Airline Business Intelligence Database

End-to-End BI System Using SQL, ETL, PostgreSQL & Python

Grace Polito

Final Capstone Project - MSDS DTSC 691
Eastern University

PRESENTATION OVERVIEW

Introduction & Problem Domain

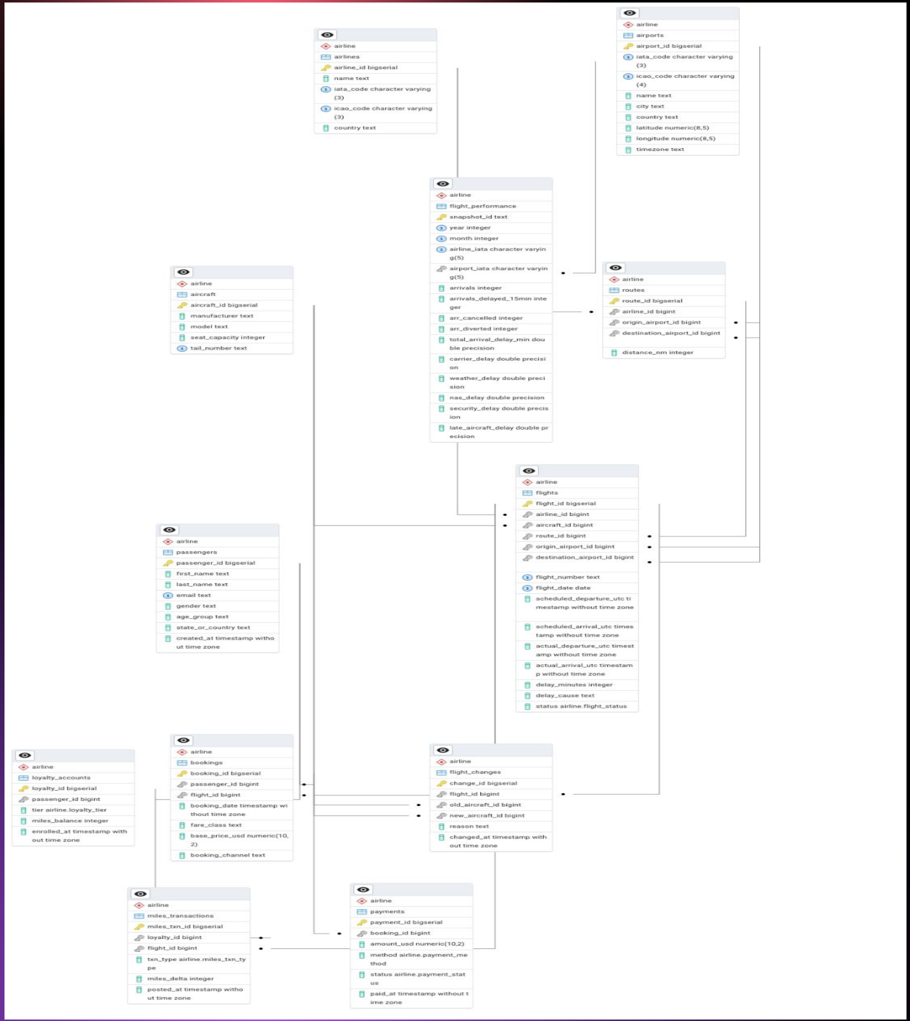
> The core problem this project tackles is the ***fragmentation*** of airline ***operational*** and ***commercial*** data.

- Airlines generate complex operational, customer, and revenue data
- Spreadsheets & disconnected sources limit analytics quality
- Need for a unified, query-ready analytical database
- Project goal: build a BI-ready airline database + analytics layer

Project Timeline (Phase 1-6)

01	Design & Setup	Schema, ERD, environment, initial constraints
02	Data Collection & Insertion	OpenFlights/BTS import, synthetic generation
03	SQL CLeaning & Constraints	DML standardization, deduplication, indexing
04	Query Development	15+ analytical SQL queries + performance testing
05	Python Analytics	Engine connection, helper functions, charts
06	Final Deliverables	Overview PDF, exported code, 20-minute presentation

Schema Design
&
Data Model Overview



Core Operational Entities

airline
airlines
airline_id bigserial
name text
iata_code character varying (3)
icao_code character varying (3)
country text

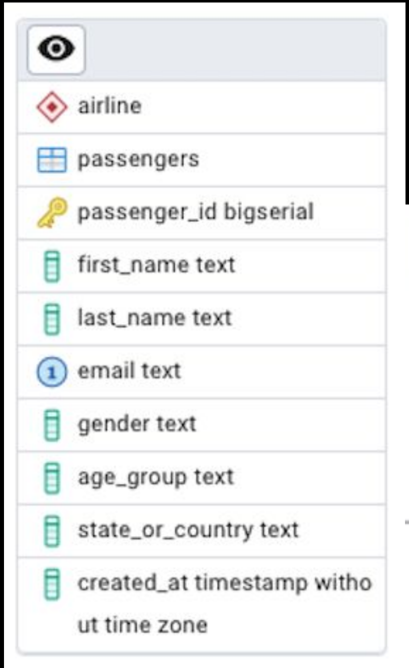
airline
airports
airport_id bigserial
iata_code character varying (3)
icao_code character varying (4)
name text
city text
country text
latitude numeric(8,5)
longitude numeric(8,5)
timezone text

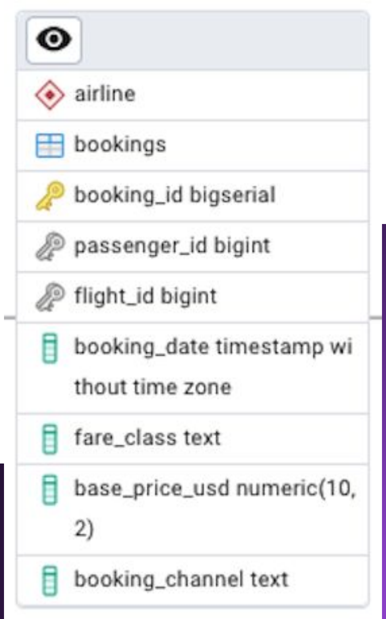
airline
routes
route_id bigserial
airline_id bigint
origin_airport_id bigint
destination_airport_id bigint
distance_nm integer

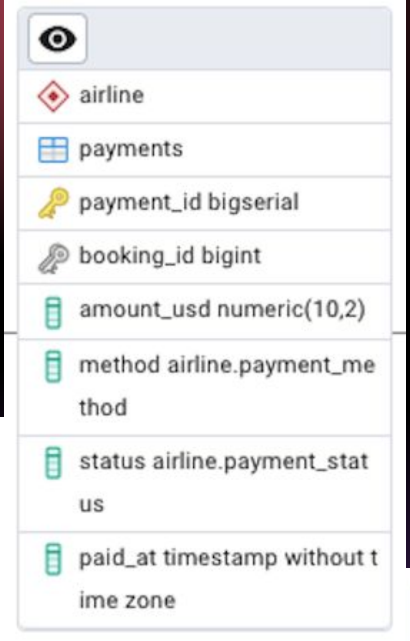
airline
aircraft
aircraft_id bigserial
manufacturer text
model text
seat_capacity integer
tail_number text

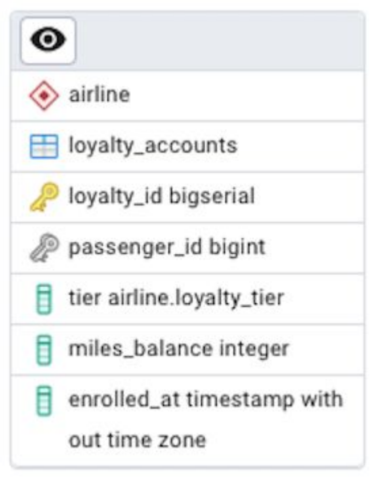
airline
flights
flight_id bigserial
airline_id bigint
aircraft_id bigint
route_id bigint
origin_airport_id bigint
destination_airport_id bigint
flight_number text
flight_date date
scheduled_departure_utc timestamp without time zone
scheduled_arrival_utc timestamp without time zone
actual_departure_utc timestamp without time zone
actual_arrival_utc timestamp without time zone
delay_minutes integer
delay_cause text
status airline.flight_status

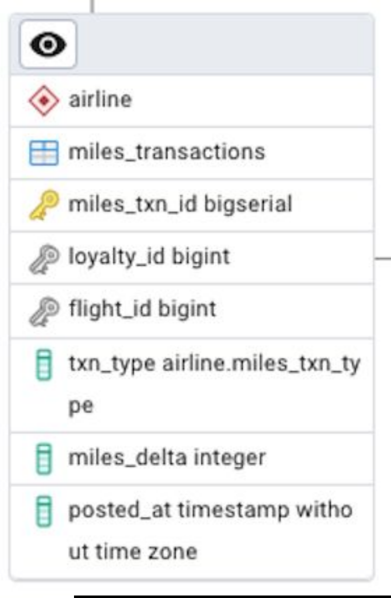
Commercial Entities: Passengers, Bookings & Payments



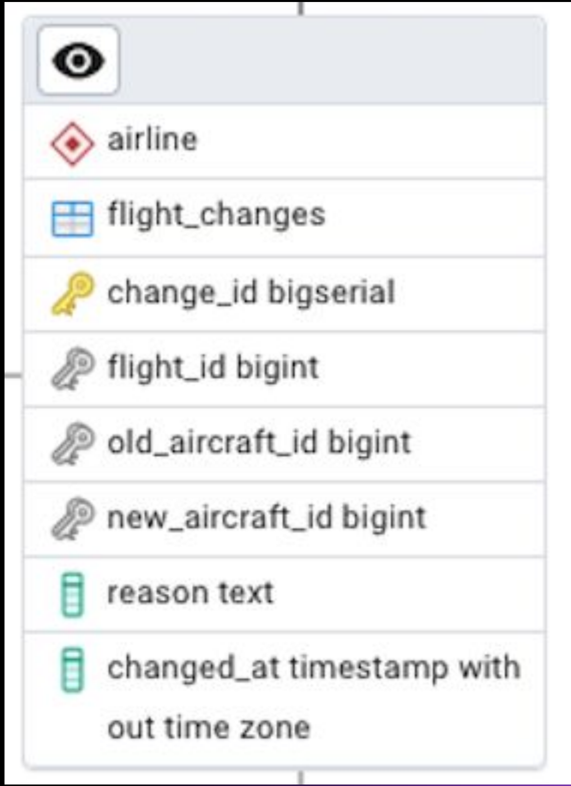


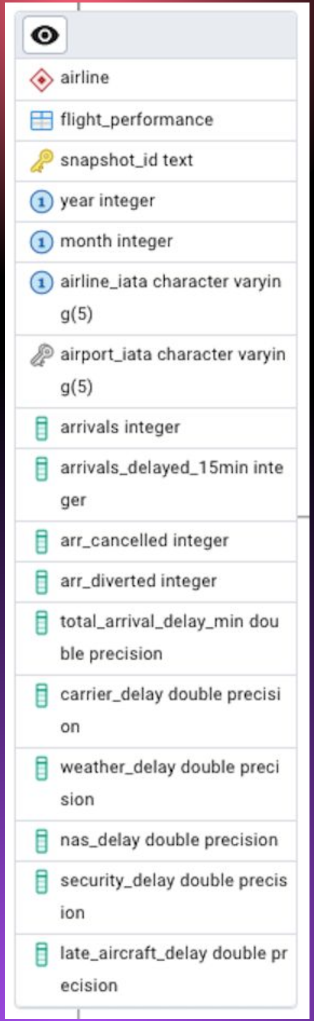


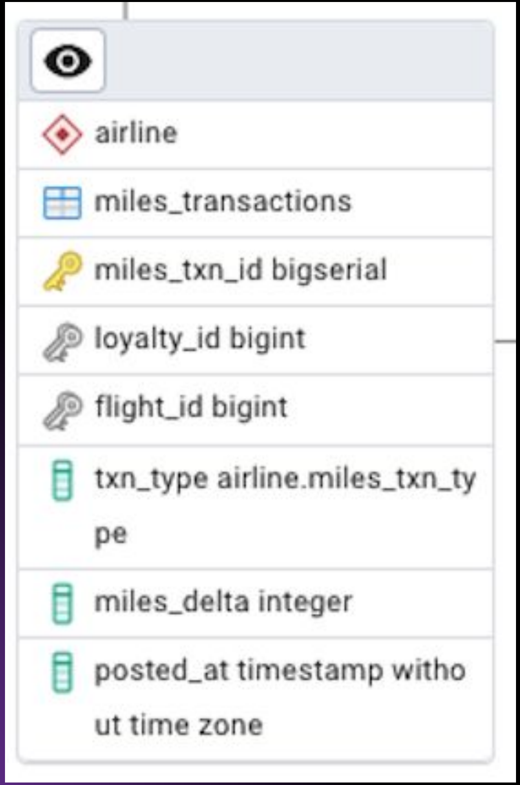




Analytical Fact Tables: Performance & Change Tracking







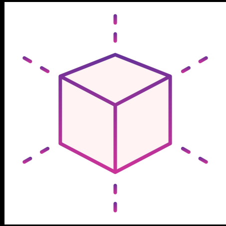
WHY 3NF?

Schema Design Principles

ATOMIC TABLES

→ Each table models one concept

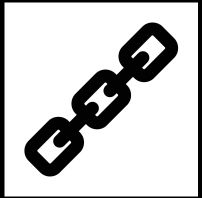
→ Eliminates duplicate fields



REFERENTIAL INTEGRITY

→ Strong PK/FK relationships

→ Prevents orphaned or inconsistent records



CONTROLLED BUSINESS RULES

→ ENUMs for statuses & tiers

→ CHECK constraints protect data quality

PERFORMANCE OPTIMIZATION

→ Indexed join keys

→ Supports fast analytical queries

DATA SOURCES

OpenFlights
real-world operational data

provides **real-world** reference data for airports and airlines

→ Airports & airlines reference tables

→ Global identifiers (IATA/ICAO)

→ Used for route + flight generation

→ Real Data

BTS On-Time Performance
real-world delay data

contains monthly airline **delay** metrics

→ Monthly delay performance metrics

→ Carrier, weather, security, NAS delays

→ Used to enrich operational analysis

→ Real Data

Synthetic Data
flights, passengers, bookings, payments, loyalty

generated to complete the **commercial side** of the model

→ Flights, passengers, bookings, payments

→ Loyalty accounts & miles transactions

→ Enables complete operational + commercial coverage

→ Generated in Python with faker

Sources: OpenFlights.org; U.S. DOT BTS (transtats.bts.gov)

ETL PIPELINE

The ETL pipeline follows a structured Extract–Transform–Load workflow.

01 Extraction

- Import OpenFlights & BTS CSVs
- Generate synthetic datasets in Python
- Standardize column names & formats

02 Transformation

- Clean nulls, trim strings, enforce types
- Normalize time formats & airports/airlines
- Create route mappings & flight schedules

03 Loading

- Insert into PostgreSQL via SQLAlchemy
- UPSERTs to avoid duplicates
- Ensure PK/FK integrity during load

04 Validation

- Row count checks
- FK consistency tests
- Spot-checks for anomalies & outliers

SQL ANALYTICS LAYER

15 analytical SQL queries built for operations, revenue, network, and loyalty insights.

Window Functions

Ranking airlines by average delay

```
SELECT airline_name,
       iata_code,
       AVG(delay_minutes) AS avg_delay_minutes,
       DENSE_RANK() OVER (ORDER BY AVG(delay_minutes) DESC) AS delay_rank
FROM flights
GROUP BY airline_name, iata_code;
```

CTEs & Aggregations

Multi-hop airport connectivity

```
WITH RECURSIVE connections AS (
  SELECT origin_iata, dest_iata, 1 AS hops, ARRAY[origin_iata, dest_iata] AS path
  FROM routes
  WHERE origin_iata = 'YCK'
  UNION ALL
  SELECT c.origin_iata, r.dest_iata, c.hops + 1,
         path || r.dest_iata
  FROM connections c
  JOIN routes r ON c.dest_iata = r.origin_iata
  WHERE c.hops < 3)
SELECT * FROM connections;
```

Advanced Analytical Queries

Revenue, fare class, payment success, CLV-style revenue

```
SELECT fare_class,
       COUNT(*) AS num_bookings,
       SUM(amount_usd) AS total_revenue
FROM bookings
JOIN payments USING (booking_id)
GROUP BY fare_class
ORDER BY total_revenue DESC;
```

Python Integration & Analytics Layer

Python, SQLAlchemy, Pandas, Matplotlib, and Plotly power the BI analysis.

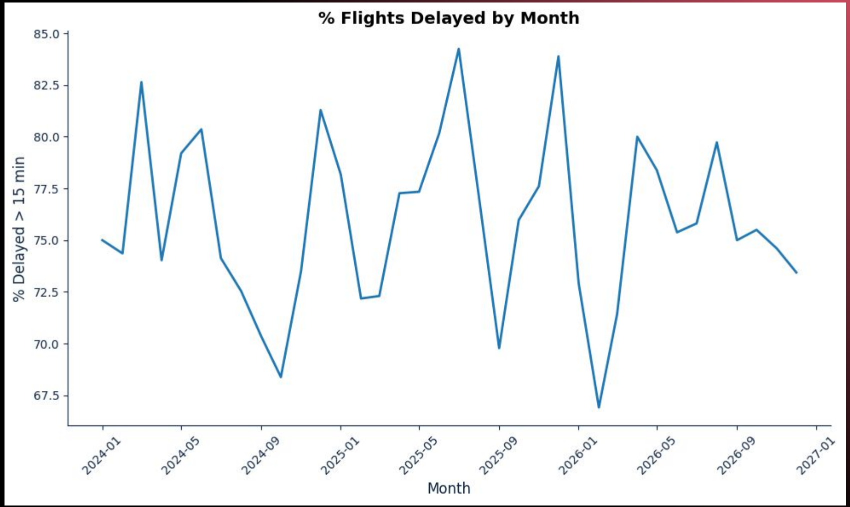
SQLAlchemy Engine → Query Help Functions → Pandas DataFrames

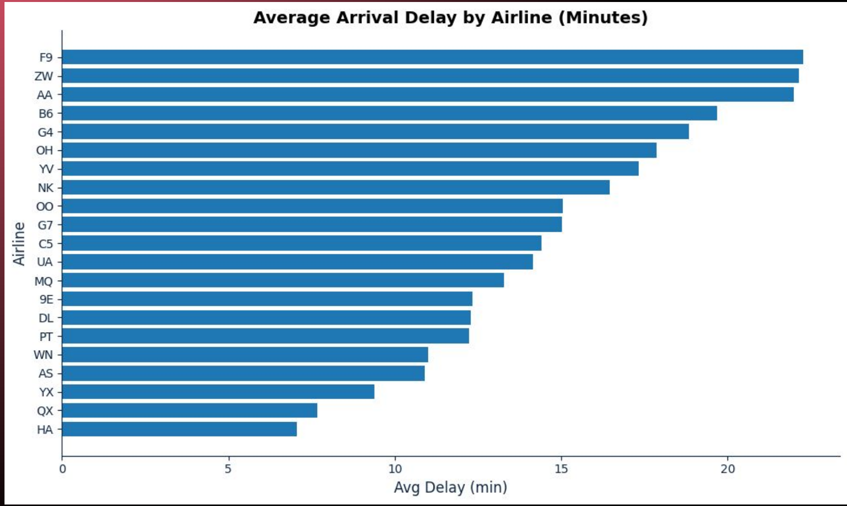
```
def get_df(sql: str, params=None):
    engine = get_engine()
    with engine.connect() as conn:
        return pd.read_sql(text(sql), conn, params=params)
```

- Connects Python ↔ PostgreSQL
- Reusable query wrappers (e.g., get_revenue_by_fare_class)
- Supports operational, revenue, and loyalty analytics
- Drives visualizations (Matplotlib + Plotly)

Code produced in 04_python_analytics.ipynb

Operational Performance Visuals



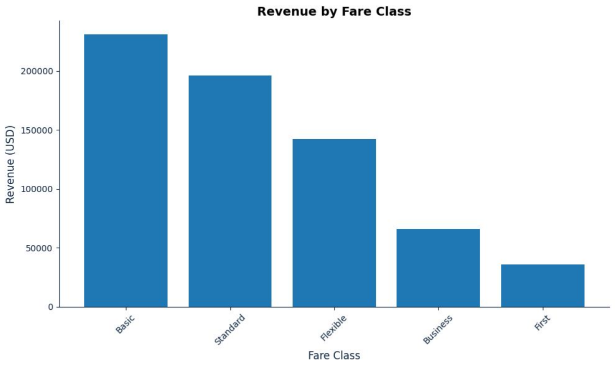


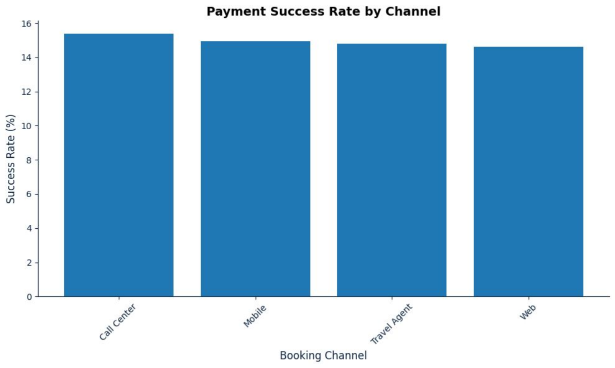
Delay rates fluctuate **seasonally**, with spring and early winter showing the **highest disruption** levels. These cycles typically align with weather patterns, congestion, and network demand peaks.

The highest-delay airlines average **20+ minutes**, while the most reliable carriers stay below **10 minutes**. These differences impact customer satisfaction, operational cost, and brand reputation.

Visualizations produced in 04_python_analytics.ipynb

Commercial Performance: Revenue, Payments & CLV





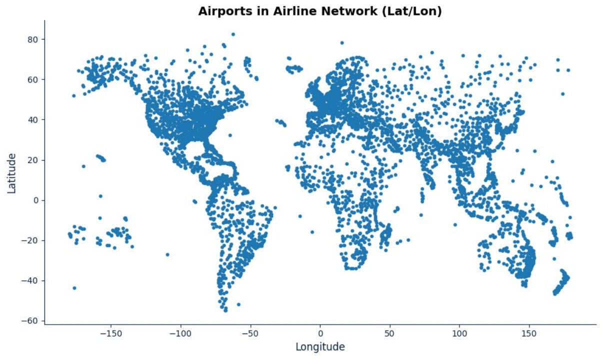
Basic and Standard fares drive the majority of revenue volume, reflecting price-sensitive demand in the synthetic dataset.

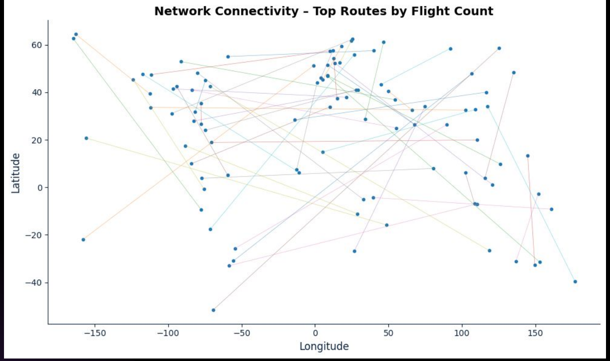
Revenue trends illustrate cyclical demand, confirming peak travel periods and slower off-season months.

All channels exhibit similar success rates, with Call Center slightly outperforming digital channels.

Visualizations produced in 04_python_analytics.ipynb

Network & Geographic Visualizations





Airport coordinates plotted globally to reveal the geographic footprint of the modeled network.

High-volume OD pairs—such as ALN→BFI and BUR→FET—highlight the strongest traffic flows in the network.

Most-frequent routes drawn as direct coordinate links, illustrating the core structure of the airline's route system.

Visualizations produced in 04_python_analytics.ipynb

KEY BUSINESS INSIGHTS

Operational Insights 🛫

★ Delays show clear **seasonal** patterns

★ Airline **reliability** varies significantly

Revenue Insights 💰

★ **Basic** and **Standard** fares drive most revenue volume

★ Revenue exhibits cyclical trends

Payment & Booking Insights 💳

★ Payment success rates are **consistent** across channels

★ Synthetic booking patterns still **reflect realistic** conversion behavior

Customer & Loyalty Insights 🎯

★ Top **5%** of customers generate **~13%** of total revenue

★ Loyalty tiers show balanced distribution

Network Insights 🗺️

★ Top **OD pairs** form key network corridors

★ Connectivity highlights major route clusters

Insights derived from SQL (Phase 4) and Python analytics (04_python_analytics.ipynb).

CHALLENGES, LIMITATIONS & LESSONS LEARNED

🧹 Data Challenges

- Required heavy ***cleaning*** & ***standardization***
- Multiple sources needed ID/time alignment

⚙️ Synthetic Data Limits

- Some airports unmapped → limited geography
- Future dates used to model f***ull-year trends***

🔗 Workflow Constraints

- PK/FK integrity required ***iterative cleanup***
- Recursive CTEs & large joins needed tuning

💡 Lessons Learned

- Clean schema drives ***better analytics***
- Reusable pipelines reduce rework

FUTURE ENHANCEMENTS

Data Expansion

Integrate **real airline** schedules, fares, and ancillary data for **richer modeling**

Advanced Analytics

Develop CLV forecasting, demand prediction, and **anomaly detection** models

Enhanced Visualizations

Build **interactive dashboards** and basemap-backed route maps using **Tableau**

Production Readiness

Containerize the pipeline and add **automated testing** for reproducible workflows

Conclusion & Next Steps

Project Summary

Built a unified, BI-ready airline database integrating operational, booking, loyalty, and payment data.

Key Outcomes

Delivered end-to-end analytics uncovering operational reliability, revenue drivers, and customer value.

Next Steps

Expand with real schedules, predictive models, and interactive dashboards.

THANK YOU

Grace Polito

Airline Business Intelligence Database