

**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY**  
**HO CHI MINH UNIVERSITY OF TECHNOLOGY**  
**FACULTY OF COMPUTER SCIENCE AND ENGINEERING**



**Data Engineering**

---

**Assignment Report**

**Flight Operation Analysis**

---

**Mentor:** Võ Th Ngọc Châu

**Student:** Trn Hà Tun Kit – 2011493  
Lã Minh Đức – 2110132  
Cao Đức Dũng – 2110971  
Nguyễn Minh Thun – 2010663

Ho Chi Minh City, 11/2024



## **Table of Contents**

# 1 Introduction about Aviation Industry

The aviation industry is a critical sector that relies heavily on data and technology to maintain high levels of operational efficiency, ensure optimal fleet and crew scheduling, and support data-driven decision-making. With some modern aircraft capable of generating up to 20 terabytes of data per hour, the industry faces both opportunities and challenges in handling vast amounts of information. This data supports crucial operations managed by organizations like the FAA's Air Traffic Organization, which coordinates services for over 45,000 flights daily. In this presentation, we will explore the impact of data on aviation efficiency and highlighting the industry's reliance on advanced data management and analysis.

## 2 Problem statement

### 2.1 Description

The aviation industry is currently facing a set of interconnected challenges, limitations, and impacts that hinder its ability to operate efficiently and maintain customer satisfaction. These issues stem primarily from data management difficulties, limited predictive capabilities, and the resulting operational inefficiencies.

- **Current Challenges:** The industry struggles with inefficient data handling and analysis, which leads to delays in decision-making. With modern aircraft generating vast amounts of data, the ability to process and interpret this information in real-time is critical. However, existing systems often fail to keep up, resulting in bottlenecks and slower response times.
- **Limitations:** One of the main constraints in the aviation sector is the inability to process large volumes of data in real-time. This limitation restricts the use of predictive analytics, which could otherwise help in foreseeing and mitigating potential operational issues. The lack of predictive analytics means that the industry remains largely reactive rather than proactive in managing its operations.
- **Impact:** The combined effect of these challenges and limitations leads to increased operational costs, as inefficiencies drive up expenses. Additionally, customer satisfaction is

negatively affected due to delays and a lack of seamless service. When data processing and decision-making are delayed, passengers experience disruptions, which can diminish trust and satisfaction with airline services.

This problem statement underscores the need for innovative solutions that can handle and analyze large data volumes efficiently, enabling the industry to transition to predictive, data-driven decision-making for better operational outcomes and enhanced customer experience.

## **2.2 Objectives**

To address the challenges and limitations in the aviation industry, several key objectives have been identified to enhance efficiency, reduce costs, and harness the full potential of data:

- **Real-time Data Processing:** Achieving real-time data processing capabilities is essential for the aviation industry to make timely decisions and respond proactively to dynamic conditions. By processing data as it is generated, airlines and air traffic controllers can anticipate issues, optimize operations, and improve safety.
- **Improve Operational Efficiency:** Increasing operational efficiency is a primary goal, as it allows airlines to better manage their resources, streamline processes, and reduce delays. Enhanced efficiency leads to smoother operations, lower costs, and a more satisfactory experience for passengers.
- **Cost Optimization through Resource Management:** Effective resource management is crucial for cost optimization. By strategically allocating resources—such as fuel, crew, and equipment—airlines can minimize expenses without compromising service quality. Optimized costs contribute to a more sustainable business model and allow airlines to remain competitive in a challenging industry landscape.

These objectives serve as the foundation for a transformative approach in the aviation industry, aiming to create a more responsive, efficient, and cost-effective operational framework powered by data-driven insights.

## 2.3 Proposal

To address the challenges in the aviation industry, this proposal focuses on leveraging data from multiple sources, advanced analytics, and user-friendly interfaces to improve decision-making, operational efficiency, and overall service quality. The key components of the proposal include:

- **Real-time Data Ingestion from Multiple Sources** By integrating real-time data from various sources such as FlightAware, Flightradar24, and Flight Air Map, the aviation industry can access comprehensive and up-to-date information on flights, weather, and air traffic. This continuous data stream allows for improved situational awareness and facilitates faster responses to changing conditions.
- **Advanced Analytics and Predictive Models:** Employing advanced analytics and predictive modeling can help the industry move from reactive to proactive decision-making. Predictive models can forecast potential disruptions, optimize flight schedules, and support fuel and resource management. This data-driven approach enables better preparation for operational challenges, contributing to more efficient and reliable services.
- **User-friendly Dashboards for Reporting:** Developing intuitive and user-friendly dashboards allows stakeholders to easily interpret complex data. These dashboards can present key metrics, trends, and predictive insights, making it easier for decision-makers to act on the information at hand. Clear and accessible data visualization improves operational transparency and enhances collaboration across teams.

This proposal aims to create a robust, data-centric system that supports real-time decision-making and predictive insights, ultimately leading to higher operational efficiency, cost savings, and improved customer satisfaction in the aviation industry.

## 2.4 Data Characteristics

### 2.4.1 Volume

The flight data involves a massive amount of information collected from various sources. This includes:

- Scheduled flight details.
- Real-time flight tracking information (e.g., location, altitude, speed).
- Historical data such as delays, cancellations, and performance metrics.

The size of the data grows continuously due to the high frequency of flights globally, with thousands of flights operating daily.

#### 2.4.2 Velocity

Flight data updates occur at high speeds, particularly with real-time data feeds from sources like FlightRadar24 and FlightAware.

For example:

- GPS coordinates and flight status are updated every few seconds during flight.
- Weather and air traffic data also change dynamically, impacting the speed of data inflow.

#### 2.4.3 Variety

The data encompasses diverse types and formats:

- **Structured Data:** Tabular datasets like Airline On-Time Statistics include columns for flight numbers, departure/arrival times, delays, and cancellations.
- **Semi-Structured Data:** JSON or XML data from APIs (e.g., FlightRadar24) includes hierarchical information like aircraft models, routes, and live positions.
- **Unstructured Data:** Visual data from maps (e.g., Flight Air Map) or textual weather reports.

Each source contributes unique attributes and formats, reflecting the complexity of integration.

#### 2.4.4 Visualization

Flight data is inherently spatial and temporal, making it suitable for visualization in various forms:

- **Geographical Maps:** Real-time flight paths and airport locations provide spatial insights.

- **Charts and Graphs:** Flight volumes, delay distributions, and airline performance metrics can be presented effectively in bar charts or time-series plots.
- **Interactive Dashboards:** Combining different data points, such as weather overlays and congestion heatmaps, enhances user comprehension.

Our group will transform raw data into actionable insights by presenting it through charts and graphs in interactive dashboards or reports, enabling end-users to easily interpret and make informed decisions based on the data

## 3 Technical stacks

### 3.1 MinIO (Data Lake)

MinIO is a high-performance, distributed object storage solution designed to handle large volumes of unstructured data. It is a lightweight and open-source software that provides an S3-compatible interface, which makes it highly compatible with a variety of applications and services. MinIO is often used for building scalable, high-performance data lakes, and it excels in environments where low-latency and high-throughput data access is critical.



Key features of MinIO:

- **S3 Compatibility:** MinIO fully supports the Amazon S3 API, making it easy to integrate with applications and tools that use S3 for storage. This compatibility also allows seamless migration from AWS S3 to MinIO in on-premises or hybrid-cloud environments.
- **Scalability:** MinIO is designed to scale horizontally, meaning you can add more storage nodes to handle increasing data volumes without compromising performance. It supports distributed object storage across multiple servers, ensuring data availability and fault tolerance.

- Performance: MinIO is optimized for high-performance workloads, delivering fast data throughput and low-latency access. It is particularly suitable for workloads involving large files, such as media files, backups, and big data analytics.
- Security: MinIO offers strong security features, including server-side encryption, secure access using AWS IAM (Identity and Access Management) policies, and end-to-end encryption for data in transit. It supports integration with various identity providers for access control.
- Simple & Lightweight: Unlike traditional enterprise-grade object storage systems, MinIO is simple to install, configure, and manage. Its lightweight architecture makes it ideal for use in environments with resource constraints, such as edge computing.

Differences between **MinIO** and other popular data lake solutions:



Feature	MinIO	Amazon S3	Google Cloud Storage	HDFS	Azure Blob Storage
<b>Type</b>	Open-source, Distributed Object Storage	Managed Object Storage	Managed Object Storage	Distributed File System (DFS)	Managed Object Storage
<b>Deployment</b>	On-premises or Any Cloud	AWS Cloud only	Google Cloud only	On-premises or Cloud	Microsoft Azure only
<b>Cost</b>	Low (No recurring costs, Open-source)	Pay-per-use (Can be expensive with large data volumes)	Pay-per-use (Can be expensive with large data volumes)	Expensive setup and management	Pay-per-use (Can be expensive with large data volumes)
<b>Vendor Lock-in</b>	No (Can be deployed anywhere)	Yes (AWS dependency)	Yes (Google Cloud dependency)	No (on-premises, but complex setup)	Yes (Microsoft Azure dependency)
<b>API Compatibility</b>	S3 Compatible	S3 API	S3 API	Custom API	S3 API
<b>Performance</b>	High-performance, low-latency access	High-performance with AWS integration	High-performance with Google Cloud integration	High-performance for batch processing	High-performance with Azure integration
<b>Scalability</b>	Horizontally scalable, suitable for unstructured data	Highly scalable within AWS ecosystem	Highly scalable within Google Cloud ecosystem	Scalable but complex to manage	Highly scalable within Azure ecosystem
<b>Data Handling</b>	Ideal for unstructured data	Ideal for unstructured data	Ideal for unstructured data	Better suited for structured or semi-structured data	Ideal for unstructured data
<b>Integration with PySpark</b>	Seamless integration with PySpark and other tools	Seamless with AWS services (e.g., AWS Lambda)	Seamless with Google Cloud tools	Requires custom integration with Hadoop ecosystem	Seamless with Azure services (e.g., Azure Databricks)

Why MinIO is Suitable for the Project:

- **Cost-efficiency:** MinIO is open-source and allows for on-premises or cloud deployments, making it a more cost-effective solution compared to the pay-per-use pricing models of S3, Google Cloud Storage, and Azure Blob Storage.
- **Flexibility:** MinIO is not tied to a specific cloud provider, providing flexibility for deploying in multi-cloud or hybrid environments. This suits your project's need to integrate multiple data sources.
- **Performance:** MinIO provides high throughput and low-latency access to unstructured data, which is crucial for handling large flight data logs and sensor data.

- **Scalability:** MinIO can scale horizontally to handle growing data volumes, making it a suitable solution as your data lake expands with more flight data.

This makes MinIO the optimal choice for building your data lake, ensuring efficient data storage and processing for flight data analytics.

## 3.2 PySpark (Processing)

PySpark is the Python API for Apache Spark, which is a distributed computing framework designed for processing large datasets in parallel across a cluster of computers. Apache Spark is one of the most widely used big data processing engines because of its ability to handle batch processing, real-time streaming, machine learning, and graph processing efficiently. PySpark allows Python developers to leverage Spark's capabilities with a familiar Python interface.



Key Features of PySpark:

- **Distributed Processing:** PySpark can distribute tasks across a cluster, enabling parallel processing of large datasets.
- **In-memory Processing:** It utilizes in-memory computation, which significantly boosts performance for certain workloads.
- **Support for Various Data Formats:** PySpark supports a wide range of file formats such as CSV, Parquet, JSON, Avro, and more.
- **DataFrames and SQL:** It includes DataFrames (similar to Pandas) for structured data processing and supports SQL-like queries through the Spark SQL API.
- **Machine Learning:** It integrates MLlib for machine learning tasks, making it easy to run ML models on large datasets.

- **Scalability:** PySpark is capable of handling data at petabyte scale across distributed clusters.
- **Integration:** PySpark integrates well with other data storage systems like HDFS, MinIO, S3, YugabyteDB, and databases, making it versatile for different use cases.

#### Comparison with other data processing technologies

Feature/Aspect	PySpark	Hadoop (MapReduce)	Apache Flink	Dask	Apache Beam
<b>Programming Language</b>	Python (also supports Scala, Java)	Java, Python (via Hadoop Streaming)	Java, Scala, Python	Python (Dask API)	Java, Python, Go
<b>Processing Model</b>	Batch, Streaming, MLlib	Batch (MapReduce)	Stream processing (real-time)	Batch, Streaming	Batch, Streaming
<b>Ease of Use</b>	Easy to use for Python developers	More complex, requires Java/MapReduce knowledge	Easy to use, with a focus on stream processing	Easy to use for Python users	Simplified API for batch and stream tasks
<b>Scalability</b>	High scalability, distributed	High scalability, distributed	High scalability, low latency	Scalable, often used for in-memory tasks	Scalable, designed for cloud-native workflows
<b>Fault Tolerance</b>	Built-in fault tolerance	Built-in fault tolerance	Built-in fault tolerance	Built-in fault tolerance	Built-in fault tolerance
<b>Real-time Processing</b>	Yes (via Spark Streaming)	No (Hadoop MapReduce is batch-only)	Yes (designed for stream processing)	Yes (via Dask's distributed scheduler)	Yes (supports real-time processing)
<b>Batch Processing</b>	Yes	Yes	Yes	Yes	Yes
<b>Integration with Data Sources</b>	Easily integrates with HDFS, S3, MinIO, Databases	Integration with HDFS, HBase, etc	Integration with Kafka, HDFS, etc.	Integration with local and distributed file systems	Integration with multiple sources, including Google Cloud
<b>Machine Learning</b>	Integrated MLlib	Not designed for machine learning	Supports stream-based ML (via FlinkML)	Limited (can use external libraries)	Limited (supports batch ML via external systems)
<b>Cost and Resources</b>	Can be resource-heavy but optimized for large datasets	Resource-heavy, especially for MapReduce jobs	Generally lighter, especially for stream processing	More efficient for small to medium-sized clusters	Optimized for cloud environments, cost-effective for cloud workloads
<b>Community and Ecosystem</b>	Large community, many integrations	Large community, but outdated in terms of modern frameworks	Growing community, strong in stream processing	Smaller community, but growing rapidly	Strong community in cloud-native ecosystems

PySpark is ideal for our flight data collection and delay analysis project due to its scalability and ability to process large datasets efficiently. It supports both batch and real-time processing, seamlessly integrating with our data lake (MinIO) and data warehouse (Yugabyte). PySpark's powerful data transformation and analysis capabilities enable effective handling of flight delay data, making it a crucial tool for extracting actionable insights.

### **3.3 Yugabyte (Data warehouse)**

Yugabyte is a high-performance, distributed SQL database designed for cloud-native applications. It offers strong consistency and horizontal scalability, making it suitable for handling large volumes of transactional data. Yugabyte combines the best features of traditional relational databases and NoSQL systems, supporting both SQL (PostgreSQL-compatible) and NoSQL APIs. This flexibility allows it to be used for a variety of use cases, including the storage of structured and semi-structured data in our project.



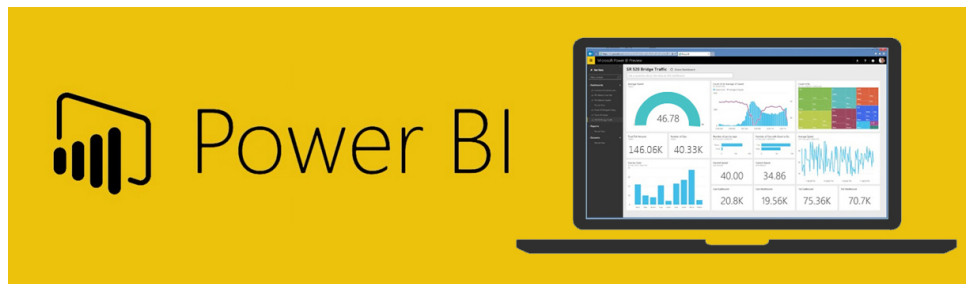
Comparison with other technologies

Feature	Yugabyte	Amazon Redshift	Google BigQuery	MySQL
Type	Distributed SQL Database	Data Warehouse	Data Warehouse	Relational Database
Data Model	Relational (PostgreSQL-compatible)	Columnar Storage for Analytics	Columnar Storage for Analytics	Relational
Scalability	Horizontal Scalability, Distributed	Scalable (AWS-only)	Scalable (Google Cloud-only)	Vertical Scaling (Limited)
Performance	High for OLTP, Low Latency	Optimized for OLAP	Optimized for OLAP	Good for OLTP
Consistency	Strong Consistency, ACID	Eventual Consistency (Analytics)	Eventual Consistency (Analytics)	Strong Consistency, ACID
Query Language	SQL (PostgreSQL)	SQL (Redshift-Specific)	SQL (BigQuery SQL)	SQL (PostgreSQL-specific)
Use Case	OLTP	OLAP	OLAP	OLTP, Complex Queries
Deployment	On-premises/Cloud	AWS-only	Google Cloud only	On-premises/Cloud
Data Integrity	ACID, Multi-region Replication	Strong Consistency (Single-region)	Strong Consistency (Single-region)	ACID (Single-node or Cluster)
Integration with PySpark	Seamless Integration	Limited (AWS services)	Limited (Google Cloud services)	Good (via JDBC/ODBC)

Yugabyte is ideal for this project due to its distributed SQL architecture, providing both scalability and high availability. It combines the reliability of relational databases with the flexibility of NoSQL, making it well-suited for handling large flight delay datasets while ensuring consistency and integration with other technologies in the stack.

### 3.4 Power BI (Analysis and visualization)

Power BI is a business analytics tool developed by Microsoft that allows users to visualize data, share insights, and make data-driven decisions. It offers a range of features for data transformation, exploration, and presentation in a visually appealing and interactive format. With its cloud-based services, Power BI enables easy integration with various data sources, including cloud storage, databases, and even real-time data streams. Its intuitive drag-and-drop interface makes it accessible to both technical and non-technical users.



#### Key Features:

- **Data Connectors:** Integrates with a wide range of data sources, including databases, cloud services, and real-time data.
- **DirectQuery:** Allows real-time access to large datasets without importing data into Power BI, improving performance and scalability.
- **Customizable Dashboards:** Users can create personalized dashboards to monitor key metrics like flight delays.
- **Real-time Data Streaming:** Capabilities to visualize real-time data for up-to-date decision-making.
- **Advanced Analytics:** Supports machine learning models, DAX (Data Analysis Expressions), and custom calculations for deeper insights.
- **Interactive Visualizations:** Offers a variety of visualizations like bar charts, line graphs, and heatmaps to represent trends and data patterns.

Power BI's DirectQuery feature is especially beneficial for big data analysis, allowing real-time access to large and continuously changing datasets. This capability, combined with Power BI's other features like customizable dashboards and advanced analytics, makes it an excellent choice for visualizing and analyzing flight delay data in this project.

## 4 How the application works

### 4.1 Workflow

The aviation data application operates through a structured workflow that involves **Data Collection, Data Storage and Processing, and Analytics and Reporting**. This approach ensures that vast amounts of real-time data are collected, efficiently processed, and converted into actionable insights.

#### Step 1: Data collection

The application's workflow begins with **Data Collection**, where information is gathered from multiple aviation-related sources such as FlightAware, Flightradar24, Airline On-Time Statistics, and Flight Air Map. These sources provide essential data, including flight schedules, delays, and real-time positioning of aircraft. To create a comprehensive view of operations, the application integrates this data in real-time from additional sources like air traffic control systems and weather services, which helps in forming a complete picture of flight conditions necessary for informed decision-making.

#### Step 2: Data storage and processing

Once collected, data is organized and processed in the **Data Storage and Processing** stage. Initially, raw data from diverse sources is stored in a data lake (blob storage), which can accommodate various unstructured data types. To enable real-time data transformation, the application employs PySpark, allowing for the rapid processing of large datasets. The processed data is then stored in a Yugabyte data warehouse, optimized for complex querying and analysis, providing an efficient setup for generating analytical insights. This data is structured using a star schema, a modeling technique that organizes information into fact and dimension tables, which facilitates efficient retrieval and supports detailed analysis.

#### Step 3: Analytics and reporting

In the **Analytics and Reporting** stage, the application analyzes historical and real-time flight data to identify potential flight delays. This data-driven approach enables the aviation

industry to shift from reactive to proactive operations, allowing better anticipation of delays. Additionally, the application visualizes key performance indicators (KPIs) and other relevant metrics related to delays through user-friendly dashboards. These visualizations make complex data accessible and interpretable, supporting quick, data-driven decision-making and improving overall operational efficiency.

## 4.2 Benefits to stakeholders

The aviation data application provides significant benefits to various stakeholder groups, each of which plays a crucial role in ensuring smooth and efficient operations:

- **Operational Teams:** With access to real-time insights, operational teams can improve scheduling and resource allocation. This capability allows for better alignment of flights, staff, and equipment, leading to enhanced efficiency and minimizing potential disruptions.
- **Maintenance Crews:** Management teams benefit from data-driven strategic planning based on both historical and real-time flight operation data. This comprehensive insight allows managers to make informed decisions, optimize resource allocation, and plan for the future, ultimately driving organizational success and resilience.
- **Management:** Management teams benefit from data-driven strategic planning based on both historical and real-time flight operation data. This comprehensive insight allows managers to make informed decisions, optimize resource allocation, and plan for the future, ultimately driving organizational success and resilience.
- **Passengers:** Passengers experience an enhanced travel experience due to fewer delays, optimized scheduling, and reduced cancellations. By improving operational efficiency and reliability, the application contributes to a smoother journey for travelers, increasing customer satisfaction and building loyalty.

Overall, the application's capabilities in real-time data processing, predictive maintenance, and strategic planning empower each stakeholder group, fostering a more efficient, proactive, and customer-centric aviation industry.



## 5 Implementation plan

The implementation plan for the aviation data application is divided into four phases: **Planning and Requirements Gathering, Development, Testing, and Deployment**. Each phase is designed to ensure that the application is built, tested, and deployed effectively, meeting both technical and business objectives.

### Phase 1: Planning and Requirements Gathering

This initial phase, lasting 1.5 weeks (Week 3 to mid-Week 4), involves defining the project scope and objectives. It includes identifying the necessary data sources, such as FlightAware, Flightradar24, and Flight Air Map, and outlining infrastructure requirements. This foundational phase sets the direction for the subsequent development work.

### Phase 2: Development

Spanning 2.5 weeks (mid-Week 4 to Week 7), this phase focuses on setting up the big data infrastructure. This involves configuring Yugabyte for data warehousing, establishing blob storage for the data lake, and utilizing PySpark for distributed data processing. Additionally, data ingestion and processing pipelines are developed to handle real-time data from various sources, ensuring the system is equipped for efficient data handling.

### Phase 3: Testing

Lasting 2 weeks (Week 7 to Week 9), this phase includes pilot testing with selected flight routes to validate data accuracy and system performance. Stress testing is also conducted to ensure that the application can scale effectively under heavy data loads, confirming its robustness before full deployment.

### Phase 4: Deployment

The final phase, taking 2 weeks (Week 9 to Week 11), involves the full-scale implementation of the system, making it fully operational. This deployment phase ensures that the application is accessible to stakeholders, with all functionalities optimized for real-world use.

## **6 Result**

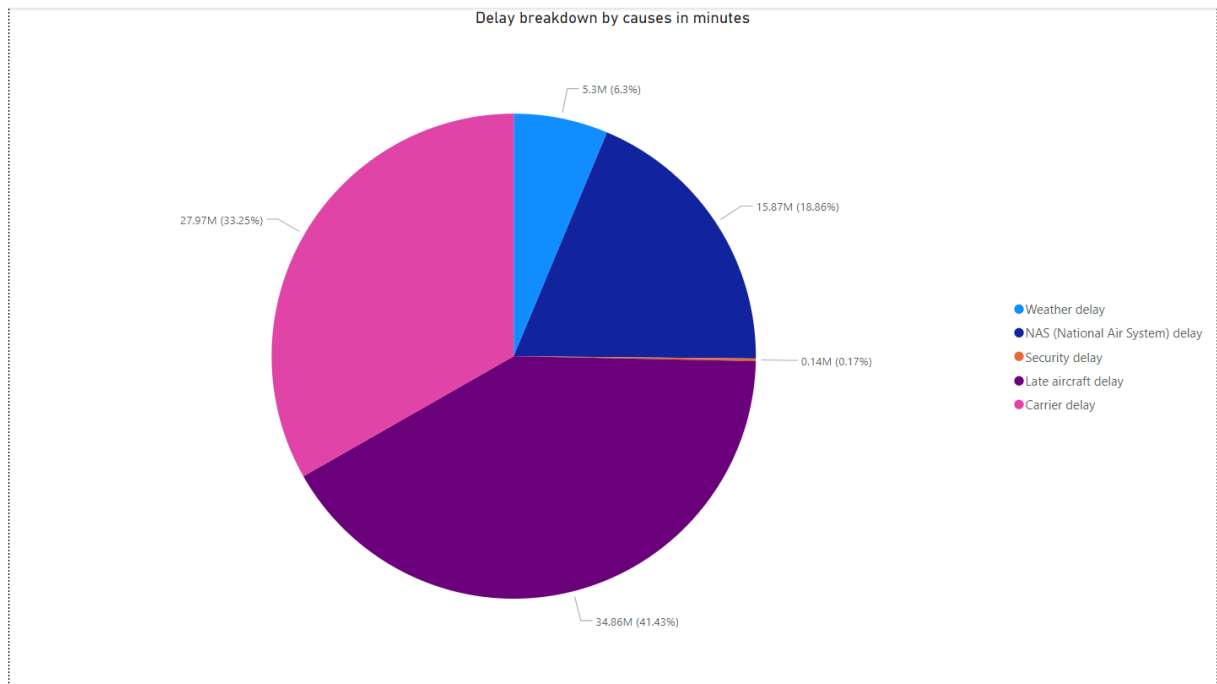
Our group has developed an application designed to streamline the collection, processing, and analysis of flight data. The application collects data from multiple trusted sources, ensuring the accuracy and reliability of the information. These raw datasets are then stored in a Data Lake, which serves as a centralized repository for efficient and scalable storage, allowing future use and exploration.

To process the stored data, we utilized PySpark, a powerful tool for distributed data processing, ensuring that the data is cleaned, transformed, and prepared for analysis. After processing, the structured data is stored in Yugabyte, a distributed SQL database known for its scalability and fault tolerance. This enables fast, reliable querying and supports advanced analytical operations.

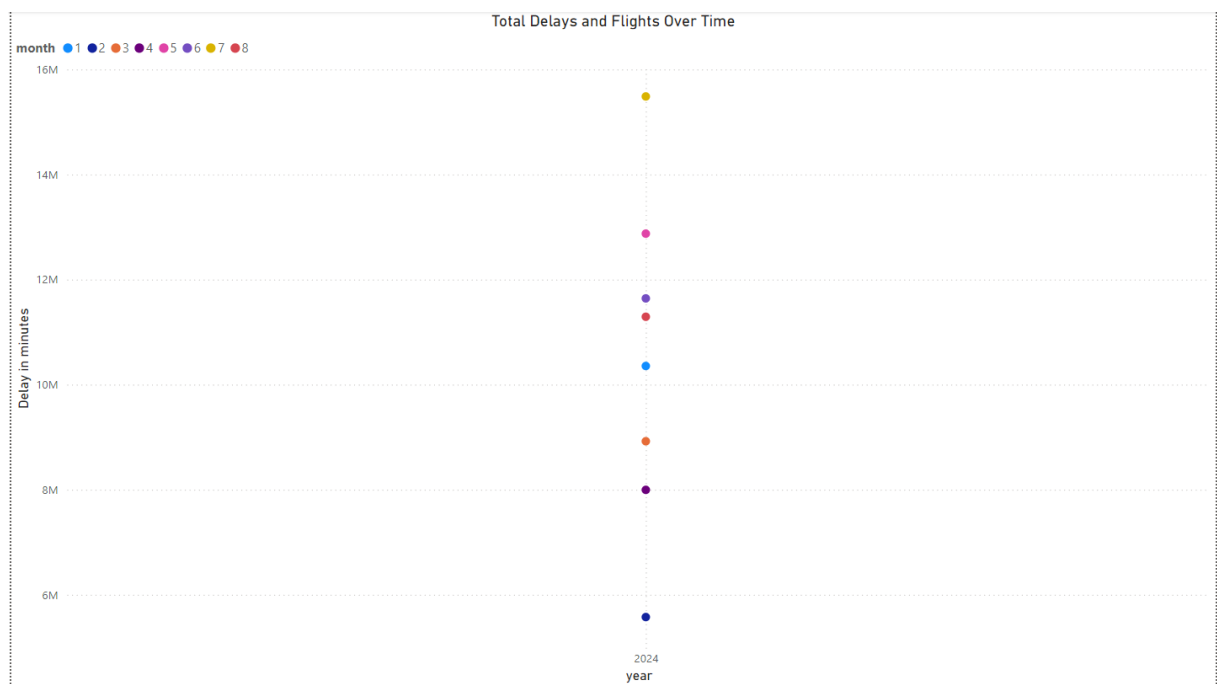
For visualization and reporting, our team integrated Power BI to create customized dashboards. These dashboards are tailored to display critical insights such as flight delays, operational trends, and carrier performance. By leveraging the dynamic capabilities of Power BI, users can interact with the data in real-time, drill down into specific metrics, and generate reports suited to their requirements.

This workflow demonstrates an end-to-end solution for flight data analysis, incorporating robust data engineering practices and intuitive analytics tools to deliver actionable insights.

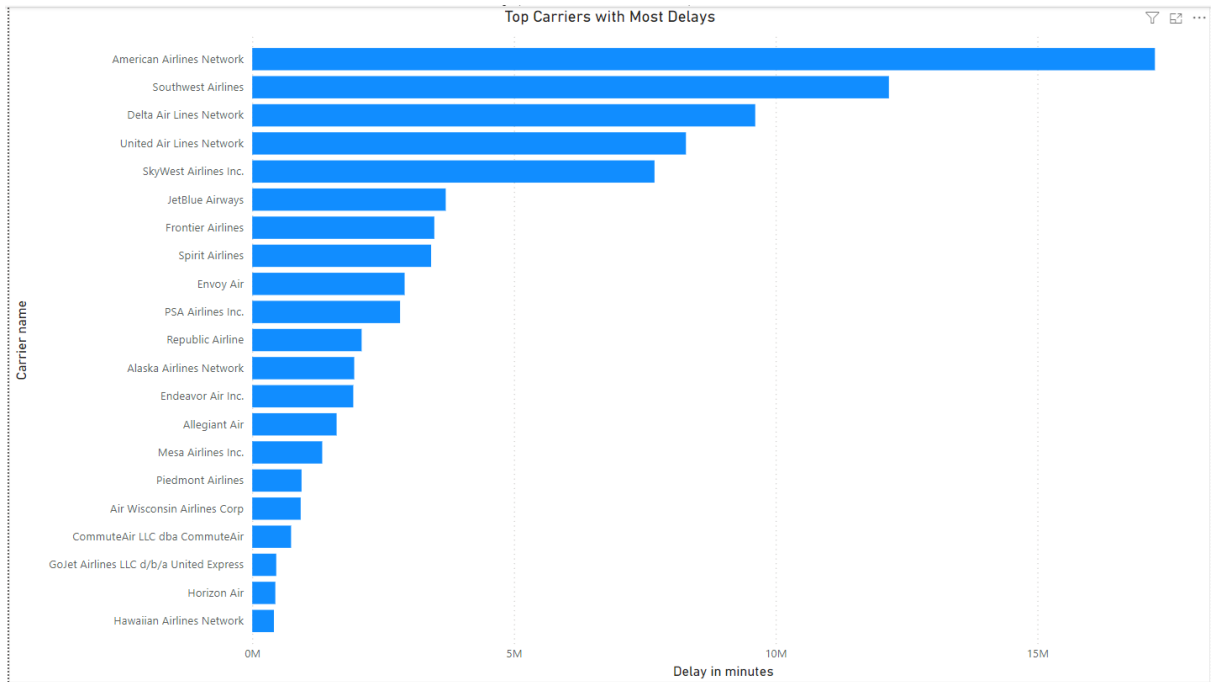
Below are some charts and graphs created with powerBI:



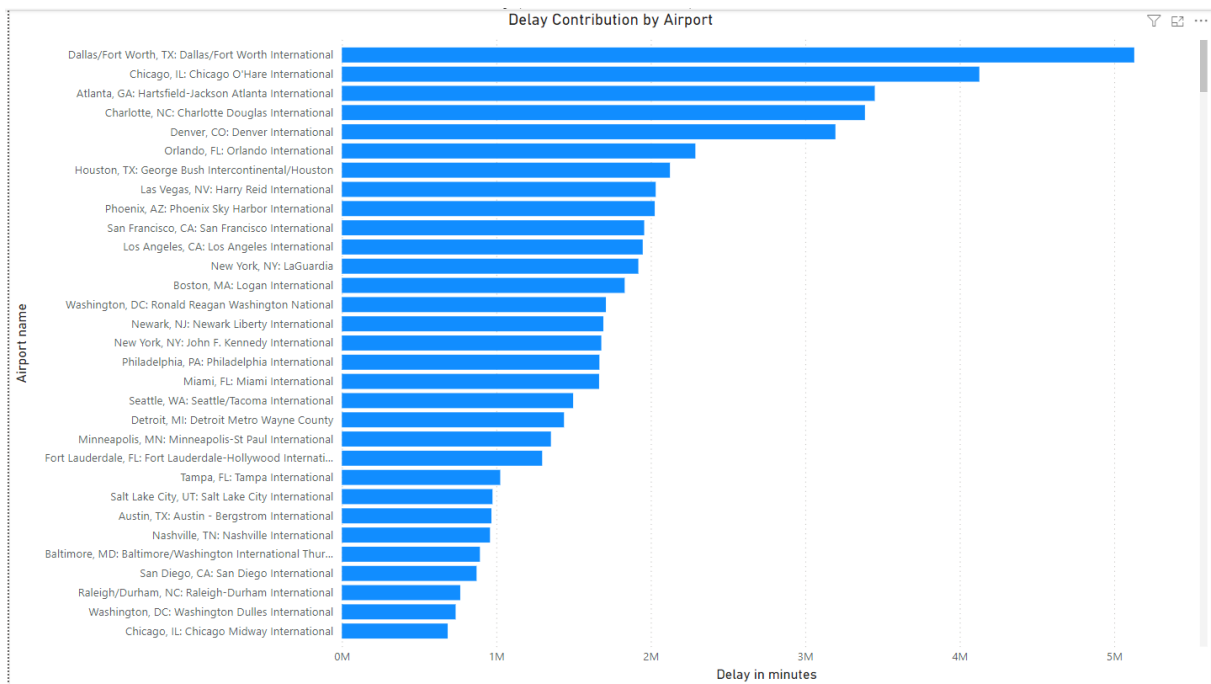
**Figure 1:** *Delay breakdown by causes in minutes*



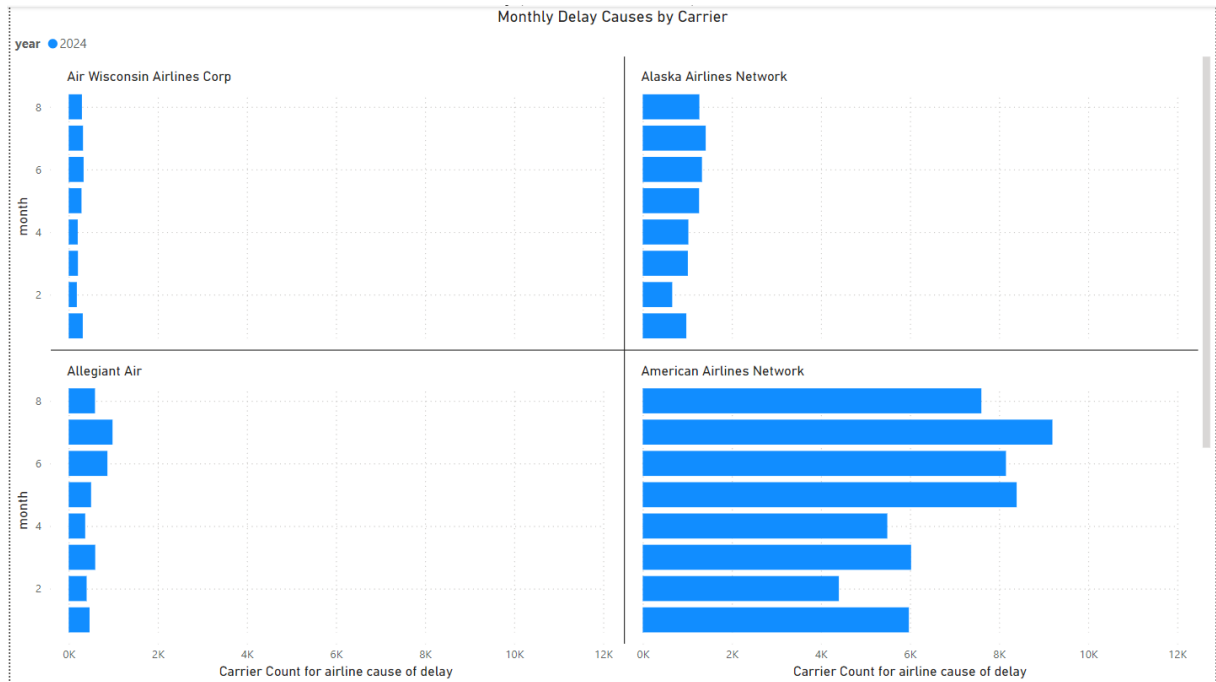
**Figure 2:** *Delay over time*



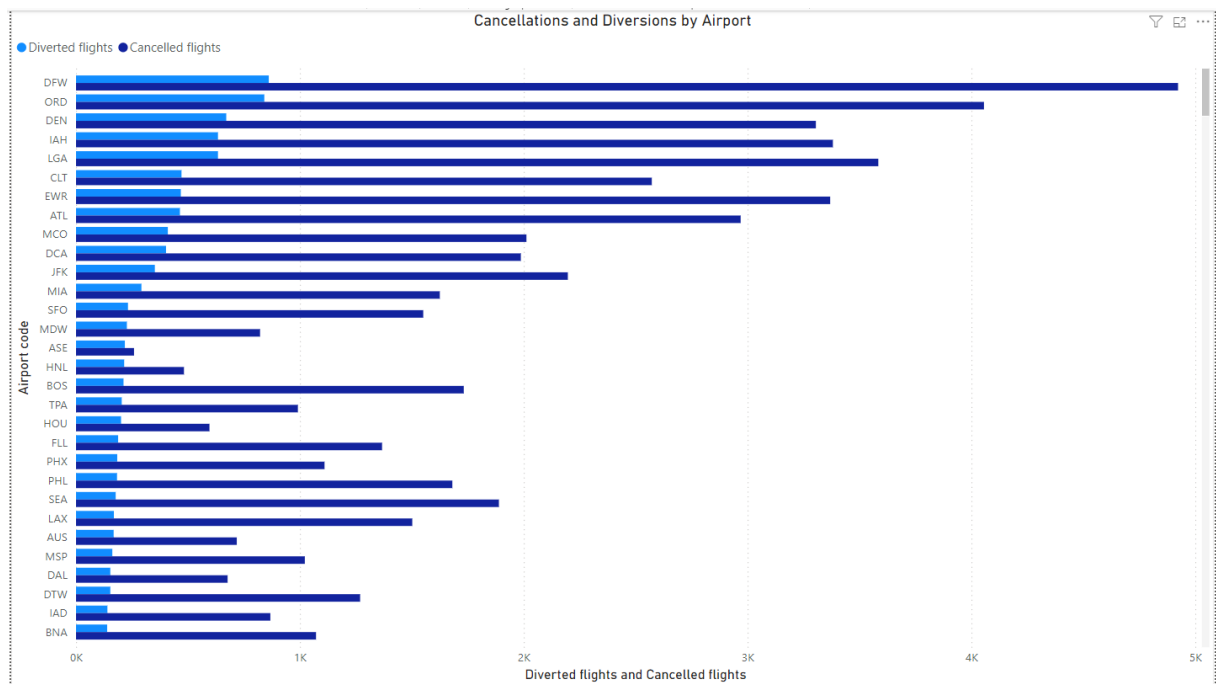
**Figure 3: Delay by carrier**



**Figure 4: Delay by airport**



**Figure 5: Monthly delay cause by carrier**



**Figure 6: Cancellations and diversions by airport**

Flights and Delays by Year and Airport				
year	2024		Total	
Airport name	Flights	Delay in minutes	Flights	Delay in minutes
Atlanta, GA: Hartsfield-Jackson Atlanta International	227,270.00	3,449,917.00	227,270.00	3,449,917.00
Dallas/Fort Worth, TX: Dallas/Fort Worth International	209,988.00	5,129,876.00	209,988.00	5,129,876.00
Chicago, IL: Chicago O'Hare International	209,176.00	4,127,016.00	209,176.00	4,127,016.00
Denver, CO: Denver International	209,093.00	3,195,297.00	209,093.00	3,195,297.00
Charlotte, NC: Charlotte Douglas International	167,279.00	3,386,429.00	167,279.00	3,386,429.00
Los Angeles, CA: Los Angeles International	130,061.00	1,947,743.00	130,061.00	1,947,743.00
Phoenix, AZ: Phoenix Sky Harbor International	128,445.00	2,025,453.00	128,445.00	2,025,453.00
Las Vegas, NV: Harry Reid International	125,885.00	2,031,645.00	125,885.00	2,031,645.00
Seattle, WA: Seattle/Tacoma International	124,308.00	1,497,259.00	124,308.00	1,497,259.00
Houston, TX: George Bush Intercontinental/Houston	111,632.00	2,123,901.00	111,632.00	2,123,901.00
Orlando, FL: Orlando International	110,229.00	2,288,544.00	110,229.00	2,288,544.00
New York, NY: LaGuardia	108,873.00	1,919,321.00	108,873.00	1,919,321.00
Washington, DC: Ronald Reagan Washington National	95,751.00	1,709,355.00	95,751.00	1,709,355.00
Boston, MA: Logan International	95,489.00	1,830,760.00	95,489.00	1,830,760.00
Newark, NJ: Newark Liberty International	95,136.00	1,692,754.00	95,136.00	1,692,754.00
San Francisco, CA: San Francisco International	87,705.00	1,957,288.00	87,705.00	1,957,288.00
Detroit, MI: Detroit Metro Wayne County	87,428.00	1,437,995.00	87,428.00	1,437,995.00
Minneapolis, MN: Minneapolis-St Paul International	85,358.00	1,353,368.00	85,358.00	1,353,368.00
New York, NY: John F. Kennedy International	81,836.00	1,679,601.00	81,836.00	1,679,601.00
Philadelphia, PA: Philadelphia International	80,717.00	1,667,172.00	80,717.00	1,667,172.00
Salt Lake City, UT: Salt Lake City International	76,357.00	975,029.00	76,357.00	975,029.00
Miami, FL: Miami International	75,086.00	1,665,246.00	75,086.00	1,665,246.00
Nashville, TN: Nashville International	67,221.00	959,001.00	67,221.00	959,001.00
Baltimore, MD: Baltimore/Washington International Thurgood Marshall	66,384.00	893,567.00	66,384.00	893,567.00
Fort Lauderdale, FL: Fort Lauderdale-Hollywood International	63,542.00	1,296,614.00	63,542.00	1,296,614.00
San Diego, CA: San Diego International	63,110.00	871,953.00	63,110.00	871,953.00
Austin, TX: Austin - Bergstrom International	57,992.00	968,138.00	57,992.00	968,138.00
Chicago, IL: Chicago Midway International	54,188.00	685,019.00	54,188.00	685,019.00
Tampa, FL: Tampa International	53,867.00	1,025,180.00	53,867.00	1,025,180.00
Washington, DC: Washington Dulles International	52,012.00	736,293.00	52,012.00	736,293.00
Dallas, TX: Dallas Love Field	48,730.00	650,034.00	48,730.00	650,034.00
Portland, OR: Portland International	47,240.00	578,091.00	47,240.00	578,091.00
St. Louis, MO: St Louis Lambert International	45,439.00	624,072.00	45,439.00	624,072.00
Total	5,022,488.00	84,147,494.00	5,022,488.00	84,147,494.00

Figure 7: Flights and delay by year and airport

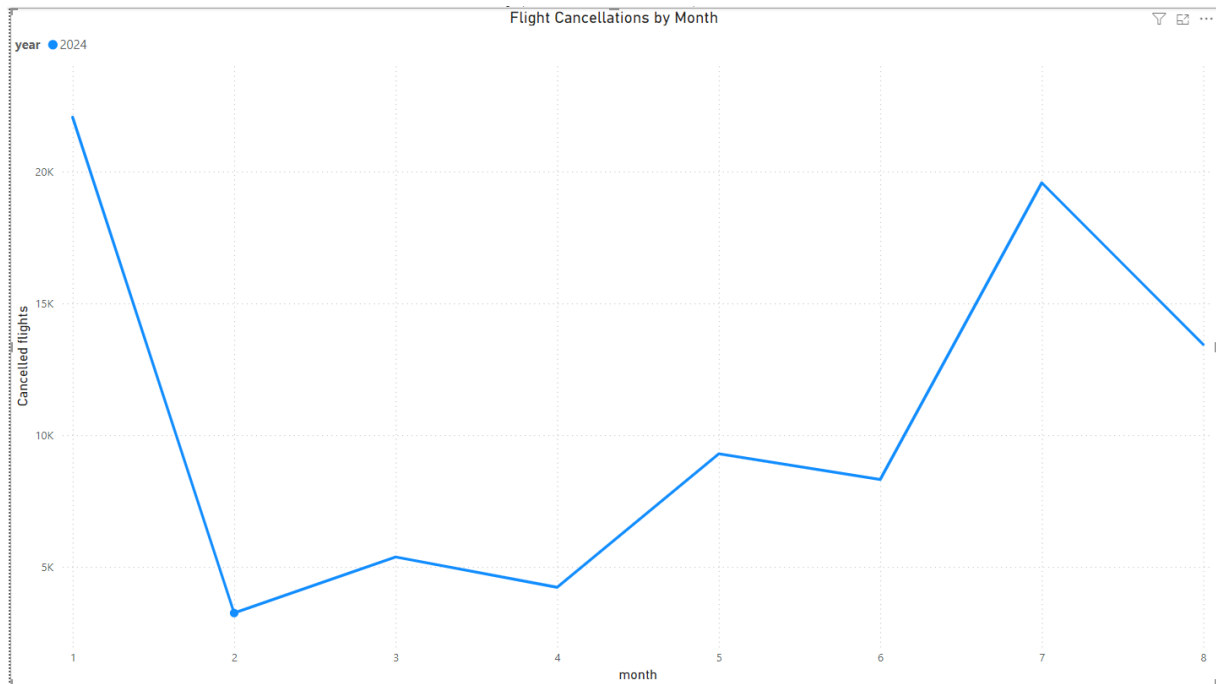


Figure 8: Flight cancellations by month