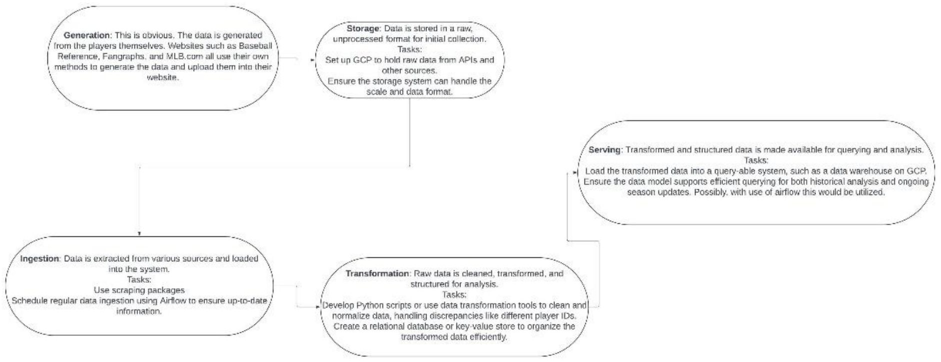
Scott Silverstein

Project Writeup

Why a baseball data pipeline?

I am what every person would consider a baseball fanatic. Every aspect of the game from playing it through college to being a observer post baseball career has created a meteor sized shape in who I am as a person. In fact, it was baseball that drove me to analytics. Baseball is by far the best sport for any analyst. It has thousands if not millions of datasets, a ton of games, and a very robust system of data collection. This proposes a challenge for me however. I love baseball so much that I want to start a baseball blog that does analytical research to quantify my baseball opinions. I tried to start this earlier but struggled. I did not have the knowledge necessary to give myself the stats I needed in a database in order for me to be successful. My goal for this project is for me to have a dataset that I can constantly query anytime I want to write an article.

<u>Steps to accomplish this:</u>



Generation: This is obvious. The data is generated from the players themselves. Websites such as Baseball Reference, Fangraphs, and MLB.com all use their own methods to generate the data and upload them into their website.

Storage: Data is stored in a raw, unprocessed format for initial collection.
Tasks:
Set up GCP to hold raw data from APIs and other sources.
Ensure the storage system can handle the scale and data format.

Ingestion: Data is extracted from various sources and loaded into the system.
Tasks:
Use scraping packages
Schedule regular data ingestion using Airflow to ensure up-to-date information.

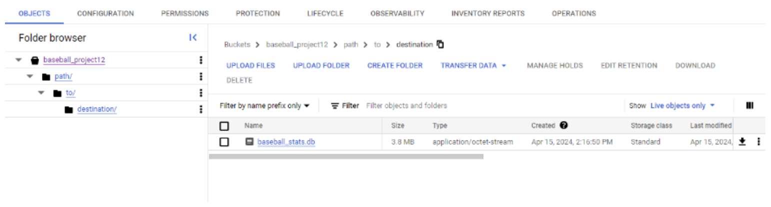
Transformation: Raw data is cleaned, transformed, and structured for analysis.
Tasks:
Develop Python scripts or use data transformation tools to clean and normalize data, handling discrepancies like different player IDs.
Create a relational database or key-value store to organize the transformed data efficiently.

Serving: Transformed and structured data is made available for querying and analysis.
Tasks:
Load the transformed data into a query-able system, such as a data warehouse on GCP.
Ensure the data model supports efficient querying for both historical analysis and ongoing season updates. Possibly, with use of airflow this would be utilized.

Data Generation: The actual generation of this data is the easy part. This is handled by websites that are able to gather game data in real time and have them in location that is able to be scooped up by an API or scraping package.

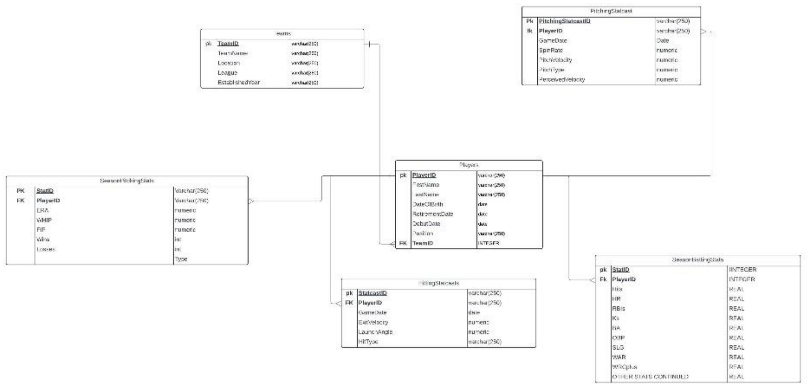
I ended up using pybaseball package, which is a package incorporating BeautifulSoup in order to scrape the different websites. The main struggle, however, is that every website has their own ID for the Players and different information. Something that I wasn't able to figure out how to rectify. For instance, I was looking forward to using the MLB stats api, however, the mlb website is terrible for anything but the old outdated stats our fathers would have used for analysis. My solution was to use just the Fangraphs data as they have every advanced stat that I could have imagined and a robust ID format.

Storage: I used GCP in order to store my data. This was relatively easy as Google Collab has a pretty well integrated method for this.

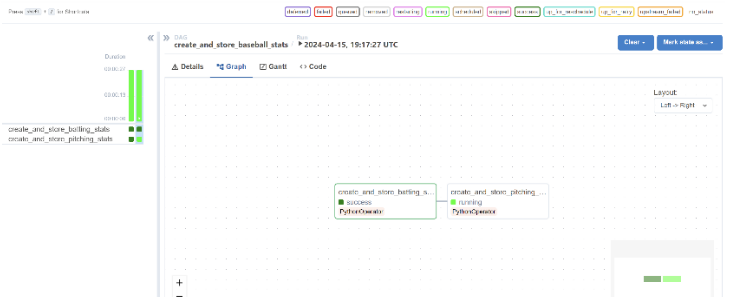


Name	Size	Type	Created	Storage class	Last modified
baseball_stats.db	3.8 MB	application/octet-stream	Apr 15, 2024, 2:16:50 PM	Standard	Apr 15, 2024,

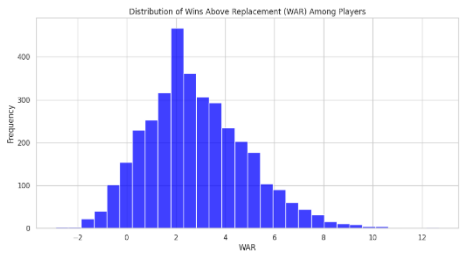
Ingestion: This was complicated as I wanted them to be able to have a specific layout to fit within the ERD. I had to make sure that as the data was being inserted it fit the overall guidelines of the database structure. This entailed changing a lot of the column names and making sure that every column matched the correct variable type. Due to the challenges in the ID's and different information on different sites among others such as not having a dataset to call on for specific statcast stats, I really had to trim down my ERD. This is the final idea shown below:

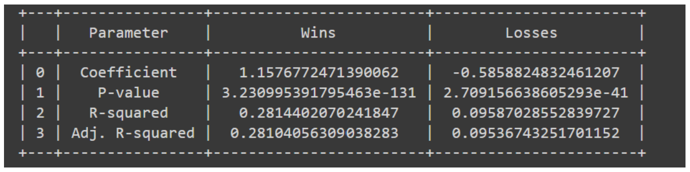


Transformation: This part was ingrained into the Ingestion stage. I would have a function that would create a table and then one that would then transform the ingested table into the correct format. This is now made easier for future use with AIRFLOW.



Serving: Serving was really nice and easy. Now that it is in a 3NF relational database it is easily queried and ready for analysis!





```
+---+----------------+-----------------------+-----------------------+
|   |   Parameter    |         Wins          |        Losses         |
+---+----------------+-----------------------+-----------------------+
| 0 |  Coefficient   |   1.1576772471390062  |  -0.5858824832461207  |
| 1 |    P-value     | 3.2309953917954463e-131 | 2.709156638605293e-41 |
| 2 |   R-squared    |   0.2814402070241847  |   0.09587028552839727  |
| 3 | Adj. R-squared |  0.28104056309038283  |   0.09536743251701152  |
+---+----------------+-----------------------+-----------------------+
```

Retrospective: I loved this project. It allowed me to take what I have learned in the classroom and apply it to something that will help me in the future. I plan to use this database religiously. I also plan to see what I can do to expand it and overcome the struggles I was having with some of the table building that caused me to really shrink my ERD. I tried to make a database like this before this class and failed drastically to the point where I was manually copy and pasting things into an excel spreadsheet. This is exactly what I needed and hope that in the future you keep the flexibility to craft ones own project in the same manner.