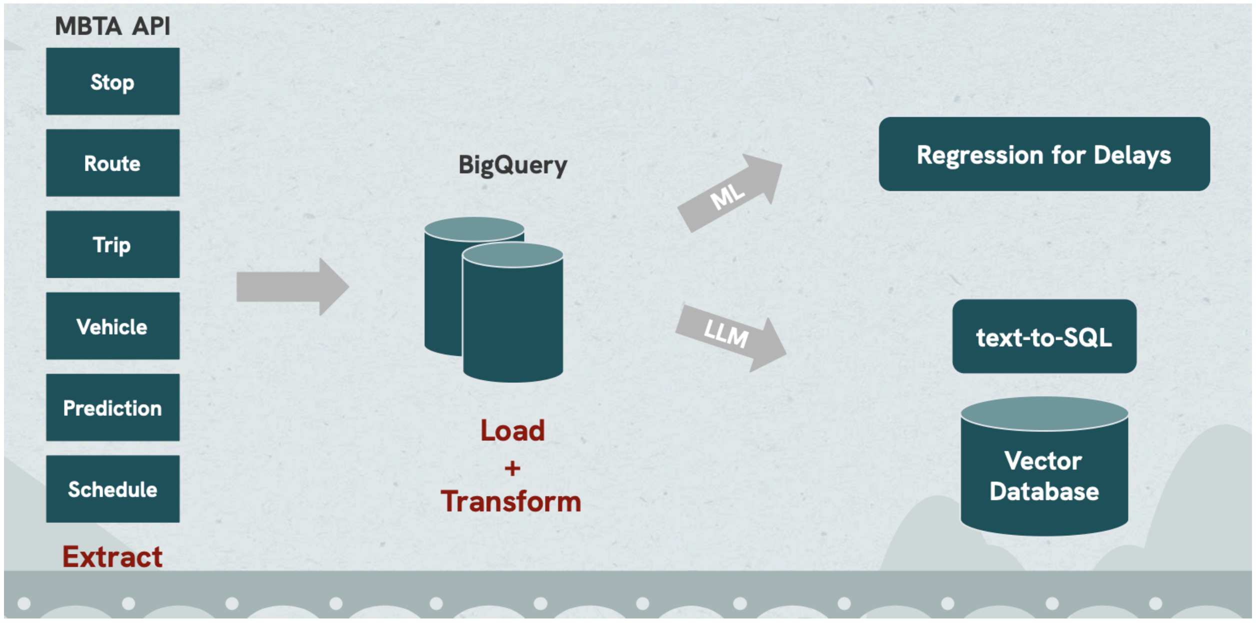
Group - 8: Optimizing Public Transit using MBTA Data

Team Members: Aishwarya Rauthan, Atharva Lokhande, Yu-Chun(Lila) Su, Neeharika Kamireddy

Project Overview

In this project, we have been working with the MBTA dataset, initially extracting data through the MBTA API, which is updated weekly. This dataset was then used to populate an Apache Superset dashboard for visualization and analysis. In the subsequent phase, we leveraged AutoML techniques to predict delays within the transit system. For the current phase of the project, we have introduced several new tools and approaches. Additionally, we have implemented Text2SQL using Streamlit, enabling users to query the database using natural language, and incorporated Generative AI to enhance our system's capabilities, providing more advanced insights and automation in the analysis of MBTA data.



Vector Database

To establish a Retrieval Augmented Generation (RAG) framework for analyzing MBTA transport delays, we leveraged Pinecone to create a vector database for storing data as embeddings. Our focus was to explore delay patterns based on factors such as the day of the week and the time of day. Using an initialized embeddings model, we encoded data from columns such as delay, day, and time_of_day into vector representations and stored them in Pinecone for efficient retrieval.

When a user submits a query, such as *"What are the typical delays on Saturday nights?"*, the framework processes it by converting the query into an embedding vector using a sentence transformer model. The vector is then queried against the Pinecone database to retrieve the most relevant data points. For each match, metadata, including day, time_of_day, and delay, is extracted, along with a relevance score. This information is formatted into a contextual prompt, which is passed to the Gemini generative model to produce a conversational summary of the retrieved data points. The system ensures that the generated response is both accurate and user-friendly, making complex delay patterns comprehensible for end-users.

<u>LLM Modeling</u>
We developed a vector database and designed a text-to-SQL application powered by a Large Language Model (LLM). This interactive platform enables users to input natural language queries, automatically generating and executing the corresponding SQL queries on a BigQuery database. The application provides users with both the generated SQL query and the resulting table, making database access intuitive even for those without SQL expertise.

When a user inputs a query, such as *"Show the top 5 municipalities in Boston with the largest average delays,"* the application transforms the natural language query into a valid SQL statement using the GenAI model, displays the generated SQL query for transparency and learning, executes the query on the database, and presents the results in a user-friendly table format. This approach bridges the gap between natural language and SQL, creating a seamless interface for database interaction.

To enable this functionality, we created a unified table by joining the prediction, schedule, route, and stop tables. The combined table includes key features such as delay minutes, delay time, delay day, route, stop, and municipality for each delay record. We automated table updates with a Cloud Run function, scheduled weekly using Cloud Scheduler on Google Cloud Platform. We also customized a text-to-SQL script from a professor's Colab example to suit our database schema and user requirements. This involved scripting a requirements.txt file and a Dockerfile for dependency management and containerization. Finally, we deployed the application through Cloud Shell for secure and scalable usage.

Once deployed, users can interact with the application using natural language prompts, such as:

- *"Return the route name and the average delay for each route. Show only the top 5 routes with the largest average delay."*
- *"Return the stop name and the average delay for each stop. Show only the top 10 stops with the largest average delay on Mondays."*
- *"Return the municipality name and the average delay for each route. Show only the bottom 5 municipalities with the smallest average delay."*

The application leverages the text-to-SQL model to convert these prompts into SQL queries, executes them on BigQuery, and displays the results. This system not only makes complex database interactions accessible but also provides users with SQL query examples for educational purposes. The modular design and adaptability of the application enable users to extend its functionality to similar datasets, offering a versatile and scalable solution for text-to-SQL integration.

<u>Challenges</u>
We faced a challenge with data retrieval accuracy in the RAG framework, leading us to discontinue its use. For instance, when querying for average delay times on Monday nights, the system sometimes returned results for unrelated contexts, like Tuesday evenings. This issue likely stemmed from limitations in the embeddings model's ability to distinguish subtle differences, overlapping patterns in the data, or noise introduced by time-related variables. These challenges underscored the need for more refined embeddings or enhanced query logic, which exceeded the scope of our current setup.

<u>Conclusions</u>
This phase introduced innovative tools for analyzing MBTA transit delays, leveraging a vector database and generative AI to streamline complex queries. The LLM-powered text-to-SQL tool further simplified

database interaction, making transit data accessible to non-technical users. While the RAG framework faced challenges with data retrieval accuracy, the progress made highlights the potential for scalable and interactive solutions to improve transit analysis and decision-making.

By building a real-time data pipeline and applying predictive modeling, we uncovered critical delay patterns, allowing for targeted interventions to improve service reliability. These insights can help MBTA optimize resource allocation, reduce commuter frustration, and enhance operational efficiency. By providing actionable recommendations, our analysis directly supports strategic decision-making to achieve better performance and higher customer satisfaction, aligning public transit services with the growing demands of urban mobility.

Future Steps

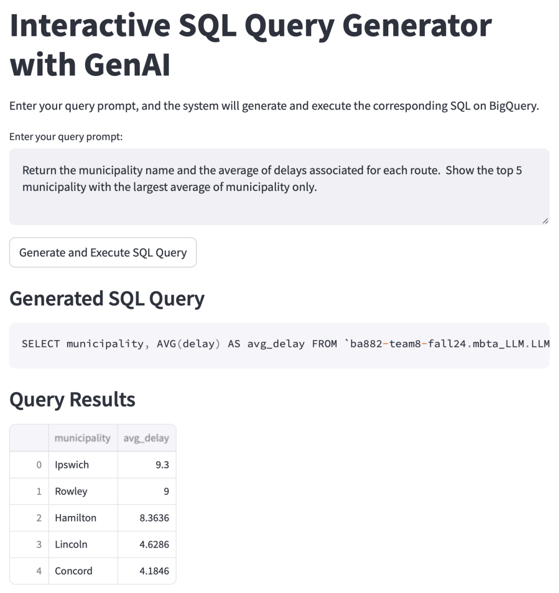
- Collaborating with MBTA to enhance data granularity and expanding predictive capabilities to include weather and event impacts are key next steps.
- Enhancing our LLM-powered text-to-SQL tool to incorporate additional features, such as visualizing delays or scheduled timings, to provide more comprehensive and user-friendly insights.

References

GitHub Repository: https://github.com/ajrauthan/BA882-Team-8
Text-to-SQL Application: https://streamlit-rag-app-1029824820040.us-central1.run.app
Google Drive Link for Superset Reporting Video:
https://drive.google.com/drive/folders/12aQCD6VTjlEkvmNOYjs7AmI6Gm6bIpOP?usp=share_link
Google Drive Link for Text-to-SQL Video:
https://drive.google.com/drive/folders/1UrVJ9rOx_C3xQCDni2E1-zu2BtTpaaJX?usp=sharing
Google Drive Link for Cloud Scheduler Video:
https://drive.google.com/drive/folders/1vM1s6mdrCMOOO-Q00v-YSl4ztXddV9UV?usp=share_link

Appendix

Interactive Application:



Interactive SQL Query Generator with GenAI

Enter your query prompt, and the system will generate and execute the corresponding SQL on BigQuery.

Enter your query prompt:

Return the municipality name and the average of delays associated for each route. Show the top 5 municipality with the largest average of municipality only.

Generated SQL Query

```
SELECT municipality, AVG(delay) AS avg_delay FROM `ba882-team8-fall24.mbta_LLM.LLM
```

Query Results

	municipality	avg_delay
0	Ipswich	9.3
1	Rowley	9
2	Hamilton	8.3636
3	Lincoln	4.6286
4	Concord	4.1846