Batch Processing Architecture for GTFS Dataset

Development Phase

### Introduction

The development phase focused on building a robust, end-to-end batch data processing pipeline for GTFS (General Transit Feed Specification) data, fully aligned with the cloud-native, online-first approach outlined in the conception phase. Leveraging Google Colab, Google Drive, BigQuery, and Looker Studio, the system was designed for maximum scalability, reproducibility, and ease of use—while maintaining high standards for data quality, modularity, and transparency.

All code, detailed workflow documentation, and analytical outputs are available in the [project GitHub repository](https://github.com/Ahsan97Javed/gtfs-batch-pipeline).

### Development Environment and Tools

To minimize complexity and maximize accessibility, the entire pipeline was developed and deployed using **fully managed cloud services**, eliminating the need for local software installs, manual server management, or Docker containerization. This strategic choice allowed the project to:

* + Ensure platform independence and reproducibility for any user with a browser and Google account.
  + Leverage Google’s built-in security, access control, and compute resources.
  + Focus on business and data engineering goals rather than infrastructure maintenance.
  + **Key tools and platforms:**
  + **Google Colab:** For all data engineering, batch processing, cleaning, and aggregation.
  + **Google Drive:** For raw, cleaned, and intermediate dataset storage.
  + **Google BigQuery:** For cloud-based storage of aggregated results and advanced SQL analytics.
  + **Google Looker Studio:** For real-time dashboarding and data visualization.
  + **GitHub:** For version control, documentation, and code sharing.
  + **Note:** As all processing was executed in managed cloud environments, containerization (Docker) was not necessary and thus not implemented.

### Modular Microservices Pipeline

The development strictly followed a modular, microservices-inspired architecture, mapping each major function to a dedicated, well-documented Jupyter/Colab notebook. This promotes code reusability, clear separation of concerns, and simplifies future maintenance or scaling.

#### Ingestion Microservice

**Objective:**

Efficiently and reliably load all GTFS .txt files (including trips.txt, routes.txt, stops.txt, and related files) from Google Drive, while performing schema and data integrity validation.

**Implementation:**

* [01\_ingestion\_gtfs.ipynb](https://github.com/Ahsan97Javed/gtfs-batch-pipeline/blob/main/notebooks/ingestion_gtfs.ipynb)
* Validated presence and format of mandatory GTFS columns.
* Automated identification of missing, duplicate, or malformed records, and logged validation issues for transparency.
* Loaded data frames serve as the foundation for downstream preprocessing.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 1: Previews of several key GTFS tables (calendar.txt, feed\_info.txt, stops.txt, routes.txt, calendar\_dates.txt, agency.txt) loaded from Google Drive. Each table is displayed with its first few rows and column names to confirm successful ingestion

A computer screen shot of a black screen

AI-generated content may be incorrect.

Figure 2: Additional GTFS tables (trips.txt, attributions.txt, stop\_times.txt) with initial records previewed. Early inspection of sample data helps catch missing or malformed entries at the ingestion stage.

#### Preprocessing Microservice

**Objective:**

Deliver clean, standardized GTFS data suitable for reliable analytics and machine learning.

**Implementation:**

* [02\_preprocessing\_gtfs.ipynb](https://github.com/Ahsan97Javed/gtfs-batch-pipeline/blob/main/notebooks/preprocessing_gtfs.ipynb)
* Applied systematic data cleaning: removed duplicate rows, handled missing/null values, and standardized timestamp formats and categorical values per GTFS specification.
* Thorough documentation and logging at each cleaning step, enabling transparency and reproducibility.
* Output stored as clean CSVs in a dedicated Google Drive folder for further processing.

****

Figure 3Loaded and validated shapes of cleaned GTFS tables after preprocessing.

#### Aggregation Microservice

**Objective:**

Extract key operational and analytical metrics from the cleaned GTFS data for strategic and operational insights.

**Implementation:**

* [03\_aggregation\_gtfs.ipynb](https://github.com/Ahsan97Javed/gtfs-batch-pipeline/blob/main/notebooks/aggregation_gtfs.ipynb)
* Aggregated multiple critical KPIs, including:
  + Total number of trips per route, joined with descriptive route names for clarity.
  + Most frequently visited stops (stop popularity).
  + Trip frequency distribution by day of the week, enabling temporal trend analysis.
  + Route distribution by transit agency (for multi-agency datasets).
* All outputs are saved as CSV files and prepared for direct ingestion by BigQuery.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 4: Aggregation scripts and outputs showing key GTFS metrics, including trip counts, stop popularity.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 5: Aggregation scripts and outputs showing key GTFS metrics, including temporal patterns, and agency route shares.

#### Output Microservice

**Objective:**

Deliver the batch-aggregated results in a format suitable for advanced analytics, visualization, and business intelligence.

**Implementation:**

* [04\_output\_gtfs.ipynb](https://github.com/Ahsan97Javed/gtfs-batch-pipeline/blob/main/notebooks/output_gtfs.ipynb)
* Loaded all aggregation outputs, performed final QA checks, and exported them to Google BigQuery (cloud-based data warehouse) for durable storage and instant SQL access.
* All major result tables were verified in the BigQuery UI, confirming schema and row counts.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 6Summary of output microservice steps, with QA-checked aggregations exported to Google BigQuery for cloud storage and instant analysis.

#### Visualization Microservice

**Objective:**

Transform raw data outputs into actionable, user-friendly visualizations for business and operational decision-making.

**Implementation:**

* Built an interactive dashboard in Google Looker Studio, directly connected to the BigQuery dataset.
* Created and documented visualizations for:
  + Top 10 most popular stops (bar chart)
  + Top 10 busiest routes (bar chart)
  + Distribution of routes per agency (pie chart)
  + Trips per day of week (bar chart)
* All charts are titled and described for clarity.

A screenshot of a graph

AI-generated content may be incorrect.

Figure 7: Interactive analytics dashboard displaying top stops, busiest routes, agency distribution, and weekly trip patterns for actionable transit insights.

### Modular Microservices Pipeline

* **Reproducibility:**  
  Every step, from raw data ingestion to dashboard generation, can be reproduced by anyone with access to Google Colab and Drive.  
  The [README.md](https://github.com/Ahsan97Javed/gtfs-batch-pipeline/blob/main/README.md) details how to run each phase, including environment setup and required permissions.
* **Collaboration:**  
  All code and documentation are version controlled via GitHub, facilitating team contributions, peer review, and iterative development.

### Data Security and Governance

* All processing was performed within Google-managed cloud platforms, inheriting strong authentication and data protection practices.
* Access to raw data and results is controlled through Google Drive and BigQuery IAM settings, ensuring only authorized users can access sensitive information.

### Reflections, Challenges, and Lessons Learned

* **Cloud-First Design:**The move to fully managed cloud services significantly reduced operational friction and increased reproducibility, compared to local or containerized alternatives.
* **No Docker Required:**   
  Containerization (Docker) was not necessary due to the inherent reproducibility and managed nature of Colab, Drive, and BigQuery environments.  
  This also aligns with industry shifts toward serverless and managed data engineering platforms.
* **Adaptability:**The pipeline was developed iteratively, with flexibility to handle data quirks, schema variations, and the technical limitations of initial local attempts.

### Summary

The development phase resulted in a fully modular, cloud-native GTFS batch processing and analytics platform. Each stage of the pipeline—ingestion, preprocessing, aggregation, output, and visualization—was carefully implemented as a dedicated microservice using Google Colab and managed Google Cloud tools. This design ensures **scalability**, **reproducibility**, and **ease of use**, requiring only a browser and Google account for full re-execution.  
All code and documentation are openly available in the GitHub repository, and every major analytical output is visually summarized in an interactive dashboard. The platform not only demonstrates best practices in modern data engineering but also provides actionable insights for transit planners and analysts through clear, user-focused visualizations.

All code, documentation, and visual outputs are publicly available at:  
<https://github.com/Ahsan97Javed/gtfs-batch-pipeline>