Automated Audit: NYC Government Jobs Postings & Payroll Data

Presented by Michael Galo

Project Overview & Agenda

Today, I'll explore my journey in building a lightweight, scaleable data engineering pipeline for NYC government job postings and payroll data, augmented with external analytics.

1 Tech Stack & Rationale

The tools I chose and why they're essential.

2 Data Architecture

A visual walkthrough of the data flow.

3 Challenges & Iterations

Challenges and iterations from V1.0 to V2.1.

4 Live Demonstration

Seeing the pipeline in action.

5 Stretch Goals & Lessons Learned

Key takeaways and future plans.

Tech Stack & Rationale

My tech stack was chosen to be lightweight, free (save for DataGrip) and scaleable, a modern data lakehouse approach.

Data Exploration & Preparation:

- Numbers (macOS): For initial data exploration and converting Lightcast sheets to CSVs.
- DataGrip: For ad-hoc data querying and validating my SQL

Storage:

- MinIO: Chosen for its S3 compatibility, enabling seamless future deployment to AWS for scalable object storage.

Data Ingestion & Transformation:

- Python Scripts: Custom scripts for API data ingestion and fuzzy matching transformations.
- External Python App for CSV → Parquet conversion
- Custom SQL: For analytical queries to display for data visualization based on business logic.

Data Querying & Cataloging:

- DuckDB & Ducklake: Fast for larger data. A powerful combination for efficient SQL query execution and maintaining a structured lakehouse catalog.

Orchestration:

- Prefect: Manages workflow automation, scheduling, and monitoring for reliable pipeline execution.

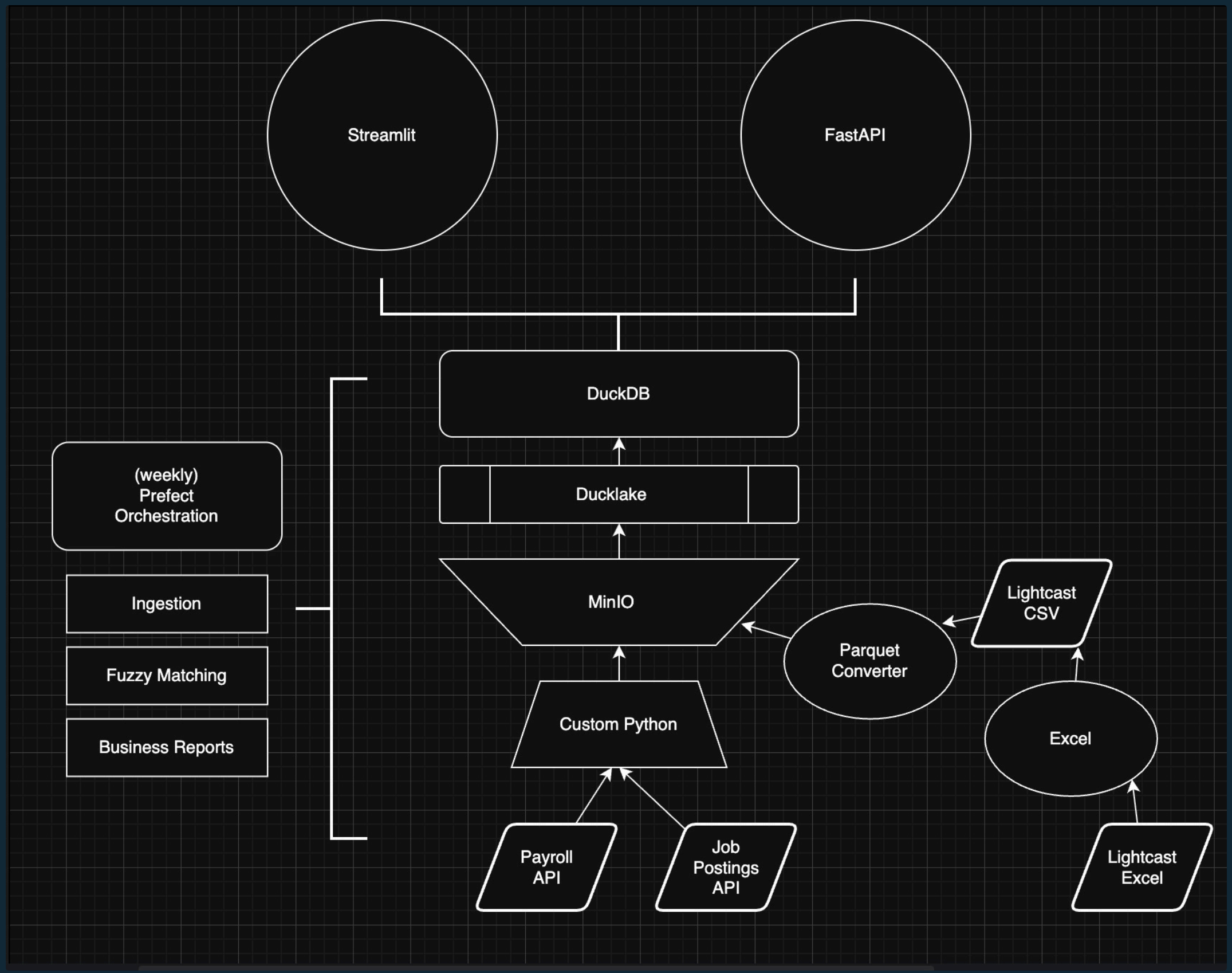
Data Access & API:

- FastAPI: Provides a high-performance, easy-to-use API for accessing processed data.
- Swagger: Integrated for automatic API documentation and interactive testing, streamlining developer experience.

Data Visualization:

- Streamlit: Chosen for rapid development of interactive data visualizations, making insights accessible to all stakeholders.

Data Pipeline Architecture



Challenges

It was never going to be smooth-sailing.

Lightcast Table Selection Dilemma

Initially, choosing the right Lightcast table was ambiguous at best—four nearly identical tables, each with subtle differences. I researched SOC & O*NET and decided to go with the largest and most standardized dataset from what was offered.

Optimizing Ingestion & Matching Efficiency

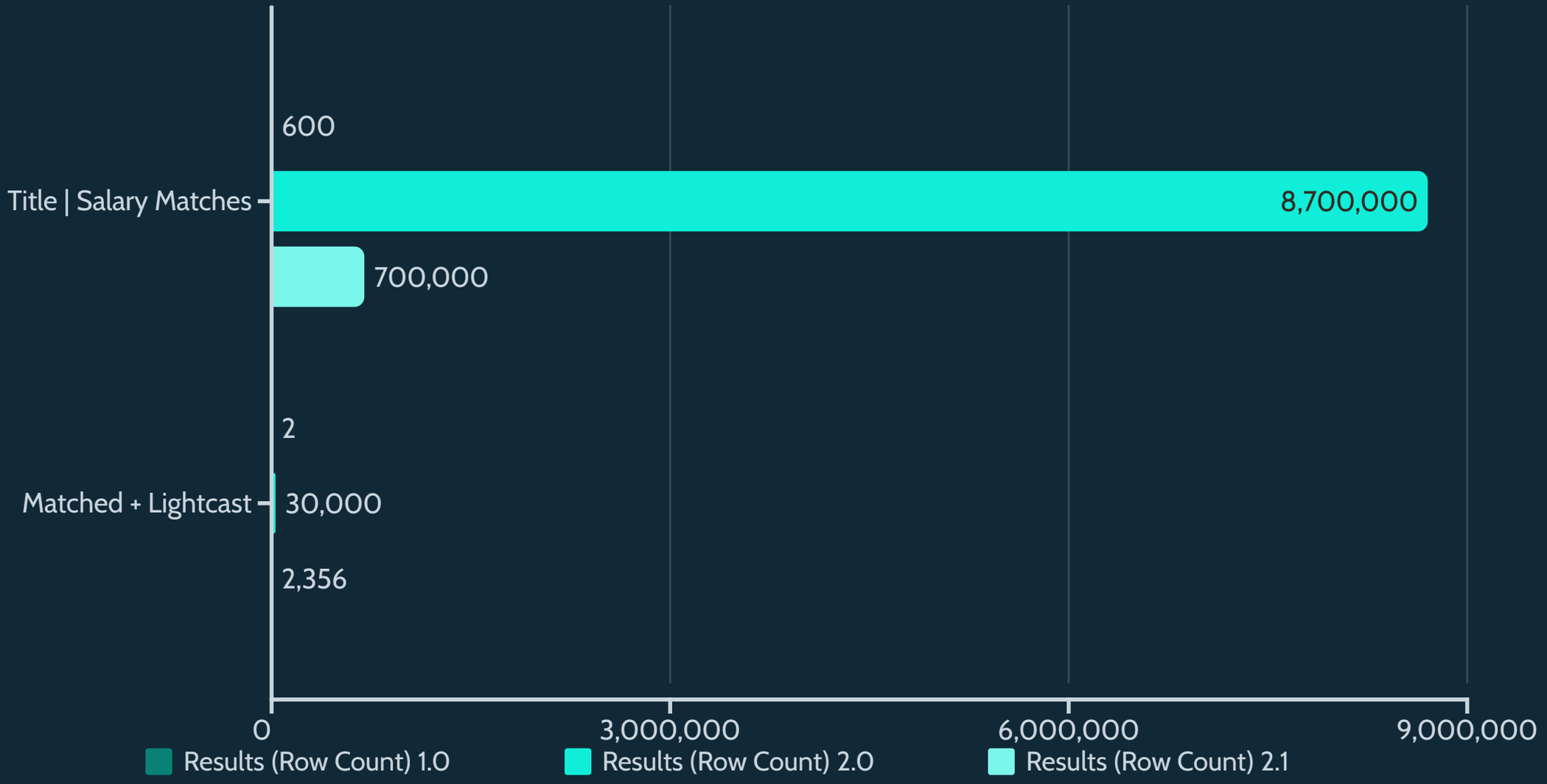
I threw all my tools at this. Through adoption of Polars, leveraging Parquet files, lazy frames, parallelization and applying vectorization techniques in my matching process, I dramatically cut processing time.

Fuzzy Matching 1.0 → 2.0 → 2.1

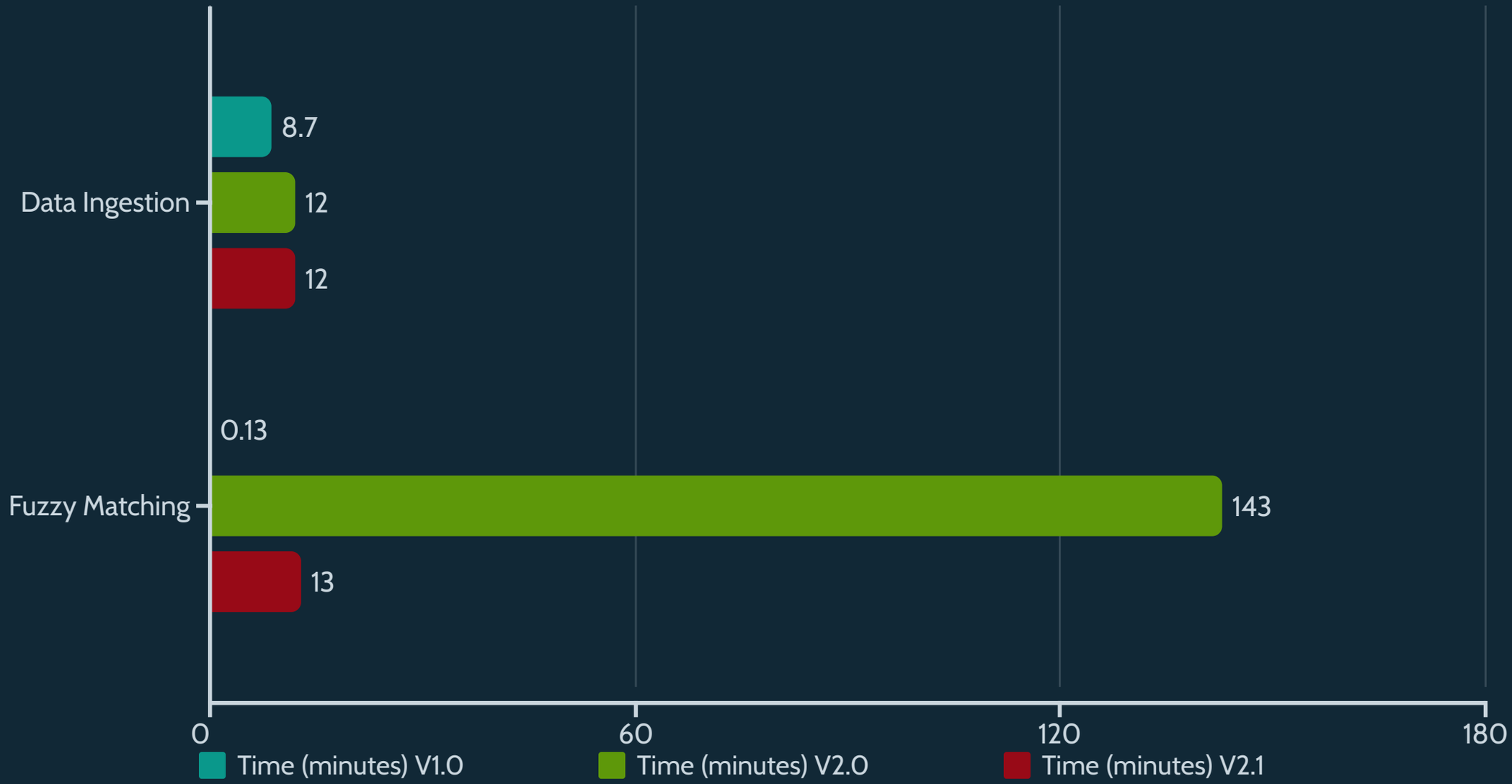
With a deep dive in fuzzy matching, I wasn't satisfied with the data I was returning. After a MVP was hit, I decided to better drill down on fuzzy matching.

Version 1.0 to 2.1: Match Results

The main difference between 1.0 and 2.0 was using vector-based fuzzy matching with no limits.



Version 1.0 to 2.1: Time



Demonstrating the Live Pipeline

This section is dedicated to showcasing our working solution. I'll walk through the deployed API and the interactive data visualization, highlighting key features.

FastAPI & Swagger

A live tour of the data access layer. I'll demonstrate how to query the processed data via the API and explore the auto-generated Swagger documentation.

Streamlit Visualization

Experience the data firsthand through the interactive Streamlit dashboard. We'll explore job posting trends, payroll discrepancies, and integrated Lightcast analytics.

Stretch Goals

What would I do if I had more time?

Deployment

I'd love to see this project deployed in a way that could utilize an S3 bucket.

Improved Data Vis

With so much time in Fuzzy Matching, I would have loved to learn more about Sreamlit.

Testing

I ran out of time to add a comprehensive unit test suite, but it would be a good idea to implement.

What I Learned

Trust your gut. If you're returning data that doesn't make sense, keep drilling down if you have time. Sometimes you can just smell that it's wrong.

Create test datasets. Testing on a dataset that has 6.3M rows is crazy. I did that. I am crazy. But I'm a little less crazy for next time now.

No lone wolves. Even on an individual project its important to surround yourself with people, if only to have a laugh or vent.