TrackMan API to PostgreSQL ETL Pipeline for Pomona-Pitzer Baseball

Capstone in Data Science

Z Skigen

November 4, 2025

1 Introduction

1.1 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 the current workflow—manual CSV downloads obtained through FileZilla and 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. The centralized database is hosted on **Railway**, which provides a managed, persistent PostgreSQL instance accessible from anywhere. Analysts can run queries either locally (e.g., through VS Code) or through a Streamlit-based SQL playground that will be deployed on **Streamlit Cloud**.

To ensure the database remains continuously up to date, automated ingestion will be scheduled using **PythonAnywhere or GitHub Actions**, allowing the ETL script to run nightly without manual intervention.

Long-term maintainability is supported through a dedicated analyst hub at **sagehenanalytics.sites.pomona.edu**, which will provide: - onboarding materials for new analysts,
- a SQL primer tailored to baseball data,
- a data dictionary describing every field and table,
- example queries and reproducible workflows, and
- documentation for sustaining and extending the ETL pipeline.

Together, the cloud infrastructure, centralized storage, and documentation ecosystem create a maintainable, MLB-style data environment that supports automated reporting, advanced modeling, and long-term continuity for the program.

1.2 Motivation: Current Workflow and Limitations

TrackMan is an optical radar- and camera-based tracking system that measures pitch trajectories, velocities, spin rates, and batted-ball outcomes. Pomona–Pitzer Baseball currently relies on manual processes to create TrackMan-based scouting reports, visualizations of key baseball statistics. 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 often 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 hosted on Railway.

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. Providing shared access through SQL, Python, and a Streamlit-based web interface.

5. Supporting long-term onboarding and sustainability through documentation at **sagehenanalytics.sites.pomona.edu**, including SQL examples and database-maintenance guides.

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

2 Data: Structure, Scope, and Organization

The ETL pipeline currently loads TrackMan data into three core PostgreSQL tables: `sessions`, `plays`, and `balls`. These tables mirror the structure of the TrackMan Data API and contain the essential components needed to reconstruct full pitch sequences and game contexts.

2.1 TrackMan API Structures

TrackMan exposes three main JSON collections:

- **`/sessions`** — metadata for each event (games, scrimmages, bullpens).
- **`/plays`** — one row per plate appearance, including batter, pitcher, inning, count progression, and pitch order.
- **`/balls`** — one row per pitch, including release metrics, spin characteristics, movement, extension, and location measurements.

These objects contain nested fields and irregular hierarchies, so the ETL process must standardize names, flatten structures, and enforce consistent typing.

2.2 Current Database Tables

After cleaning and transformation, the pipeline writes each collection into one of three relational tables:

- **`sessions`**
  Contains event-level context such as date, venue, session ID, and the teams involved.
- **`plays`**
  Contains plate-appearance–level information (batter, pitcher, inning, count, pitch numbers).
  Each row includes a unique **`playID`**, TrackMan's identifier for a plate appearance.
- **`balls`**
  Contains pitch-level tracking data, including release speed, spin rate, spin axis, extension, vertical/horizontal break, and movement characteristics.
  Each row includes **`playId`** (matching `plays.playID`) and a pitch number within the plate appearance.

This schema is intentionally minimal and mirrors TrackMan's native structure. It can be expanded later (e.g., splitting `players`, normalizing pitch types, modeling sessions → plays → pitches), but the three-table design already supports complete reconstruction of pitch-level datasets.

2.3 How Tables Connect (Joins)

Analysts can combine tables using TrackMan's built-in keys:

- **`plays.playID`** = **`balls.playId`** connects plate appearances to their corresponding pitches.
- **`plays.sessionId`** = **`sessions.sessionId`** links plate appearances to their session metadata.
- **`balls.sessionId`** also links directly to sessions when needed.

The current pipeline supports queries like:

```
SELECT
    p."taggerBehavior_pitchNo"           AS pitch_no,
    to_timestamp(p."localDateTime", 'MM/DD/YYYY HH24:MI:SS') AS date,
    p."pitcher_name",
    b."pitch_release_relSpeed",
    b."pitch_movement_inducedVertBreak"
FROM plays p
JOIN balls b
    ON p."playID" = b."playId"
WHERE to_timestamp(p."localDateTime", 'MM/DD/YYYY HH24:MI:SS') >= '2024-01-01'
ORDER BY date, pitch_no;
```

This join structure is the backbone of all future extensions. As the database grows, additional normalized tables—such as player, pitch_type, or plate_appearance_metadata—can be layered on top without changing these fundamental relationships.

2.4 API Acquisition and Data Cleaning

The TrackMan Range API uses an OAuth2 client-credentials flow to authenticate access. The ETL process automates the full sequence of acquiring, cleaning, and loading data:

1. **Retrieve an access token** from the authentication endpoint.

2. **Fetch available sessions** in 30-day windows to remain within API rate limits (TrackMan 2025).

3. **Request `/plays` and `/balls` JSON objects** for each session.

4. **Flatten nested JSON** using the `requests` library and `pandas.json_normalize()` (Goel 2020).

5. **Standardize all fields**, including:

   - converting names to `snake_case`,
   - parsing timestamps into UTC,
   - enforcing numeric data types, and
   - removing duplicate or null rows.

6. **Load cleaned DataFrames into PostgreSQL** using SQLAlchemy (GeeksforGeeks 2021).

This pipeline converts TrackMan's deeply nested JSON responses into a structured, queryable relational model suitable for cloud storage and analytics workflows.

A temporary note: I am currently investigating a limitation in which only a subset of 2024 sessions return complete play- and ball-level data through the API. This appears related to access permissions rather than the ETL logic, and I am in the process of confirming the correct API scope with TrackMan support.

2.5 Note on Limited 2024 Data

At present, the TrackMan API only returns a small subset of 2024 sessions with complete pitch-level (`balls`) and plate-appearance (`plays`) data. This limitation appears to stem from **account permissions or session-type filtering**, rather than from any issue in the ETL code itself. I am actively working with TrackMan support to confirm the correct API access scope and to ensure that all eligible 2024 sessions are retrievable.

Once the access configuration is resolved, the current ETL pipeline will be able to ingest the full historical dataset without requiring architectural changes.

2.6 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.

2.7 Cloud Architecture and Deployment Workflow

The ETL pipeline is designed to run on lightweight, modular cloud services that simplify deployment and make the system accessible to multiple analysts without requiring local installations. Three platforms play distinct roles in the proposed architecture:

2.7.1 Railway (Managed PostgreSQL Hosting)

Railway is used to host the centralized PostgreSQL database. Its managed infrastructure removes the need for local PostgreSQL installation, manual server provisioning, or SSL configuration. Railway provides:

- a persistent PostgreSQL instance,
- simple credential and environment variable management,
- automatic backups, and
- the ability to connect from local machines, VS Code, or Python scripts.

This allows analysts to query the same live database regardless of device or location.

2.7.2 Streamlit Cloud (Interactive SQL Playground)

Streamlit Cloud will host a lightweight web application that serves as an interactive SQL playground. The app connects securely to the Railway database and allows analysts to:

- run parameterized queries,
- visualize pitch-level or player-level data,
- export results for reporting, and
- interact with trackman data without writing boilerplate Python.

This interface mirrors MLB-style internal dashboards and lowers the barrier to entry for new analysts.

2.7.3 PythonAnywhere (Optional Scheduled ETL Worker)

While Railway can run serverless functions, PythonAnywhere provides a simple environment for automated Python execution. It can run the ETL script on a fixed schedule (e.g., nightly), pulling new sessions from the TrackMan API and appending them to the Railway database. PythonAnywhere is well-suited for:

- scheduled CRON-like jobs,
- lightweight API ingestion,
- log monitoring,
- isolated execution of ETL tasks without needing a local machine.

If PythonAnywhere is used, the workflow would be:

1. A scheduled Python script authenticates with TrackMan.

2. The script fetches new sessions, plays, and balls.

3. Cleaned data are appended to the Railway PostgreSQL database.

4. Streamlit dashboards automatically reflect new data without redeployment.

Together, Railway, Streamlit Cloud, and PythonAnywhere form a small but effective cloud ecosystem that supports automated ingestion, centralized storage, and analyst-facing tools while maintaining reproducibility and minimizing the operational burden on students.

3 Analysis

The goal of this project is not to build statistical models or conduct performance analytics, but to create the infrastructure that enables those tasks. To demonstrate the practical value of the centralized PostgreSQL database, this section showcases examples of queries and workflows that were previously time-consuming or impossible under the manual CSV-based system.

Rather than presenting full analytic results, these examples illustrate how the new schema supports fast, reproducible, and consistent access to pitch-level data.

3.1 Example 1: Accessing All Pitches for a Game

A query that once required manually merging multiple CSVs can now be executed in a single step:

```sql
SELECT *
FROM balls
WHERE session_id = 'some_game_id';
```

##Example 2: Linking Pitches to Plate Appearances

The relational structure allows analysts to connect events without manual joins:

```sql
SELECT p.pitch_number, p.pitch_type, pa.batter_id, pa.inning
FROM balls p
JOIN plays pa USING (session_id, pitch_number)
LIMIT 50;
```

##Example 3: Checking Data Completeness

Because ingestion is automated, analysts can quickly verify which sessions have full play and pitch data:

```sql
SELECT session_id,
       COUNT(*) AS n_pitches
FROM balls
GROUP BY session_id;
```

##Example 4: Preparing Data for Scouting Reports

Analysts can build consistent scouting templates directly from the database:

```sql
SELECT pitcher_id,
       AVG(release_speed) AS avg_velo,
       STDDEV(release_speed) AS velo_sd
FROM balls
GROUP BY pitcher_id;
```

These examples are not intended as full statistical analyses. Instead, they demonstrate that the new database functions as a reliable foundation for future analytics work—reducing time spent on data wrangling and ensuring that all analysts work from the same, reproducible source of truth.

4 Ethical Considerations

Building a live data pipeline from the TrackMan API introduces important ethical considerations related to data privacy, ownership, contractual compliance, access control, and responsible use. Because the pipeline automates acquisition and centralizes all pitch-level data, it is essential to ensure that athlete information is handled securely and used only for appropriate internal purposes.

4.1 Data Ownership and Privacy

TrackMan data belong to Pomona–Pitzer Baseball and its athletes. Although the ETL system streamlines ingestion, it does not change who controls the data or how the data may be shared. The database is designed only for internal use within the Pomona–Pitzer Baseball program.

To maintain privacy and comply with institutional expectations:

- API credentials are stored securely through environment variables, never in source code.
- Database access is restricted to verified analysts through password-protected accounts.
- Personally identifiable information (PII), such as student names or ID numbers, is never included in exports or external reports; all analysis is conducted using anonymized player identifiers.
- No raw or processed TrackMan data are publicly distributed or indexed on the open internet.

These safeguards ensure that the pipeline enhances accessibility without compromising privacy or exposing sensitive athlete performance information.

4.2 Contractual and Compliance Considerations

TrackMan requires users of its Data API to agree to a formal Terms of Service (ToS). As of now, the team is awaiting a copy of the ToS from our TrackMan representative. Once received, I will review the document to determine:

- whether data may be shared internally across analysts,
- what forms of storage or distribution are prohibited,
- whether the database must remain private,
- any restrictions on derived metrics or visualization,
- any expectations regarding secure data handling or retention.

Until the ToS is reviewed, the system is intentionally designed to restrict access to authorized internal users only. This conservative default minimizes the risk of accidental violation of TrackMan's contractual requirements.

4.3 Fairness and Responsible Use

Performance data can influence evaluation and playing-time decisions, which makes responsible interpretation essential. Raw pitch metrics do not capture context such as injury history, mechanical adjustments, or coaching strategy, and could be misused if treated as objective rankings.

To promote fair use:

- The database supports development-focused analysis, not public comparison, ranking, or punitive evaluation.
- Summaries and dashboards are contextualized with coaching input.
- Metrics are interpreted as tools for improvement rather than indicators of athlete value.
- Analysts are encouraged to consider uncertainty, variability, and sampling issues in their conclusions.

This approach reduces the risk of overinterpreting noisy performance data or unintentionally creating inequitable evaluation environments.

4.4 Security and Access Control

Centralizing data introduces technical risks that must be mitigated. To ensure secure operation:

- Access follows least-privilege principles: each analyst receives only the permissions necessary for their role.
- Backups are encrypted, and the database is never left open to public traffic.
- API usage logs and ingestion events are monitored for anomalies.
- Credentials are rotated when personnel change.

These safeguards protect both athletes and the institution from data breaches, unauthorized access, or unintentional exposure.

4.5 Cloud Platform Considerations

The infrastructure supporting this project relies on modern cloud services, including Railway for database hosting, Streamlit Cloud for interactive SQL exploration, and potentially PythonAnywhere for scheduled ingestion or lightweight API tasks. Each of these services offers different strengths—Railway provides a simple and secure managed PostgreSQL instance, Streamlit Cloud enables accessible web-based dashboards for analysts, and PythonAnywhere supports automated Python execution without requiring local resources.

However, using third-party cloud platforms introduces additional ethical responsibilities. To prevent accidental public exposure of athlete data:

- all Railway database instances are configured to require password-authenticated connections,

- the Streamlit Cloud application will authenticate users before granting access to any query functionality, and

- no TrackMan-derived data will be written to publicly accessible URLs, buckets, or web endpoints.

If PythonAnywhere or similar services are used for scheduled ingestion, credentials will remain stored in environment variables, not in source code, and logs will not contain sensitive player information. These platform decisions are made for operational simplicity, but each is configured to preserve the privacy and confidentiality of athlete performance data and to prevent unauthorized external access.

4.6 Potential Risks With Automation

The pipeline is designed for automated updates, which introduces new ethical and operational risks:

- If TrackMan mislabels sessions (e.g., bullpens vs. games), automated ingestion may import incorrect or unintended data.
- Errors in the API or incomplete sessions could propagate into the database without being manually reviewed.
- Analysts could rely on real-time metrics without understanding their limitations or uncertainty.

To address these risks:

- Automated updates will include logging, flags for new sessions, and procedures for rollback.
- Analysts will receive documentation explaining how ingestion works, what assumptions are made, and how to detect anomalies.
- Any automated reporting tools will emphasize transparency and uncertainty to avoid overconfidence in raw metrics.

5 Next Steps and Automation

The core ETL pipeline has been implemented and tested locally, but several major components remain before the system can operate as a complete, production-ready analytics platform. The following next steps focus on resolving current data-access limitations, building analyst-facing tools, and establishing long-term maintainability for the program.

5.1 1. Resolve the Limited 2024 Data Returned by the TrackMan API

The most immediate priority is diagnosing why the TrackMan API is returning only a small subset of 2024 sessions with complete pitch-level data. I am currently:

- reviewing the TrackMan Data API Terms of Service once it is provided,
- confirming the access scope associated with the Pomona–Pitzer TrackMan account,
- contacting our TrackMan representative to clarify available endpoints and permissions, and
- testing alternative approaches to querying sessions or session types.

Once access details are confirmed and expanded, the existing ETL code will be able to ingest the full historical dataset without requiring structural changes.

5.2 2. Build a Front-End Interface for Analysts

A lightweight Streamlit Cloud application will serve as the primary interface for team analysts who may not want to write SQL or Python. Planned features include:

- an authenticated login page,
- a SQL query sandbox connected to the central Railway database,
- built-in visualizations of pitch-level metrics,
- the ability to export data for scouting reports, and
- tools for monitoring data completeness and ingestion logs.

This web app will mirror internal MLB-style dashboards and significantly lower the barrier to using TrackMan data effectively.

5.3 3. Create a Persistent Analyst Hub (sagehenanalytics.sites.pomona.edu)

To ensure long-term sustainability and institutional memory, I will build a documentation hub hosted at:

sagehenanalytics.sites.pomona.edu

This site will contain:

- detailed documentation on how to access, query, and maintain the PostgreSQL database,

- a simple introduction to SQL for baseball analytics,

- a data dictionary for all tables produced by the ETL pipeline,

- setup instructions for new analysts,
- version-controlled documentation for future modifications, and
- guidance on data privacy, permissions, and operational best practices.

This hub will serve as the permanent home for Sagehen Baseball's analytics workflows.

5.4 4. Implement Automated Data Refreshes

A major goal of the system is to remove the need for manual data downloads entirely. Scheduled ingestion will be implemented using GitHub Actions or PythonAnywhere to automatically run the ETL script on a nightly basis (Sahu 2024). This automated process will:

- authenticate with the TrackMan API,
- detect and ingest new sessions,
- append updated play and pitch data,
- log ingestion events for debugging, and
- notify analysts if anomalies arise.

Automation is a crucial step: once implemented, the database will remain up to date without requiring analyst intervention.

5.5 5. Finalize Cloud Deployment Workflow

The last stage of development involves fully transitioning from local testing to secure cloud deployment:

- deploying the ingestion workflow to GitHub Actions or PythonAnywhere for scheduled execution (Joe Nelson, Steve Chavez 2025),
- hosting the SQL playground and visualization tools on Streamlit Cloud (Streamlit, Inc. 2025),
- ensuring secure connectivity between Streamlit, PythonAnywhere, and Railway,
- rotating credentials and storing them via environment variables, and

- performing multi-user testing to confirm reliability, latency, and usability.

Together, these steps will produce a robust, maintainable, and accessible cloud-based analytics ecosystem for the Pomona–Pitzer Baseball program (Plotly Technologies Inc. 2025).

6 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.

6.1 Incomplete 2024 TrackMan Data (Under Investigation)

During development, I discovered that the TrackMan API was only returning a small subset of 2024 sessions with complete play- and ball-level data. Most of the pitch-level data currently available through the `/plays` and `/balls` endpoints falls between January 25–28, 2024, even though the API reports thousands of session identifiers for the year.

This does not appear to be an error in the ETL pipeline itself. Instead, it is likely related to the scope of the Pomona–Pitzer TrackMan account, the API's session-type filtering behavior, or access permissions defined in the TrackMan Data API Terms of Service. I am actively working to diagnose the cause by reviewing the ToS, communicating with TrackMan support, and testing alternative session-querying strategies.

As this investigation continues, the current database reflects the subset of 2024 data that the API makes available with the present access configuration. Once access details are confirmed and expanded, the pipeline will be able to ingest the full historical dataset without requiring structural changes.

7 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/.