CSE 560: Data Models and Query Language
Spring 2025
Project Milestone 2
MLB Data (2015–2024)

Luke Baylon
Jiun Kim
State University of New York at Buffalo
{lukebayl, jiunkim}@buffalo.edu

I. Project Details

A. Project Overview

Project Name: MLB Data (2015–2024)
Team Number: 3
Members and UBIDs: Luke Baylon (lukebayl), Jiun Kim (jiunkim)

B. Problem Statement

Being able to query baseball data from a database is helpful to retrieve player statistics for various uses. A database such as this one, which includes ways to connect teams, players, and their respective managers throughout the years (or in this case for the sake of not having an enormous dataset, from 2015-2024) in a quick manner is a nice tool for fans and possibly teams to use. An Excel file would potentially struggle to meet the needs of someone's search, such as finding players that played from 2017-2019 who had a batting average above .280 and played in the AL, for example.

C. Target Users

MLB teams use player statistics all the time when evaluating players, deciding what parts of their team need improvement and which players outside of the organization would help them the most. This information helps teams' front offices decide what trades to make, and how much money to spend on free agent players. In addition, querying multiple players at one time can aid in contextualizing the production of a given player, as comparing players is critical for normalizing and defining what a "good" player is.

Perhaps more common is the die-hard baseball fan who finds joy in reminiscing about their favorite players or searching for various statistics of players that they vaguely remember. A database serves as a baseball history book to such people, where being able to look up data about different teams and their rosters is entertaining to them. Some fans maybe even play fantasy baseball or simply want to develop opinions about certain players, which means that a baseball database is even more useful.

If this database were to be widely accessible, the administrator should be someone willing to update the database with new statistics as they are produced during the MLB season. That way, users can get the most up-to-date information and analyze player performance mid-season.

II. Phase 1: Initial Data Collection and E/R Design

MLB data was collected from various sources, such as ESPN, Stathead, and baseball-reference.com (see readme.txt). The goal of this project is to store the data of hitter, pitcher, and managerial statistics, as well as team attendance. With these four areas covered via four data collections from the previously mentioned sources, a natural set of initial relations is obtained.

See figure 1 in the Appendix for the phase 1 E/R Diagram. This version of the E/R diagram has 4 relations stemming from the 4 data sources, before the relations were decomposed into Boyce-Codd normal form. The relations include information about MLB hitters, pitchers, managers, and team attendance. The following describes the information stored within each relation, as well as a given attribute's nullity and data type:

Hitter: Contains information about individual hitters, including their name, age, position, season, team, and various hitting statistics. Default values for all the counting hitting statistics (AB, PA, H, 2B, 3B, HR, RBI, SB, CS, BB, SO, TB, GIDP, HBP, SH, SF, IBB) is 0, whereas rate stats (BA, OBP, SLG, OPS, OPS+) can be null. Other attributes such as the player's name, team, league, season, age, and position must all hold a value (no nulls allowed).

Pitcher: Stores data on pitchers, such as their season, team, name, age, and pitching statistics (e.g., IP, ERA, W, L, SO, WHIP). Counting statistics (IP, Pit, W, L, Dec, G, GS, CG, SHO, SV, H, R, ER, HR, BB, IBB, SO, HBP, BK, WP, Str) have a default value of 0 and are never null, whereas rate statistics (W-L

Manager: Holds details about baseball managers, including their team, name, win/loss records, and information about challenges, overturns, and ejections. The entities Hitter, Pitcher,

and Attendance have a many-to-many relationship with Manager through the attributes Team and Year. As for the Manager relation itself, year, team and name should never be null with no default value; wins, losses, ties, challenges, overturns, and ejections are never null with default values of 0; games are never null with a default value of 1; Wpost and Lpost can be null; finally, W-L

Attendance: Records attendance figures for each team, including the year, team, number of home/away games, and total/average attendance. All the attributes in this relation are never null.

The relations are connected through foreign keys like "Team" and "Year" (which is synonymous with "Season") indicating how player statistics, manager information, and attendance data are connected within the database. It is worth noting that while transitioning from phase 1 to phase 2, almost all the rate statistics are dropped due to being derivable from the counting statistics.

III. Phase 2: Normalization and Database Design

Normalization

All the relations are altered to follow Boyce-Codd normal form (BCNF) as per the milestone 2 guidelines. The rest of this section describes the reasoning behind removing or decomposing different attributes. All meaning and nullability for the attributes that are still present in the final relations are the same as from phase 1 (in other words, removing or decomposing the original relations does not impact the information itself).

For the original "Hitter" relation, many attributes (particularly hitting statistics) in the table were functional dependencies of other attributes via mathematical formulas. For example, the attribute BA, which is the hitter's batting average, can be computed from the player's hits (H) and at-bats (AB). Therefore, rate stats such as BA were removed from the relation since it can be calculated from the counting stats attributes; a user of the database could calculate these values with a simple query. The only non-counting stat that is maintained is OPS+, as this is derived from information that is outside the scope of the relation and is not part of a functional dependency. Lg and Age were removed from the "hitter" relation due to decomposition into two other relations, which will be explained later. The slightly renamed and new "hitters" relation is now in BCNF, with a primary key of (player, season, team) where joins to other relations are possible through the attributes player, season and team.

Similarly, the original "pitcher" relation is changed by removing pitcher rate stats that can be obtained from the already present counting stats; again, a user of the database can calculate rate stats with a simple query on the counting stats. The only non-counting stat that is maintained is ERA+, as it is derived from information outside the scope of the data and is not part of a functional dependency. Lg and Age were removed from the "pitcher" relation due to decomposition into two other relations. Like the "hitters" relation, the primary key of the "pitcher" relation is (player, season, team) with the same capabilities for joining via player, season and team.

A new relation is introduced in phase 2, which is the "ages" relation. It contains only three columns for the player's name, the season, and the age (which is the age of the player in that season). This relation is created to preserve information about player ages and exists to break the functional dependency involving the player's name, the season, and age that existed in the "hitter" and "pitcher" relations in phase 1, so that those relations can be in BCNF. The primary key of the "ages" relation is (player, season), which are the attributes that are joinable with other relations.

The "managers" relation lost some attributes due to decomposition into other relations (which will be outlined later), which were win, lose, ties, finish, wpost, and lpost. Other attributes were then dropped due to being derivable from the lost attributes from decomposition (thus being derivable in another relation), such as $w_{p,ct}$, g, and $u_{p,ct,ost}$. The attribute p_{ct} is removed due to being a formulaic functional dependency, which was challenged and overturned. With these changes, the new "managers" relation is in BCNF with the primary key of (year, manager); team is not needed in the primary key of manager, as the same manager has never managed two teams in the same season in the history of MLB (though in theory this is possible). The "managers" relation is joinable via the attributes team and year (which is the same as season).

Information about the wins and losses of managers is transferred to a new relation called "teams", which is fitting as the record of a sports team should be associated with the name of that team rather than the manager. Again, attributes such as win/loss percentage and postseason win/loss percentage are not included in this relation either, since it can be calculated from other attributes. Manager names are included in this relation because there are instances where a team in a single season has two different managers, where each instance has a different number of wins and losses. Therefore, the primary key is (year, team, manager) with attributes describing the team's performance being included.

To decompose the functional dependency with team and league in the relations "hitters" and "pitchers", a new relation called "divisions" is introduced. This relation has only two attributes which is the team and the name of the division that the team plays in. Divisions were chosen instead of leagues because the name of the division includes the name of the league, and it is more useful in queries to have information about a team's division rather than just the league; even if a user wanted to query based on the league, this is doable with the LIKE command. The primary key of this relation is quite trivial, being (team).

With the relations in the database all being in BCNF, a new E/R diagram is made to reflect the changes. See figure 2 in the Appendix for the final (phase 2) E/R Diagram, which also shows where common attributes can be linked. As for creating the tables in SQL, figure 3 to figure 9 show the table creation queries for each relation. Loading the data into the tables directly from the associated .csv files was quick, indicating

that handling this dataset is not an issue and indexing concepts are not needed.

Figure 10 to figure 19 in the appendix shows various queries being applied to the database, including inserts, deletions, updates, and selects. The completion time is shown in each figure, as well as the result of the query. All the completion times are quite fast, with all of them being about 100 milliseconds or faster

IV. Contribution Summary

This project was completed by Luke Baylon and Jiun Kim, with equal contributions:

- **Luke Baylon:** Collected MLB data from ESPN, Stathead, and Baseball-Reference.com. Designed the Phase 1 E/R diagram and initial relations. Implemented SQL table creation and data loading scripts. Contributed to the report by writing the problem statement, target user, and Phase 1 sections.
- **Jiun Kim:** Normalized relations to BCNF, designed the Phase 2 E/R diagram, and developed query scripts for inserts, deletions, updates, and selects. Generated figures for the Appendix and wrote the Phase 2 and contribution sections of the report.

A. Bonus

https://cse560-team3-frontend.onrender.com/

V. Appendix

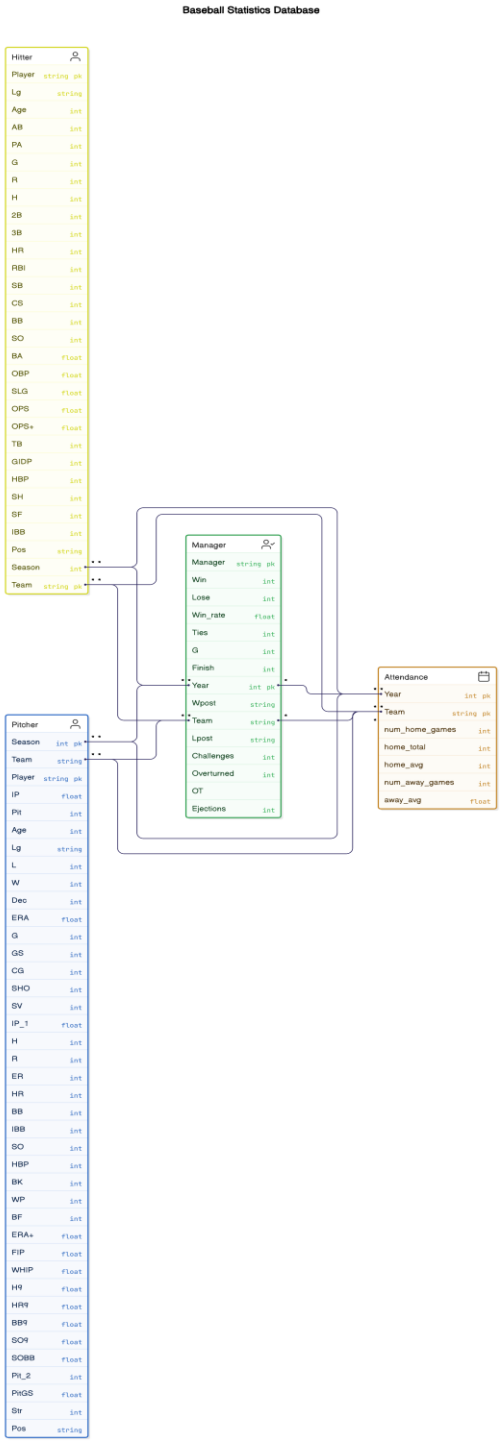

Figure 1. Phase 1 E/R Diagram

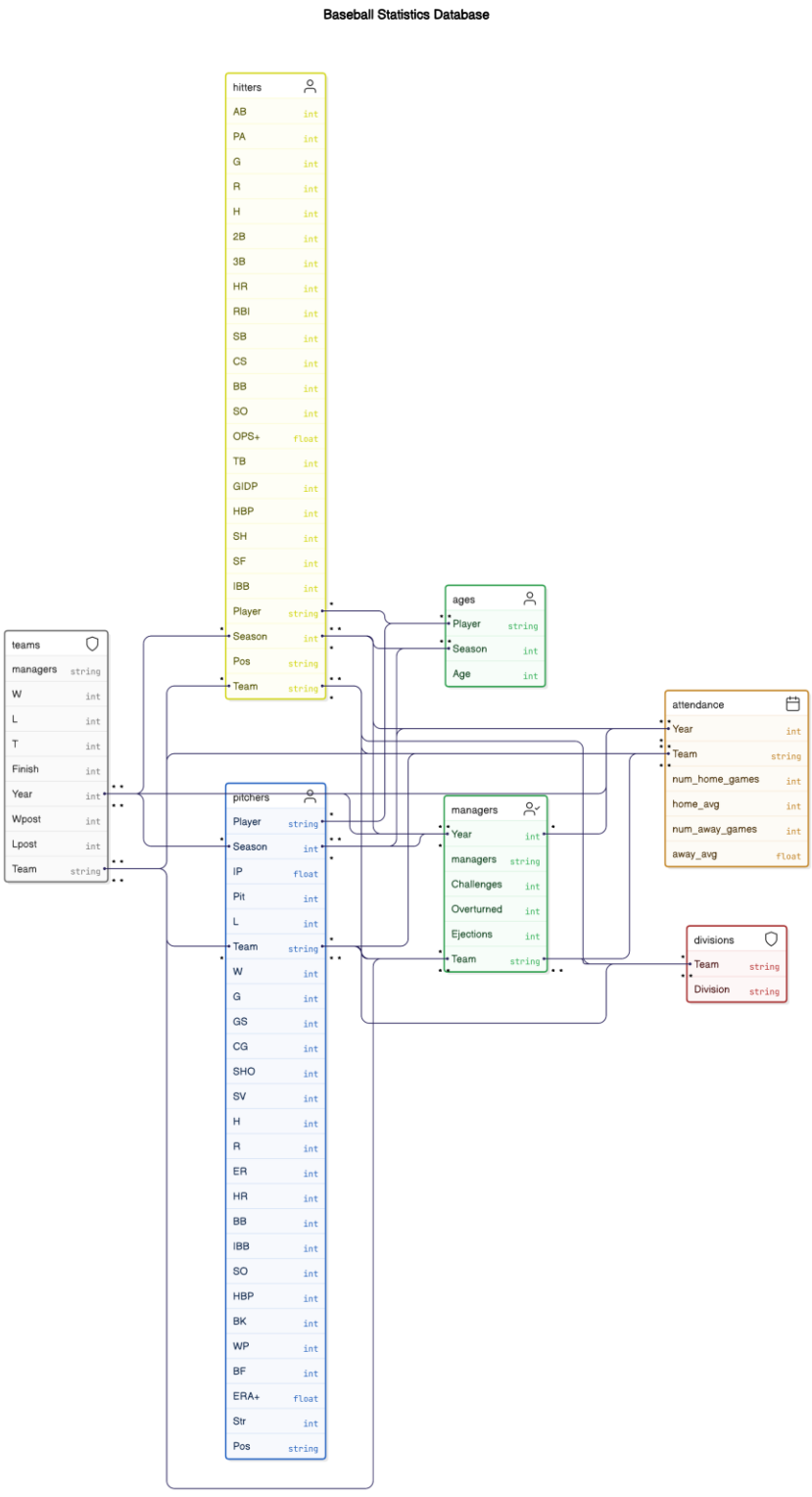

Figure 2. Final E/R Diagram

```sql
CREATE TABLE teams (
    year INT NOT NULL,
    team TEXT NOT NULL,
    manager TEXT NOT NULL,
    win INT NOT NULL DEFAULT 0,
    lose INT NOT NULL DEFAULT 0,
    ties INT NOT NULL DEFAULT 0,
    finish INT,
    wpost INT,
    lpost INT,
    PRIMARY KEY (year, team, manager)
);
```

Figure 3. SQL CREATE TABLE query to make the "teams" relation.

```sql
CREATE TABLE managers (
    year INT NOT NULL,
    manager TEXT NOT NULL,
    team TEXT NOT NULL,
    challenges INT NOT NULL DEFAULT 0,
    overturned INT NOT NULL DEFAULT 0,
    ejections INT NOT NULL DEFAULT 0,
    PRIMARY KEY (year, manager)
);
```

Figure 4. SQL CREATE TABLE query to make the "managers" relation.

```sql
CREATE TABLE pitchers (
    player TEXT NOT NULL,
    ip FLOAT NOT NULL DEFAULT 0 ,
    pit INT NOT NULL DEFAULT 0,
    season INT NOT NULL,
    team TEXT NOT NULL,
    w INT NOT NULL DEFAULT 0,
    l INT NOT NULL DEFAULT 0,
    g INT NOT NULL DEFAULT 0,
    gs INT NOT NULL DEFAULT 0,
    cg INT NOT NULL DEFAULT 0,
    sho INT NOT NULL DEFAULT 0,
    sv INT NOT NULL DEFAULT 0,
    h INT NOT NULL DEFAULT 0,
    r INT NOT NULL DEFAULT 0,
    er INT NOT NULL DEFAULT 0,
    hr INT NOT NULL DEFAULT 0,
    bb INT NOT NULL DEFAULT 0,
    ibb INT NOT NULL DEFAULT 0,
    so INT NOT NULL DEFAULT 0,
    hbp INT NOT NULL DEFAULT 0,
    bk INT NOT NULL DEFAULT 0,
    wp INT NOT NULL DEFAULT 0,
    bf INT NOT NULL DEFAULT 0,
    era_plus FLOAT,
    str INT NOT NULL DEFAULT 0,
    pos TEXT NOT NULL DEFAULT '1',
    PRIMARY KEY (player, season, team)
);
```

Figure 5. SQL CREATE TABLE query to make the “pitchers” relation.

```sql
CREATE TABLE hitters (
    player TEXT NOT NULL,
    team TEXT NOT NULL,
    season INT NOT NULL,
    ab INT NOT NULL DEFAULT 0,
    pa INT NOT NULL DEFAULT 0,
    g INT NOT NULL DEFAULT 0,
    r INT NOT NULL DEFAULT 0,
    h INT NOT NULL DEFAULT 0,
    "2B" INT NOT NULL DEFAULT 0,
    "3B" INT NOT NULL DEFAULT 0,
    hr INT NOT NULL DEFAULT 0,
    rbi INT NOT NULL DEFAULT 0,
    sb INT NOT NULL DEFAULT 0,
    cs INT NOT NULL DEFAULT 0,
    bb INT NOT NULL DEFAULT 0,
    so INT NOT NULL DEFAULT 0,
    ops_plus FLOAT,
    tb INT NOT NULL DEFAULT 0,
    gidp INT NOT NULL DEFAULT 0,
    hbp INT NOT NULL DEFAULT 0,
    sh INT NOT NULL DEFAULT 0,
    sf INT NOT NULL DEFAULT 0,
    ibb INT NOT NULL DEFAULT 0,
    pos TEXT NOT NULL,
    PRIMARY KEY (player, season, team)
);
```

Figure 6. SQL CREATE TABLE query to make the “hitters” relation.

```sql
CREATE TABLE attendance (
    year INT NOT NULL,
    team TEXT NOT NULL,
    num_home_games INT NOT NULL,
    home_avg INT NOT NULL,
    num_away_games INT NOT NULL,
    away_avg INT NOT NULL,
    PRIMARY KEY (year, team)
);
```

Figure 7. SQL CREATE TABLE query to make the “attendance” relation.

```sql
CREATE TABLE ages (
    player TEXT NOT NULL,
    season INT NOT NULL,
    age INT NOT NULL
);
```

Figure 8. SQL CREATE TABLE query to make the “ages” relation.

```sql
CREATE TABLE divisions (
    team TEXT NOT NULL,
    division TEXT NOT NULL,
    PRIMARY KEY (team)
);
```

Figure 9. SQL CREATE TABLE query to make the “divisions” relation.

```sql
INSERT INTO hitters (player, team, season, ab, pa, g, r, h, "2B", "3B", hr, rbi, sb, cs, bb, so, ops_plus, tb, gidp, hbp, sh, sf, ibb, pos)
VALUES ('Test Hitter', 'NYY', 2024, 100, 110, 30, 20, 35, 8, 1, 5, 25, 5, 2, 10, 30, 120.5, 50, 3, 2, 1, 1, 0, 'RF');
```

```
INSERT 0 1

Query returned successfully in 63 msec.
```

Figure 10. SQL INSERT query applied to the “hitters” relation.

```sql
INSERT INTO pitchers (player, ip, pit, season, team, w, l, g, gs, cg, sho, sv, h, r, er, hr, bb, ibb, so, hbp, bk, wp, bf, era_plus, str, pos)
VALUES ('Test Pitcher', 75.2, 1200, 2024, 'NYY', 5, 3, 20, 15, 0, 1, 2, 50, 25, 20, 6, 15, 2, 80, 3, 1, 2, 300, 110.3, 500, 'P');
```

```
INSERT 0 1

Query returned successfully in 59 msec.
```

Figure 11. SQL INSERT query applied to the “pitchers” relation.

```
DELETE FROM hitters
WHERE player = 'Test Hitter' AND season = 2024;
```

```
DELETE 1

Query returned successfully in 57 msec.
```

Figure 12. SQL DELETE query applied to the “hitters” relation.

```
DELETE FROM pitchers
WHERE player = 'Test Pitcher' AND season = 2024;
```

```
DELETE 1

Query returned successfully in 56 msec.
```

Figure 13. SQL DELETE query applied to the “pitchers” relation.

```
UPDATE managers
SET ejections = ejections + 1
WHERE manager = 'Aaron Boone' AND year = 2023;
```

```
UPDATE 1

Query returned successfully in 59 msec.
```

Figure 14. SQL UPDATE query applied to the “managers” relation.

```
UPDATE attendance
SET home_avg = home_avg + 500
WHERE team = 'NYY' AND year = 2023;
```

```
UPDATE 1

Query returned successfully in 62 msec.
```

Figure 15. SQL UPDATE query applied to the “attendance” relation.

```sql
SELECT h.player, h.team, h.season, d.division
FROM hitters h
JOIN divisions d ON h.team = d.team
WHERE h.season = 2023
ORDER BY h.team
LIMIT 10;
```

	player text	team text	season integer	division text
1	Christian Walker	ARI	2023	NL West
2	Ketel Marte	ARI	2023	NL West
3	Corbin Carroll	ARI	2023	NL West
4	Lourdes Gurriel Jr.	ARI	2023	NL West
5	Geraldo Perdomo	ARI	2023	NL West
6	Alek Thomas	ARI	2023	NL West
7	Gabriel Moreno	ARI	2023	NL West
8	Jake McCarthy	ARI	2023	NL West
9	Emmanuel Rivera	ARI	2023	NL West
10	Evan Longoria	ARI	2023	NL West

Total rows: 10 Query complete 00:00:00.064

Figure 16. SQL SELECT query using JOIN and ORDER BY.

```sql
SELECT team, AVG(home_avg) AS avg_home_attendance
FROM attendance
GROUP BY team
ORDER BY avg_home_attendance DESC
LIMIT 10;
```

	team text	avg_home_attendance numeric
1	LAD	45902.333333333333
2	STL	39633.777777777778
3	NYY	38929.333333333333
4	CHC	35540.666666666667
5	SF	34638.666666666667
6	LAA	33304.666666666667
7	BOS	33273.111111111111
8	COL	32776.777777777778
9	ATL	32266.888888888889
10	HOU	32025.333333333333

Total rows: 10 Query complete 00:00:00.067

Figure 17. SQL SELECT query that defines a new attribute while using GROUP BY and ORDER BY.

```
SELECT player, team, season, so
FROM hitters
WHERE so < (SELECT AVG(so) FROM hitters WHERE season = 2023)
ORDER BY so ASC
LIMIT 10;
```

	player [PK] text	team [PK] text	season [PK] integer	so integer
1	Steven Brault	PIT	2016	0
2	Cole Figueroa	NYY	2015	0
3	Tim Cooney	STL	2015	0
4	Myles Straw	HOU	2018	0
5	Joe Hudson	LAA	2018	0
6	Austin Dean	SFG	2022	0
7	Jason Delay	PIT	2024	0
8	Steven Brault	PIT	2017	0
9	Breyvic Valera	STL	2017	0
10	Juan Graterol	LAA,MIN	2018	0

Total rows: 10 Query complete 00:00:00.071

Figure 18. SQL SELECT query that uses a subquery in a WHERE clause.

```
SELECT player, team, season, so
FROM pitchers
WHERE season = 2023
ORDER BY so DESC
LIMIT 5;
```

	player [PK] text	team [PK] text	season [PK] integer	so integer
1	Spencer Strider	ATL	2023	281
2	Kevin Gausm…	TOR	2023	237
3	Blake Snell	SDP	2023	234
4	Gerrit Cole	NYY	2023	222
5	Zac Gallen	ARI	2023	220

Total rows: 5 Query complete 00:00:00.115

Figure 19. SQL SELECT query that uses a WHERE clause.

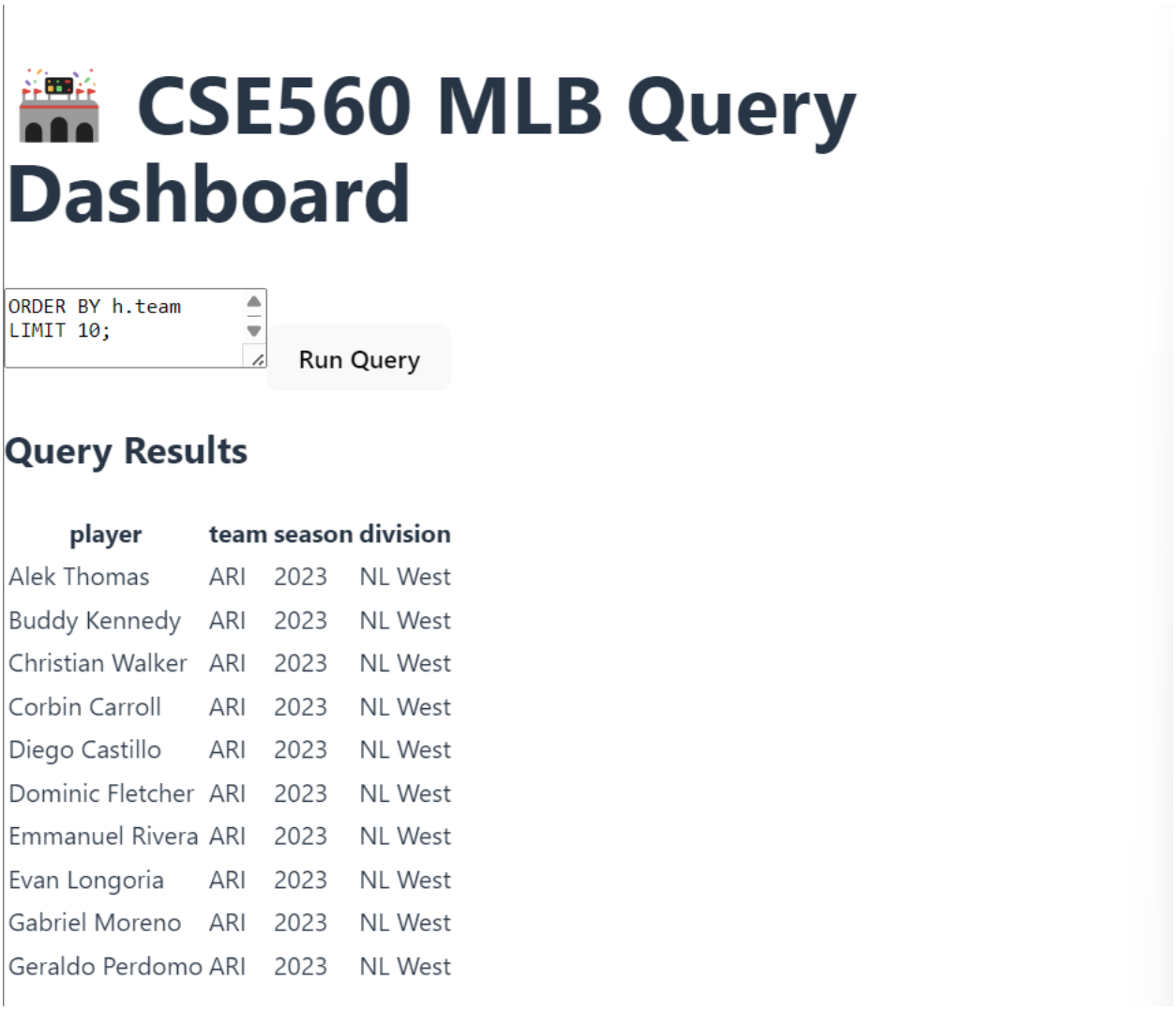

Figure 20. Screenshot of a running website that links the MLB database and can receive a query and display the result.