TrackMan API to PostgreSQL ETL Pipeline for Pomona-Pitzer Baseball

Capstone in Data Science

Z Skigen

November 4, 2025

1 Executive Summary

This project creates an automated Extract–Transform–Load (ETL) pipeline that connects the Pomona–Pitzer Baseball TrackMan system directly to a PostgreSQL database.
The goal is to move away from our current workflow, manual CSV downloads obtained from FileZilla, stored in Google Drive, toward a reproducible, cloud-based database that can be accessed and queried by multiple analysts simultaneously.

The pipeline authenticates with the TrackMan API, extracts raw JSON data, parses it into structured tables with Python and pandas, and loads it into PostgreSQL using SQLAlchemy. Once the data is centralized, analysts can query live data directly from VS Code or through a cloud-based SQL playground Streamlit app.

The deliverable result is a maintainable, queryable database that reflects Major League Baseball's standard of data infrastucture, providing a foundation for automated report generation and advanced modeling.

2 Motivation: Current Workflow and Limitations

Pomona–Pitzer Baseball currently relies on manual processes to create TrackMan-based scouting reports. After each game, analysts download separate CSV files containing pitch-level data, many of which contain thousands of rows. These files must be cleaned, merged, and analyzed individually, with every analyst maintaining their own local copy of the data.
This approach is slow, error-prone, and difficult to reproduce.

TrackMan's proprietary software provides visual summaries and leaderboards, but it does not grant analysts direct, flexible access to underlying pitch-level data.

As a result, analysts rely on manual R/Python scripts to generate metrics like pitch usage, release velocity trends, or command consistency.
These analyses are valuable but isolated, different students may compute similar metrics differently, leading to inconsistent results across reports.

The ETL pipeline addresses these issues by:

1. Centralizing all TrackMan data into a relational PostgreSQL database.
2. Automating data extraction and cleaning directly from the TrackMan API.
3. Standardizing field names, data types, and relationships between games, players, and pitches.
4. Enabling collaborative, cloud-based access through SQL and Streamlit interfaces.

Once the pipeline is deployed, every analyst will query the same database. This reduces redundant data wrangling and ensures reproducible, transparent analytics across seasons and analysts.

3 Data: Structure, Scope, and Organization

The pipeline ingests TrackMan data through authenticated API calls.
Each endpoint provides nested JSON data with slightly different hierarchies, requiring extensive parsing before analysis.

3.1 Key Endpoints

- `/games`: metadata such as teams, date, and venue.
- `/plays`: plate appearances and pitch sequences.
- `/balls`: detailed pitch-tracking metrics (velocity, spin, movement, release position, location).
- `/players`: player roster and handedness information.

3.2 Data Model

To make these data usable, the project implements a normalized relational schema with four main tables:

- `game`: one row per game, with keys for date, venue, and teams.

- `player`: one row per player, linked by TrackMan ID.
- `plate_appearance`: contextual details such as inning, batter, pitcher, and count.
- `pitch`: pitch-level tracking data, linked to both the game and plate appearance.

This design mirrors how baseball events are structured in professional data systems, enabling queries such as:

```
SELECT batter_id, AVG(release_speed)
FROM pitch
WHERE pitch_type = 'FF'
GROUP BY batter_id;
```

Analysts no longer need to manually merge tables or maintain separate copies for each report, queries are now reproducible.

3.3 API Acquisition and Data Cleaning

The TrackMan Range API uses an OAuth2 authentication flow.
The pipeline first retrieves an access token using client credentials, then requests all available game sessions in 30-day windows to stay within API rate limits (TrackMan 2025).

Each session returns JSON objects from multiple endpoints, which are parsed in Python using *requests* and pandas.json_normalize() (Goel 2020).

Cleaning and transformation steps include:

1. Flattening nested fields into tabular format.
2. Standardizing timestamps to UTC.
3. Renaming variables to snake_case.
4. Removing duplicate or null rows.
5. Aligning foreign keys across tables (e.g., linking pitches to players and games).

After transformation, the cleaned DataFrames are loaded into PostgreSQL using SQLAlchemy (GeeksforGeeks 2021).
This step converts the raw JSON stream into a structured, queryable format suitable for cloud storage and analysis.

3.4 PostgreSQL Database Creation

I will follow the official PostgreSQL documentation (PostgreSQL Global Development Group 2025), which explains how to properly create and configure new databases for persistent storage.

4 Next Steps and Automation

At present, the database prototype has been created locally but not yet deployed to a cloud server.
The next phase will involve:

1. Setting up a cloud-hosted PostgreSQL instance using Render or AWS RDS (TBD)
2. Scheduling automated daily updates using GitHub Action or cron (Sahu 2024).
3. Building a playground SQL Streamlit app (Streamlit, Inc. 2025).

Automation is crucial step: once achieved, the system will refresh nightly and remove the need for manual downloads altogether.
(Joe Nelson, Steve Chavez 2025; Streamlit, Inc. 2025; Plotly Technologies Inc. 2025)

5 Limitations

1. TrackMan API documentation changes periodically, requiring manual field validation. 2.Large JSON files can exceed local memory, necessitating batch ingestion.
2. Historical data from earlier seasons may use inconsistent field names or units.
3. Operational data (e.g., bullpen sessions, scrimmages) may need manual tagging to distinguish from official games.

Despite these limitations, the project's modular structure will allow straightforward debugging and schema expansion as the database scales.

6 Conclusion

The TrackMan ETL pipeline will modernize Pomona–Pitzer Baseball's data operations.
Instead of dozens of fragmented CSVs, analysts will have a unified database accessible through SQL, Python, or a web app.
Once fully implemented, the system will:

1. Eliminate redundant manual cleaning.
2. Provide live, reproducible access to pitch-level data.
3. Serve as a foundation for advanced models and automated scouting reports.

This infrastructure will not only support the current analytics team but also create a sustainable framework for future seasons and future analysts.

References

GeeksforGeeks. 2021. "Python SQLAlchemy Introduction." https://www.geeksforgeeks.org/python/sqlalchemy-introduction/.

Goel, Ankit. 2020. "How to Parse JSON Data with Python Pandas?" *Medium: TDS Archive*. https://medium.com/data-science/how-to-parse-json-data-with-python-pandas-f84fbd0b1025.

Joe Nelson, Steve Chavez. 2025. *PostgREST Documentation*. https://postgrest.org/.

Plotly Technologies Inc. 2025. *Dash User Guide and Documentation*. https://dash.plotly.com/tutorial.

PostgreSQL Global Development Group. 2025. *PostgreSQL: `CREATE DATABASE` — SQL Command*. https://www.postgresql.org/docs/current/sql-createdatabase.html.

Sahu, Satyam. 2024. "Automate Data Ingestion: Connecting REST APIs to Your Data Warehouse with Python." Medium. https://medium.com/towards-data-engineering/automate-data-ingestion-connecting-rest-apis-to-your-data-warehouse-with-python-eb889fb668f7.

Streamlit, Inc. 2025. *Streamlit Documentation*. https://docs.streamlit.io/.

TrackMan. 2025. *TrackMan Range API Documentation*. https://docs.trackmanrange.com/.