HOMEWORK #1 - SQL

Last Updated: Sep 07, 2025

Overview

The first homework is to construct a set of SQL queries for analyzing a dataset that will be provided to you. For this, you will look into Lahman Database. This homework is an opportunity to: (1) learn basic and certain advanced SQL features, and (2) get familiar with using a full-featured DBMS DuckDB that can be useful for you in the future.

This is a single-person project that will be completed individually (i.e., no groups).

- **Release Date:** Aug 27, 2025
- **Due Date:** Sep 07, 2025 @ 11:59pm

Specification

The homework contains six questions in total and is graded out of 100 points. For each question, you will need to construct a SQL query that fetches the desired data. Your answer for each question should contain only one statement. It will likely take you approximately six hours to complete the questions.

Placeholder Folder

Create the placeholder submission folder with the empty SQL files that you will use for each question:

```
$ mkdir placeholder
$ cd placeholder
$ touch q1_sample.duckdb.sql \
        q2_hr.duckdb.sql \
```

```
        q3_mvp.duckdb.sql \
        q4_award.duckdb.sql \
        q5_hof.duckdb.sql \
        q6_upsert.duckdb.sql
```

After filling in the queries, you can compress the folder by running the following command:

```
$ zip -j submission.zip placeholder/*.sql
```

The `-j` flag lets you compress all the SQL queries in the zip file without path information. The grading scripts will **not** work correctly unless you do this.

Instructions

1. Download the database dump file:

   ```
   $ wget https://15445.courses.cs.cmu.edu/fall2025/files/l
   ```

   Check its MD5 checksum to ensure that you have correctly downloaded the file:

   ```
   $ md5sum lahman-cmudb2025.db.gz
   a4fb302c11f80030f26bb6bb59d68cac  lahman-cmudb2025.db.gz
   ```

2. Unzip the database from the provided database dump by running the following commands on your shell. Note that the database file be **40MB** after you decompress it.

   ```
   $ tar zxf lahman-cmudb2025.db.gz
   ```

Then follow the instructions below to install DuckDB.

DuckDB

Please follow the instructions to install DuckDB for the command line environment.

We can directly load the dataset when we start DuckDB:

```
$ duckdb lahman-cmudb2025.db
DuckDB v1.3.2 (Ossivalis) 0b83e5d2f6
Enter ".help" for usage hints.
◗
```

1. You can check the contents of the database by running the `.tables` command on the DuckDB terminal. You should see **11 tables**. The output should look like this:

```
◗ .tables
appearances     collegeplaying  leagues         schools
awardsmanagers  divisions       managers        teams
awardsplayers   halloffame      people
```

Check the schema

Get familiar with the schema (structure) of the tables (what attributes do they contain, what are the primary and foreign keys). Run the `DESCRIBE $TABLE_NAME;` command on the `duckdb` terminal for each table. The output should look like the example below for each table.

appearances

```
◗ DESCRIBE appearances
┌──────────────┬──────────────┬─────────┬─────────┬─────────┐
│ column_name  │ column_type  │  null   │   key   │ default │
│   varchar    │   varchar    │ varchar │ varchar │ varchar │
├──────────────┼──────────────┼─────────┼─────────┼─────────┤
│ yearID       │ SMALLINT     │ NO      │ PRI     │ NULL    │
│ teamID       │ VARCHAR      │ NO      │ PRI     │ NULL    │
│ lgID         │ VARCHAR      │ YES     │ NULL    │ NULL    │
│ playerID     │ VARCHAR      │ NO      │ PRI     │ NULL    │
│ G_all        │ SMALLINT     │ YES     │ NULL    │ NULL    │
│ G_batting    │ SMALLINT     │ YES     │ NULL    │ NULL    │
│ HR           │ SMALLINT     │ YES     │ NULL    │ NULL    │
└──────────────┴──────────────┴─────────┴─────────┴─────────┘
```

Contains details for player appearances. For example, this is a row from the table:

```
2004|SFN|NL|aardsda01|11|11|0
```

The important fields are `G_all` (11) which describes the number of games played, `G_batting` (11) which describes the number of games batted, and `HR` (0) which describes the number of home runs hit.

awardsmanagers

```
⚪◗ DESCRIBE awardsmanagers
┌──────────────┬──────────────┬──────────┬──────────┬──────────┐
│ column_name  │ column_type  │   null   │   key    │ default  │
│   varchar    │   varchar    │ varchar  │ varchar  │ varchar  │
├──────────────┼──────────────┼──────────┼──────────┼──────────┤
│ playerID     │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ awardID      │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ yearID       │ SMALLINT     │ NO       │ PRI      │ NULL     │
│ lgID         │ VARCHAR      │ NO       │ PRI      │ NULL     │
└──────────────┴──────────────┴──────────┴──────────┴──────────┘
```

Contains details of awards given to managers. For example, this is a row from the table:

```
mccarjo99|TSN Manager of the Year|1936|ML
```

awardsplayers

```
⚪◗ DESCRIBE awardsplayers
┌──────────────┬──────────────┬──────────┬──────────┬──────────┐
│ column_name  │ column_type  │   null   │   key    │ default  │
│   varchar    │   varchar    │ varchar  │ varchar  │ varchar  │
├──────────────┼──────────────┼──────────┼──────────┼──────────┤
│ playerID     │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ awardID      │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ yearID       │ SMALLINT     │ NO       │ PRI      │ NULL     │
│ lgID         │ VARCHAR      │ NO       │ PRI      │ NULL     │
└──────────────┴──────────────┴──────────┴──────────┴──────────┘
```

Contains details of awards given to players. For example, this is a row from the table:

```
bondto01|Pitching Triple Crown|1877|NL
```

collegeplaying

```
⚪◗ DESCRIBE collegeplaying
┌──────────────┬──────────────┬──────────┬──────────┬──────────┐
│ column_name  │ column_type  │   null   │   key    │ default  │
│   varchar    │   varchar    │ varchar  │ varchar  │ varchar  │
├──────────────┼──────────────┼──────────┼──────────┼──────────┤
│ playerID     │ VARCHAR      │ NO       │ NULL     │ NULL     │
│ schoolID     │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ yearID       │ SMALLINT     │ YES      │ NULL     │ NULL     │
└──────────────┴──────────────┴──────────┴──────────┴──────────┘
```

Contains details of which players played for which school in a given year. For example, this is a row from the table:

```
aardsda01|pennst|2001
```

divisions

```
⚪◗ DESCRIBE divisions
┌──────────────┬──────────────┬──────────┬──────────┬──────────┐
│ column_name  │ column_type  │   null   │   key    │ default  │
│   varchar    │   varchar    │ varchar  │ varchar  │ varchar  │

├──────────────┼──────────────┼──────────┼──────────┼──────────┤
│ divID        │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ lgID         │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ division     │ VARCHAR      │ NO       │ NULL     │ NULL     │
│ active       │ VARCHAR      │ NO       │ NULL     │ NULL     │
└──────────────┴──────────────┴──────────┴──────────┴──────────┘
```

Contains information about divisions. For example, this is a row from the table:

```
E|AL|AL East|Y
```

halloffame

```
⚪◗ DESCRIBE halloffame
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar
playerID	VARCHAR	NO	PRI	NULL
yearid	SMALLINT	NO	PRI	NULL
inducted	VARCHAR	NO	PRI	NULL

Contains details of hall of fame players. For example, this is a row from the table:

```
abbotji01|2005|N
```

leagues

```
⚪◗ DESCRIBE leagues
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar
lgID	VARCHAR	NO	PRI	NULL
league	VARCHAR	NO	NULL	NULL
active	VARCHAR	NO	NULL	NULL

Contains information about leagues. For example, this is a row from the table:

```
ML|Major League|Y
```

managers

```
⚪◗ DESCRIBE managers
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar
playerID	VARCHAR	YES	NULL	NULL

```
│ yearID       │ SMALLINT     │ NO       │ PRI      │ NULL     │
│ teamID       │ VARCHAR      │ NO       │ PRI      │ NULL     │
│ lgID         │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ inseason     │ SMALLINT     │ NO       │ PRI      │ NULL     │
└──────────────┴──────────────┴──────────┴──────────┴──────────┘
```

Contains details about managers. For example, this is a row from the table:

```
actama99|2007|WAS|NL|1
```

people

```
⚪◗ DESCRIBE people

┌──────────────┬──────────────┬──────────┬──────────┬──────────┐
│ column_name  │ column_type  │   null   │   key    │ default  │
│   varchar    │   varchar    │ varchar  │ varchar  │ varchar  │
├──────────────┼──────────────┼──────────┼──────────┼──────────┤
│ ID           │ INTEGER      │ NO       │ NULL     │ NULL     │
│ playerID     │ VARCHAR      │ NO       │ PRI      │ NULL     │

│ birthYear    │ INTEGER      │ YES      │ NULL     │ NULL     │
│ birthMonth   │ INTEGER      │ YES      │ NULL     │ NULL     │
│ birthDay     │ INTEGER      │ YES      │ NULL     │ NULL     │
│ birthCountry │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ birthState   │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ birthCity    │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ deathYear    │ INTEGER      │ YES      │ NULL     │ NULL     │
│ deathMonth   │ INTEGER      │ YES      │ NULL     │ NULL     │
│ deathDay     │ INTEGER      │ YES      │ NULL     │ NULL     │
│ deathCountry │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ deathState   │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ deathCity    │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ nameFirst    │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ nameLast     │ VARCHAR      │ YES      │ NULL     │ NULL     │
│ nameGiven    │ VARCHAR      │ YES      │ NULL     │ NULL     │
├──────────────┼──────────────┼──────────┼──────────┼──────────┤
```

Contains details about players. For example, this is a row from the table:

```
1|aardsda01| … |David|Aardsma|David Allan
```

For us, the important fields are `playerID` (e.g., aardsda01), `nameFirst` (e.g., David), `nameLast` (e.g., Aardsma), and `nameGiven` (e.g., David Allan).

schools

```
⚪◗ DESCRIBE schools
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar
schoolID	VARCHAR	NO	PRI	NULL
name_full	VARCHAR	YES	NULL	NULL
city	VARCHAR	YES	NULL	NULL
state	VARCHAR	YES	NULL	NULL
country	VARCHAR	YES	NULL	NULL

Contains details about schools. For example, this is a row from the table:

```
abilchrist|Abilene Christian University|Abilene|TX|USA
```

teams

```
⚪◗ DESCRIBE teams
```

column_name varchar	column_type varchar	null varchar	key varchar	default varchar
yearID	SMALLINT	NO	PRI	NULL
lgID	VARCHAR	NO	PRI	NULL
teamID	VARCHAR	NO	PRI	NULL
franchID	VARCHAR	YES	NULL	NULL
divID	VARCHAR	YES	NULL	NULL
teamRank	SMALLINT	YES	NULL	NULL
name	VARCHAR	YES	NULL	NULL
attendance	INTEGER	YES	NULL	NULL

The table contains details about teams. For example, this is a row from the table:

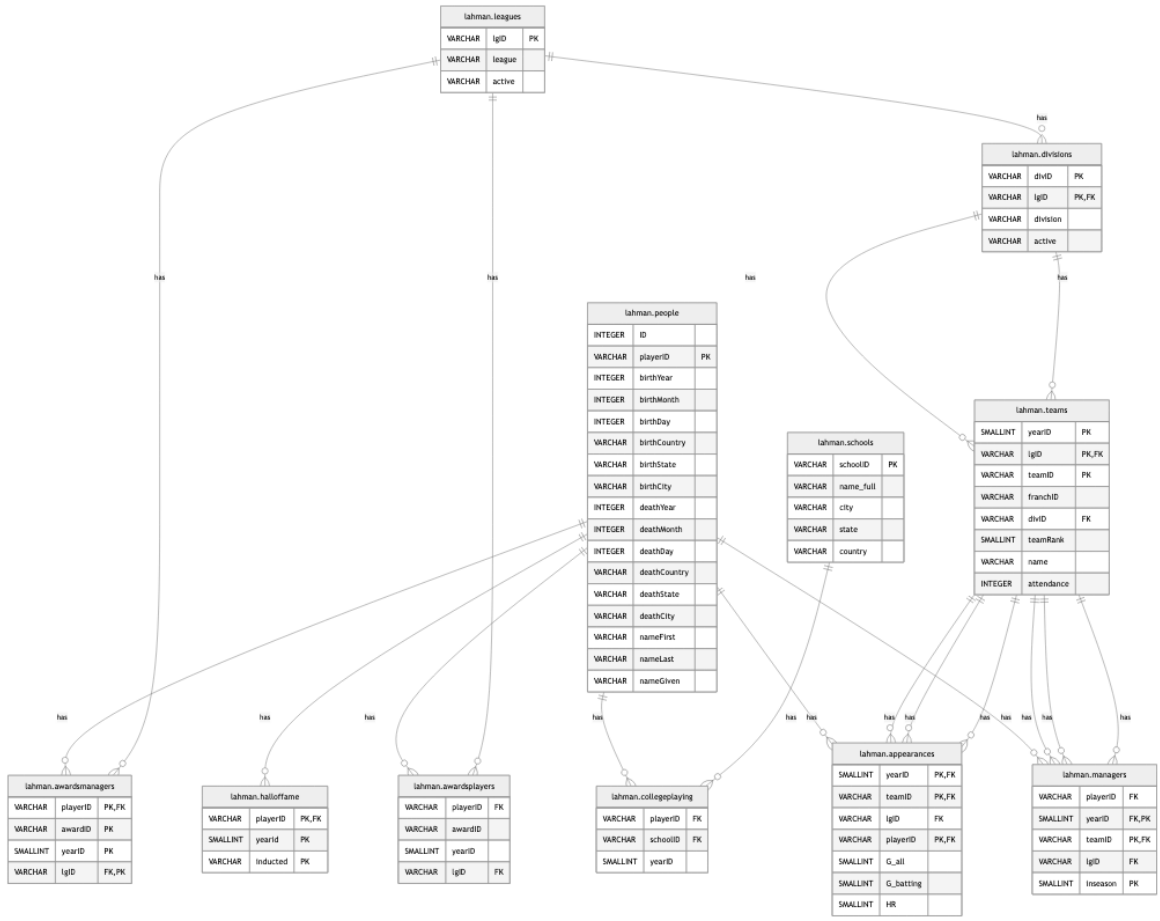
```
1884|UA|ALT|ALT|NULL|10|Altoona Mountain City|NULL
```

Sanity Check

Count the number of rows in the table

```
D select count(*) from appearances;
┌──────────────┐
│  count(1)    │
│   int64      │
├──────────────┤
│   115355     │
└──────────────┘
```

The following figure illustrates the schema of these tables:



Construct the SQL Queries

It's time to start constructing the SQL queries and put them into the placeholder files. You can start with using SQLite.

Q1 [0 points]

The purpose of this query is to make sure that the formatting of your output matches exactly the formatting of our auto-grading script.

File: `q1_sample.duckdb.sql`

Details: List all active divisions ordered alphabetically.

Answer: Here's the correct SQL query and expected output:

```
D SELECT DISTINCT division FROM divisions WHERE active
AL Central
AL East
AL West
NL Central
NL East
NL West
```

You should put this SQL query into the appropriate file (`q1_sample.duckdb.sql`) in the submission directory (`placeholder`).

Q2 [20 points]

Find ten people with the highest home-runs (HR) in any single appearance, irrespective of year or team. Only consider people who have played at some point for a school in `PA`. Order the results from most to least home runs, and then by **first name** alphabetically.

File: `q2_hr.duckdb.sql`

Hints: The `||` might be useful for constructing the name field.

The name should be assembled as `{FIRST} ({GIVEN}) {LAST}`. A player's highest home-runs in any appearance can be found by aggregating over all years and teams of a given player's `appearances`.

Your output should look like this:

```
name|max_hr_appearance
```

Your first row should look like this:

```
Bobby (Robert Leigh) Higginson|30
```

Q3 [20 points]

Find ten (player, team) pairs where the player won the `Gold Glove` award in an active league after 1999 and batted in more games than the player's team's average since 1999. Order by the number of distinct award-winning years from most to least, and then by given name alphabetically.

File: `q3_mvp.duckdb.sql`

Hints: Only consider awardID that matches `Gold Glove`. The team average can be computed by taking the average over the batted games by each team player's `appearances` since 1999. You might find Correlated Subqueries in DuckDB useful.

Your output should look like this:

```
nameGiven|teamID|distinct_years
```

Your first row should look like this:

```
Ichiro|SEA|10
```

Q4 [30 points]

Consider the event `E`: for a given (team, year), the team has more than 5 distinct players win an award and some manager has won

than 5 distinct players win an award and some manager has won an award in the same year. For all active leagues, find the teams where the event `E` has happened more than once. Order the results by the number of event `E` occurrences from most to least, and then by team name alphabetically.

File: `q4_award.duckdb.sql`

Hints: You might find Correlated Subqueries and CTEs in DuckDB useful. Consider breaking the problem into sub-problems and combining their results.

Your output should look like this:

```
league|team_name|distinct_years
```

One of the rows in the output looks like the following:

```
National League|Atlanta Braves|3
```

Q5 [20 points]

Find 10 Hall of Fame players who were inducted. For each inducted player, find the earliest teammate that they appeared with. If there are multiple teammates, pick the first teammate ordered by the teammate's constructed name alphabetically. Order by the hall of fame player's constructed name alphabetically.

File: `q5_hof.duckdb.sql`

Details: To find a player's teammates, find `appearances` where both players were playing for the same team in the same year.

Hints: You might find Lateral Joins in DuckDB useful. All names should be assembled as `{FIRST} ({GIVEN}) {LAST}`.

Your output should look like this:

```
hof_player_name|earliest_teammate_name|earliest_teammat
```

Your first row should look like this:

```
Addie (Adrian) Joss|Bill (William Henry) Bernhard|1902
```

Q6 [10 points]

The league director wants you to insert new records into the `teams` table. For all teams in 2024, insert a new record using the `franchID` as the `teamID`. If the insert fails due to a conflict, set the existing record's `attendance` to -1.

File: `q6_upsert.duckdb.sql`

Details: We will evaluate by selecting all teams in 2024 from the table and sorting by each column in order.

```
SELECT yearID, lgID, teamID, franchID, teamRank, attend
FROM teams
WHERE yearID = 2024
ORDER BY lgID, teamID, franchID, teamRank, attendance;
```

Hints Consider how you might be able to insert and update existing rows in a single statement.

After running the select statement, the following two rows should exist:

```
2024|AL|HOU|HOU|1|-1
2024|AL|KCA|KCR|2|1658347
```

Grading Rubric

Each submission will be graded based on whether the SQL queries fetch the expected sets of tuples from the database. Only one statement is allowed in each SQL query. Note that your SQL queries will be auto-graded by comparing their outputs (i.e. tuple sets) to the correct outputs. For your queries, the **order** of the output columns is important; their names are not.

Late Policy

See the late policy in the syllabus.

Submission

We use the Autograder from Gradescope for grading in order to provide you with immediate feedback. After completing the homework, you can submit your compressed folder `submission.zip` (only one file) to Gradescope:

- **https://www.gradescope.com/courses/1074751**

Important: Use the Gradescope course code announced on Piazza.

We will be comparing the output files using a function similar to `diff`. You can submit your answers as many times as you like.

Collaboration Policy

- Every student has to work individually on this assignment.
- Students are allowed to discuss high-level details about the project with others.
- Students are **not** allowed to copy the contents of a white-board after a group meeting with other students.
- Students are **not** allowed to copy the solutions from another colleague.

⚠ **WARNING:** All of the code for this project must be your own. You may not copy source code from other students or other sources that you find on the web. Plagiarism **will not** be tolerated. See CMU's Policy on Academic Integrity for additional information.