COMS W4111: Introduction to Databases
Spring 2024, Sections 002/V02

Homework 2: Nonprogramming

Introduction

This notebook contains HW2 Nonprogramming. **Only students on the nonprogramming track should complete this part.** To ensure everything runs as expected, work on this notebook in Jupyter.

Submission instructions:

- You will submit **PDF and ZIP files** for this assignment. Gradescope will have two separate assignments for these.
- For the PDF:
  - The most reliable way to save as PDF is to go to your browser's menu bar and click `File -> Print`. **Switch the orientation to landscape mode**, and hit save.
  - **MAKE SURE ALL YOUR WORK (CODE AND SCREENSHOTS) IS VISIBLE ON THE PDF. YOU WILL NOT GET CREDIT IF ANYTHING IS CUT OFF.** Reach out for troubleshooting.
- For the ZIP:
  - Zip the folder that contains this notebook and any screenshots.

Setup

SQL Magic

In [1]:
```
%load_ext sql
```

You may need to change the password below.

In [2]:
```
%sql mysql+pymysql://root:dbuserdbuser@localhost
```

In [3]:
```
%sql SELECT 1
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[3]:

1
1

Python Libraries

In [4]:
```
import os

from IPython.display import Image
import pandas
from sqlalchemy import create_engine
```

You may need to change the password below.

In [5]:
```
engine = create_engine("mysql+pymysql://root:dbuserdbuser@localhost")
```

Load Data

- We're going to load data into a new database called `s24_lahmans_hw2`
- The data is stored as CSV files in the `data/` directory.

In [6]:
```
%sql DROP SCHEMA IF EXISTS s24_lahmans_hw2
%sql CREATE SCHEMA s24_lahmans_hw2
```

```
 * mysql+pymysql://root:***@localhost
6 rows affected.
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[6]:
```
[]
```

In [7]:
```
def load_csv(data_dir, file_name, schema, table_name=None):
    """
    :param data_dir: The directory containing the file.
    :param file_name: The file name.
    :param schema: The database for the saved table.
    :param table_name: The name of the table to create. If the name is None, the function uses the name of
        the file before '.csv'. So, file_name 'cat.csv' becomes table 'cat'.
    :return: None
    """

    if table_name is None:
        table_name = file_name.split(".")
        table_name = table_name[0]

    full_file_name = os.path.join(data_dir, file_name)

    df = pandas.read_csv(full_file_name)
    df.to_sql(table_name, con=engine, schema=schema, if_exists="replace", index=False)
```

```
In [8]: data_dir = "data"
        csv_files = [
            "People.csv",
            "Appearances.csv",
            "Batting.csv",
            "Pitching.csv",
            "Teams.csv",
            "Managers.csv",
        ]
        schema = "s24_lahmans_hw2"

        for f in csv_files:
            load_csv(data_dir, f, schema)
            print("Loaded file:", f)
```

```
Loaded file: People.csv
Loaded file: Appearances.csv
Loaded file: Batting.csv
Loaded file: Pitching.csv
Loaded file: Teams.csv
Loaded file: Managers.csv
```

Data Cleanup

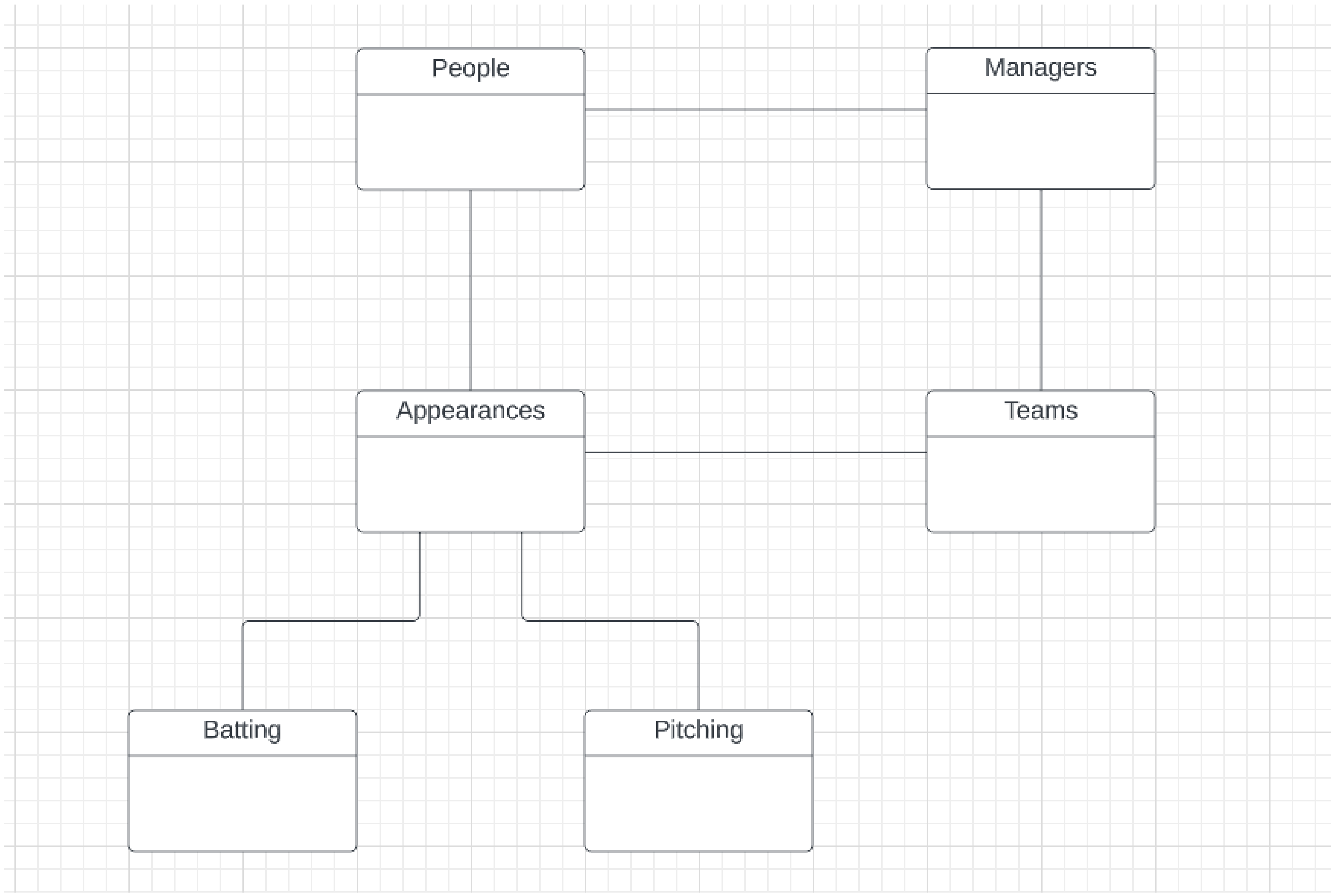
- The `load_csv` function above created new tables and inserted data into them for us
- Unfortunately, because it cannot guess our intentions, the tables have generic data types and are not related to each other
- You will fix these issues

```
In [9]: %sql USE s24_lahmans_hw2
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

```
Out[9]: []
```

Below is an overview of the six tables that we inserted and how they should be related.



Lahmans Database

People

- The `People` table is defined as

```
create table People
(
    playerID     text   null,
    birthYear    double null,
    birthMonth   double null,
    birthDay     double null,
    birthCountry text   null,
    birthState   text   null,
    birthCity    text   null,
    deathYear    double null,
    deathMonth   double null,
    deathDay     double null,
    deathCountry text   null,
    deathState   text   null,
    deathCity    text   null,
    nameFirst    text   null,
    nameLast     text   null,
    nameGiven    text   null,
    weight       double null,
    height       double null,
    bats         text   null,
    throws       text   null,
    debut        text   null,
    finalGame    text   null,
    retroID      text   null,
    bbrefID      text   null
);
```

1. Convert `playerID`, `retroID`, and `bbrefID` to **minimally sized** `CHAR`

   A. Minimally sized means that the length passed into `CHAR` must be as small as possible while still being able to contain a `playerID` (i.e., don't simply choose a random large number)
   B. `playerID`, `retroID`, and `bbrefID` may have different minimal sizes
   C. You don't need to show how you got the minimal sizes
2. Convert the `DOUBLE` columns to `INT`
3. Convert `bats and throws to ENUM`
4. Create two new columns, `dateOfBirth and dateOfDeath` of type `DATE`. Populate these columns based on `birthYear`, `birthMonth`, `birthDay`, `deathYear`, `deathMonth`, and `deathDay`. If any of these columns are null, you can set the corresponding new column to null (i.e., only keep full dates).
5. Convert `debut and finalGame to DATE`

In [10]:
```
%%sql

/*SELECT MAX(CHAR_LENGTH(playerID)), MAX(CHAR_LENGTH(retroID)), MAX(CHAR_LENGTH(bbrefID))
FROM People*/

ALTER TABLE People
MODIFY COLUMN playerID CHAR(9), -- part 1
MODIFY COLUMN retroID CHAR(8), -- part 1
MODIFY COLUMN bbrefID CHAR(9), -- part 1
MODIFY COLUMN birthDay INT, -- part 2
MODIFY COLUMN birthMonth INT, -- part 2
MODIFY COLUMN birthYear INT, -- part 2
MODIFY COLUMN deathDay INT, -- part 2
MODIFY COLUMN deathMonth INT, -- part 2
MODIFY COLUMN deathYear INT, -- part 2
MODIFY COLUMN height INT, -- part 2
MODIFY COLUMN weight INT, -- part 2
MODIFY COLUMN bats ENUM('R','L','B'), -- part 3
MODIFY COLUMN throws ENUM('R','L','S'), -- part 3
MODIFY COLUMN debut DATE, -- part 5
MODIFY COLUMN finalGame DATE; -- part 5
```

```
 * mysql+pymysql://root:***@localhost
20370 rows affected.
```

Out[10]:
```
[]
```

In [11]:
```
%%sql
-- part 4
ALTER TABLE People
ADD dateOfBirth DATE,
ADD dateOfDeath DATE;

UPDATE People
SET
dateOfBirth=IF(
    birthYear+birthMonth+birthDay,
    DATE(CONCAT(birthYear,'-',birthMonth,'-',birthDay)),
    NULL
),
dateOfDeath=IF(
    deathYear+deathMonth+deathDay,
    DATE(CONCAT(deathYear,'-',deathMonth,'-',deathDay)),
    NULL
);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
20370 rows affected.
```

Out[11]:
```
[]
```

In [12]:
```
%%sql
-- part 3
select distinct bats from People
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[12]:

bats
R
L
None
B

```
In [13]: %%sql
         -- part 3
         select distinct throws from People
```

```
 * mysql+pymysql://root:***@localhost
4 rows affected.
```

Out[13]:

throws
R
L
None
S

Managers

- The `Managers` table is defined as

```
create table Managers
(
    playerID text   null,
    yearID   bigint null,
    teamID   text   null,
    lgID     text   null,
    inseason bigint null,
    G        bigint null,
    W        bigint null,
    L        bigint null,
    `rank`   bigint null,
    plyrMgr  text   null
);
```

1. Convert `playerID`, `teamID`, and `lgID` to minimally sized `CHAR`
2. Convert `yearID` to `CHAR(4)`
3. Convert `plyrMgr` to `BOOLEAN`. This may require creating a temporary column.

- You should use `ALTER TABLE` to modify attributes (columns) and `UPDATE TABLE` to modify data (rows)

```
In [14]: %%sql
/*SELECT MAX(CHAR_LENGTH(playerID)), MAX(CHAR_LENGTH(teamID)), MAX(CHAR_LENGTH(lgID))
FROM Managers*/

ALTER TABLE Managers
MODIFY COLUMN playerID CHAR(9), -- part 1
MODIFY COLUMN teamID CHAR(3), -- part 1
MODIFY COLUMN lgID CHAR(2), -- part 1
MODIFY COLUMN yearID CHAR(4) -- part 2
```

```
 * mysql+pymysql://root:***@localhost
3684 rows affected.
```

```
Out[14]: []
```

```
In [15]: %%sql
-- part 3
AlTER TABLE Managers
ADD COLUMN tmp_plyrMgr BOOLEAN
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

```
Out[15]: []
```

```
In [16]: %%sql
-- part 3
UPDATE Managers
SET tmp_plyrMgr = (plyrMgr = 'Y'); -- part 3
```

```
 * mysql+pymysql://root:***@localhost
3684 rows affected.
```

```
Out[16]: []
```

```
In [17]: %%sql
         -- part 3
         ALTER TABLE Managers
         DROP COLUMN plyrMgr; -- part 3

          * mysql+pymysql://root:***@localhost
         0 rows affected.

Out[17]: []
```

```
In [18]: %%sql
         -- part 3
         ALTER TABLE Managers
         RENAME COLUMN tmp_plyrMgr TO plyrMgr; -- part 3

          * mysql+pymysql://root:***@localhost
         0 rows affected.

Out[18]: []
```

Bonus point: MySQL has a `YEAR` type, but we choose to not use it for `yearID`. Can you figure out why?

The reason is `YEAR` type can only have range 1901 to 2155 and `yearID` has values smaller than 1901

Appearances

- The `Appearances` table is defined as

```
        create table Appearances
        (
            yearID    bigint null,
            teamID    text   null,
            lgID      text   null,
            playerID  text   null,
            G_all     bigint null,
            GS        double null,
            G_batting bigint null,
            G_defense double null,
            G_p       bigint null,
            G_c       bigint null,
            G_1b      bigint null,
            G_2b      bigint null,
            G_3b      bigint null,
            G_ss      bigint null,
            G_lf      bigint null,
            G_cf      bigint null,
            G_rf      bigint null,
            G_of      bigint null,
```

In [19]:
```
%%sql
ALTER TABLE Appearances
MODIFY COLUMN yearID CHAR(4), -- part 1
MODIFY COLUMN teamID CHAR(3), -- part 2
MODIFY COLUMN lgID CHAR(2), -- part 2
MODIFY COLUMN playerID CHAR(9); -- part 2
```

```
 * mysql+pymysql://root:***@localhost
110422 rows affected.
```

Out[19]:
```
[]
```

```
In [20]: %%sql
         -- part 2
         SELECT MAX(CHAR_LENGTH(teamID)), MAX(CHAR_LENGTH(lgID)), MAX(CHAR_LENGTH(playerID))
         FROM Appearances
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[20]:

MAX(CHAR_LENGTH(teamID))	MAX(CHAR_LENGTH(lgID))	MAX(CHAR_LENGTH(playerID))
3	2	9

Batting

- The `Batting` table is defined as

```
create table Batting
(
    playerID text   null,
    yearID   bigint null,
    stint    bigint null,
    teamID   text   null,
    lgID     text   null,
    G        bigint null,
    AB       bigint null,
    R        bigint null,
```

In [21]:
```
%%sql
-- part 1
SELECT MAX(CHAR_LENGTH(teamID)), MAX(CHAR_LENGTH(lgID)), MAX(CHAR_LENGTH(playerID))
FROM Batting
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[21]:

MAX(CHAR_LENGTH(teamID))	MAX(CHAR_LENGTH(lgID))	MAX(CHAR_LENGTH(playerID))
3	2	9

In [22]:
```
%%sql
ALTER TABLE Batting
MODIFY COLUMN playerID CHAR(9), -- part 1
MODIFY COLUMN teamID CHAR(3), -- part 1
MODIFY COLUMN lgID CHAR(2), -- part 1
MODIFY COLUMN yearID CHAR(4); -- part 2
```

```
 * mysql+pymysql://root:***@localhost
110493 rows affected.
```

Out[22]:
```
[]
```

Pitching

- The `Pitching` table is defined as

```
create table Pitching
(
    playerID text   null,
    yearID   bigint null,
    stint    bigint null,
    teamID   text   null,
    lgID     text   null,
    W        bigint null,
    L        bigint null,
    G        bigint null,
    GS       bigint null,
    CG       bigint null,
    SHO      bigint null,
    SV       bigint null,
    IPouts   bigint null,
    H        bigint null,
    ER       bigint null,
    HR       bigint null,
    BB       bigint null,
    SO       bigint null,
    BAOpp    double null,
    ERA      double null,
    IBB      double null,
    WP       bigint null,
    HBP      double null,
    BK       bigint null,
    BFP      double null,
    GF       bigint null,
    R        bigint null,
    SH       double null,
    SF       double null,
    GIDP     double null
);
```

1. Convert `playerID`, `teamID`, and `lgID` to minimally sized `CHAR`
2. Convert `yearID` to `CHAR(4)`

- You should use `ALTER TABLE` to modify attributes (columns) and `UPDATE TABLE` to modify data (rows)

In [23]:
```
%%sql
-- part 1
SELECT MAX(CHAR_LENGTH(teamID)), MAX(CHAR_LENGTH(lgID)), MAX(CHAR_LENGTH(playerID))
FROM Pitching
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[23]:

MAX(CHAR_LENGTH(teamID))	MAX(CHAR_LENGTH(lgID))	MAX(CHAR_LENGTH(playerID))
3	2	9

In [24]:
```
%%sql
ALTER TABLE Pitching
MODIFY COLUMN playerID CHAR(9), -- part 1
MODIFY COLUMN teamID CHAR(3), -- part 1
MODIFY COLUMN lgID CHAR(2), -- part 1
MODIFY COLUMN yearID CHAR(4); -- part 2
```

```
 * mysql+pymysql://root:***@localhost
49430 rows affected.
```

Out[24]:
```
[]
```

Teams

- The `Teams` table is defined as

```
create table Teams
(
    yearID          bigint null,
    lgID            text   null,
    teamID          text   null,
    franchID        text   null,
    divID           text   null,
    `Rank`          bigint null,
    G               bigint null,
    Ghome           double null,
    W               bigint null,
    L               bigint null,
    DivWin          text   null,
    WCWin           text   null,
    LgWin           text   null,
    WSWin           text   null,
    R               bigint null,
    AB              bigint null,
    H               bigint null,
    `2B`            bigint null,
    `3B`            bigint null,
    HR              bigint null,
    BB              double null,
    SO              double null,
    SB              double null,
    CS              double null,
    HBP             double null,
    SF              double null,
    RA              bigint null,
    ER              bigint null,
    ERA             double null,
    CG              bigint null,
    SHO             bigint null,
    SV              bigint null,
    IPouts          bigint null,
    HA              bigint null,
    HRA             bigint null,
    BBA             bigint null,
    SOA             bigint null,
    E               bigint null,
    DP              bigint null,
    FP              double null,
    name            text   null,
    park            text   null,
    attendance      double null,
    BPF             bigint null,
    PPF             bigint null,
    teamIDBR        text   null,
```

In [25]:
```
%%sql
-- part 1
SELECT MAX(CHAR_LENGTH(teamID)), MAX(CHAR_LENGTH(franchID)), MAX(CHAR_LENGTH(divID))
FROM Teams
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[25]:

MAX(CHAR_LENGTH(teamID))	MAX(CHAR_LENGTH(franchID))	MAX(CHAR_LENGTH(divID))
3	3	1

In [26]:
```
%%sql
ALTER TABLE Teams
MODIFY COLUMN teamID CHAR(3), -- part 1
MODIFY COLUMN franchID CHAR(3), -- part 1
MODIFY COLUMN divID CHAR(1), -- part 1
MODIFY COLUMN yearID CHAR(4); -- part 2
```

```
 * mysql+pymysql://root:***@localhost
2985 rows affected.
```

Out[26]:
```
[]
```

Primary Keys

- You will now add primary keys to the tables
- The PKs for the tables are
  - People: `playerID`
  - Managers: `(playerID, yearID, inseason)`
  - Appearances: `(playerID, yearID, teamID)`
  - Batting: `(playerID, yearID, stint)`
  - Pitching: `(playerID, yearID, stint)`
  - Teams: `(teamID, yearID)`
- Write and execute statements showing why `(playerID, yearID, teamID)` is a valid PK for Appearances
  - You should show that the PK is non-null for all rows and unique across all rows

In [27]:
```
%%sql
-- This shows (playerID, yearID, teamID) does not have null values for all rows

SELECT COUNT(DISTINCT playerID, yearID, teamID) FROM Appearances
WHERE playerID IS NULL OR yearID IS NULL OR teamID IS NULL;
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[27]:

COUNT(DISTINCT playerID, yearID, teamID)
0

In [28]:
```
%%sql
/*Since the number of unique PK (distinct playerID, yearID, teamID) in Appearances
is the same as the number of rows in Appearances,
(distinct playerID, yearID, teamID) is unique*/

SELECT
    COUNT(DISTINCT playerID, yearID, teamID) AS distinct_count,
    COUNT(*) AS total_count
FROM Appearances
WHERE playerID IS NOT NULL AND yearID IS NOT NULL AND teamID IS NOT NULL;
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[28]:

distinct_count	total_count
110422	110422

- Write and execute `ALTER TABLE` statements to add the primary keys to the tables

In [29]:
```
%%sql
ALTER TABLE People
ADD PRIMARY KEY (playerID)
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[29]: `[]`

In [30]:
```
%%sql
ALTER TABLE Managers
ADD PRIMARY KEY (playerID, yearID, inseason)
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[30]: `[]`

```
In [31]: %%sql
         ALTER TABLE Appearances
         ADD PRIMARY KEY (playerID, yearID, teamID)

          * mysql+pymysql://root:***@localhost
         0 rows affected.

Out[31]: []
```

```
In [32]: %%sql
         ALTER TABLE Batting
         ADD PRIMARY KEY (playerID, yearID, stint)

          * mysql+pymysql://root:***@localhost
         0 rows affected.

Out[32]: []
```

```
In [33]: %%sql
         ALTER TABLE Pitching
         ADD PRIMARY KEY (playerID, yearID, stint)

          * mysql+pymysql://root:***@localhost
         0 rows affected.

Out[33]: []
```

```
In [34]: %%sql
         ALTER TABLE Teams
         ADD PRIMARY KEY (teamID, yearID)

          * mysql+pymysql://root:***@localhost
         0 rows affected.

Out[34]: []
```

Foreign Keys

- You will now add foreign keys to the tables

- **The conceptual ER diagram above should indicate to you which tables are related by foreign keys**
  - You need to figure out which table in a relationship has the foreign key
- Write and execute statements showing why `Appearances.playerID` is a valid FK referencing `People.playerID`
  - You should show that all the values in `Appearances.playerID` appear in `People.playerID`

In [35]:
```
%%sql
/*This statement shows the number of Appearances.playerID not in People.playerID is 0,
which shows all the values in Appearances.playerID appear in People.playerID*/
SELECT COUNT(DISTINCT Appearances.playerID)
FROM Appearances
WHERE Appearances.playerID NOT IN (SELECT People.playerID FROM People WHERE People.playerID IS NOT NULL);
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

Out[35]:

COUNT(DISTINCT Appearances.playerID)
0

- Write and execute `ALTER TABLE` statements to add foreign keys to the tables

In [36]:
```
%%sql
ALTER TABLE Appearances
ADD FOREIGN KEY (playerID) REFERENCES People(playerID)
```

```
 * mysql+pymysql://root:***@localhost
110422 rows affected.
```

Out[36]:
```
[]
```

In [37]:
```
%%sql
ALTER TABLE Batting
ADD FOREIGN KEY (playerID, yearID, teamID) REFERENCES Appearances(playerID, yearID, teamID)
```

```
 * mysql+pymysql://root:***@localhost
110493 rows affected.
```

Out[37]:
```
[]
```

```
In [38]: %%sql
ALTER TABLE Pitching
ADD FOREIGN KEY (playerID, yearID, teamID) REFERENCES Appearances(playerID, yearID, teamID)

 * mysql+pymysql://root:***@localhost
49430 rows affected.

Out[38]: []
```

```
In [39]: %%sql
ALTER TABLE Appearances
ADD FOREIGN KEY (teamID, yearID) REFERENCES Teams(teamID, yearID)

 * mysql+pymysql://root:***@localhost
110422 rows affected.

Out[39]: []
```

```
In [40]: %%sql
ALTER TABLE Managers
ADD FOREIGN KEY (teamID, yearID) REFERENCES Teams(teamID, yearID)

 * mysql+pymysql://root:***@localhost
3684 rows affected.

Out[40]: []
```

```
In [41]: %%sql
ALTER TABLE Managers
ADD FOREIGN KEY (playerID) REFERENCES People(playerID)

 * mysql+pymysql://root:***@localhost
3684 rows affected.

Out[41]: []
```

SQL Queries

On-Base Percentage and Slugging

- The formula for `onBasePercentage` is

$$\frac{(H - 2B - 3B - HR) + 2 \times 2B + 3 \times 3B + 4 \times HR)}{AB}$$

- `2B`, `3B`, `HR`, and `AB` are their own columns, not multiplication
- Write a query that returns a table of form

  ```
  (playerID, nameFirst, nameLast, yearID, stint, H, AB, G, onBasePercentage)
  ```

- Your table should be sorted on `onBasePercentage` from highest to lowest, then on last name alphabetically (if there are any ties in `onBasePercentage`)
- **To avoid freezing your notebook, add a `LIMIT 10` to the end of your query to display only the first 10 rows**
- You may use the `Batting` and `People` tables

In [42]:
```
%%sql
SELECT
    People.playerID AS playerID,
    People.nameFirst AS nameFirst,
    People.nameLast AS nameLast,
    Batting.yearID AS yearID,
    Batting.stint AS stint,
    Batting.H AS H,
    Batting.AB AS AB,
    Batting.G AS G,
    ((Batting.H - Batting.2B - Batting.3B - Batting.HR) +
     2 * Batting.2B + 3 * Batting.3B + 4 * Batting.HR) / Batting.AB AS onBasePercentage
FROM
    People JOIN Batting ON
        People.playerID = Batting.playerID
ORDER BY
    onBasePercentage DESC,
    nameLast ASC
LIMIT 10
```

```
 * mysql+pymysql://root:***@localhost
10 rows affected.
```

Out[42]:

playerID	nameFirst	nameLast	yearID	stint	H	AB	G	onBasePercentage
chacigu01	Gustavo	Chacin	2010	1	1	1	44	4.0000
hernafe02	Felix	Hernandez	2008	1	1	1	31	4.0000
lefebbi01	Bill	LeFebvre	1938	1	1	1	1	4.0000
motagu01	Guillermo	Mota	1999	1	1	1	51	4.0000
narumbu01	Buster	Narum	1963	1	1	1	7	4.0000
perrypa02	Pat	Perry	1988	2	1	1	35	4.0000
quirkja01	Jamie	Quirk	1984	2	1	1	1	4.0000
rogered01	Eddie	Rogers	2005	1	1	1	8	4.0000
sleatlo01	Lou	Sleater	1958	1	1	1	4	4.0000
yanes01	Esteban	Yan	2000	1	1	1	43	4.0000

Players and Managers

- A person in `People` was a player if their `playerID` appears in `Appearances`
- A person in `People` was a manager if their `playerID` appears in `Managers`
- A person could have been both a player and manager
- Write a query that returns a table of form

  ```
  (playerID, nameFirst, nameLast, careerPlayerGames, careerManagerGames)
  ```

- `careerPlayerGames` is the sum of `Appearances.G_all` for a single player
  - It should be 0 if the person was never a player
- `careerManagerGames` is the sum of `Managers.G` for a single manager
  - It should be 0 if the person was never a manager
- Your table should be sorted on `careerPlayerGames + careerManagerGames` from highest to lowest
- **To avoid freezing your notebook, add a `LIMIT 10` to the end of your query to display only the first 10 rows**
- You may use the `People` , `Appearances` , and `Managers` tables.

```
In [43]: %%sql
SELECT
    P.playerID AS playerID,
    P.nameFirst AS nameFirst,
    P.nameLast AS nameLast,
    COALESCE(A.G_all, 0) AS careerPlayerGames,
    COALESCE(M.G, 0) AS careerManagerGames
FROM
    People P
LEFT JOIN
    (SELECT
         playerID,
         SUM(G_all) AS G_all
     FROM
         Appearances
     GROUP BY
         playerID) A ON P.playerID = A.playerID
LEFT JOIN
    (SELECT
         playerID,
         SUM(G) AS G
     FROM
         Managers
     GROUP BY
         playerID) M ON P.playerID = M.playerID
ORDER BY
    careerPlayerGames + careerManagerGames DESC
LIMIT 10
```

```
 * mysql+pymysql://root:***@localhost
10 rows affected.
```

Out[43]:

	playerID	nameFirst	nameLast	careerPlayerGames	careerManagerGames
	mackco01	Connie	Mack	724	7755
	torrejo01	Joe	Torre	2209	4323
	mcgrajo01	John	McGraw	1105	4769
	bakerdu01	Dusty	Baker	2039	3704
	harribu01	Bucky	Harris	1262	4410
	larusto01	Tony	LaRussa	132	5248
	durocle01	Leo	Durocher	1637	3739
	pinielo01	Lou	Piniella	1747	3536
	dykesji01	Jimmy	Dykes	2283	2962
	clarkfr01	Fred	Clarke	2246	2829

- Copy and paste your query from above. Modify it to only show people who were never managers.
    - This should be a one-line change

```
In [44]: %%sql
SELECT * FROM (
    SELECT
        P.playerID AS playerID,
        P.nameFirst AS nameFirst,
        P.nameLast AS nameLast,
        COALESCE(A.G_all, 0) AS careerPlayerGames,
        COALESCE(M.G, 0) AS careerManagerGames
    FROM
        People P
    LEFT JOIN
        (SELECT
             playerID,
             SUM(G_all) AS G_all
         FROM
             Appearances
         GROUP BY
             playerID) A ON P.playerID = A.playerID
    LEFT JOIN
        (SELECT
             playerID,
             SUM(G) AS G
         FROM
             Managers
         GROUP BY
             playerID) M ON P.playerID = M.playerID
    ORDER BY
        careerPlayerGames + careerManagerGames DESC
) tmp
WHERE
    careerManagerGames = 0
LIMIT 10
```

```
 * mysql+pymysql://root:***@localhost
10 rows affected.
```

Out[44]:

playerID	nameFirst	nameLast	careerPlayerGames	careerManagerGames
yastrca01	Carl	Yastrzemski	3308	0
aaronha01	Hank	Aaron	3298	0
henderi01	Rickey	Henderson	3081	0
musiast01	Stan	Musial	3026	0
murraed02	Eddie	Murray	3026	0
ripkeca01	Cal	Ripken	3001	0
mayswi01	Willie	Mays	2992	0
bondsba01	Barry	Bonds	2986	0
winfida01	Dave	Winfield	2973	0
pujolal01	Albert	Pujols	2971	0

In [ ]: