Airline Business Intelligence Database – Presentation Script

Slide 1 — Title Slide

Hello, my name is Grace Polito, and welcome to my capstone presentation: the Airline Business Intelligence Database.

This project models airline operations, customer behavior, and commercial performance using a fully integrated data pipeline—from schema design, to ETL, data quality, SQL analytics, and Python visualizations.

Slide 2 — Presentation Overview

Here's a quick overview of what I'll be covering today.

I'll begin with the introduction and the problem domain—why this project matters and the real-world airline challenges it's designed to address.

Then I'll walk through the full project timeline and each phase at a high level.

Next, I'll cover the schema design and data model, including the ERD and how the system was normalized into a clean analytical structure.

From there, I'll move into the data sources and ETL pipeline—how real OpenFlights and BTS data were combined with synthetic flights, customers, bookings, and revenue.

After the pipelines, I'll showcase the SQL analytics layer and some of the advanced analytical queries developed during Phase 4.

Then we'll transition to the Python integration and visual analytics, where I'll highlight the BI visualizations and Pandas-based analysis.

Toward the end, I'll summarize the key business insights uncovered through the analysis, followed by some of the challenges and lessons learned along the way.

Finally, I'll wrap up with future enhancements and closing next steps.

Slide 3 - Intro & Problem Domain

To begin, the core problem this project tackles is the fragmentation of airline operational and commercial data. Airlines generate thousands of touchpoints—flights, delays, customers, loyalty accounts, bookings, payments—and these datasets are often stored across different systems or delivered as messy, semi-structured files. This makes it difficult to answer even basic questions like ‘What drives delays?’ or ‘Which routes are most profitable?’

This project addresses that by building a unified, PostgreSQL-based analytical database that models airline operations end-to-end. The goal was not only to consolidate these data sources, but to engineer a clean schema, enforce data quality, and provide a full analytical layer that supports business intelligence workflows.

Ultimately, the purpose was to create a realistic, BI-ready environment that mirrors what airlines rely on for performance reporting, network planning, revenue analysis, and loyalty insights.

Slide 4 — Project Timeline

This project followed a structured, six-phase lifecycle that mirrors a real BI engineering workflow.

Phase 1 was Design and Setup, where I established the PostgreSQL environment, created the schema, produced the ERD, and defined the initial constraints that form the backbone of the system.

Phase 2 focused on Data Collection and Insertion. This included importing real OpenFlights and BTS datasets and generating synthetic flights, passengers, bookings, and payments so that the system had realistic operational and commercial coverage.

Phase 3 was SQL Cleaning and Constraints. In this phase, I standardized all tables using DML cleanup scripts, removed duplicates, enforced primary and foreign keys, added CHECK and UNIQUE constraints, and created indexes to support analytical workloads.

Phase 4 covered Query Development. I created 15 analytical SQL queries, including window functions, CTEs, aggregations, and performance-tested joins, producing insights across operations, revenue, loyalty, and network performance.

Phase 5 focused on Python Analytics. Here I built engine connectors, reusable helper functions, and a full set of visualizations using Pandas and Matplotlib to demonstrate how the analytical layer integrates with Python-based BI workflows.

And finally, **Phase 6 wrapped everything together** into a cohesive set of deliverables—an overview PDF, exported SQL and Python code, and this video presentation.

Together, these phases form a complete end-to-end BI system, from raw data ingestion all the way through insights and visualization.

Slide 5 — Schema Overview

This is the full entity–relationship diagram for the Airline Business Intelligence Database.

The schema is fully normalized and organized into several logical domains: airline and airport reference data, operational flight data, customer and loyalty data, commercial booking and payment data, and historical performance data.

In total, the system includes 12 interconnected tables. Each table represents a clean, atomic concept, and the relationships across the schema allow the database to answer complex operational, commercial, and customer-focused analytical questions.

Slide 6 — Core Operational Entities

Here is a closer look at the core operational entities.

Airlines and airports form the main reference tables, with unique IATA (eye-AH-tuh) and ICAO (eye-KAY-oh) identifiers.

Routes map origin and destination airports with a calculated distance, which is used for both flight generation and later analytics like route profitability and delay pattern analysis.

Aircraft captures seat capacity, tail number, and manufacturer, which links into the flights table.

The flights table is the central fact table for airline operations, storing scheduled vs. actual times, delay minutes, and flight status.

Together, these operational tables provide the backbone for performance, routing, and scheduling analytics.

Slide 7 — Commercial Entities: Passengers, Bookings & Payments

On the commercial side, the schema models passengers, bookings, payments, and loyalty activity.

Passengers store individual customer attributes and are linked to both bookings and loyalty accounts.

Bookings represent the purchase of a flight, including fare class, booking channel, and base fare.

Payments model the financial transaction itself—whether the customer paid by card, points, voucher, or cash—and track both payment status and paid timestamp.

The loyalty_accounts and miles_transactions tables model a simplified version of an airline loyalty program. These tables allow analytics such as point accrual patterns, tier distribution, and customer lifetime value proxies.

These commercial entities support revenue analysis, customer segmentation, and payment reliability insights.

Slide 8 — Analytical Fact Tables: Performance & Change Tracking

The final set of tables are analytical fact tables that enrich the operational and commercial pillars.

flight_performance contains monthly aggregated BTS performance metrics, including arrival delay, cancellation rates, and carrier-specific delay categories. This dataset provides historical context and allows for trend and reliability analytics.

flight_changes logs operational changes such as aircraft swaps or schedule changes. This is important for understanding the operational disruptions that may correlate with delays or downstream revenue impacts.

Finally, miles_transactions captures loyalty earning and redemption activity, enabling customer-centric analytics such as engagement, value, and tier progression.

These fact tables significantly expand the analytical capabilities of the system.

Slide 9 — Why 3NF: Design Principles Behind the Model

The schema follows Third Normal Form to ensure data cleanliness, consistency, and analytical reliability.

Each table represents a single atomic concept—for example, a passenger, a booking, or a flight—without redundant or duplicated information.

All business rules are enforced through foreign keys, CHECK constraints, and ENUM types for controlled vocabularies like flight status, payment method, and loyalty tiers.

Indexes were added on high-usage analytical fields—such as flight_date, airline_id, airport_id, and booking timestamps—to ensure efficient query performance.

This design ensures that the system behaves like a true analytical warehouse: clean, reliable, scalable, and easy to query.

Slide 10 — Data Sources

The database integrates three main categories of data sources.

First, OpenFlights provides real-world reference data for airports and airlines. This gives the project accurate global identifiers—like IATA (eye-AH-tuh) and ICAO (eye-KAY-oh) codes—and forms the foundation for building realistic routes and flights.

Second, the project incorporates BTS On-Time Performance data, which contains monthly airline delay metrics broken down by categories such as carrier, weather, security, and airspace congestion. This dataset allows the system to capture patterns in operational reliability and seasonal delays.

Finally, synthetic data was generated to complete the commercial side of the model. This includes flights, passengers, bookings, payments, loyalty accounts, and miles transactions.

The real datasets provide a grounded operational context, while the synthetic datasets allow for a full end-to-end BI environment, including revenue, customer behavior, loyalty, and payment flows.

Slide 11 — ETL Pipeline

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

During the extraction stage, real datasets from OpenFlights and BTS were imported as CSV files, while Python scripts generated synthetic flights, passengers, bookings, payments, and loyalty activity. At this stage, raw fields were standardized so they could flow cleanly into the transformation process.

In the transformation stage, the data was cleaned and normalized. This included trimming whitespace, fixing nulls, standardizing casing, enforcing numeric types, and aligning timestamps. Route mappings and flight schedules were also generated here to connect airports, airlines, and aircraft into a unified operational model.

The load stage inserted all data into PostgreSQL using SQLAlchemy. UPSERT patterns ensured no duplicates were created, and foreign key relationships were validated during the load to maintain referential integrity.

Finally, the validation stage performed row-count comparisons, FK consistency checks, and anomaly detection to confirm that the entire dataset was correctly ingested and structurally sound.

Together, this ETL pipeline ensures the system starts with clean, trustworthy data—critical for accurate analytics later on.

Slide 12 - SQL Analytics Layer

In Phase 4, I developed 15 analytical SQL queries that make up the core of the BI environment. This slide highlights 3 representative examples.

On the left is a window-function query. Here I use tools like DENSE_RANK, rolling averages, and cumulative sums to rank airlines by delay performance and support metrics such as customer lifetime value and running revenue totals.

In the center is a recursive CTE example. Standard CTEs help structure route-level analysis, while the recursive version builds multi-hop connectivity graphs—showing all airports reachable from a hub within three connections.

On the right is an example of multi-table commercial analytics. These joins calculate revenue by fare class, evaluate payment success rates across channels, and support loyalty segmentation using window-based percentile logic.

Together, these SQL queries power the system's operational dashboards, revenue reporting, customer analytics, and network insights—forming the analytical backbone of the entire database.

Slide 13 - Python Integration Architecture

Phase 5 connects the analytical SQL layer into Python using SQLAlchemy.

I created a reusable set of helper functions — like get_df() — that connect to PostgreSQL, execute a SQL query, and return a clean Pandas data frame.

On top of this base, I built a library of analytics functions such as get_revenue_by_fare_class, get_busiest_airports, get_airline_punctuality, and get_clv_samples.

These functions allow Python to drive operational, network, commercial, and customer analytics using live data from the warehouse.

Finally, the results are visualized using both Matplotlib for static plots and Plotly for interactive dashboards.

Slide 14 - Operational Performance Visuals

Using the Python helper functions, I visualized key operational metrics.

Here, the monthly delay chart shows how the percent of flights delayed over 15 minutes fluctuates over time. The synthetic dataset still produces clear seasonality-like patterns.

On the right, airline-level performance is calculated using the BTS delay dataset. Some carriers average under 10 minutes of delay, while others exceed 20 minutes.

These visuals reflect exactly how operations teams and network planners monitor performance in real airline environments.

Slide 15 - Revenue, Payments & CLV Analytics

Python also drives commercial analytics by joining bookings, payments, and passenger data.

Revenue by fare class clearly shows Basic and Standard fares driving volume, while premium products deliver higher per-passenger yield.

Monthly revenue trends help illustrate demand cycles and how the BI system supports revenue forecasting.

Payment success analysis highlights funnel performance by channel — in this synthetic dataset success is lower than real-world figures, but the analysis demonstrates the reporting workflow.

Finally, customer lifetime value calculations show that the top 5% of passengers generate around 13% of total revenue, reflecting moderate loyalty concentration.

Slide 16 - Network & Geographic Visualizations

To understand the structure of the airline network, I generated several geographic and route-level visualizations using the airport latitude/longitude data.

This **first visualization** plots every airport in the dataset by latitude and longitude.

Because the OpenFlights dataset contains global airport reference points, this scatter map effectively shows the world map using only airport coordinates.

You can clearly see the concentration of airports across North America, Europe, and Asia, and the sparser coverage across the Southern Hemisphere.

This establishes the geographic footprint available for route modeling.

The n**ext visualization is a Sankey** diagram showing the busiest origin–destination airport pairs based on flight count.

On the left are the origin airports, and on the right are their corresponding destinations. The thickness of each flow line represents how frequently that route appears in the modeled schedule.

In this dataset, several routes stand out as high-volume corridors. For example, St. Louis (ALN) to Seattle (BFI) and Burbank (BUR) to Fremont (FET) all appear as heavily traveled pairs.

These origin destination pairs represent the strongest traffic flows in the simulated network and help visualize where aircraft and capacity are concentrated.

The **final chart** shows the top routes by flight count, drawn as straight lines between origin and destination coordinates.

Each point is an airport, and each line represents one of the most frequently flown routes in the dataset.

Because this visualization uses direct latitude/longitude plotting without a basemap, the lines may appear abstract when viewed as a static PNG. Even without a basemap, the structure of the network still emerges: in the upper left, the cluster represents airports in Alaska; in the middle region you see dense connectivity across the continental United States, and other major flows extend into Europe and Asia.

Together, these lines highlight the core skeleton of the airline's route system and the strongest traffic corridors.

Slide 17 - Key Business Insights

Pulling together insights across operations, revenue, customers, and network structure reveals several meaningful patterns.

Operationally, delay rates show clear seasonal fluctuation — with the heaviest disruption occurring in the early spring and winter months. Airline reliability also varies significantly; some carriers average over twenty minutes of delay, while the most reliable remain below ten.

Commercially, revenue is driven primarily by Basic and Standard fare products, which mirrors real-world price-sensitive demand. Revenue also exhibits cyclical behavior, with mid-year peaks and off-season troughs.

When looking at payments, success rates are relatively consistent across channels, with call center transactions slightly outperforming digital channels, suggesting differences in customer support or payment behavior.

On the customer side, the top five percent of passengers generate about thirteen percent of total revenue — moderate concentration that still aligns with typical airline loyalty distributions.

Finally, from a network perspective, a small number of high-volume airport pairs form the backbone of the route system, and the connectivity map reveals the major geographic clusters driving the airline's flow structure.

Collectively, these insights demonstrate how the BI system supports operations, revenue management, customer analytics, and network planning in a unified environment.

Slide 18 — Challenges, Limitations & Lessons Learned

This project introduced several challenges across data preparation, modeling, and analytics.

On the data side, both BTS and OpenFlights required extensive cleaning and alignment. The synthetic datasets I created—bookings, payments, and loyalty—needed realistic behavior patterns, which required fine-tuning. Some synthetic IATA (eye-AH-tuh) codes do not map to real airports, and that added limitations when interpreting a few network visuals.

You'll also notice some dates extend into the future. That was intentional. The synthetic generator uses randomized future windows so the dataset includes a complete annual cycle. This allowed for meaningful trend and seasonality analysis, rather than being constrained to partial historical data.

Technically, maintaining referential integrity during bulk inserts was demanding, and recursive CTEs and multi-table joins required optimization.

The biggest takeaways were the importance of a clean schema, the power of reproducible ETL and analytics pipelines, and the need for clarity and context when creating BI visualizations. Balancing engineering, analysis, and communication was key to building a coherent end-to-end system.

Slide 19 — Future Enhancements

Looking ahead, there are several meaningful ways this project could continue to evolve.

On the data side, a next step would be integrating additional real-world datasets, such as live schedules or historical fare and revenue data. The synthetic generator could also be enhanced to model more complex loyalty, payment, and customer behavior.

Analytically, future work could include more advanced models like delay anomaly detection or network optimization algorithms.

On the BI layer, Tableau could be used to develop fully interactive dashboards, and incorporate real geographic basemaps for the route visualizations.

Finally, from a technical standpoint, containerizing the pipeline and adding automated testing would make the system more production-ready, helping ensure long-term reproducibility and maintainability.

Slide 20 — Conclusion & Next Steps

To wrap up, this project delivered a complete end-to-end airline analytics environment — from schema design and ETL pipelines to SQL analytics and Python-based BI visuals.

By unifying fragmented operational and commercial data, the system supports clear insights into reliability, revenue drivers, customer value, and network performance.

Looking forward, next steps could include expanding the dataset with real schedules and fare data, adding predictive models, and developing interactive dashboards that build on this foundation.

Slide 21 — Thank You

Thank you for taking the time to review my project.

This has been an exciting opportunity to bring together database design, ETL engineering, SQL analytics, and Python-driven insights into one unified solution.

I appreciate your time and look forward to your feedback.