**Tài liệu dự án**

# **Chương 1: Design**

Lấy dữ liệu về các chuyến bay trong năm 2015 để xây dựng 1 ETL Pipeline và visualize liệu rằng hãng máy bay nào sẽ có thời gian delay ít nhất, ….

Nguồn dữ liệu: [2015 Flight Delays and Cancellations](https://www.kaggle.com/datasets/usdot/flight-delays)

Các bước thực hiện:

1. **Trích xuất dữ liệu từ file CSV**
2. **Biến đổi, làm sạch dữ liệu**
3. **Lưu dữ liệu sạch vào data-warehouse**
4. **Visualize dữ liệu trong data-warehouse**
5. **Đẩy dự án lên Github**

Công cụ thực hiện:

1. **Python và thư viện Pandas**: Trích xuất dữ liệu và biến đổi dữ liệu
2. **DuckDB:** Lưu dữ liệu vào data-warehouse của dự án
3. **PowerBI**: Visualize dữ liệu
4. **Docker**: Đóng gói để người khác dễ sử dụng
5. **Github:** Lưu dự án

## **Hiểu dữ liệu**

### **Data source**

Cục Thống kê Giao thông của Bộ Giao thông Vận tải Hoa Kỳ (DOT) theo dõi hiệu suất đúng giờ của các chuyến bay trong nước được vận hành bởi các hãng hàng không lớn. Dữ liệu nguồn là file CSV bao gồm thông tin tóm tắt về số lượng các chuyến bay đúng hạn, bị trì hoãn, bị hủy và chuyển hướng được công bố trong Báo cáo người tiêu dùng về du lịch hàng không và trong bộ dữ liệu 2015 Flight Delays and Cancellations.

### **Schema**

* **airlines.csv**
* **IATA\_CODE:** Mã định danh 2 ký tự đại diện hãng hàng không
* **AIRLINE:** Tên hãng hàng không
* **airports.csv**
* **IATA\_CODE:** Mã định danh 3 ký tự đại diện sân bay
* **AIRPORT:** Tên sân bay
* **CITY:** Tên thành phố
* **STATE:** Tên bang
* **COUNTRY:** Tên quốc gia
* **LATITUDE:** Vĩ độ
* **LONGITUDE:** Kinh độ
* **flights.csv**
* **YEAR:** Năm chuyến bay
* **MONTH:** Tháng chuyến bay
* **DAY:** Ngày chuyến bay
* **DAY\_OF\_WEEK:** Ngày chuyến bay là THỨ mấy
* **AIRLINE:** Mã định danh 2 ký tự đại diện hãng hàng không
* **FLIGHT\_NUMBER:** Số hiệu chuyến bay (Nhận dạng chuyến bay)
* **TAIL\_NUMBER:** Định danh máy bay
* **ORIGIN\_AIRPORT:** Sân bay khởi hành
* **DESTINATION\_AIRPORT:** Sân bay đích
* **SCHEDULED\_DEPARTURE:** Thời điểm (giờ, phút) khởi hành theo kế hoạch
  + **WHEELS\_OFF:** Thời điểm (giờ, phút) bánh xe máy bay rời khỏi mặt đất
  + **TAXI\_OUT:** Số phút từ lúc máy bay rời khỏi cổng ở sân bay khởi hành đến khi bánh máy bay rời mặt đất
* **DEPARTURE\_TIME:** Thời điểm (giờ, phút) khởi hành = WHEELS\_OFF - TAXI\_OUT
* **DEPARTURE\_DELAY:** Số phút sớm/trễ so với thời gian khởi hành dự kiến
* **SCHEDULED\_TIME:** Số phút dự kiến cho toàn bộ chuyến bay
  + **AIR\_TIME:** Số phút máy bay trên không
* **TAXI\_IN:** Số phút từ lúc bánh xe máy bay tiếp xúc mặt đất đến khi đến cổng tại sân bay đích
* **ELAPSED\_TIME:** Số phút tính từ lúc rời khỏi cổng + bay + đến cổng đích = AIR\_TIME + TAXI\_IN + TAXI\_OUT
* **DISTANCE:** Khoảng cách (km) giữa 2 sân bay
* **WHEELS\_ON:** Thời điểm (giờ, phút) bánh máy bay tiếp xúc với đường băng
* **SCHEDULED\_ARRIVAL:** Thời điểm (giờ, phút) dự kiến đến sân bay đích
* **ARRIVAL\_TIME:** Thời điểm (giờ, phút) máy bay có mặt ở sân bay đích = WHELLS\_ON + TAXI\_IN
* **ARRIVAL\_DELAY:** Số phút sớm/trễ so với thời gian đến dự kiến = ARRIVAL\_TIME - SCHEDULED\_ARRIVAL
* **DIVERTED:** Biến nhị phân cho biết máy bay có hạ cánh tại sân bay ngoài kế hoạch không
* **CANCELLED:** Biến nhị phân cho biết chuyến bay có bị hủy không (1 = hủy)
* **CANCELLATION\_REASON:** Lý do chuyến bay bị hủy
* A: Do hãng hàng không
* B: Do thời tiết
* C: Do vấn đề từ hệ thống điều hành bay quốc gia
* D: Vì lý do an ninh
* **AIR\_SYSTEM\_DELAY:** Số phút bị trễ do hệ thống điều hành bay quốc gia
* **SECURITY\_DELAY:** Số phút bị trễ vì lý do an ninh
* **AIRLINE\_DELAY:** Số phút bị trễ vì lý do hãng hàng không
* **LATE\_AIRCRAFT\_DELAY:** Số phút bị trễ do máy bay từ chuyến trước về muộn → chuyến bay hiện tại không thể khởi hành đúng giờ
* **WEATHER\_DELAY:** Số phút bị trễ do thời tiết

### **Lý do chọn công cụ**

#### **Python và thư viện Pandas**

* Vì chạy trên bộ nhớ RAM, nên rất hiệu quả với dữ liệu vừa và nhỏ (thường < vài triệu dòng) → Tốc độ xử lý nhanh hơn Pyspark đối với dữ liệu nhỏ
* Pyspark xử lý dữ liệu lớn bằng cách phân tán công việc qua nhiều nút tính toán (cluster computing) → khởi tạo session và thực thi lệnh sẽ lâu hơn Pandas với dữ liệu nhỏ
* Pandas có cú pháp đơn giản hơn PySpark
* Pandas dễ dùng trong môi trường học tập (máy đơn lẻ) còn PySpark cần môi trường phức tạp hơn (Spark Cluster,...)

#### **DuckDB**

* DuckDB cung cấp khả năng lưu trữ và truy vấn dữ liệu hiệu suất cao ngay trên file hệ thống (hoạt động như 1 thư viện trong Python, lưu toàn bộ dữ liệu trong 1 file .db), không cần cài đặt server phức tạp như PostgreSQL. Với kiến trúc lưu trữ theo cột và khả năng tích hợp trực tiếp với Pandas và file CSV, DuckDB giúp đơn giản hóa ETL Pipeline, đồng thời đáp ứng tốt các truy vấn phân tích dữ liệu lớn (OLAP)

#### **Power BI**

* Giao diện thân thiện, khả năng tích hợp tốt với nhiều nguồn dữ liệu và hỗ trợ biểu đồ nhiều phân tích tương tác. Power BI cho phép xây dựng dashboard động dễ dàng, giúp người dùng hiểu nhanh xu hướng delay, số lượng chuyến bay và lý do hủy chuyến bay theo thời gian, hãng bay hoặc sân bay,...

#### **Github**

* Hỗ trợ chia sẻ mã nguồn, tài liệu hướng dẫn và trình bày sản phẩm một cách chuyên nghiệp. Đây là công cụ không thể thiếu trong các dự án thực tế hoặc học thuật, đồng thời giúp thể hiện kỹ năng kỹ thuật trong hồ sơ cá nhân.

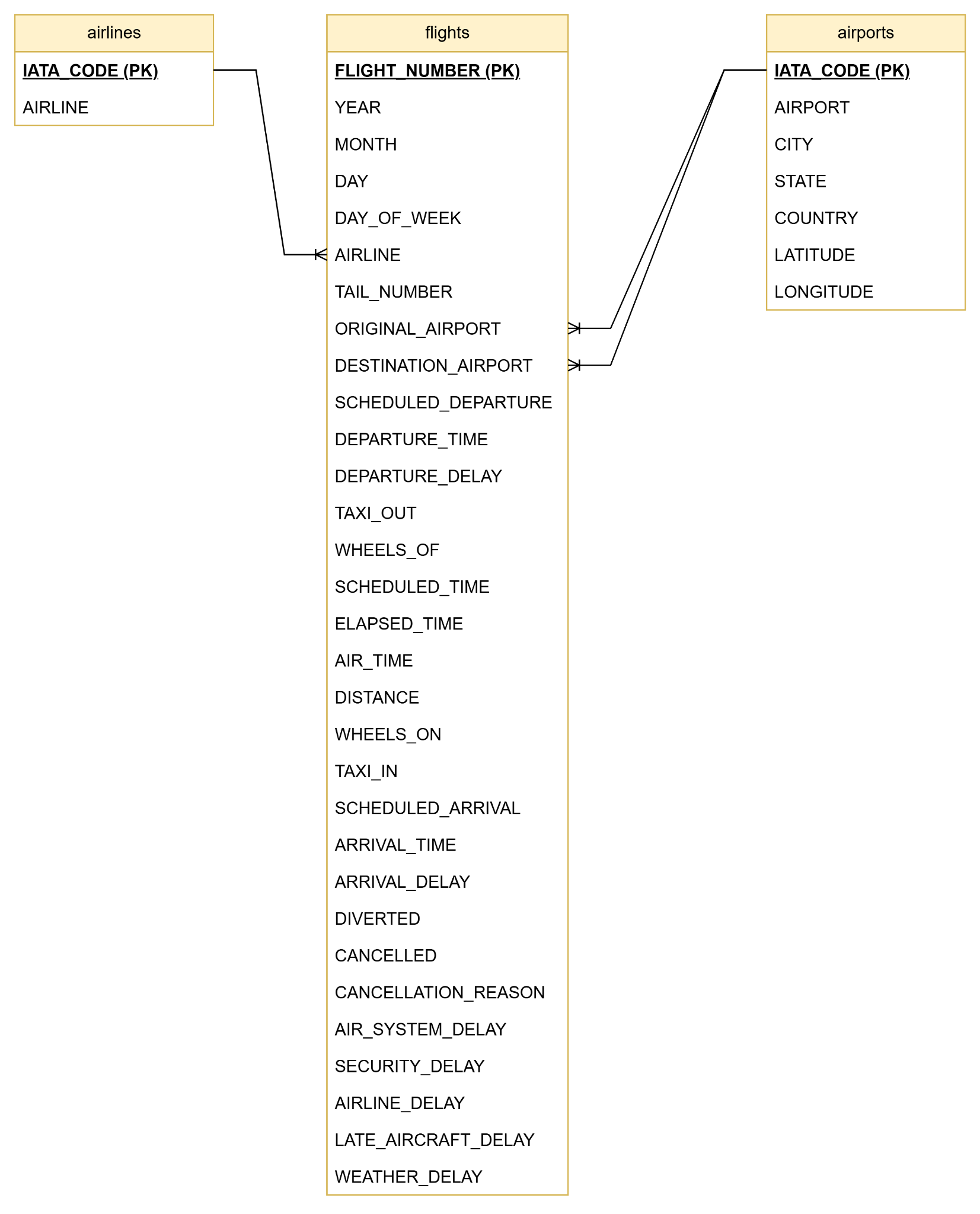
#### **Docker**

* Giúp đóng gói toàn bộ môi trường dự án ETL vào 1 container, đảm bảo tính nhất quán giữa các máy phát triển, triển khai và chia sẻ. Với docker, người dùng không cần phải cài đặt thù công Python, thư viện hay cấu hình môi trường – chỉ cần chạy 1 lệnh là toàn bộ ETL pipeline có thể thực thi tự động. Điều này đặc biệt hữu ích khi làm việc nhóm, triển khai trên cloud hoặc tích hợp vào các hệ thống CI/CD.

### **Xác định các bài toán**

* **1. So s ánh hiệu suất các hãng hàng không: Hãng hàng không nào có tỷ lệ delay thấp nhất trong năm 2015?**
  + Mục tiêu: Hỗ trợ hành khách lựa chọn hãng bay tốt hơn hoặc giúp hãng bay cải thiện dịch vụ.
* **2. Phân tích nguyên nhân delay: Đâu là lý do chính gây ra các chuyến bay bị trễ?**
  + Mục tiêu: Giúp cơ quan quản lý hoặc hãng bay xác định đâu là nút thắt cần giải quyết (thời tiết, an ninh, hệ thống,...).
* **3. Phân tích theo thời gian: Các tháng hoặc ngày trong tuần nào có xu hướng delay nhiều hơn?**
  + Mục tiêu: Lập kế hoạch tăng nhân sự, tối ưu vận hành tại các khung thời gian cao điểm.
* **4. Hiểu về tình trạng hủy chuyến: Có bao nhiêu chuyến bị hủy và lý do phổ biến nhất là gì?**
  + Mục tiêu: Đánh giá tác động của điều kiện thời tiết, an ninh hay nội bộ hãng lên việc hủy chuyến.
* **5. Xác định sân bay có hiệu suất kém: Sân bay nào thường xuyên bị delay hoặc có số lượng lớn chuyến bay bị hủy?**
  + Mục tiêu: Giúp cơ quan vận hành sân bay cải thiện quy trình hoặc tăng đầu tư hạ tầng.
* **6. Khả năng mở rộng báo cáo**
  + Mục tiêu: Thiết kế pipeline ETL linh hoạt để dễ dàng tích hợp thêm dữ liệu các năm sau (2016, 2017,…) hoặc áp dụng vào quốc gia khác.

### **Thiết kế ERD Diagram lưu trữ dữ liệu từ 3 file CSV**



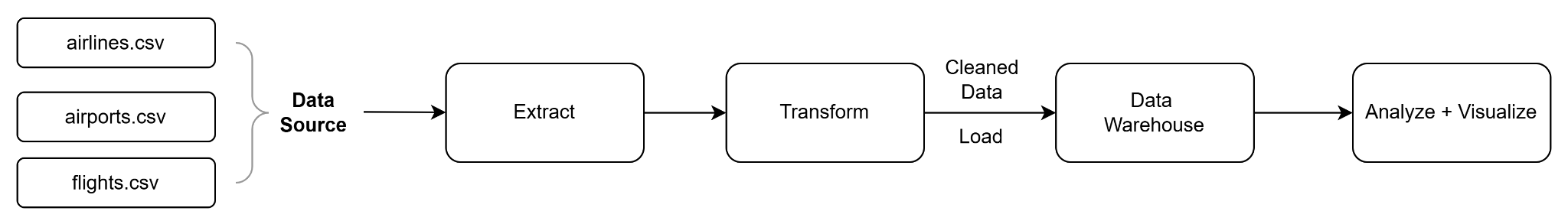
### **Trích xuất dữ liệu gì để giải quyết bài toán?**

* Bảng airlines
  + AIRLINE
* Bảng flights
  + DEPARTURE\_DELAY
  + ARRIVAL\_DELAY
  + AIRLINE
  + AIRLINE\_DELAY
  + AIR\_SYSTEM\_DELAY
  + SECURITY\_DELAY
  + LATE\_AIRCRAFT\_DELAY
  + WEATHER\_DELAY
  + MONTH
  + DAY\_OF\_WEEK
  + CANCELLED
  + CANCELLATION\_REASON
  + ORIGINAL\_AIRPORT
  + DESTINATION\_AIRPORT
* Bảng airports
  + AIRPORT

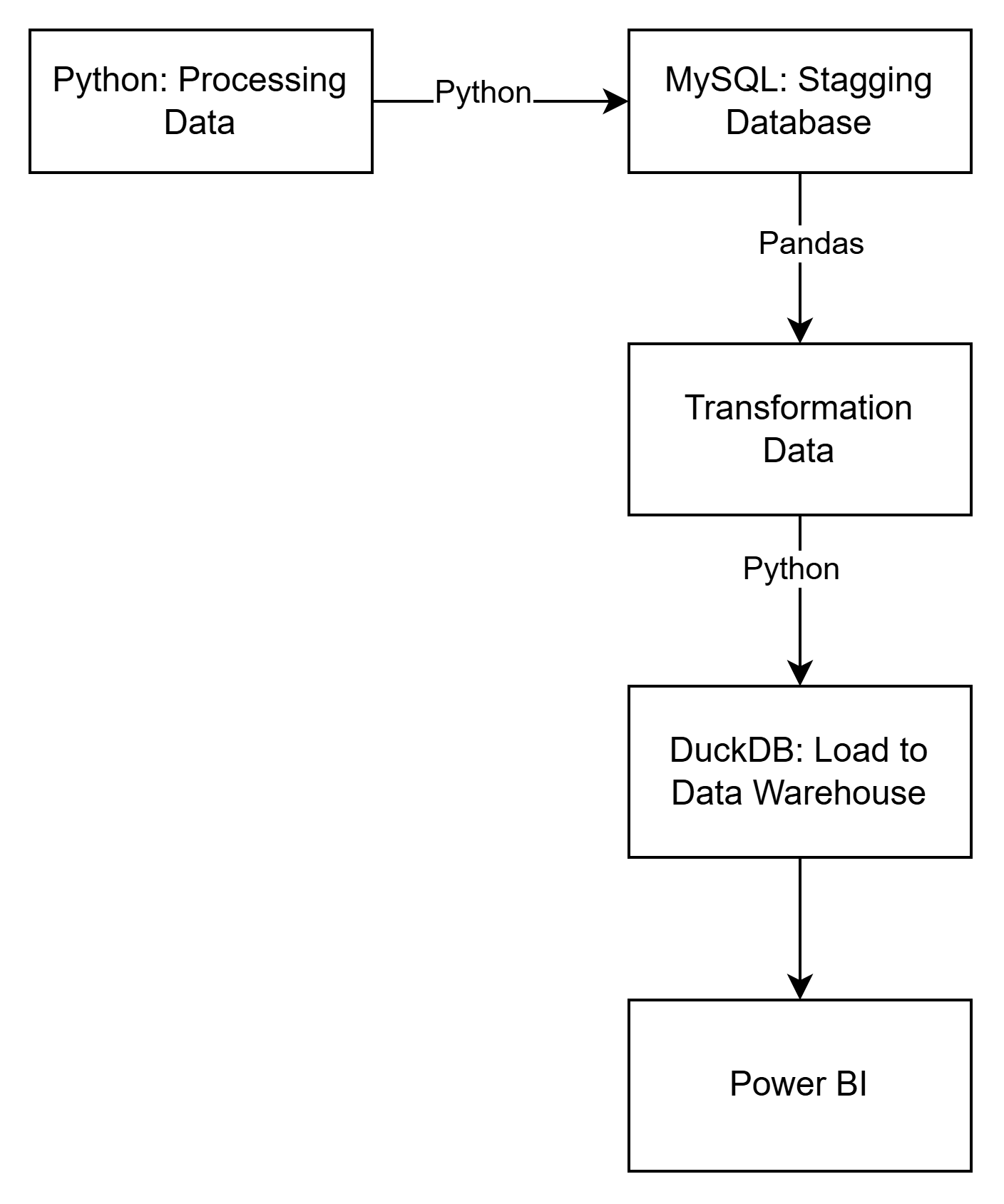
# **Chi tiết**

## **Thiết kế hệ thống ETL**

* 1. **ETL diagram**



* 1. **Taskflow**



* 1. **File System**

*└── flight\_delay\_etl\_project/*

*├── .gitignore*

*├── docker-compose.yaml*

*├── Dockerfile*

*├── main\_etl.py*

*├── requirements.txt*

*├── reports/*

*│ └── flight\_dashboard.pbix*

*├── notebooks/*

*│ ├── 01\_extract.ipynb*

*│ ├── 02\_clean\_transform.ipynb*

*│ └── 03\_load\_duckdb.ipynb*

*├── etl/*

*│ ├── extract.py*

*│ ├── load.py*

*│ ├── transform.py*

*│ └── utils.py*

*├── docs/*

*│ └── README.md*

*├── diagram/*

*│ ├── erd-diagram.drawio*

*│ ├── etl-diagram.drawio*

*│ └── taskflow.drawio*

*├── db/*

*│ └── flight\_data.duckdb*

*└── data/*

*├── raw/*

*│ ├── airlines.csv*

*│ ├── airports.csv*

*│ └── flights.csv*

*└── processed/*

*├── clean\_airlines.csv*

*├── clean\_airports.csv*

*└── clean\_flights.csv*

# **Chương 2: Triển khai**