CS 121: Relational Databases — Winter 2026

Project 00: Introduction to SQL

Project Overview

Before we begin implementing a database, we should understand our end goal of executing SQL queries. In this project, you will write various queries to familiarize yourself with the relational model and different capabilities of SQL.

Setup

Register for the assignment by going to the Airtable link on the course website, which will create you a repository on GitLab with the starter code.

Then follow the SQL setup guide to install SQLite and familiarize yourself with the repo and database.

Task 1: Basic Queries (10 points)

1.i In the `People` table, find the `namefirst`, `namelast`, and `birthyear` for all players with a weight greater than 300 pounds.

1.ii Find the `namefirst`, `namelast`, and `birthyear` of all players whose `namefirst` field contains a space. Order the results by `namefirst`, breaking ties with `namelast`, both in ascending order.

1.iii Group players by `birthyear` and report the `birthyear`, average height (rounded to 4 decimal places), and number of players for each group. Order the results by `birthyear` in ascending order.

1.iv Modify the previous query to include only birthyears with an average height greater than 70. Again, order by `birthyear` in ascending order.

Task 1

Write a single SQL query to answer each of the above questions in: `q1i.sql`, `q1ii.sql`, `q1iii.sql` and `q1iv.sql`.

Task 2: Hall of Fame (20 points)

2.i Find the `namefirst`, `namelast`, `playerid`, and `yearid` of all people who were successfully inducted into the Hall of Fame in descending order of `yearid`. Break ties on `yearid` by `playerid` (ascending).

2.ii Find the people who were successfully inducted into the Hall of Fame and played in college at a school located in the state of California. For each person, return their `namefirst`, `namelast`, `playerid`, `schoolid`, and `yearid` in descending order of `yearid`. Break ties on `yearid` by `schoolid`, then `playerid` (ascending). *Note: `yearid` refers to the year of induction. A player may appear multiple times.*

2.iii Find the `playerid`, `namefirst`, `namelast`, and `schoolid` of all people who were successfully inducted into the Hall of Fame, whether or not they played in college. Return people in descending order of `playerid`. Break ties on `playerid` by `schoolid` (ascending). *Note: `schoolid` should be `NULL` if they did not play in college.*

Task 2

Write a single SQL query to answer each of the above questions in: `q2i.sql`, `q2ii.sql`, and `q2iii.sql`.

Task 3: SaberMetrics (30 points)

3.i Find the `playerid`, `namefirst`, `namelast`, `yearid`, and single-year `slg` rounded to 4 decimal places (Slugging Percentage) of the players with the 10 best annual Slugging Percentage recorded over all time.

- A player can appear multiple times.
- Only include players with more than 50 at-bats in the season.
- Order by `slg` descending, break ties by `yearid`, then `playerid` (both ascending).

Formula for Slugging Percentage:

$$\text{SLG} = \frac{(1\text{B}) + (2 \times 2\text{B}) + (3 \times 3\text{B}) + (4 \times \text{HR})}{\text{AB}}$$

*Data Note: You should compute `slg` as a floating point number. You will need to figure out how to convince SQL to do this! The `batting` table column `H` represents **all hits** (Singles + Doubles + Triples + Home Runs). Columns `2B` and `3B` are renamed to `H2B` and `H3B` in your local database (columns starting with numbers are tedious to write queries on). Finally, if a player played on multiple teams during the same season (for example `anderma02` in 2006) treat their time on each team separately for this calculation.*

3.ii Following the results from Question 3.i, find the `playerid`, `namefirst`, `namelast`, and `lslg` rounded to 4 decimal places (Lifetime Slugging Percentage) for the players with the top 10 Lifetime Slugging Percentage.

- Use the same formula as above, but aggregate hits and at-bats over the player's entire career. You will need to convert to total information across all time (earliest date recorded to last date recorded).
- Only include players with more than 50 at-bats across their lifetime.
- Order by `lslg` descending, break ties by `playerid` (ascending).

3.iii Find the `namefirst`, `namelast`, and Lifetime Slugging Percentage (`lslg` rounded to 4 decimal places) of batters whose lifetime slugging percentage is higher than that of San Francisco favorite Willie Mays.

- You may include Willie Mays' `playerid` (`mayswi01`) in your query logic, but do not hardcode his slugging percentage—you must calculate this as part of the query. Test your query by replacing `mayswi01` with another player's `playerid`, it should work for them as well.
- Only include players with more than 50 at-bats lifetime.
- Order by `namefirst, namelast` (ascending)

Task 3

Write a single SQL query to answer each of the above questions in: `q3i.sql`, `q3ii.sql`, and `q3iii.sql`.

Task 4: Salaries (40 points)

4.i Find the `yearid`, `min`, `max`, and `average` (rounded to 4 decimal places) of all player salaries for each year recorded, ordered by `yearid` in ascending order.

4.ii Compute the Year-over-Year change in min, max, and average player salary. For each year (after the first), return the `yearid`, `mindiff`, `maxdiff`, and `avgdiff` (rounded to 4 decimal places) with respect to the previous year. Order by `yearid` in ascending order. *Note: You should omit the very first year of recorded salaries from the result.*

4.iii In 2001, the max salary went up by over $6 million. Find the players that had the max salary in 2000 and 2001. Return the `playerid`, `namefirst`, `namelast`, `salary`, and `yearid`. If multiple players tied for max, return all of them.

4.iv Each team has at least one All-Star. For each team in 2016, give the `teamid` and `diffAvg` rounded to 4 decimal places (the difference in salary between the team's highest-paid All-Star and lowest-paid All-Star).
- Due to some discrepancies in the database, please draw your team names from the All-Star table (so use `allstarfull.teamid` in the SELECT statement for this).

Task 4

Write a single SQL query to answer each of the above questions in: `q4i.sql`, `q4ii.sql`, `q4iii.sql`, and `q4iv.sql`.

Submitting

Once you are finished, make sure your code is pushed to GitLab. Then, submit the commit hash you would like to be reviewed on Gradescope.

For more details, see the Project Submission instructions on the course website.

Acknowledgements

This project was initially developed for Berkeley's CS186 by their course staff. We've used it with permission.