Dimensional Modelling and ETL Data Processing for Flight Delays Analysis

Project Report

Group Components:
Lascaro Gianluca, Morbidelli Filippo, Trincia Elio

Course Name and University:
Data Engineering, University Of Coruna

GitHub Repository:
https://github.com/byluca/flight-delays-etl

26 settembre 2025

Indice

1 Introduction and Analytical Objectives

1.1 Introduction

This project focuses on building a robust data pipeline, underpinned by a dimensional model, designed to support future analytical insights into flight delay patterns. By structuring complex raw flight data into an optimized schema, we aim to enable efficient querying and reporting, laying the groundwork for in-depth performance analysis.

1.2 Analytical Objectives

The primary goal of this dimensional model is to provide a robust and efficient foundation for analyzing flight delay patterns. By structuring the data effectively, we aim to support the following analytical objectives:

1. **Performance Analysis of Airlines and Airports**

   Description: To identify which airlines and airports consistently experience the highest or lowest average delays. This includes analyzing delays by origin, destination, and specific airline routes.

   Justification: This analysis is critical for multiple stakeholders. Airlines can use this information to identify operational inefficiencies and improve customer satisfaction. Airports can better allocate resources for ground control and gate management. For consumers, this data helps in making informed travel choices.

2. **Temporal Pattern Analysis of Delays**

   Description: To understand how flight delays vary over different time periods, such as by month, day of the week, or between weekdays and weekends.

   Justification: Identifying temporal trends is key to proactive planning. Airlines can adjust schedules, and airports can manage staffing more effectively during predictably busy periods (e.g., holiday seasons or weekends), thereby mitigating potential delays and improving operational efficiency.

3. **Correlation of Flight Characteristics with Delays**

   Description: To analyze the relationship between physical flight characteristics, such as flight distance and air time, and the duration of arrival or departure delays.

   Justification: Understanding if longer-haul flights are more susceptible to delays, for instance, helps in optimizing route planning and scheduling. This objective provides valuable insights for operational adjustments that can directly impact delay mitigation strategies.

2 Data Model Design

2.1 Star Schema Explanation

The chosen data model is a classic **Star Schema**. This design is optimized for fast analytical queries, separating the quantitative measures of events (the "facts") from their descriptive context (the "dimensions"). The `f_flights` fact table centralizes all quantitative measures related to individual flights, such as `ARRIVAL_DELAY`, `DEPARTURE_DELAY`, `AIR_TIME`, and `DISTANCE`. This fact table is then linked to several descriptive dimension tables:

- `d_airlines`: Provides context about the operating airline, including `IATA_CODE` and `AIRLINE` name.
- `d_airports`: Offers detailed information about origin and destination airports, such as `IATA_CODE`, `AIRPORT` name, `CITY`, `STATE`, `COUNTRY`, `LATITUDE`, and `LONGITUDE`.
- `d_time`: Enriches temporal analysis with attributes like date, year, month, `day_of_week`, `day_name`, and `is_weekend`.

This structure significantly improves query performance for analytical workloads by minimizing the number of joins required for common aggregations.

2.2 Key Design Decisions

The model was designed to be both efficient and resilient, based on the following key decisions:

- **Choice of Star Schema:** A star schema was selected over a more normalized model (like those used in transactional systems) because it dramatically improves query performance for analytical workloads. By centralizing the quantitative data in a fact table and linking to descriptive dimensions, complex aggregations (e.g., `GROUP BY`, `SUM`, `AVG`) become significantly faster as they require fewer table joins.
- **Fact Table Granularity:** The "grain" of the fact table, `f_flights`, is a single, individual flight. This provides the most detailed level of analysis possible, allowing for aggregations at any level (per day, per airport, per airline, etc.). The table exclusively contains numeric measures and foreign keys, making it compact and optimized for mathematical operations.
- **Surrogate Keys vs. Natural Keys:** A crucial design decision was the use of numeric, auto-incrementing surrogate keys (e.g., `airline_id`, `airport_id`) as the primary keys for all dimension tables. While natural keys (e.g., the IATA codes) were available, surrogate keys were chosen for three main reasons:
  - *Performance:* Joining tables on integer keys is significantly faster than joining on string-based keys.
  - *Stability:* A surrogate key is immutable; it will never change. A natural key, like an airline's name or code, could theoretically change (e.g., due to a merger), which would require complex and risky updates across the entire database.
  - *Data Consistency:* Using a single, clean dimension table for airports ensures that an airport is represented consistently, even if its name contained typos in the original source data.
- **Creation of the Time Dimension:** The source data provided date components (`YEAR`, `MONTH`, `DAY`) but did not include a dedicated calendar dimension. We programmatically created the `d_time` dimension to enrich the data. This allows for powerful and intuitive time-based analysis, such as filtering or grouping by month, day of the week, or whether a day is a weekend, without needing to perform complex date calculations in every query.

- **Ensuring Referential Integrity:** The design enforces strict relationships between the fact and dimension tables through foreign key constraints. As discovered during the data pipeline implementation, nearly 500,000 flight records in the source data referred to non-existent airport codes. The ETL process was designed to cleanse these "orphaned" records, ensuring that every flight loaded into the `f_flights` table corresponds to a valid airline, airport, and date in the dimension tables.

2.3 Database Creation Script (SQL Script)

The following script creates all the necessary tables, keys, and constraints for the dimensional model.

```sql
-- ====================================================================
-- SQL Script for Creating the Flight Delays Star Schema
--
-- Project: Dimensional Modelling and ETL Data Processing
-- Author: Lascaro Gianluca, Morbidelli Filippo, Trincia Elio
--
-- Instructions:
-- 1. Create a new schema (database) named 'flight_delays_db'.
-- 2. Run this script to create all the necessary tables.
-- ====================================================================

-- Use the correct database
USE `flight_delays_db`;

-- Drop existing tables in reverse order of creation to avoid foreign key
    issues.
-- This allows the script to be run multiple times safely.
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS `f_flights`;
DROP TABLE IF EXISTS `d_time`;
DROP TABLE IF EXISTS `d_airports`;
DROP TABLE IF EXISTS `d_airlines`;
SET FOREIGN_KEY_CHECKS = 1;


-- ====================================================================
-- Dimension Tables Creation
-- ====================================================================

-- Dimension Table: d_airlines
-- Stores descriptive information about each airline.
CREATE TABLE `d_airlines` (
  `airline_id` INT NOT NULL AUTO_INCREMENT,
  `IATA_CODE` VARCHAR(3) NOT NULL,
  `AIRLINE` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`airline_id`),
  UNIQUE INDEX `iata_code_UNIQUE` (`IATA_CODE` ASC) VISIBLE
) ENGINE=InnoDB;


-- Dimension Table: d_airports
-- Stores descriptive information about each airport.
CREATE TABLE `d_airports` (
  `airport_id` INT NOT NULL AUTO_INCREMENT,
  `IATA_CODE` VARCHAR(3) NOT NULL,
  `AIRPORT` VARCHAR(255) NOT NULL,
  `CITY` VARCHAR(100),
  `STATE` VARCHAR(2),
  `COUNTRY` VARCHAR(3),
  `LATITUDE` DECIMAL(11, 8),
  `LONGITUDE` DECIMAL(11, 8),
  PRIMARY KEY (`airport_id`),
  UNIQUE INDEX `iata_code_UNIQUE` (`IATA_CODE` ASC) VISIBLE
) ENGINE=InnoDB;


-- Dimension Table: d_time
-- A calendar table to store date-related attributes for analysis.
CREATE TABLE `d_time` (
  `time_id` INT NOT NULL AUTO_INCREMENT,
  `date` DATE NOT NULL,
  `year` INT,
  `month` INT,
  `day` INT,
  `day_of_week` INT,
  `is_weekend` BOOLEAN,
  PRIMARY KEY (`time_id`),
  UNIQUE INDEX `date_UNIQUE` (`date` ASC) VISIBLE
) ENGINE=InnoDB;


-- ==================================================================
-- Fact Table Creation
-- ==================================================================

-- Fact Table: f_flights
-- Stores the numeric measures for each flight and links to the dimensions via
   foreign keys.
CREATE TABLE `f_flights` (
  `flight_id` INT NOT NULL AUTO_INCREMENT,
  `time_id` INT NOT NULL,
  `airline_id` INT NOT NULL,
  `origin_airport_id` INT NOT NULL,
  `destination_airport_id` INT NOT NULL,
  `DEPARTURE_DELAY` INT NULL,
  `ARRIVAL_DELAY` INT NULL,
  `AIR_TIME` FLOAT NULL,
  `DISTANCE` INT NULL,
  `CANCELLED` TINYINT NULL,
  `DIVERTED` TINYINT NULL,
  PRIMARY KEY (`flight_id`),
  INDEX `fk_time_idx` (`time_id` ASC) VISIBLE,
  INDEX `fk_airline_idx` (`airline_id` ASC) VISIBLE,
  INDEX `fk_origin_airport_idx` (`origin_airport_id` ASC) VISIBLE,
  INDEX `fk_destination_airport_idx` (`destination_airport_id` ASC) VISIBLE,
  CONSTRAINT `fk_flights_time`
    FOREIGN KEY (`time_id`)
    REFERENCES `d_time` (`time_id`),
  CONSTRAINT `fk_flights_airline`
    FOREIGN KEY (`airline_id`)
    REFERENCES `d_airlines` (`airline_id`),
  CONSTRAINT `fk_flights_origin_airport`
    FOREIGN KEY (`origin_airport_id`)
    REFERENCES `d_airports` (`airport_id`),
  CONSTRAINT `fk_flights_destination_airport`
    FOREIGN KEY (`destination_airport_id`)
    REFERENCES `d_airports` (`airport_id`)
) ENGINE=InnoDB;

-- ==================================================================
-- End of Script
-- ==================================================================
```

Listing 1: SQL script for database schema creation.

2.4 Data Model Diagram (EER Diagram)

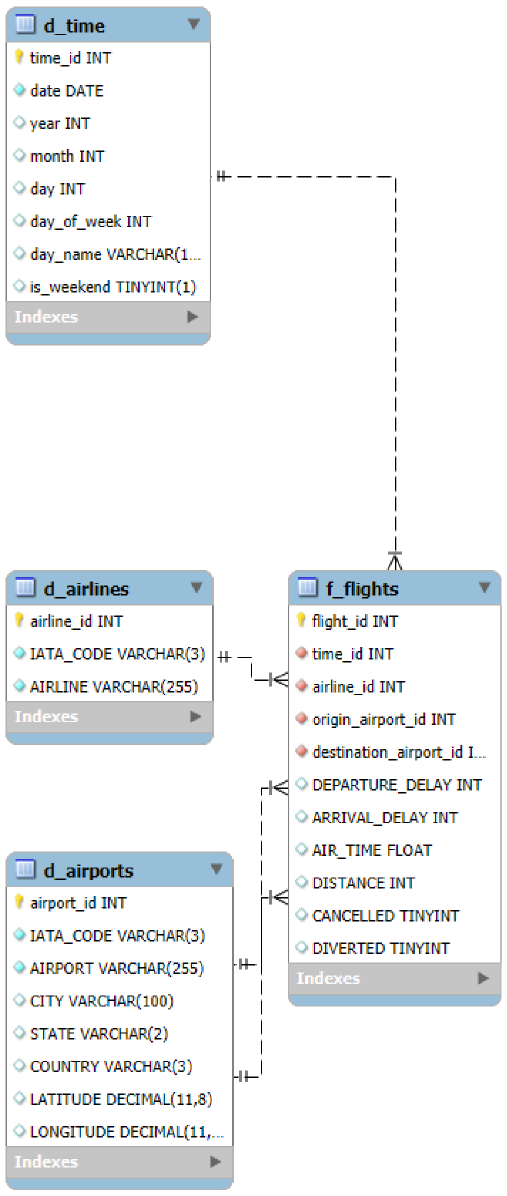

Figura 1: EER Diagram of the Star Schema for Flight Delays Analysis.

3 ETL Pipeline Implementation

3.1 General Flow Description

The ETL (Extract, Transform, Load) pipeline is designed to ingest raw flight data from multiple CSV files, transform it into the star schema model, and load it into a MySQL database.

Extract: Data is read from three separate CSV files (`flights.csv`, `airlines.csv`, `airports.csv`) using the Pandas library in Python.

Transform: The extracted data undergoes several transformations to conform to the star schema. This involves creating the dimension tables (`d_airlines`, `d_airports`, `d_time`) by identifying unique entities and enriching the `d_time` table with additional temporal attributes (e.g., `is_weekend`). The main `f_flights` fact table is then constructed by merging and joining the raw flight records with the newly created dimension keys.

Load: The transformed data, now structured according to the star schema, is loaded into the MySQL database using SQLAlchemy, ensuring efficient and robust insertion into the `d_airlines`, `d_airports`, `d_time`, and `f_flights` tables.

3.2 Data Quality Management and Justification of Choices

A critical step in our ETL process involved a thorough audit of the source data's integrity. Initial checks revealed significant inconsistencies:

```
    FAILURE: Found 307 invalid airport codes... 486,165 orphaned
    ↪ records.
```

Listing 2: Output from the data integrity audit script.

This audit demonstrated a severe violation of referential integrity within the original dataset, where a substantial number of flight records referenced non-existent airport codes. To guarantee a consistent and reliable database for analytical purposes, the design decision was made to remove these "orphaned" rows from the `f_flights` fact table during the transformation phase. This approach ensures that every record loaded into the database is fully attributable to valid dimension entities. Other data cleaning operations included standardizing data types (e.g., converting float to integer for delay values) and handling missing values appropriately to maintain data quality.

3.3 Initial Exploratory Data Analysis

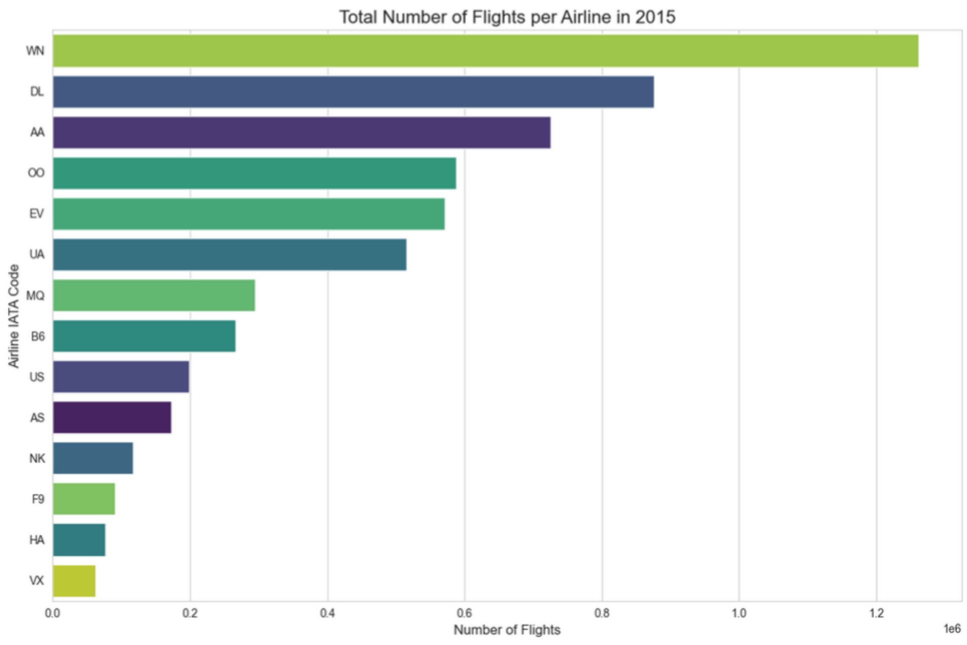

Figura 2: Example EDA: Number of flights per airline.

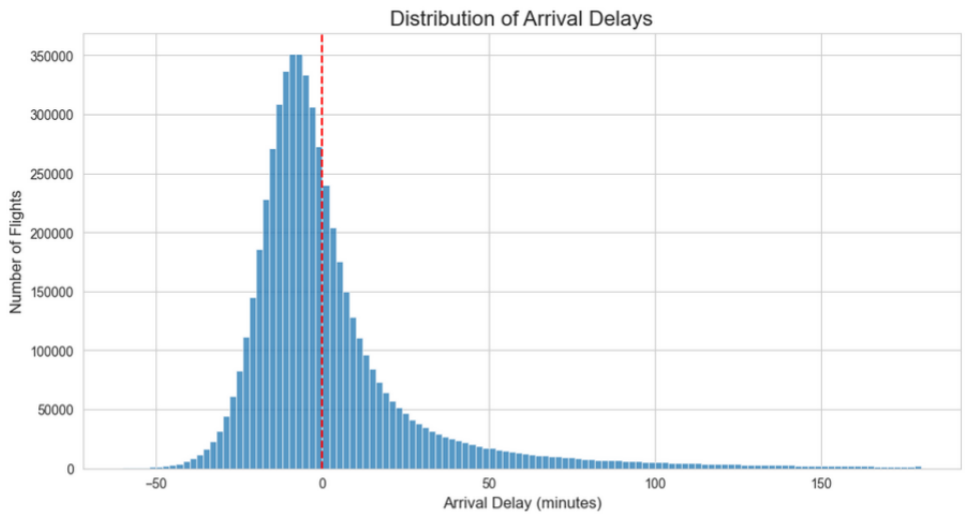

Figura 3: Example EDA: Distribution of arrival delays.

3.4 Load Optimization

To robustly handle the loading of the `f_flights` fact table, which contains over 5 million rows, the `chunksize` parameter was utilized during the SQLAlchemy `to_sql` operation. This approach breaks down the large dataset into smaller, manageable chunks, which are then inserted into the database iteratively. This method significantly improves the stability and efficiency of the loading process, preventing memory issues and database timeouts that could arise from attempting to insert all records in a single transaction.

4 Execution Instructions

4.1 Software Prerequisites

To run the ETL pipeline, the following software and libraries are required:

- Python 3.x
- MySQL Database Server
- Python Libraries: All necessary Python libraries (e.g., Pandas, SQLAlchemy, pymysql) are listed in the `requirements.txt` file.

4.2 Environment Setup

Follow these steps to set up your environment:

1. **Install Python Libraries:** Navigate to the project's root directory in your terminal and install the required Python libraries using pip:

   ```
   pip install -r requirements.txt

   ```

2. **Create MySQL Database:** Ensure a MySQL server is running. Create a new database schema named `flight_delays_db` (or your chosen name) in MySQL Workbench or via the MySQL command line.

3. **Execute SQL Script:** Run the `create_tables.sql` script (provided in Section 2.4) against the `flight_delays_db` schema to create all dimension and fact tables.

4.3 Configuration

The database connection credentials need to be configured. The user must modify the `DB_PASSWORD` variable within the `pipeline.py` file to match their MySQL root password (or the password of the user configured for `flight_delays_db`).

4.4 Execution

Once the environment is set up and configured, execute the main ETL pipeline script from your terminal:

```
python pipeline.py
```

This command will initiate the full ETL process, from data extraction to loading into the MySQL database.

5 Verification and Final Results

This section demonstrates the successful execution of the ETL pipeline and the efficacy of the dimensional model by presenting query results that directly address the analytical objectives outlined in Section 1.

5.1 Query Results 1

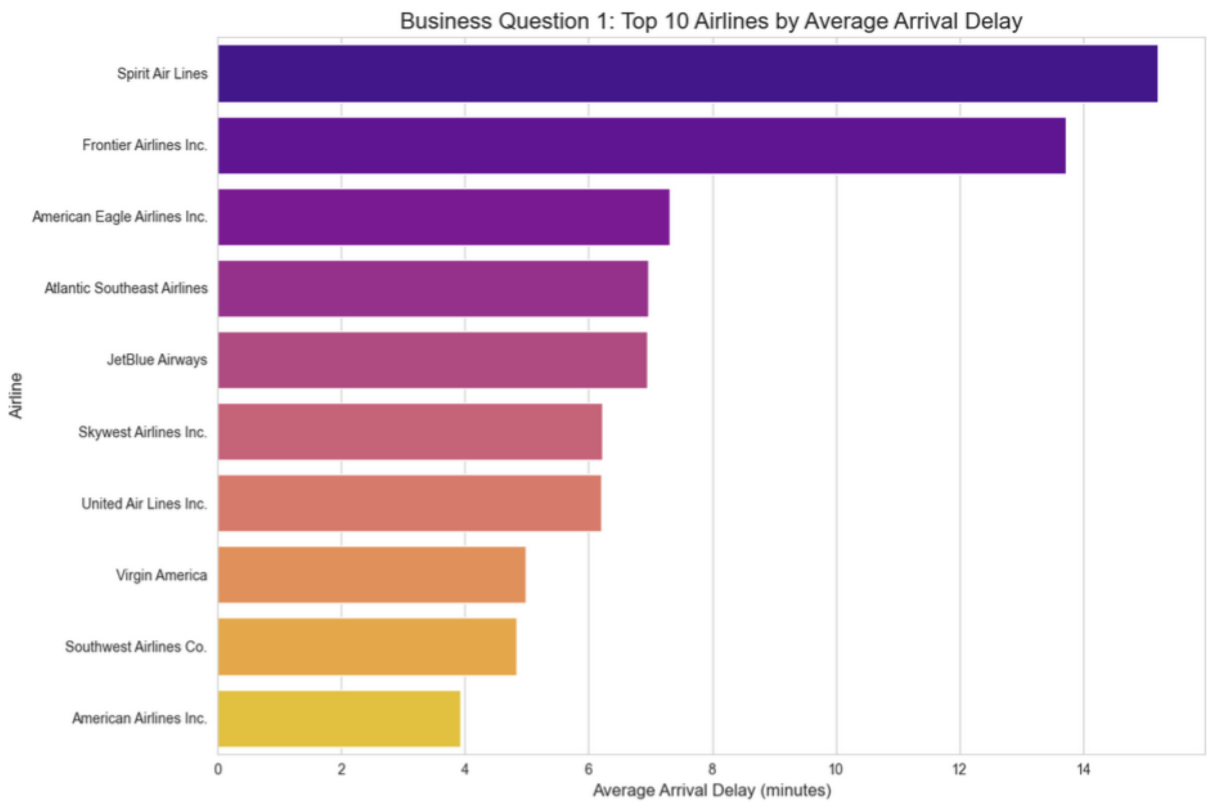

Figura 4: Average Departure Delays for Top 10 Airlines, illustrating operational inefficiencies.

5.2 Query Results 2

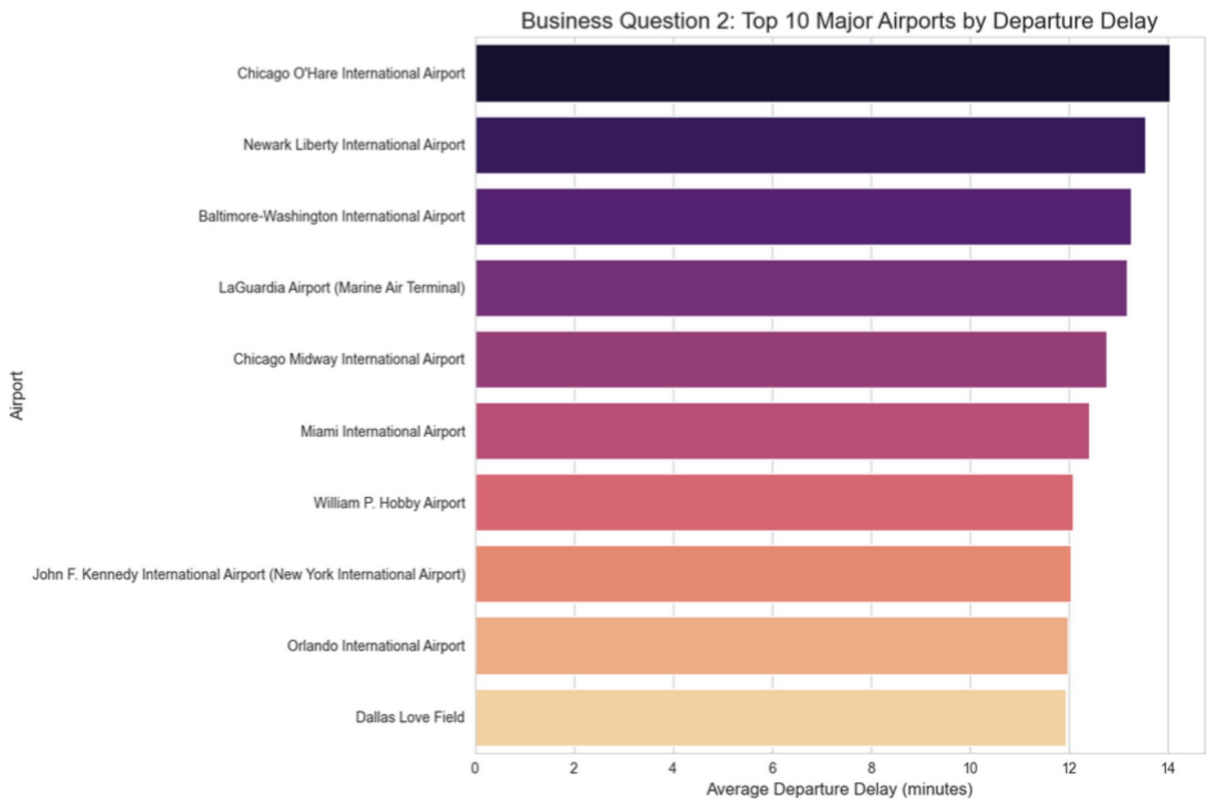

Figura 5: Average Arrival Delays for Top 10 Busiest Airports, highlighting areas for resource reallocation.

5.3 Query Results 3

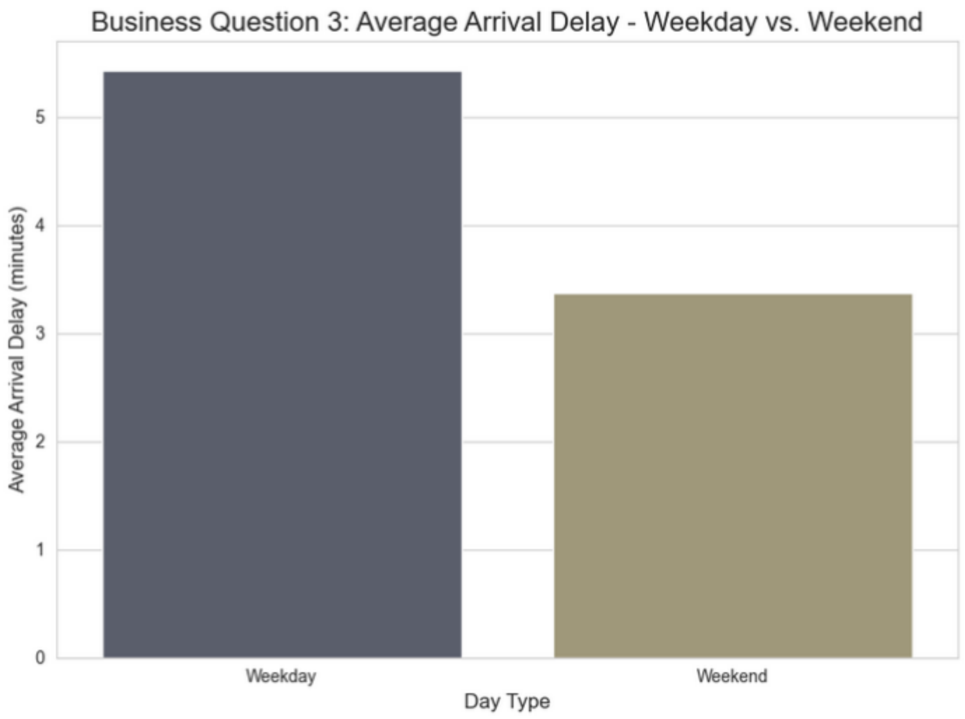

Figura 6: Comparison of Average Delays between Weekdays and Weekends, revealing temporal delay patterns.