác vụ khi máy tính đang sử dụng nguồn điện AC (không phải pin) hoặc khi mạng đang kết nối với Internet. Điều này giúp đảm bảo rằng các tác vụ không gây ảnh hưởng đến hiệu suất của máy tính trong các tình huống không phù hợp.

Chạy các tác vụ định kỳ

- **Tự động hóa nhiệm vụ lặp đi lặp lại:** Người dùng có thể thiết lập các tác vụ chạy theo lịch định kỳ, giúp tự động hóa các nhiệm vụ thường xuyên mà không cần phải can thiệp thủ công. Chẳng hạn, một kế toán có thể lập lịch cho một tác vụ chạy báo cáo tài chính hàng tháng hoặc hàng quý.
- **Tùy chọn linh hoạt về tần suất:** Task Scheduler cung cấp nhiều tùy chọn cho việc thiết lập tần suất chạy tác vụ, bao gồm hàng ngày, hàng tuần, hàng tháng, hoặc thậm chí theo các khoảng thời gian tùy chỉnh (ví dụ: mỗi 5 ngày). Người dùng có thể dễ dàng điều chỉnh các tham số này để đáp ứng nhu cầu cụ thể của mình.
- **Cập nhật và sửa đổi dễ dàng:** Nếu có sự thay đổi trong lịch làm việc hoặc quy trình công việc, người dùng có thể dễ dàng cập nhật các tác vụ đã lập kế hoạch. Ví dụ, nếu một tác vụ cần phải chạy vào một thời điểm khác hoặc với các tham số khác, người dùng có thể điều chỉnh một cách nhanh chóng mà không cần phải xóa và tạo lại tác vụ.

2.3.2 Ghi lại sự kiện trong Task Scheduler

Ghi lại sự kiện là một trong những chức năng quan trọng của Task Scheduler, giúp người dùng quản lý và theo dõi các tác vụ đã được lập lịch một cách hiệu quả.

Theo dõi lịch sử thực hiện

- **Thông tin chi tiết về tác vụ:** Task Scheduler ghi lại các thông tin quan trọng về mỗi lần thực hiện tác vụ, bao gồm:
 - * **Thời gian bắt đầu:** Thời điểm mà tác vụ được khởi động, cho phép người dùng biết được khi nào tác vụ được thực hiện.
 - * **Thời gian kết thúc:** Thời điểm mà tác vụ hoàn tất, giúp người dùng đánh giá thời gian cần thiết để hoàn thành tác vụ.
 - * **Trạng thái:** Thông tin về kết quả của tác vụ (thành công hay thất bại). Việc theo dõi trạng thái giúp người dùng nhận biết ngay những tác vụ không thực hiện được, từ đó có thể đưa ra các quyết định kịp thời.
- **Phân tích hiệu suất:** Việc ghi lại lịch sử thực hiện không chỉ giúp người dùng theo dõi các tác vụ mà còn cung cấp dữ liệu để phân tích hiệu suất tổng thể của các tác vụ. Người dùng có thể tìm ra các mẫu, chẳng hạn như các tác vụ thường xuyên không thành công hoặc mất quá nhiều thời gian để thực hiện. Từ đó, họ có thể điều chỉnh để tối ưu hóa quy trình làm việc.

Ghi chú lỗi

- **Thông tin lỗi chi tiết:** Khi một tác vụ không thành công, Task Scheduler sẽ ghi lại thông tin chi tiết về lỗi, bao gồm:
 - * **Mã lỗi:** Một mã số cụ thể giúp xác định loại lỗi.
 - * **Mô tả lỗi:** Giúp người dùng hiểu rõ hơn về nguyên nhân gây ra lỗi.
 - * **Điều kiện xảy ra lỗi:** Thông tin về các điều kiện khi lỗi xảy ra, có thể liên quan đến tài nguyên hệ thống hoặc các tác vụ khác.
- **Hỗ trợ khắc phục sự cố:** Nhờ vào thông tin chi tiết về lỗi, người dùng có thể nhanh chóng xác định nguyên nhân của sự cố và thực hiện các biện pháp khắc phục cần thiết. Điều này giúp giảm thiểu thời gian và công sức để tìm ra giải pháp cho các vấn đề phát sinh, đồng thời cải thiện tính ổn định và hiệu suất của hệ thống.

Thông báo và cảnh báo

- **Thiết lập thông báo:** Người dùng có thể cấu hình Task Scheduler để tự động gửi thông báo qua email hoặc hiển thị thông báo trên màn hình khi một tác vụ hoàn thành hoặc không thành công. Việc này giúp người dùng nhận biết kịp thời về tình trạng của các tác vụ mà họ đã lập lịch.
- **Cảnh báo khi có sự cố:** Task Scheduler cũng cho phép người dùng thiết lập cảnh báo cho các tác vụ không thực hiện được. Điều này cực kỳ quan trọng, đặc biệt với các tác vụ liên quan đến dữ liệu quan trọng hoặc các quy trình tự động hóa phức tạp, vì nó giúp người dùng kịp thời xử lý và ngăn chặn các vấn đề lớn hơn có thể xảy ra.
- **Tùy chọn tùy chỉnh:** Ngoài các thiết lập thông báo và cảnh báo mặc định, người dùng có thể tùy chỉnh nội dung và cách thức thông báo để phù hợp với nhu cầu cụ thể của mình. Ví dụ, họ có thể thiết lập thông báo cho từng tác vụ riêng lẻ hoặc nhóm tác vụ theo các tiêu chí nhất định, giúp quản lý và giám sát hiệu quả hơn.

2.3.3 Tùy chỉnh điều kiện thực hiện trong Task Scheduler

Tùy chỉnh điều kiện thực hiện là một trong những tính năng quan trọng của Task Scheduler, cho phép người dùng tối ưu hóa cách mà các tác vụ được thực hiện dựa trên các yêu cầu và điều kiện cụ thể.

Thiết lập điều kiện

- **Điều kiện thực hiện:** Người dùng có thể xác định các điều kiện cần thiết để một tác vụ có thể được thực hiện. Một số điều kiện phổ biến bao gồm:
 - * **Kết nối Internet:** Người dùng có thể thiết lập tác vụ chỉ chạy khi máy tính đang kết nối với Internet. Điều này rất hữu ích cho các tác vụ cần truy cập tài nguyên trực tuyến, chẳng hạn như tải xuống tệp hoặc thực hiện các truy vấn từ cơ sở dữ liệu trên cloud.
 - * **Trạng thái máy tính:** Task Scheduler cho phép người dùng xác định tác vụ chỉ chạy khi máy tính đang ở trạng thái nhất định, ví dụ như không hoạt động trong một khoảng thời gian cụ thể. Điều này giúp tránh tình trạng máy tính quá tải do thực hiện nhiều tác vụ đồng thời, từ đó nâng cao hiệu suất hệ thống.
- **Kết hợp nhiều điều kiện:** Người dùng có thể kết hợp nhiều điều kiện khác nhau để xác định chính xác khi nào tác vụ nên được thực hiện. Việc này giúp

đảm bảo rằng các tác vụ không chỉ chạy trong các điều kiện lý tưởng mà còn tránh gây ra xung đột hoặc giảm hiệu suất của hệ thống.

Thời gian chờ

- **Thiết lập thời gian chờ:** Task Scheduler cho phép người dùng thiết lập thời gian chờ cho các tác vụ. Điều này có nghĩa là nếu một tác vụ không hoàn thành trong thời gian quy định, nó sẽ tự động bị ngừng lại. Tính năng này giúp quản lý tài nguyên của hệ thống một cách hiệu quả và ngăn chặn các tác vụ kéo dài không cần thiết.
- **Quản lý hiệu suất:** Bằng cách đặt thời gian chờ, người dùng có thể tránh được tình trạng một tác vụ chiếm dụng quá nhiều tài nguyên trong khi không còn khả năng hoàn thành. Điều này đặc biệt quan trọng đối với các tác vụ có thể gặp phải sự cố hoặc bị treo, vì nó giúp giảm thiểu tác động tiêu cực đến các tác vụ khác đang chạy trên hệ thống.
- **Chức năng tự động:** Nếu tác vụ không hoàn thành trong khoảng thời gian đã được thiết lập, Task Scheduler sẽ tự động ghi lại sự kiện và thông báo cho người dùng. Thông tin này bao gồm trạng thái của tác vụ và lý do nó không hoàn thành, giúp người dùng dễ dàng xác định và khắc phục sự cố.

Kết hợp các tùy chỉnh

- **Tùy chỉnh toàn diện:** Bằng cách kết hợp các điều kiện thực hiện và thời gian chờ, người dùng có thể tạo ra một môi trường làm việc tối ưu hơn cho các tác vụ tự động. Họ có thể xác định thời điểm cụ thể cho từng tác vụ dựa trên các điều kiện hiện tại của hệ thống, từ đó tối ưu hóa việc sử dụng tài nguyên và hiệu suất tổng thể.
- **Theo dõi và điều chỉnh:** Sau khi thiết lập các điều kiện và thời gian chờ cho tác vụ, người dùng có thể theo dõi hiệu suất thực hiện và điều chỉnh các thiết lập nếu cần thiết. Việc này giúp đảm bảo rằng các tác vụ được thực hiện đúng cách và phù hợp với nhu cầu thay đổi của công việc hoặc môi trường.

2.4 Tableau

Tableau là một công cụ phân tích và trực quan hóa dữ liệu mạnh mẽ, cho phép người dùng tạo ra các biểu đồ, bảng điều khiển (dashboards) và báo cáo để trực

quan hóa và hiểu rõ hơn về dữ liệu của họ. Với khả năng kết nối đến nhiều nguồn dữ liệu khác nhau và dễ sử dụng, Tableau đã trở thành một trong những công cụ phổ biến nhất trong lĩnh vực Business Intelligence (BI).

2.4.1 Các thành phần chính của Tableau:

- **Tableau Desktop:** Phần mềm dùng để tạo và phát triển các báo cáo và biểu đồ. Nó cung cấp giao diện người dùng thân thiện cho phép người dùng kéo và thả để tạo trực quan hóa dữ liệu.
- **Tableau Server:** Nền tảng cho phép chia sẻ và phân phối các báo cáo được tạo ra trên Tableau Desktop cho người dùng khác trong tổ chức. Nó cũng cung cấp các tính năng bảo mật và quản lý quyền truy cập.
- **Tableau Online:** Phiên bản đám mây của Tableau Server, cho phép người dùng truy cập các báo cáo và bảng điều khiển từ bất kỳ đâu mà không cần cài đặt phần mềm.
- **Tableau Public:** Phiên bản miễn phí cho phép người dùng chia sẻ các bảng điều khiển và báo cáo trên web. Tuy nhiên, dữ liệu trong Tableau Public là công khai và không bảo mật.

2.4.2 Nguyên lý hoạt động của Tableau

Tableau hoạt động theo mô hình truy vấn dữ liệu và trực quan hóa, cho phép người dùng dễ dàng kết nối đến nhiều nguồn dữ liệu và chuyển đổi chúng thành các biểu đồ, bảng điều khiển tương tác. Các bước cơ bản trong quy trình làm việc với Tableau bao gồm:

- **Kết nối dữ liệu:** Người dùng có thể kết nối Tableau với nhiều loại nguồn dữ liệu khác nhau như cơ sở dữ liệu SQL, file Excel, Google Analytics, và các nguồn dữ liệu đám mây khác.
- **Tạo trực quan hóa:** Sau khi kết nối với dữ liệu, người dùng có thể tạo ra các trực quan hóa bằng cách kéo và thả các trường dữ liệu vào không gian làm việc. Tableau hỗ trợ nhiều loại biểu đồ khác nhau như biểu đồ thanh, biểu đồ đường, bản đồ, và nhiều loại trực quan hóa khác.
- **Xây dựng bảng điều khiển (Dashboard):** Các trực quan hóa có thể được kết hợp lại để tạo ra các bảng điều khiển, giúp người dùng có cái nhìn tổng

quan về dữ liệu. Bảng điều khiển có thể bao gồm nhiều biểu đồ và báo cáo để hiển thị thông tin một cách trực quan và dễ hiểu.

- **Chia sẻ và cộng tác:** Người dùng có thể chia sẻ các bảng điều khiển và báo cáo qua Tableau Server hoặc Tableau Online, giúp các thành viên trong tổ chức có thể truy cập và tương tác với dữ liệu.

2.4.3 Khả năng trực quan hóa của Tableau

Tableau nổi bật với khả năng trực quan hóa dữ liệu phong phú, cho phép người dùng:

- **Tương tác trực tiếp:** Người dùng có thể tương tác với các trực quan hóa bằng cách lọc dữ liệu, chọn các thành phần và khám phá thông tin sâu hơn một cách dễ dàng.
- **Tạo báo cáo động:** Tableau hỗ trợ tạo các báo cáo động, nơi người dùng có thể thay đổi điều kiện và xem sự thay đổi của dữ liệu trong thời gian thực.
- **Kết nối dữ liệu thời gian thực:** Tableau có khả năng kết nối với các nguồn dữ liệu thời gian thực, giúp người dùng theo dõi và phân tích dữ liệu ngay khi nó thay đổi.

2.4.4 Các khái niệm quan trọng trong Tableau

- **Dimension và Measure:**
 - * **Dimension** (Chiều) là các thuộc tính dùng để phân loại hoặc nhóm dữ liệu, chẳng hạn như tên sản phẩm, khu vực, hoặc thời gian. (cột)
 - * **Measure** (Đo lường) là các giá trị số mà người dùng muốn phân tích, như doanh thu, số lượng, hoặc chi phí. (hàng)
- **Calculated Field:** Người dùng có thể tạo ra các trường tính toán để thực hiện các phép toán phức tạp trên dữ liệu, cho phép họ tùy chỉnh và mở rộng các phân tích.
- **Filters:** Tableau cho phép người dùng áp dụng bộ lọc để giới hạn dữ liệu hiển thị trong trực quan hóa, giúp tập trung vào những thông tin quan trọng.

2.5 Streamlit

Streamlit là một công cụ mã nguồn mở giúp các nhà phát triển dễ dàng tạo ra các ứng dụng web dành cho khoa học dữ liệu và máy học chỉ với vài dòng mã Python. Được thiết kế để trực quan, nhanh chóng và hiệu quả, Streamlit giúp bạn biến các mô hình và tập dữ liệu phức tạp thành các ứng dụng tương tác một cách dễ dàng mà không cần kinh nghiệm về lập trình web.

2.5.1 Giới thiệu lý thuyết về Streamlit

Streamlit được xây dựng trên triết lý rằng các ứng dụng dành cho khoa học dữ liệu không nhất thiết phải phức tạp hoặc mất nhiều thời gian để phát triển. Bằng cách kết hợp giao diện trực quan và cú pháp đơn giản của Python, Streamlit giúp các nhà khoa học dữ liệu tập trung vào việc trình bày và phân tích dữ liệu thay vì phải làm việc với các công cụ lập trình web phức tạp.

Các tính năng chính của Streamlit:

- **Dễ dàng thiết lập:** Không cần sử dụng HTML, CSS hay JavaScript. Chỉ cần viết mã Python là đủ.
- **Tính tương tác cao:** Hỗ trợ các thành phần giao diện như biểu đồ, nút bấm, thanh trượt, hộp nhập liệu,..., giúp người dùng dễ dàng tương tác với dữ liệu.
- **Cập nhật thời gian thực:** Ứng dụng sẽ tự động cập nhật mỗi khi bạn thay đổi mã nguồn.
- **Hỗ trợ tích hợp:** Streamlit tương thích với nhiều thư viện phổ biến như Pandas, Matplotlib, Plotly, và TensorFlow.
- **Triển khai dễ dàng:** Các ứng dụng có thể được triển khai trên các nền tảng đám mây như Heroku, AWS hoặc Streamlit Community Cloud.

2.5.2 Cách sử dụng Streamlit

Bước 1: Cài đặt Streamlit

Để bắt đầu, bạn chỉ cần cài đặt Streamlit thông qua trình quản lý gói pip:

```
pip install streamlit
```

Bước 2: Viết ứng dụng

Tạo một tệp Python (ví dụ: *app.py*) và bắt đầu viết ứng dụng. Dưới đây là ví dụ đơn giản:

```
import streamlit as st

st.title("Ứng dụng Streamlit đầu tiên")
st.write("Chào mừng bạn đến với Streamlit!")
st.line_chart([1, 2, 3, 4, 5])
```

Bước 3: Chạy ứng dụng

Chạy ứng dụng bằng lệnh sau trong terminal:

```
streamlit run app.py
```

Ứng dụng sẽ được mở trong trình duyệt web.

Bước 4: Tích hợp các thành phần nâng cao

Streamlit cho phép bạn thêm các thành phần tương tác như thanh trượt và biểu đồ phức tạp. Ví dụ:

```
import numpy as np
import pandas as pd

st.sidebar.slider("Chọn một giá trị", 0, 100)
chart_data = pd.DataFrame(np.random.randn(20, 3), columns=["A", "B", "C"])
st.area_chart(chart_data)
```

2.5.3 Ứng dụng của Streamlit

- **Trực quan hóa dữ liệu:** Streamlit rất phù hợp để trình bày dữ liệu với các biểu đồ tương tác.
- **Triển khai mô hình máy học:** Streamlit hỗ trợ tích hợp các mô hình học máy để người dùng cuối có thể thử nghiệm.
- **Ứng dụng hỗ trợ quyết định:** Các công cụ dashboard dựa trên Streamlit có thể giúp doanh nghiệp đưa ra quyết định nhanh chóng dựa trên dữ liệu thời gian thực.

- **Cộng tác và chia sẻ:** Ứng dụng Streamlit có thể dễ dàng chia sẻ với các thành viên khác thông qua các liên kết trực tuyến.

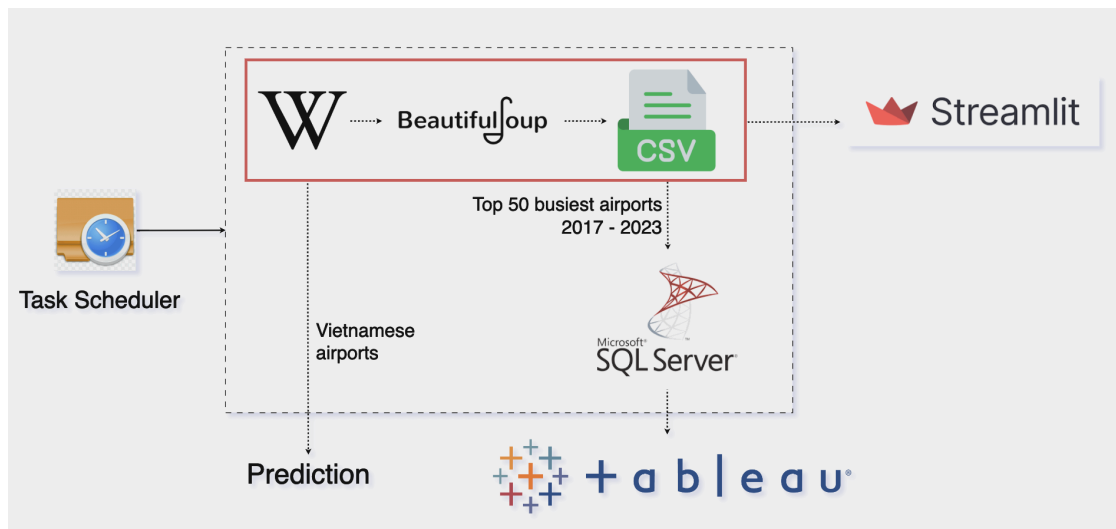
Ưu điểm của Streamlit:

- Đơn giản hóa quy trình phát triển ứng dụng khoa học dữ liệu.
- Tích hợp mạnh mẽ với các công cụ và thư viện Python.
- Cộng đồng người dùng đông đảo, tài liệu phong phú.

Với Streamlit, việc trình bày và phân tích dữ liệu trở nên dễ dàng hơn bao giờ hết, mở ra cánh cửa cho các nhà khoa học dữ liệu và lập trình viên không chuyên tham gia vào việc phát triển ứng dụng tương tác.

Chương 3

Quy trình



HÌNH 3.1: Quy trình dẫn xuất dữ liệu

3.1 Thu thập dữ liệu

Số liệu chi tiết về 50 sân bay đông đúc nhất trên toàn cầu dựa trên lưu lượng hành khách thu thập từ trang Wikipedia. https://en.wikipedia.org/wiki/List_of_busiest_airports_by_passenger_traffic

Ở đây cung cấp các thông tin chi tiết về số lượng hành khách tại mỗi sân bay, vị trí địa lý, mã IATA và ICAO, cùng với các số liệu thay đổi về xếp hạng, tỷ lệ phần trăm tăng trưởng và tổng số lượng hành khách (của 1 năm).

Để thực hiện quá trình thu thập dữ liệu, tiến hành sử dụng các công cụ BeautifulSoup và Pandas để cào dữ liệu từ trang web và xử lý thành định dạng có thể phân tích.

Quy trình cào dữ liệu cho 50 sân bay bận rộn nhất trên thế giới:

- **Gửi yêu cầu HTTP đến trang web Wikipedia:** Sử dụng thư viện requests để gửi một yêu cầu HTTP đến URL của trang Wikipedia chứa danh sách các sân bay bận rộn nhất. Phản hồi của yêu cầu này sẽ trả về mã HTML của trang, từ đó có thể tiếp tục phân tích.

```
# URL của trang Wikipedia
```

```
url = 'https://en.wikipedia.org/wiki  
↪ /List_of_busiest_airports_by_passenger_traffic'
```

```
# Gửi yêu cầu GET tới URL
```

```
page = requests.get(url)
```

- **Phân tích HTML bằng BeautifulSoup:** Sử dụng BeautifulSoup để phân tích mã HTML của trang web và tìm kiếm các bảng dữ liệu mà mình quan tâm. Trong trường hợp này, bảng thứ hai trên trang chứa thông tin về các sân bay bận rộn nhất theo lưu lượng hành khách, bao gồm các trường như thứ hạng, sân bay, vị trí, mã IATA/ICAO, số lượng hành khách và tỷ lệ thay đổi.

```
# Phân tích HTML với BeautifulSoup
```

```
soup = BeautifulSoup(page.text, 'html.parser')
```

- **Tạo DataFrame với Pandas:** Sau khi tìm được bảng dữ liệu mong muốn, tạo một DataFrame với Pandas để dễ dàng xử lý dữ liệu. Các cột trong bảng bao gồm: Thứ hạng, Tên sân bay, Vị trí, Quốc gia, Mã IATA/ICAO, Tổng số hành khách, Thay đổi thứ hạng, và Tỷ lệ thay đổi phần trăm.

```
# Tìm bảng dữ liệu (ở đây bảng thứ 2 chứa thông tin mong  
↪ muốn)
```

```
table = soup.find_all('table')[1]
```

```
# Đặt tên các cột cho DataFrame
```

```
col_table_titles = ['Rank', 'Airport', 'Location',
```

```
↪ 'Country', 'Code(IATA/ICAO)', 'Total_passengers',
```

```
↪ 'Rank_change', 'Percent_change']
```

```
df = pd.DataFrame(columns=col_table_titles)
```



```
# Lấy tất cả các hàng từ bảng
column_data = table.find_all('tr')
```

– **Làm sạch dữ liệu:** Trước khi đưa dữ liệu vào phân tích, thực hiện các bước làm sạch như:

- * Xử lý cột "Thứ hạng" để loại bỏ các ký tự không cần thiết.
- * Xử lý cột "Tổng số hành khách" và "Tỷ lệ phần trăm thay đổi" để chuyển đổi thành định dạng số.
- * Tách mã IATA và ICAO thành các cột riêng biệt để dễ phân tích.

```
# Vòng lặp để lấy dữ liệu từ các hàng
for row in column_data[1:]:
    row_data = row.find_all('td')
    individual_row_data = [data.text.strip() for data in
        ↪ row_data]
    # Làm sạch dữ liệu cột Rank
    individual_row_data[0] =
        ↪ clean_rank(individual_row_data[0])
    # Làm sạch dữ liệu cột 'Total passengers'
    individual_row_data[5] =
        ↪ clean_total_passengers(individual_row_data[5])
    # Làm sạch dữ liệu cột '% change'
    individual_row_data[7] =
        ↪ clean_percent_change(individual_row_data[7])
    # Làm sạch dữ liệu cột 'Rank change'
    individual_row_data[6] =
        ↪ clean_rank_change(individual_row_data[6])
    length = len(df)
    df.loc[length] = individual_row_data

# Tách cột 'Code(IATA/ICAO)' thành 2 cột riêng biệt
df[['IATA', 'ICAO']] =
    ↪ df['Code(IATA/ICAO)'].str.split('/', expand=True)

# Xóa cột gốc 'Code(IATA/ICAO)'
df = df.drop('Code(IATA/ICAO)', axis=1)
```

- **Lưu dữ liệu:** Dữ liệu sau khi làm sạch được xuất ra file CSV để lưu trữ, hoặc có thể được tải trực tiếp vào cơ sở dữ liệu SQL Server để quản lý và sử dụng trong các phân tích sau này.

Xuất DataFrame ra file CSV

```
df.to_csv(r'C:\Users\THIS
↪ PC\OneDrive\Desktop\data_engineerairports1.csv',
↪ index=False)
```

- **Lưu trữ trong SQL Server:** Kết nối với SQL Server để tạo bảng chứa dữ liệu đã được làm sạch. Bảng này lưu trữ thông tin về các sân bay và có thể được truy vấn hoặc tích hợp với các công cụ trực quan hóa như Tableau.

In kết quả để kiểm tra

```
conn = pyodbc.connect(
    "Driver={SQL Server};"
    "Server=LAPTOP-BVTT8GOU\SQLEXPRESS;"
    "Database= data project;"
    "Trusted_Connection=yes;"
)
cursor = conn.cursor()
```

```
CREATE TABLE demo(Rank int, Airport nvarchar(100) primary
↪ key, Location nvarchar(50), Country nvarchar(50),
↪ Total_passengers int, Rank_change int, Percent_change
↪ decimal, IATA nvarchar(50), ICAO nvarchar(50) );
```

- **Tự động cập nhật:** Mỗi lần cào dữ liệu mới thì để cập nhật trong SQL server ta sẽ tiến hành xóa dữ liệu cũ đã tồn tại và chèn dữ liệu mới vào.

Xóa dữ liệu hiện có trong bảng airport_auto_new3

```
cursor.execute("DELETE FROM airport_auto_new4")
```

Chèn dữ liệu từ DataFrame vào bảng SQL

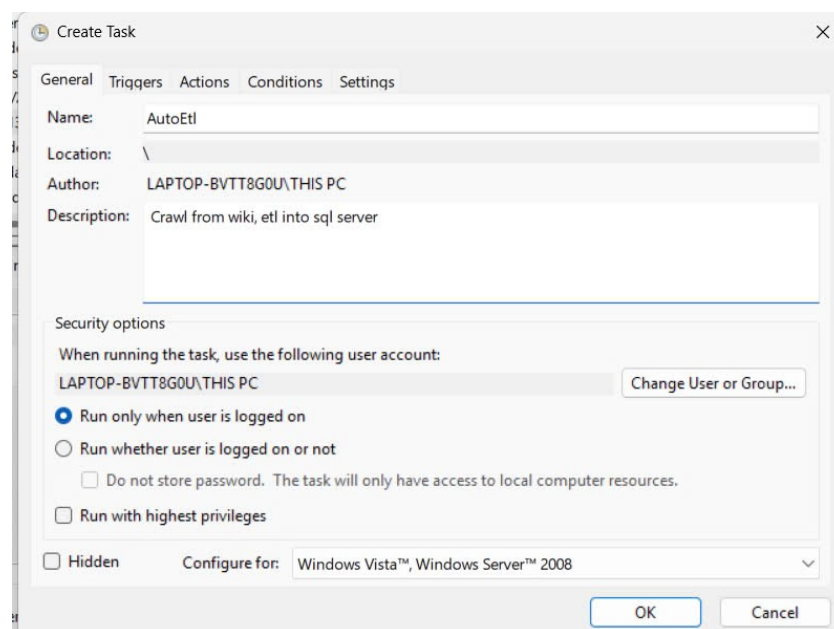
```
for row in df.itertuples():
    cursor.execute(
        """
        INSERT INTO demo (Rank, Airport, Location,
        ↪ Country, Total_passengers, Rank_change,
        ↪ Percent_change, IATA, ICAO)
```

```
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);
""",
row.Rank,
row.Airport,
row.Location,
row.Country,
row.Total_passengers,
row.Rank_change,
row.Percent_change,
row.IATA,
row.ICAO
)
```

3.2 Tự động hoá quy trình với Task Scheduler

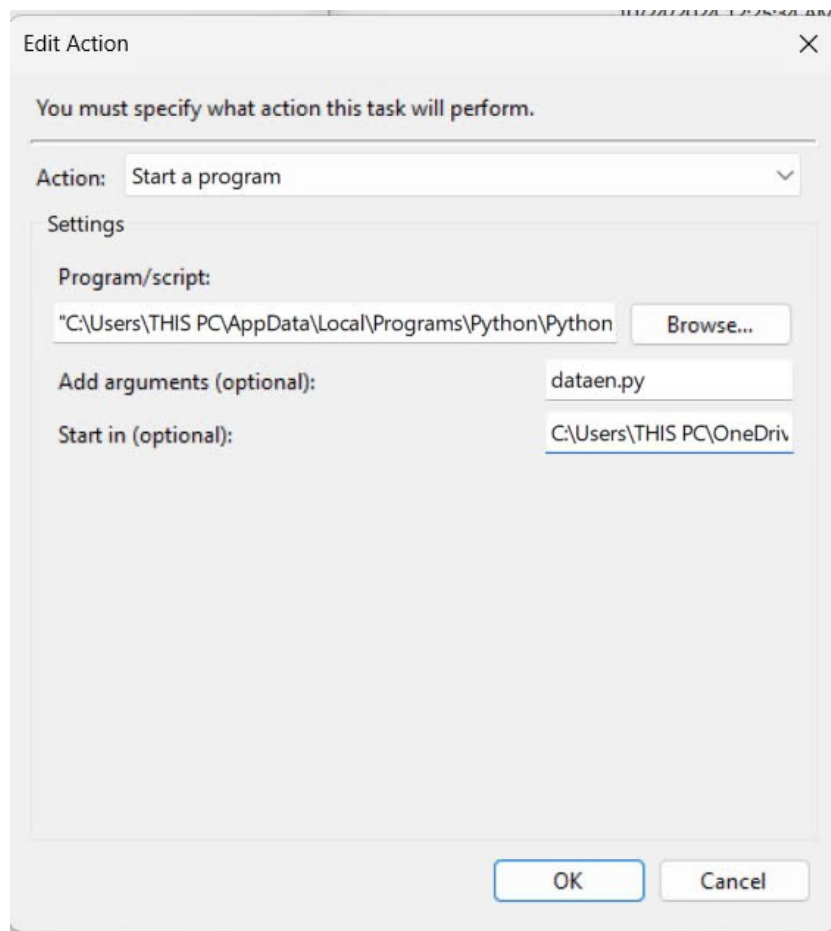
Bước 1: Tạo Task Mới

- Mở **Task Scheduler** trên Windows.
- Chọn **Create Basic Task** hoặc **Create Task** từ menu **Actions**.
- Đặt tên cho task, ví dụ: *Run dataen.py*.
- Nhấn **Next** để tiếp tục.



Bước 2: Chỉnh sửa Action

- Trong tab **Actions**, chọn **New** để thêm hành động mới.
- Chọn **Start a program**.
- Trong ô *Program/script*, duyệt và chọn tệp thực thi của Python 3.12. Ví dụ:
"C:\Users\THIS PC\AppData\Local\Programs\Python\Python312\python.exe"
(Thay đổi đường dẫn nếu Python được cài đặt ở nơi khác).
- Trong ô *Add arguments (optional)*, nhập đường dẫn đầy đủ đến file *dataen.py*, ví dụ: "C:\path\to\dataen.py"
- Start in: C:\Users\THIS PC\OneDrive\Desktop
- Nhấn **OK** để lưu hành động.

**Bước 3: Chỉnh sửa Trigger**

- Trong tab **Triggers**, chọn **New** để thêm trigger.

- Trong phần **Begin the task**, chọn **On a schedule**.
- Trong phần **Settings**, chọn **Monthly**.
- Trong phần **Months**, chọn các tháng: *January, February, March,...*
- Trong phần **Days**, chọn *Last* để chạy vào ngày cuối cùng của mỗi tháng.
- Bạn có thể điều chỉnh thời gian bắt đầu trong phần **Start**, ví dụ: ngày 13/10/2024 vào lúc 3:19:38 PM.
- Đánh dấu **Enabled** để kích hoạt trigger.
- Nhấn **OK** để lưu trigger.

The screenshot shows the 'Edit Trigger' dialog box with the following configuration:

- Begin the task:** On a schedule
- Settings:**
 - ☐ One time
 - ☐ Daily
 - ☐ Weekly
 - ☒ Monthly
- Start:** 10/13/2024 3:19:38 PM
- Months:** January, February, March...
- Days:** Last
- Advanced settings:**
 - ☐ Delay task for up to (random delay): 1 hour
 - ☐ Repeat task every: 1 hour for a duration of: 1 day
 - ☐ Stop all running tasks at end of repetition duration
 - ☐ Stop task if it runs longer than: 3 days
 - ☐ Expire: 10/23/2025 12:15:57 PM
 - ☐ Synchronize across time zones
 - ☒ Enabled
- Buttons:** OK, Cancel

Bước 4: Hoàn tất

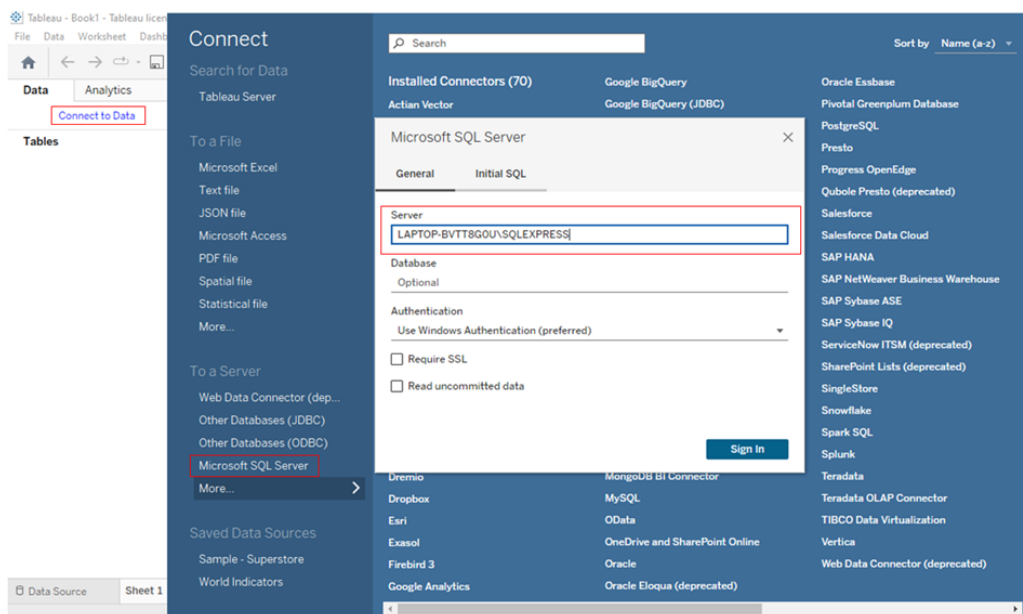
- Kiểm tra lại các cấu hình trong các tab **General**, **Actions**, **Triggers**, và **Settings**.
- Nhấn **OK** để hoàn tất việc tạo task.

Task sẽ tự động chạy file *dataen.py* với Python 3.12 theo lịch đã thiết lập.

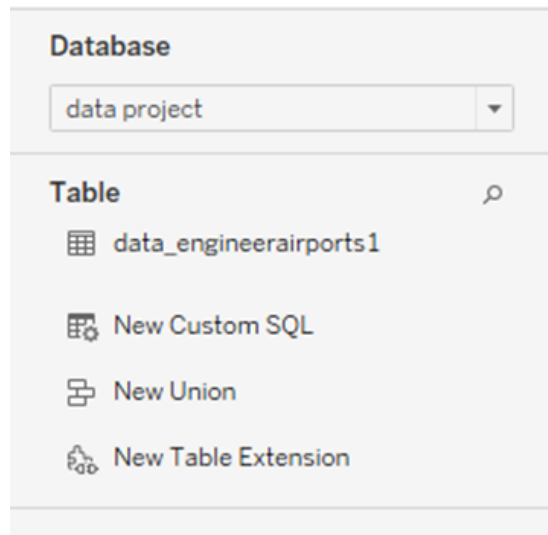
3.3 Trực quan hóa dữ liệu cho dữ liệu 50 sân bay bận rộn nhất

Các câu hỏi đặt ra cho phân tích dữ liệu 50 sân bay hoạt động tích cực nhất thế giới là:

- Nước nào có nhiều sân bay trong top 50 sân bay hoạt động tích cực nhất? Sân bay nào bận rộn nhất? Sân bay nào có phần trăm số lượng khách tăng lớn nhất?
- Số sân bay của Việt Nam lọt vào danh sách 50 sân bay bận rộn nhất là bao nhiêu? Sân bay đó tên gì?
- Số lượng khách hàng ghé thăm sân bay của Việt Nam chiếm thị phần bao nhiêu trong toàn thế giới?
- Việt Nam tăng bao nhiêu phần trăm số hành khách vào năm 2023?



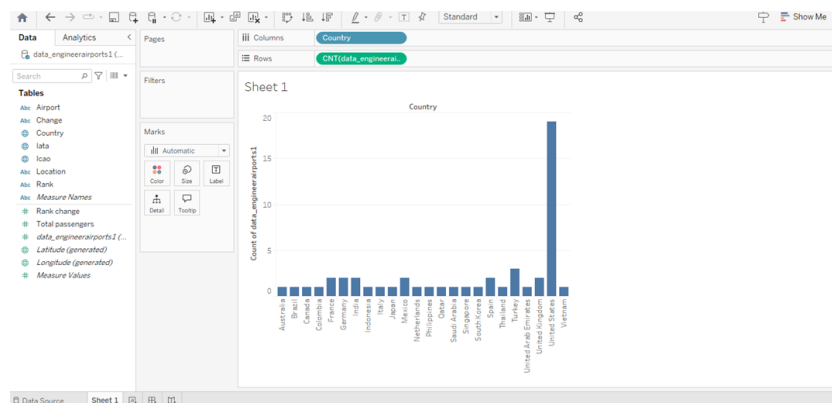
Để kết nối với SQL Server, chọn Connect to Data- Microsoft SQL Server- tên SQL server cần kết nối (LAPTOP-BVTT8G0U\SQLEXPRESS)



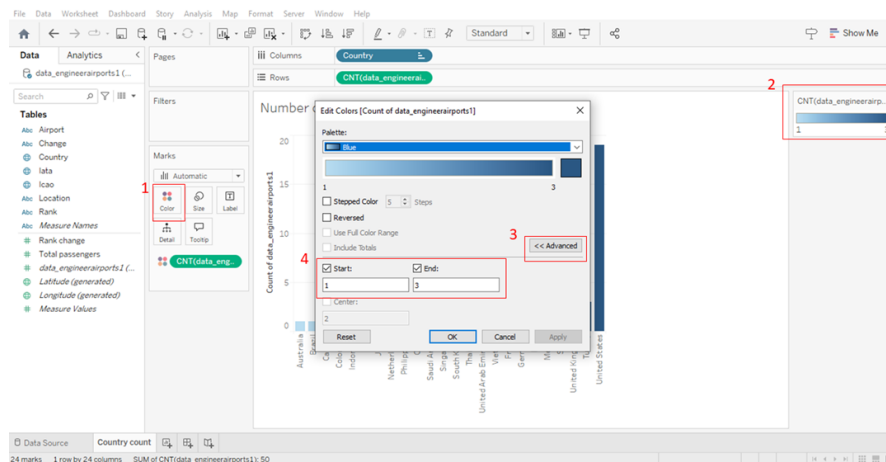
HÌNH 3.2: Chọn database đã tạo cho báo cáo này (data project)

Sau khi kết nối với database, tiếp theo thực hiện các phân tích và trực quan hóa dữ liệu.

1. Tạo Number of Airports in Top 50 by Country (sheet country count)



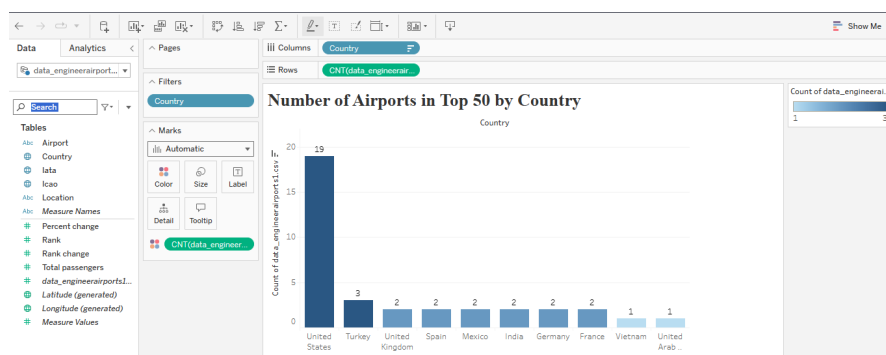
Kéo thả Columns: Country, Rows: data_engineerairports1 (với data_engineerairports1 được định nghĩa là một trường (field) được tạo để đếm số lượng bản ghi trong bảng data_engineerairports1).



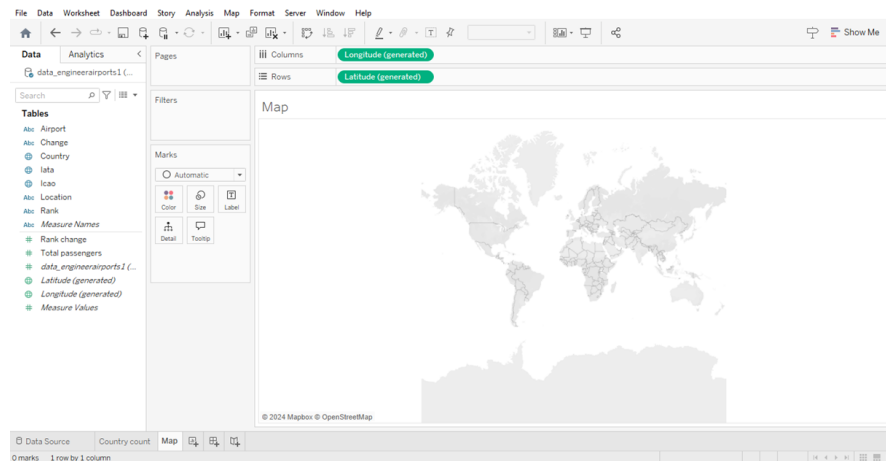
Để xếp loại bằng màu sắc: Kéo thả data_engineerairports1 vào color (1). Chọn mục edit colors (2). Sau đó chọn phần Advanced (3) và chọn số nhóm xếp loại (4).

Để sắp xếp các cột các cột theo độ lớn, điều chỉnh ở mục sort.

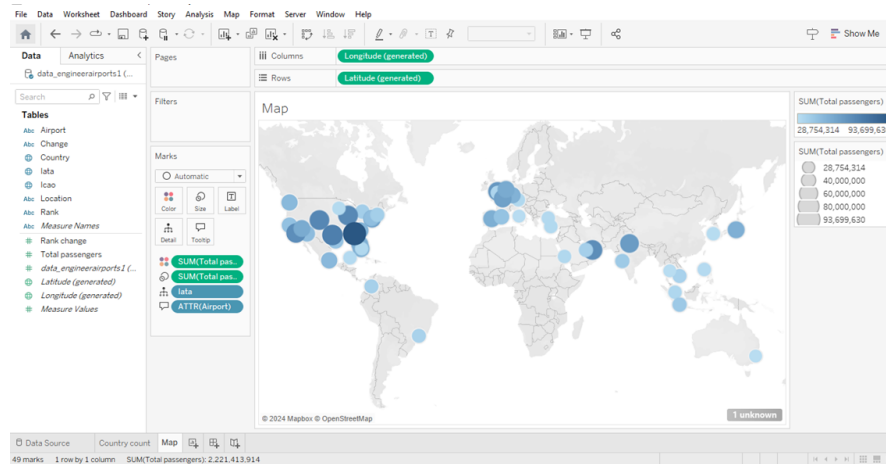
Để có thể nhìn thấy được sự chênh lệch số sân bay của Việt Nam và các nước khác lọt vào top 50, chọn Show mark label cho tất cả các cột.



2. Map



Kéo thả Columns: Longitude (Kinh độ) và Rows: Latitude (Vĩ độ).



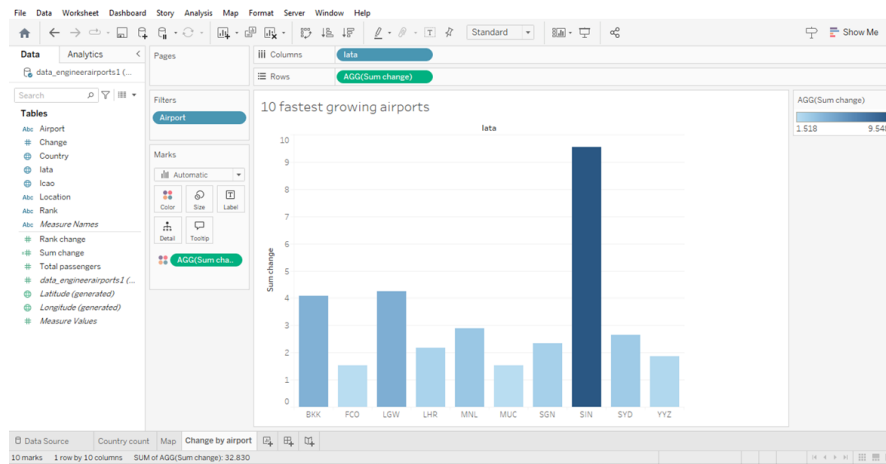
Sau khi có bản đồ, kéo thả **sum total passengers** vào color và size để biểu diễn số lượng hành khách khác nhau thông qua màu sắc và độ lớn.

Kéo thả **iata** vào detail hiển thị thông tin chi tiết của từng sân bay theo mã iata, mỗi sân bay được hiển thị riêng biệt trên bản đồ với vị trí tương ứng.

Kéo thả **airport** vào attr để tránh trùng lặp khi có nhiều tên sân bay giống nhau. Nếu chỉ có 1 tên sân bay, nó sẽ trả về đúng giá trị đó. Nếu có nhiều hơn 1 tên sân bay, nó sẽ trả về *.

Tiếp theo, điều chỉnh kích thước size của các điểm để dễ nhìn. Hiển thị mark label cho sân bay có nhiều khách ghé thăm nhất và sân bay của Việt Nam để có thể so sánh mức độ khác biệt.

3. 10 fastest growing airports (sheet: Change by airport)



Ban đầu, Change có định dạng là string. Do vậy để thực hiện được phép tính tổng thì cần chuyển định dạng string về số thập phân bằng cách chọn Change Data Type, chọn Number (decimal).

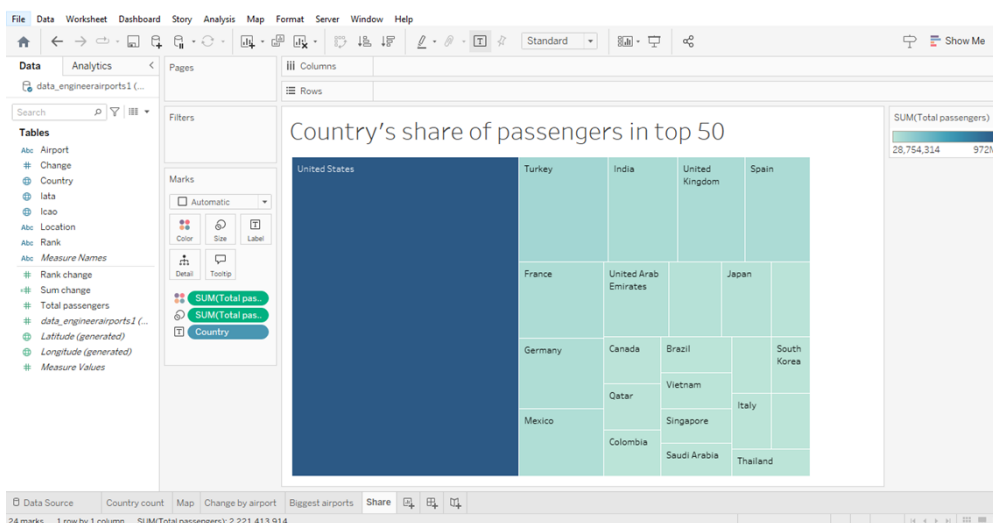
Sau đó tính tổng sự thay đổi bằng cách chọn Create Calculated field. Điền tên phép tính là Sum change, phép tính tổng bằng hàm SUM(Change).

Kéo thả Columns: Iata, Rows: Sum change. Kết quả sẽ hiện sự thay đổi của tất cả các sân bay thuộc mã Iata.

Để hiển thị top 10 sân bay phát triển nhanh nhất, kéo thả Airport vào Filters và set chọn hiển thị top 10.

Kéo thả Sum change vào Color để biểu diễn sự khác biệt màu sắc tương ứng độ thay đổi của sân bay.

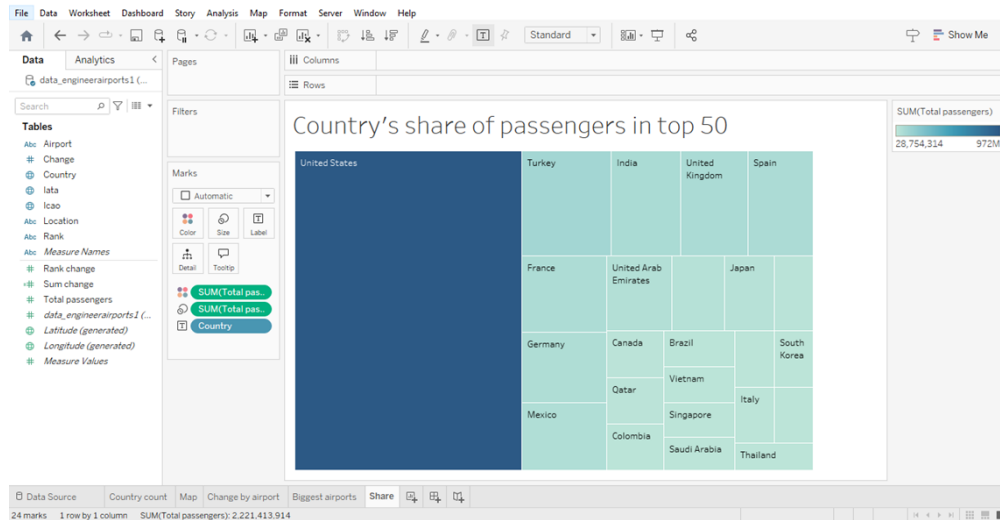
4. 10 busiest airports (sheet: Biggest airports)



Kéo thả Iata, Icao và Country vào Columns, Total passengers vào Rows. Kết quả sẽ hiện tổng tất cả các hành khách cho toàn bộ sân bay.

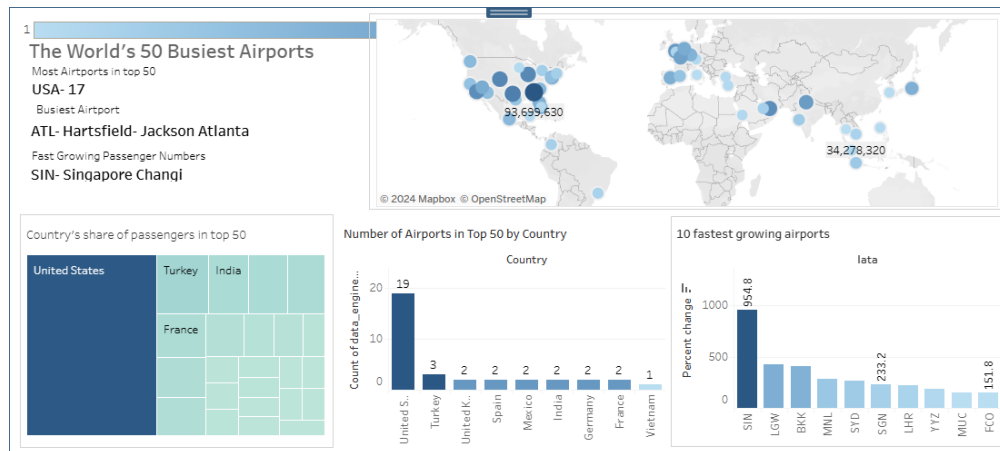
Để hiện 10 sân bay có lượng khách đông nhất (nhiều nhất) thì kéo thả Airport vào Filters, chọn hiển thị top 10.

5. Country's share of passengers in top 50 (sheet: Share)



Để số hành khách của 50 nước, kéo thả Total passengers vào color, size. Kéo thả Country vào Label để hiện tên nước.

6. Tạo Dashboard



Sắp xếp các thông tin và có được Dashboard hoàn chỉnh như trên.

Từ bản đồ có thể thấy, vào năm 2023, đa phần các sân bay (theo mã Iata) có lượng khách nhiều tập trung ở các khu vực Bắc Mỹ, Tây Âu, Trung Đông.

Một số ít đến trung bình tập trung ở vùng Tây Á và Đông Nam Á. Trong đó, Việt Nam thuộc nhóm các nước có độ lớn là 4 trên 5 nhóm; và nhỏ hơn 1/3 so với sân bay ATL- Hartsfield- Jackson Atlanta.

Các nước có sân bay lọt vào top 50 sân bay tích cực nhất bao gồm: Thailand, Australia, Italy, South Korea, Philippines, Saudi Arabia, Singapore, Vietnam, Brazil, Colombia, Qatar, Canada, Indonesia, Japan, Netherlands, United Arab Emirates, Mexico, Germany, France, Spain, United Kingdom, India, Turkey, United States- được mô tả ở biểu đồ Country's share of passengers in top 50. Trong đó, Mỹ (United States) chiếm thị phần nhiều nhất, chiếm khoảng gần 50% trên toàn thế giới. Việt Nam chỉ chiếm một thị phần phần nhỏ, khoảng 1/24 khi so với Mỹ.

Trong top 50 sân bay này, nhiều sân bay của Mỹ (United States) có số lượng khách đông nhất (19 sân bay), tiếp đến là Các Tiểu vương quốc Ả Rập Thống nhất (United Arab Emirates), Thổ Nhĩ Kỳ (Turkey), Vương quốc Anh (United Kingdom), Ấn Độ (India), Pháp (France)- được mô tả ở biểu đồ 10 busiest airports. Điều này phản ánh nhu cầu du lịch cao và khả năng cung cấp dịch vụ hàng không chất lượng tại những khu vực này. Với Việt Nam thì chỉ có 1 sân bay lọt vào trong danh sách này, là sân bay quốc tế Tân Sơn Nhất (Tan Son Nhat International Airport).

Sân bay SIN (Sân bay Changi Singapore) là sân bay phát triển nhanh chóng nhất. Tiếp theo là các sân bay được xếp theo thứ tự kế tiếp như LGW (Sân bay Gatwick London), BKK (Sân bay Suvarnabhumi Bangkok), MNL (Sân bay Ninoy Aquino Manila), LHR (Sân bay Heathrow London), YYZ (Sân bay Pearson Toronto), và MUC (Sân bay Munich). Sự phát triển nhanh chóng nhất ở sân bay Changi Singapore (SIN) không chỉ chứng tỏ sự thành công của Singapore trong việc thu hút du khách mà còn phản ánh chiến lược đầu tư vào cơ sở hạ tầng du lịch. So với sân bay Changi Singapore, sân bay quốc tế Tân Sơn Nhất chiếm 1/4. Điều này cũng phần nào cho thấy những tín hiệu tích cực từ một quốc gia nhỏ như Việt Nam.

3.4 Dự đoán số lượng khách ở tất cả sân bay của Việt Nam

Bên cạnh việc so sánh với các sân bay trên thế giới để nhìn nhận được vị thế của Việt Nam trên thị trường quốc tế. Để xây dựng các chiến lược cho sự phát triển ở năm tiếp theo cũng như phân bổ nguồn nhân lực trong nước cho phù hợp. Ở phần này sẽ thực hiện dự đoán số lượng khách ở tất cả các sân bay của Việt Nam cho năm tiếp theo (2024).

3.4.1 Thu thập dữ liệu

Dữ liệu tổng lượt khách trên tất cả sân bay ở Việt Nam từ năm 2012 đến 2023 được thu thập từ wikipedia https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_Vietnam. Để thu thập dữ liệu

- Định nghĩa URL: Định nghĩa biến url chứa địa chỉ của trang Wikipedia mà chúng ta muốn thu thập dữ liệu (địa chỉ liệt kê danh sách các sân bay bận rộn nhất ở Việt Nam).

```
url = "https://en.wikipedia.org/wiki/  
↳ List_of_the_busiest_airports_in_Vietnam"
```

- Gửi yêu cầu và nhận phản hồi để trích xuất dữ liệu: Gửi một yêu cầu GET đến URL đã định nghĩa để tải nội dung của trang. Phản hồi từ yêu cầu được lưu trong biến response. Sau đó, nội dung HTML của phản hồi được phân tích bằng BeautifulSoup, và tiến hành phân tích cú pháp HTML, tìm kiếm và trích xuất dữ liệu

```
# Gửi một yêu cầu GET đến website  
response = requests.get(url)  
soup = BeautifulSoup(response.text, 'html.parser')
```

- Tìm các bảng dữ liệu: Sử dụng phương thức find_all của BeautifulSoup để tìm tất cả các bảng trên trang có lớp witable. Kết quả là một danh sách các bảng, được lưu trong biến tables, mà trong đó sẽ tìm kiếm dữ liệu sân bay.

```
tables = soup.find_all('table', class_='witable')
```

- Định nghĩa hàm thu thập dữ liệu: Trên trang web có nhiều bảng, đều có lớp là Witable. Do vậy, để lấy được thông tin của bảng cần lấy, cần phải xét

các thành phần trong bảng. Điều kiện để xác định bảng cần lấy là bảng phải chứa đủ hết các thông tin trong các cột yêu cầu (cột mong muốn lấy).



Hàm định nghĩa scrape airport data sẽ nhận vào một danh sách các tên cột dữ liệu cần khai thác hay tiêu đề yêu cầu (required headers). Bên trong hàm, một danh sách rỗng data được khởi tạo để lưu trữ dữ liệu trích xuất. Hàm lặp qua từng bảng trong tables, lấy các tiêu đề cột và kiểm tra xem tất cả các tiêu đề yêu cầu có nằm trong tiêu đề của bảng không. Nếu có, nó sẽ lặp qua từng hàng trong bảng, bỏ qua hàng tiêu đề. Tiếp theo, tiến hành làm sạch dữ liệu, loại bỏ các số trong dấu ngoặc (biểu thị cho là tài liệu tham khảo). Chuyển đổi các giá trị 'null' hoặc 'unknown' thành '0'. Các cột đã xử lý được thêm vào danh sách data, và cuối cùng hàm trả về danh sách này.

```
# Tìm tất cả các bảng có class 'wikitable'
```

```
tables = soup.find_all('table', class_='wikitable')
```

```
# Hàm để lấy dữ liệu từ một bảng với tiêu đề đã chỉ định
```

```
def scrape_airport_data(required_headers):
```

```
    data = [] # Danh sách để lưu trữ dữ liệu
```

```
    for table in tables:
```

```
        # Lấy tiêu đề cột
```

```
        headers = [th.text.strip() for th in
```

```
            ↪ table.find_all('th')]
```

```

# Kiểm tra xem tất cả các tiêu đề yêu cầu có
→ trong tiêu đề cột hay không
if all(header in headers for header in
→ required_headers):
    # Lấy dữ liệu từ bảng
    rows = table.find_all('tr')[1:] # Bỏ qua hàng
    → tiêu đề
    for row in rows:
        cols = [td.text.strip() for td in
        → row.find_all('td')]
        # Chỉ thêm dữ liệu nếu số lượng cột khớp
        → với tiêu đề
        if len(cols) == len(required_headers):
            # Xử lý từng cột: loại bỏ số trong
            → ngoặc và chuyển đổi giá trị
            → null/unknown thành 0
            processed_cols = []
            for col in cols:
                # Loại bỏ số trong ngoặc
                col = re.sub(r'\[\d+\]', '',
                → col).strip()
                # Chuyển đổi 'null' hoặc 'unknown'
                → thành '0'
                if col.lower() in ['null',
                → 'unknown']:
                    col = '0'
                processed_cols.append(col) # Thêm
                → cột đã xử lý vào danh sách
            data.append(processed_cols)

return data

```

- Đặt tên cột cho hai bảng dữ liệu: Vì dữ liệu được tách thành 2 bảng ứng với 2 giai đoạn thu thập dữ liệu, gồm 2012 đến 2019 và 2020 đến 2023. Do vậy danh sách file thứ nhất (required headers 2020s) bao gồm các cột cho các năm từ 2020 đến 2023. File thứ hai (required headers 2012s) bao gồm các tiêu đề cho các năm từ 2012 đến 2019.

```
# Định nghĩa tiêu đề yêu cầu cho cả hai tập dữ liệu
required_headers_2020s = ['No.', 'Airport name',
    ↪ 'Province', 'City served', 'IATA', 'ICAO', '2020',
    ↪ '2021', '2022', '2023']
required_headers_2012s = ['No.', 'Airport name',
    ↪ 'Province', 'City served', 'IATA', 'ICAO', '2012[17]',
    ↪ '2013[17]', '2014[17]', '2015[18]', '2016[17]',
    ↪ '2017', '2018[19]', '2019[20][21]']
```

– Thu Thập Dữ Liệu

```
# Lấy dữ liệu cho những năm 2020 và lưu vào CSV
data_2020s = scrape_airport_data(required_headers_2020s)
if data_2020s:
    df_2020s = pd.DataFrame(data_2020s,
        ↪ columns=required_headers_2020s)
    df_2020s.to_csv('airports_data2020s.csv', index=False)
    print("Dữ liệu đã được lưu vào
        ↪ airports_data2020s.csv.")
else:
    print("Không có dữ liệu nào được tìm thấy với các tiêu
        ↪ đề cột yêu cầu cho 2020s.")

# Lấy dữ liệu cho những năm 2012 và lưu vào CSV
data_2012s = scrape_airport_data(required_headers_2012s)
if data_2012s:
    df_2012s = pd.DataFrame(data_2012s,
        ↪ columns=required_headers_2012s)
    df_2012s.to_csv('airports_data2012s.csv', index=False)
    print("Dữ liệu đã được lưu vào
        ↪ airports_data2012s.csv.")
else:
    print("Không có dữ liệu nào được tìm thấy với các tiêu
        ↪ đề cột yêu cầu cho 2012s.")
```

– Gộp các tệp CSV: kết nối hai tệp trên thành 1 file duy nhất.

```
# Hợp nhất hai tệp CSV
# Tải hai tệp CSV vào DataFrames
```



```

df_2020s = pd.read_csv('airports_data2020s.csv')
df_2012s = pd.read_csv('airports_data2012s.csv')

# Hợp nhất hai DataFrame trên một cột chung (ví dụ:
↪ 'No.')
merged_df = pd.merge(df_2012s, df_2020s, on='No.',
↪ how='outer')

```

– Làm sạch DataFrame đã gộp: bỏ các ký tự đặc biệt

```

# Loại bỏ các hậu tố khỏi các cột
merged_df.columns = [col.split('_')[0] if '_' in col else
↪ col for col in merged_df.columns]

```

```

# Loại bỏ số trong ngoặc khỏi tên cột
merged_df.columns = [re.sub(r'\(\d+\)', '', col).strip()
↪ for col in merged_df.columns]

```

```

# Loại bỏ các cột trùng lặp, giữ lại chỉ lần xuất hiện
↪ đầu tiên
merged_df = merged_df.loc[:,
↪ ~merged_df.columns.duplicated()]

```

```

# Giữ lại chỉ các cột đã chỉ định
columns_to_keep = [
    'No.', 'Airport name', 'Province', 'City served',
    ↪ 'IATA', 'ICAO',
    '2012', '2013', '2014', '2015', '2016', '2017',
    ↪ '2018', '2019',
    '2020', '2021', '2022', '2023'
]

```

```
merged_df = merged_df[columns_to_keep]
```

```

# Lưu DataFrame đã hợp nhất vào một tệp CSV mới
merged_df.to_csv('merged_airports_data.csv', index=False)

```

Kết quả sẽ như sau:

Dữ liệu đã được lưu vào airports_data2020s.csv.
Dữ liệu đã được lưu vào airports_data2012s.csv.
Dữ liệu đã được lưu vào merged_airports_data.csv.

```
[ ] 1 merged_df.head()
```

Airport name	Province	City served	IATA	ICAO	2012	2013	2014	2015	2016	2017	2018	2019	2020
Tan Son Nhat International Airport	Ho Chi Minh City	Ho Chi Minh City	SGN	VVTS	17,538,353	20,035,152	22,153,349	26,546,475	32,486,537	35,996,014	38,414,737	41,243,240	22,062,893
Noi Bai International Airport	Hanoi	Hanoi	HAN	VVNB	11,341,039	12,825,784	14,190,675	17,213,715	20,596,632	23,824,400	25,908,048	29,304,631	16,473,214
Da Nang International Airport	Da Nang	Da Nang	DAD	VVDN	3,090,877	4,376,775	4,989,687	6,722,587	8,783,429	10,801,927	13,229,663	15,543,598	Unknown
Cam Ranh International Airport	Khánh Hòa	Nha Trang	CXR	VVCR	1,095,776	1,509,212	2,062,494	2,722,833	4,858,362	6,500,000	8,250,000	9,747,172	3,305,057
Phu Quoc International Airport	Kiên Giang	Phù Quốc	PQC	VVPQ	493,434	685,036	1,002,750	1,467,043	2,278,814	3,000,000	3,200,000	3,700,205	Unknown

- Làm sạch dữ liệu: Ở các cột năm, định dạng số về số nguyên, đưa giá trị 'Unknown' và null về 0. Chuyển data về hướng tiện cho phân tích dự đoán.

```
# Chỉ định các cột để kiểm tra chuyển đổi sang kiểu số
numeric_columns = ['2012', '2013', '2014', '2015', '2016',
                    '2017',
                    '2018', '2019', '2020', '2021', '2022',
                    '2023']
```

```
# Loại bỏ các cột không cần thiết
```

```
columns_to_drop = ['No.', 'Province', 'City served',
                    'IATA', 'ICAO']
merged_df.drop(columns=columns_to_drop, inplace=True)
```

```
# Chuyển đổi các giá trị chuỗi có dấu phẩy sang kiểu số
```

```
→ nguyên và xử lý các giá trị không phải số
```

```
for col in numeric_columns:
    # Loại bỏ dấu phẩy, chuyển đổi sang kiểu số, và thiết
    → lập phân tích không hợp lệ thành NaN
    merged_df[col] =
    → pd.to_numeric(merged_df[col].str.replace(',', ''),
    → errors='coerce').fillna(0)
```

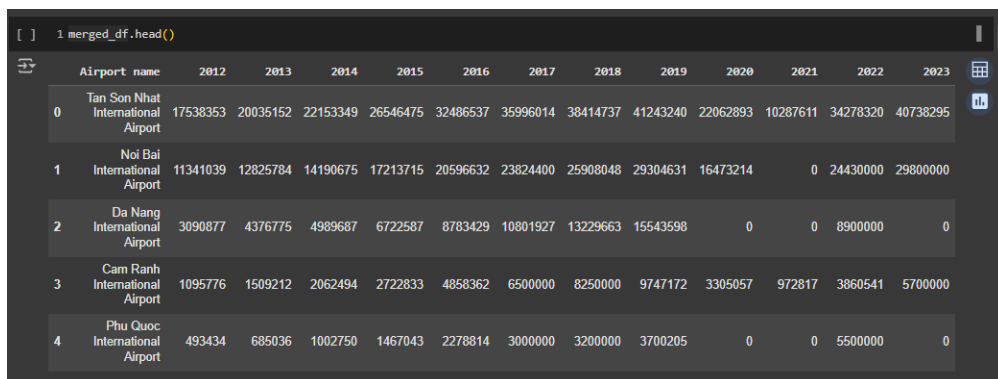
```
# Chuyển đổi các giá trị không phải số trong các cột đã
```

```
→ chỉ định thành 0 (đã được xử lý ở trên)
```

```
# Tùy chọn, chuyển các cột sang kiểu số nguyên sau khi
↪ thay thế NaN bằng 0
merged_df[numeric_columns] =
↪ merged_df[numeric_columns].astype(int)

# Hiển thị vài hàng đầu tiên của DataFrame đã được chỉnh
↪ sửa
print(merged_df.head())
```

Kết quả sẽ như sau



	Airport name	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
0	Tan Son Nhat International Airport	17538353	20035152	22153349	26546475	32486537	35996014	38414737	41243240	22062893	10287611	34278320	40738295
1	Noi Bai International Airport	11341039	12825784	14190675	17213715	20596632	23824400	25908048	29304631	16473214	0	24430000	29800000
2	Da Nang International Airport	3090877	4376775	4989687	6722587	8783429	10801927	13229663	15543598	0	0	8900000	0
3	Cam Ranh International Airport	1095776	1509212	2062494	2722833	4858362	6500000	8250000	9747172	3305057	972817	3860541	5700000
4	Phu Quoc International Airport	493434	685036	1002750	1467043	2278814	3000000	3200000	3700205	0	0	5500000	0

3.4.2 Dự đoán

Dùng thư viện LinearRegression của thư viện Scikit-learn (sklearn) để dự đoán từng sân bay của Việt Nam, với điều kiện sân bay có tối thiểu 2 điểm dữ liệu trong quá khứ.

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np

# Chuẩn bị một DataFrame để lưu trữ các dự đoán
predictions = []

# Vòng lặp qua từng sân bay để xây dựng mô hình và dự đoán
for airport in pivot_df.columns[1:]:
    # Chuẩn bị dữ liệu cho sân bay này
```

```

X = pivot_df['Year'].values.reshape(-1, 1)  # Năm là đặc
    ↪  trưng
y = pivot_df[airport].values  # Số hành khách là mục tiêu

# Xóa các giá trị NaN nếu có
if np.any(np.isnan(y)):
    valid_indices = ~np.isnan(y)
    X = X[valid_indices]
    y = y[valid_indices]

# Huấn luyện mô hình nếu có đủ dữ liệu
if len(X) >= 2:  # Đảm bảo có đủ dữ liệu để xây dựng mô
    ↪  hình
    model = LinearRegression()
    model.fit(X, y)

# Dự đoán cho năm tiếp theo
next_year = np.array([[pivot_df['Year'].max() + 1]])
predicted_passengers = model.predict(next_year)[0]

# Lưu kết quả dưới dạng số nguyên
predictions.append({'Airport name': airport,
                    'Predicted Passengers':
                        ↪  int(round(predicted_passengers))})
else:
    predictions.append({'Airport name': airport,
                        ↪  'Predicted Passengers': None})  # Không đủ dữ liệu

# Chuyển đổi các dự đoán thành DataFrame
predictions_df = pd.DataFrame(predictions)

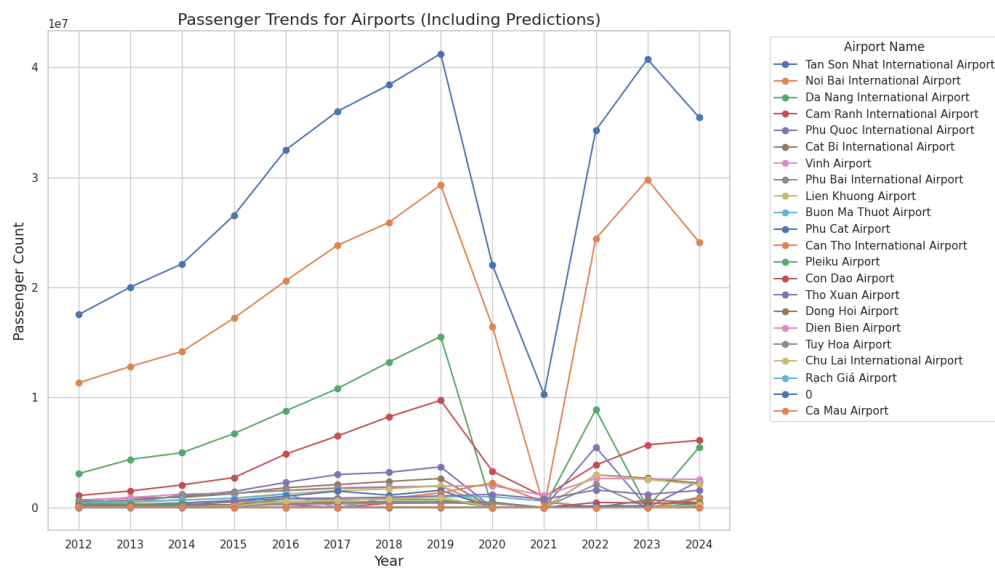
# Hiển thị các dự đoán
print(predictions_df)

```

Kết quả dự đoán như sau:

	Airport name	Predicted Passengers
0	Tan Son Nhat International Airport	35449899
1	Noi Bai International Airport	24113336
2	Cam Ranh International Airport	6113270
3	Da Nang International Airport	5480717
4	Vinh Airport	2568648
5	Phu Quoc International Airport	2414015
6	Cat Bi International Airport	2225046
7	Lien Khuong Airport	2095871
8	Tho Xuan Airport	1565522
9	Phu Bai International Airport	929769
10	Can Tho International Airport	802589
11	Dong Hoi Airport	458692
12	Con Dao Airport	399062
13	Buon Ma Thuot Airport	393873
14	Phu Cat Airport	338533
15	Chu Lai International Airport	249989
16	Tuy Hoa Airport	178673
17	Pleiku Airport	122683
18	Ca Mau Airport	-3138
19	Dien Bien Airport	-7234
20	Rach Gia Airport	-10310

Sau khi trực quan hoá dự đoán xu hướng của tất cả các sân bay. Kết quả cho thấy, sau đại dịch covid 19, 2021 bị ảnh hưởng, sụp giảm rất sâu. Bước sang năm 2022 và 2023 các sân bay hầu như đã phục hồi rất mạnh mẽ trở lại. Kết quả dự đoán ở năm 2024 cho thấy, hai sân bay lớn ở Việt Nam là Tan Son Nhat International Airport và Noi Bai International Airport lần lượt vẫn là hai sân bay dẫn đầu. Tuy nhiên, số lượng khách có thể giảm gần bằng năm 2022.



3.5 Ứng dụng Web-App: Top 50 Busiest Airports

Link github: <https://github.com/vuthithi24/List-of-busiest-airports-by-passenger-traffic.git>

Ứng dụng **Top 50 Busiest Airports** được xây dựng bằng Streamlit nhằm trực quan hóa dữ liệu về lượng hành khách tại các sân bay trên toàn cầu từ năm 2017 đến 2023. Ứng dụng cho phép người dùng tương tác và lọc dữ liệu theo năm và quốc gia, đồng thời cung cấp nhiều biểu đồ trực quan để giúp hiểu rõ xu hướng lưu lượng hành khách qua các năm.

3.5.1 Cấu trúc ứng dụng

Thư viện và tài nguyên:

- Sử dụng các thư viện: `streamlit`, `pandas`, `plotly.express`, và `math` để xử lý dữ liệu và tạo các biểu đồ.
- Dữ liệu được lưu trữ dưới dạng tệp CSV: `airport_data_2017_2023.csv`.

Cách thiết lập:

- **Tiêu đề ứng dụng:** Đặt tiêu đề `Top 50 busiest airports` và biểu tượng favicon (hình ảnh máy bay `airport.png`).
- **Tải dữ liệu:** Dữ liệu từ tệp CSV được xử lý và lưu trong bộ nhớ đệm thông qua decorator `@st.cache_data` nhằm tăng tốc độ tải.

3.5.2 Tính năng chính

Tùy chọn lọc dữ liệu (Sidebar):

- Người dùng có thể lọc dữ liệu bằng cách:
 - * Chọn khoảng thời gian từ **năm bắt đầu** đến **năm kết thúc** (từ 2017 đến 2023).
 - * Lọc các quốc gia mong muốn để phân tích (mặc định là Vietnam, United States, và United Kingdom).

Biểu đồ đường (Line Chart):

- Hiển thị xu hướng lưu lượng hành khách theo quốc gia qua các năm.
- Dữ liệu được nhóm theo năm và quốc gia, tổng số lượng hành khách được cộng dồn.

So sánh số liệu (Metric):

- So sánh số lượng hành khách tại các quốc gia được chọn giữa năm đầu tiên và năm cuối cùng trong khoảng thời gian lọc.
- Tính toán tỷ lệ tăng trưởng và hiển thị qua các chỉ số động.

Bản đồ Choropleth:

- Hiển thị lưu lượng hành khách tổng cộng theo quốc gia trên bản đồ thế giới.
- Màu sắc bản đồ thay đổi theo tổng số hành khách của từng quốc gia, sử dụng thang màu Rainbow.

3.5.3 Mã nguồn

Mã nguồn chi tiết được trình bày trong phần tiếp theo để minh họa đầy đủ cách thức hoạt động của ứng dụng.

```
import streamlit as st
import pandas as pd
import math
from pathlib import Path
import plotly.express as px

# Thiết lập tiêu đề và favicon
st.set_page_config(
    page_title='Top 50 busiest airports',
    page_icon='airport.png',
)

# Tải dữ liệu từ tệp CSV
@st.cache_data
def get_total_data():
```

```

DATA_FILENAME = Path(__file__).parent /
    ↪ 'data/airport_data_2017_2023.csv'
total_df = pd.read_csv(DATA_FILENAME)
total_df['Year'] = pd.to_numeric(total_df['Year'])
return total_df

# Gọi dữ liệu
total_df = get_total_data()

# Sidebar: Bộ lọc
st.sidebar.title("Filter Options")
st.sidebar.markdown("Use the filters below to refine the data
    ↪ displayed.")

min_value = total_df['Year'].min()
max_value = total_df['Year'].max()

from_year, to_year = st.sidebar.slider(
    'Select Year Range',
    min_value=int(min_value),
    max_value=int(max_value),
    value=[int(min_value), int(max_value)]
)

countries = total_df['Country'].unique()
selected_countries = st.sidebar.multiselect(
    'Select Countries',
    options=countries,
    default=['Vietnam', 'United States', 'United Kingdom']
)

# Lọc dữ liệu
filtered_total_df = total_df[
    (total_df['Country'].isin(selected_countries)) &
    (total_df['Year'] >= from_year) &

```



```

        (total_df['Year'] <= to_year)
    ]
    country_year_total_df = filtered_total_df.groupby(['Country',
        ↪ 'Year'])['Total_passengers'].sum().reset_index()

    # Tiêu đề chính
    st.title("Top 50 Busiest Airports")

    # Biểu đồ đường
    st.header('Passenger Traffic by Country over Time')
    st.line_chart(
        country_year_total_df,
        x='Year',
        y='Total_passengers',
        color='Country'
    )

    # So sánh số liệu giữa các quốc gia
    st.header(f'Passenger Count Comparison in {to_year}')
    cols = st.columns(4)
    for i, country in enumerate(selected_countries):
        col = cols[i % len(cols)]
        with col:
            first_total =
                ↪ country_year_total_df[(country_year_total_df['Country']
                ↪ == country) & (country_year_total_df['Year'] ==
                ↪ from_year)]['Total_passengers'].sum() / 1e6
            last_total =
                ↪ country_year_total_df[(country_year_total_df['Country']
                ↪ == country) & (country_year_total_df['Year'] ==
                ↪ to_year)]['Total_passengers'].sum() / 1e6

            if math.isnan(first_total) or first_total == 0:
                growth = 'n/a'
                delta_color = 'off'

```

```

        else:
            growth = f'{last_total / first_total:,.2f}x'
            delta_color = 'normal'

    st.metric(
        label=f'{country} Passengers',
        value=f'{last_total:,.0f}M',
        delta=growth,
        delta_color=delta_color
    )

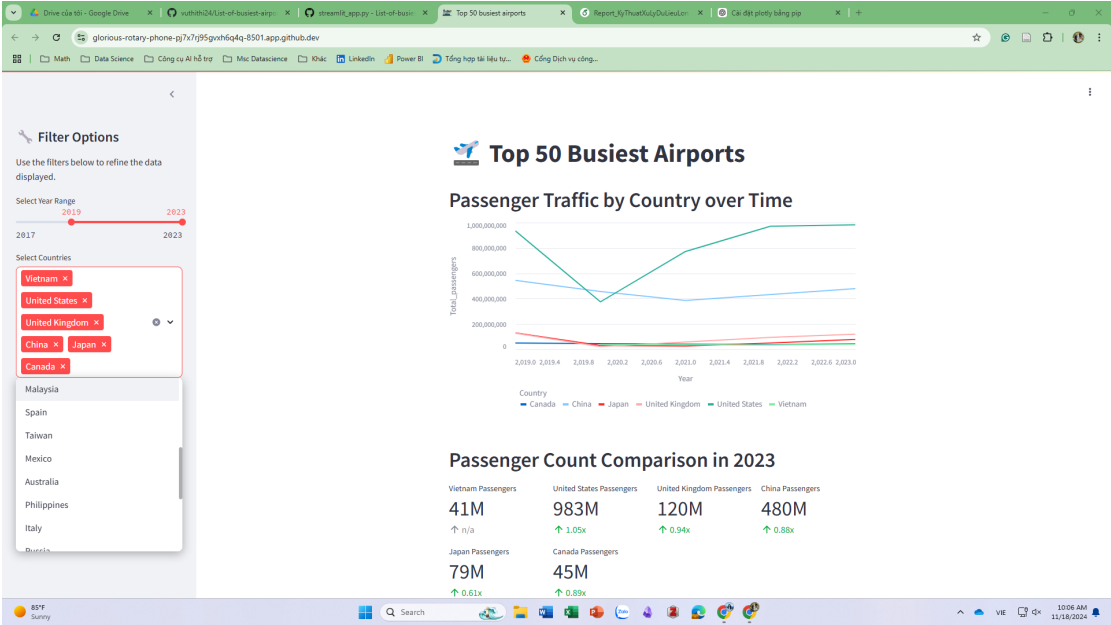
# Bản đồ Choropleth
st.header("Choropleth Map of Passenger Traffic")
choropleth_data =
    → country_year_total_df.groupby('Country')['Total_passengers']
    → .sum().reset_index()

fig = px.choropleth(
    choropleth_data,
    locations="Country",
    locationmode="country names",
    color="Total_passengers",
    hover_name="Country",
    title="Total Passenger Traffic by Country",
    color_continuous_scale="Rainbow"
)
st.plotly_chart(fig)

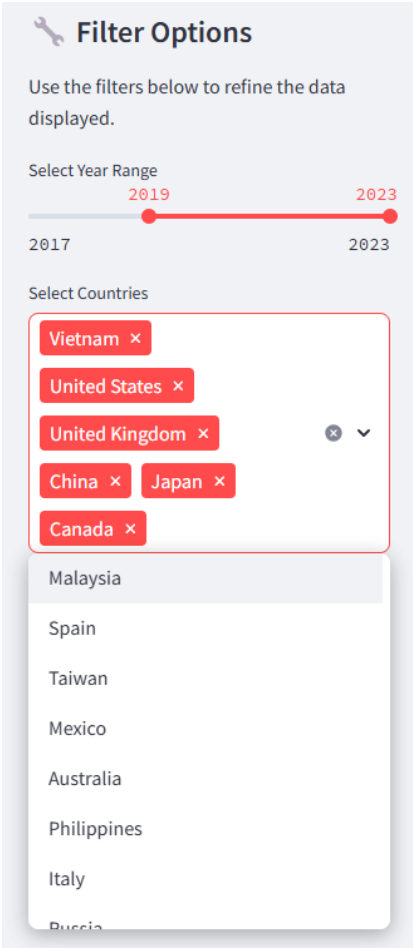
```

3.5.4 Kết quả nhận được

GIAO DIỆN CHÍNH



CỬA SỐ ĐỂ CHỌN THỜI GIAN VÀ QUỐC GIA

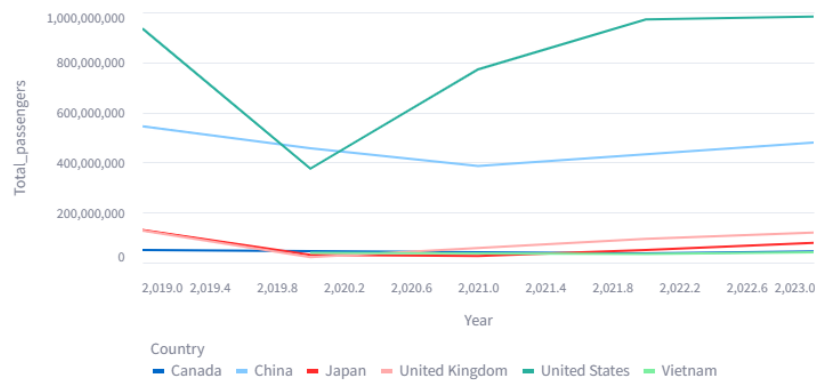


BIỂU ĐỒ LINE CHART VÀ SỐ LIỆU SO SÁNH CỦA TỪNG QUỐC GIA



Top 50 Busiest Airports

Passenger Traffic by Country over Time



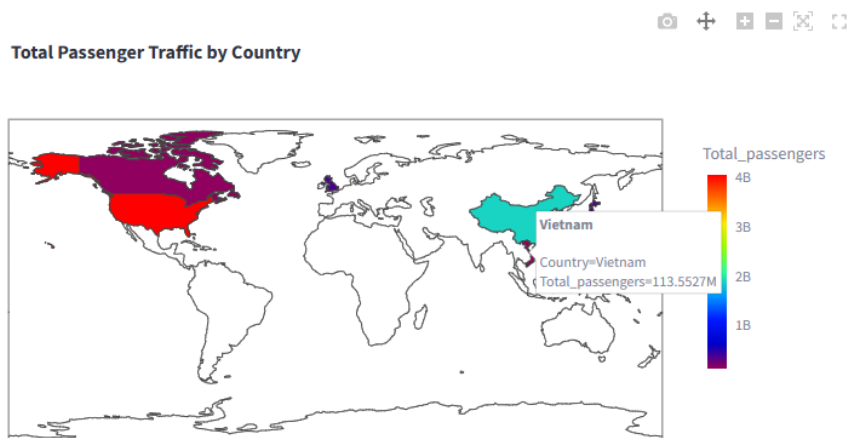
Passenger Count Comparison in 2023

Vietnam Passengers	United States Passengers	United Kingdom Passengers	China Passengers
41M	983M	120M	480M
↑ n/a	↑ 1.05x	↑ 0.94x	↑ 0.88x
Japan Passengers	Canada Passengers		
79M	45M		
↑ 0.61x	↑ 0.89x		

BẢN ĐỒ CHOROPLETH



Choropleth Map of Passenger Traffic



3.5.5 Nhận xét

Dựa trên bảng so sánh, Web-App (Streamlit) và Tableau đều có những ưu điểm riêng phù hợp với các mục đích sử dụng khác nhau. Streamlit lý tưởng cho các nhà phát triển và nhà phân tích dữ liệu có kinh nghiệm lập trình, nhờ khả năng tích hợp linh hoạt với các thư viện Python và khả năng mở rộng mạnh mẽ. Tuy nhiên, nó yêu cầu kiến thức kỹ thuật cao hơn để thiết kế giao diện và xử lý dữ liệu. Ngoài ra, Streamlit đòi hỏi người dùng phải chủ động cập nhật dữ liệu, do không tích hợp các công cụ tự động hóa như Tableau.

Ngược lại, Tableau là công cụ mạnh mẽ dành cho người dùng không chuyên về lập trình, với giao diện thân thiện, trực quan và khả năng xử lý dữ liệu lớn hiệu quả. Tableau hỗ trợ tự động hóa dữ liệu thông qua tính năng Task Scheduler, giúp cập nhật báo cáo và bảng điều khiển một cách liên tục mà không cần can thiệp thủ công. Tuy nhiên, Tableau đòi hỏi chi phí sử dụng cao và phụ thuộc vào nền tảng riêng để triển khai. Nhìn chung, Streamlit phù hợp cho các dự án tùy chỉnh, miễn phí, trong khi Tableau thích hợp cho doanh nghiệp cần công cụ tự động hóa và trực quan hóa dữ liệu chuyên nghiệp.

So sánh giữa Web-App (Streamlit) và Tableau

Tiêu chí	Web-App (Streamlit)	Tableau
Khả năng lập trình	Yêu cầu kiến thức Python, dễ dàng tùy chỉnh bằng cách viết mã.	Không yêu cầu lập trình, giao diện kéo thả thân thiện với người dùng.
Tự động hóa dữ liệu	Không hỗ trợ tự động hóa, người dùng cần chủ động cập nhật dữ liệu.	Hỗ trợ tự động hóa thông qua Task Scheduler , cập nhật dữ liệu tự động theo lịch trình.
Mức độ mở rộng	Linh hoạt tích hợp với nhiều thư viện Python khác như Pandas, NumPy, Plotly.	Mở rộng qua các kết nối dữ liệu mặc định và tích hợp với các nền tảng như Salesforce, Google Sheets.
Hiệu suất xử lý dữ liệu	Tốt với tập dữ liệu nhỏ đến vừa; hiệu suất có thể phụ thuộc vào môi trường triển khai.	Tối ưu hóa cho xử lý dữ liệu lớn và kết nối trực tiếp với cơ sở dữ liệu.
Chi phí sử dụng	Miễn phí (mã nguồn mở).	Yêu cầu giấy phép, chi phí sử dụng cao hơn.

Trực quan hóa	Cung cấp các biểu đồ tùy chỉnh thông qua các thư viện Python như Plotly.	Bộ công cụ trực quan hóa phong phú, có sẵn nhiều biểu đồ và bảng điều khiển để dùng.
Tính linh hoạt	Linh hoạt trong thiết kế và tính năng, phù hợp với dự án tùy chỉnh hoặc nghiên cứu.	Tập trung vào doanh nghiệp, phù hợp cho báo cáo nhanh và chia sẻ thông tin.
Khả năng triển khai	Dễ dàng triển khai trên nhiều nền tảng như Streamlit Cloud, Heroku hoặc máy chủ cục bộ.	Triển khai qua Tableau Desktop, Tableau Public hoặc tích hợp doanh nghiệp.