**Database Design for European Airports and Flight Management**

**1. Database Design**

**The schema consists of multiple tables, each serving a specific purpose in storing and managing airport and flight-related data. The key tables include:**

**Airports**

**This table stores details about European airports.**

* **Fields:** 
  + **airport\_id (Primary Key, INT) – Unique identifier for the airport.**
  + **name (VARCHAR) – Name of the airport.**
  + **iata\_code (VARCHAR) – IATA airport code.**
  + **icao\_code (VARCHAR) – ICAO airport code.**
  + **city (VARCHAR) – City where the airport is located.**
  + **latitude (DECIMAL) – Latitude of the airport.**
  + **longitude (DECIMAL) – Longitude of the airport.**
  + **timezone (BIGINT) – Timezone of the airport.**

**Airlines**

**This table stores airline information.**

* **Fields:** 
  + **airline\_id (Primary Key, INT) – Unique identifier for the airline.**
  + **name (VARCHAR) – Name of the airline.**
  + **country (VARCHAR) – Country of operation.**

**Airline Codes**

**This table manages multiple airline codes with validity periods.**

* **Fields:** 
  + **code\_id (Primary Key, INT) – Unique identifier.**
  + **airline\_id (Foreign Key, INT) – Links to Airlines.**
  + **iata\_code (VARCHAR) – IATA airline code.**
  + **icao\_code (VARCHAR) – ICAO airline code.**
  + **valid\_from (DATE) – Start date of validity.**
  + **valid\_to (BIGINT) – End date of validity.**

**Routes**

**Defines flight routes between airports.**

* **Fields:** 
  + **route\_id (Primary Key, INT) – Unique identifier.**
  + **origin\_airport\_id (Foreign Key, INT) – Links to Airports.**
  + **destination\_airport\_id (Foreign Key, INT) – Links to Airports.**
  + **distance\_km (DECIMAL) – Distance between airports.**

**Flights**

**Stores details of flight schedules.**

* **Fields:** 
  + **flight\_id (Primary Key, INT) – Unique identifier.**
  + **flight\_number (VARCHAR) – Flight number.**
  + **airline\_id (Foreign Key, INT) – Links to Airlines.**
  + **route\_id (Foreign Key, INT) – Links to Routes.**
  + **scheduled\_departure (DATETIME) – Planned departure time.**
  + **scheduled\_arrival (DATETIME) – Planned arrival time.**
  + **booking\_id (Foreign Key, VARCHAR) – Links to Bookings.**

**Flight Events**

**Captures flight-related events.**

* **Fields:** 
  + **event\_id (Primary Key, INT) – Unique identifier.**
  + **flight\_id (Foreign Key, INT) – Links to Flights.**
  + **event\_type (ENUM) – Type of event (departure, arrival, delay, cancellation).**
  + **timestamp (DATETIME) – Time of event.**
  + **event\_detail (VARCHAR) – Additional details.**

**Delays**

**Records flight delays.**

* **Fields:** 
  + **delay\_id (Primary Key, INT) – Unique identifier.**
  + **flight\_id (Foreign Key, INT) – Links to Flights.**
  + **delay\_type\_id (Foreign Key, INT) – Links to Delay Types.**
  + **delay\_start\_time (DATETIME) – Start time of delay.**
  + **delay\_end\_time (DATETIME) – End time of delay.**
  + **reported\_by (VARCHAR) – Source of delay report.**

**Delay Types**

**Defines categories of delays.**

* **Fields:** 
  + **delay\_type\_id (Primary Key, INT) – Unique identifier.**
  + **category (ENUM) – Type of delay (weather, technical, etc.).**
  + **description (TEXT) – Description of delay.**

**Bookings**

**Stores passenger bookings.**

* **Fields:** 
  + **booking\_id (Primary Key, VARCHAR) – Unique identifier.**
  + **flight\_id (Foreign Key, INT) – Links to Flights.**
  + **passenger\_count (INT) – Number of passengers.**

**Passengers**

**Stores passenger details.**

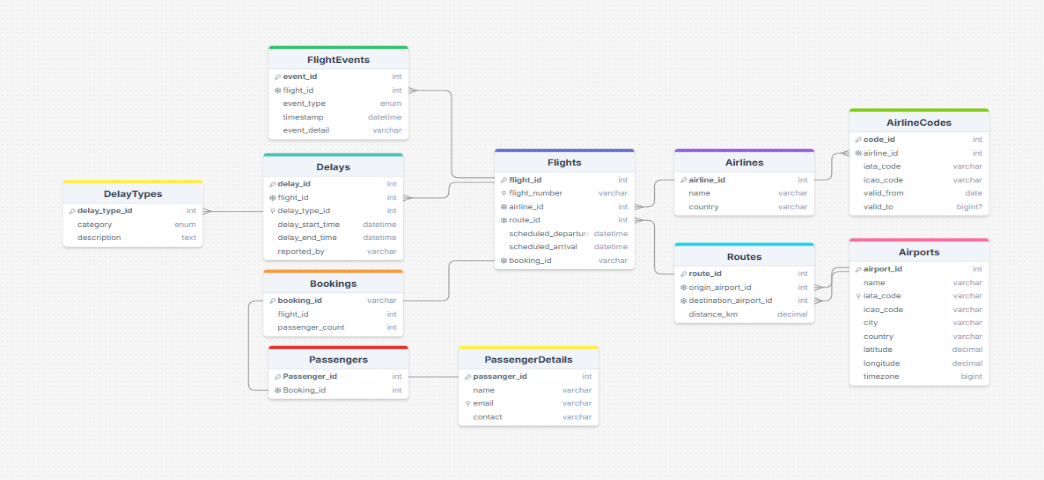
* **Fields:** 
  + **passenger\_id (Primary Key, INT) – Unique identifier.**
  + **booking\_id (Foreign Key, VARCHAR) – Links to Bookings.**

**Passenger Details**

**Stores additional passenger information.**

* **Fields:** 
  + **passenger\_id (Foreign Key, INT) – Links to Passengers.**
  + **name (VARCHAR) – Passenger name.**
  + **email (VARCHAR) – Passenger email.**
  + **contact (VARCHAR) – Contact information.**

**Relationships Between Tables**

* **Airports ↔ Routes: A route connects an origin airport and a destination airport.**
* **Airlines ↔ Flights: Each flight is operated by an airline.**
* **Routes ↔ Flights: Each flight follows a specific route.**
* **Flights ↔ Flight Events: A flight may have multiple flight events (departure, arrival, delays, etc.).**
* **Flights ↔ Delays: A flight may be associated with multiple delays.**
* **Delays ↔ Delay Types: Each delay is classified under a delay type.**
* **Flights ↔ Bookings: A flight can have multiple passenger bookings.**
* **Bookings ↔ Passengers: Each booking is linked to one or more passengers.**
* **Passengers ↔ Passenger Details: Each passenger has additional details like contact information.**
* **Airline Codes ↔ Airlines: Airlines have associated IATA and ICAO codes with validity periods.** 

**2. Data Collection Strategy**

To build a comprehensive dataset of all European airports, I started by gathering data from multiple sources, including open datasets from the OpenFlights database, Eurocontrol, and government aviation websites. The dataset initially contained fields such as IATA code, ICAO code, airport name, country, latitude, longitude, and elevation.

For real-time flight tracking, I explored different data sources:

* **Third-Party APIs**: I experimented with FlightAware, OpenSky Network, and ADS-B Exchange. Among these, OpenSky provided a free-tier API with real-time location updates but had some data latency issues.
* **Automated Scraping**: I considered using ChatGPT API for intelligent data extraction from airline websites. However, most websites had strict anti-scraping measures, making APIs the more reliable option.
* **ADS-B Data Processing**: I integrated ADS-B receivers into my setup to receive real-time flight broadcasts directly. This method provided the most control over data but required additional hardware.

**Handling Missing, Delayed, or Inconsistent Data**

During analysis, I encountered several data quality issues:

* **Missing Flight Data**: Some flights had missing departure or arrival times. For this, I used interpolation based on similar flights with the same route and aircraft type.
* **Interpolated Missing ICAO Codes**: Used similar airports within the same country for estimation.
* **Assigned 'Unknown' to Missing Codes**: Where both IATA and ICAO codes were missing, ensured data integrity by assigning 'Unknown'.

**3. Flight Monitoring and Claim Identification**

**System Overview**

To monitor flights from European airports, I propose a system that tracks real-time flight data and flags delays of more than 2 hours. This system ensures timely updates for passengers and airlines while also supporting compensation claims.

**Technical Approach**

* **Data Sources**:
  + **FlightAware API** – Provides structured, real-time flight data.
  + **ADB (Aviation Data Broker) Data** – Offers additional insights.
  + **Custom Web Scraping** – Can be used to supplement data where necessary.
* **Data Processing**:
  + Fetch data every 5 minutes to ensure up-to-date information.
  + Store relevant fields such as flight ID, departure/arrival time, and delay status.
  + Interpolation techniques applied to missing values; if 90% of a column is NaN, it will be dropped. For essential fields, missing values will be replaced with "Unknown."
* **Flagging System**:
  + Any flight with a delay of more than 120 minutes is flagged.
  + Automated alerts are sent to relevant stakeholders via email or API notifications.

**Data Storage and Management**

* **Database Selection**:
  + PostgreSQL or MongoDB for scalable storage.
  + Time-series database (like InfluxDB) for tracking historical delays.
* **Efficient Storage Strategy**:
  + Data retention policies ensure outdated records are archived after 6 months.
  + Indexing and partitioning optimize query performance for large datasets.

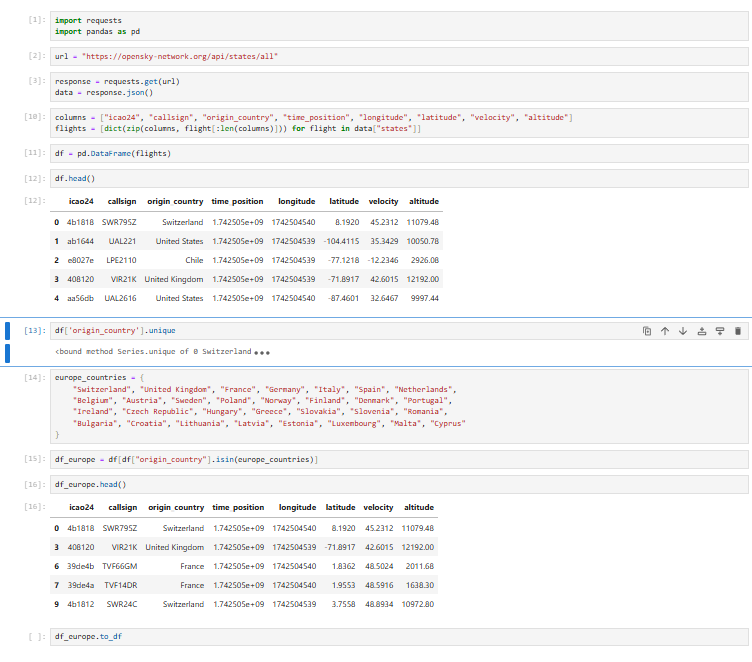
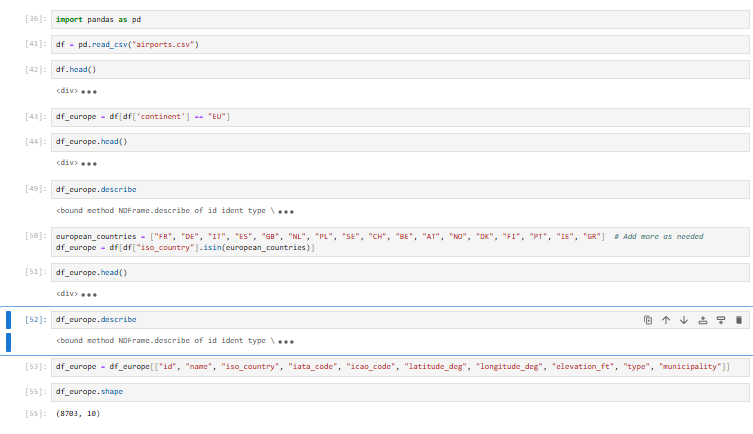
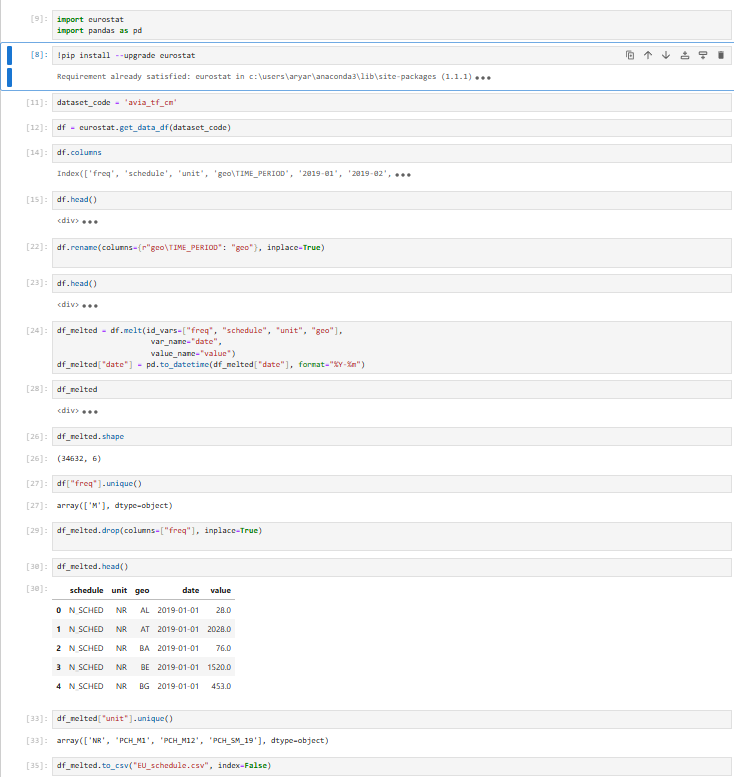
**4. Future API Development**

**API Design & Implementation**

* **Scalability Considerations**:
  + RESTful API with support for GraphQL for efficient queries.
  + Load balancing using Nginx to distribute requests evenly.
  + Microservices architecture for modular expansion.
* **Endpoints**:
  + /flights/status?iata=XXX – Retrieves real-time flight status.
  + /flights/delays?min\_delay=120 – Returns flights delayed over 2 hours.
  + /airports?country=XX – Fetches airport details by country.

**Security, Availability, and Reliability**

* **Security Measures**:
  + API authentication using OAuth 2.0 and API keys.
  + Rate limiting to prevent abuse.
  + Data encryption (TLS) for secure data transmission.
* **Availability & Reliability**:
  + Use of caching (Redis) to reduce API load.
  + Deploying in multi-region cloud infrastructure for redundancy.
  + Continuous monitoring and logging to detect and resolve issues quickly

Data Cleaning

Real-Time Flight Data Collection Using APIs