# 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 [2]:
```
%load_ext sql
```

You may need to change the password below.

In [3]:
```
%sql mysql+pymysql://root:dbuserbdbuser@localhost
```

In [4]:
```
%sql SELECT 1
```

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

Out[4]:

| 1 |
|---|
| 1 |

### Python Libraries

In [5]:
```
import os

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

You may need to change the password below.

In [6]:
```
engine = create_engine("mysql+pymysql://root:dbuserbdbuser@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 [7]:
```
%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[7]:
```
[]
```

In [8]:
```
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 [9]:
```
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 [10]:
```
%sql USE s24_lahmans_hw2
```

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

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

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`

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

```
In [35]: %%sql

# 1
ALTER TABLE People
MODIFY COLUMN playerID CHAR(9);

ALTER TABLE People
MODIFY COLUMN retroID CHAR(8);
```

```
ALTER TABLE People
MODIFY COLUMN bbrefID CHAR(9);


# 2
ALTER TABLE People
MODIFY COLUMN birthYear INT;

ALTER TABLE People
MODIFY COLUMN birthMonth INT;

ALTER TABLE People
MODIFY COLUMN birthDay INT;

ALTER TABLE People
MODIFY COLUMN deathYear INT;

ALTER TABLE People
MODIFY COLUMN deathMonth INT;

ALTER TABLE People
MODIFY COLUMN deathDay INT;

ALTER TABLE People
MODIFY COLUMN weight INT;

ALTER TABLE People
MODIFY COLUMN height INT;


# 3
ALTER TABLE People
MODIFY COLUMN bats ENUM("R", "L", "B");

ALTER TABLE People
MODIFY COLUMN throws ENUM("R", "L", "S");


# 4
ALTER TABLE People
ADD dateOfBirth DATE;

ALTER TABLE People
ADD dateOfDeath DATE;

UPDATE People
SET dateOfBirth = CONCAT(birthYear, "-", birthMonth, "-", birthDay);

UPDATE People
SET dateOfDeath = CONCAT(deathYear, "-", deathMonth, "-", deathDay);


# 5
ALTER TABLE People
MODIFY COLUMN debut DATE;

ALTER TABLE People
MODIFY COLUMN finalGame DATE;
```

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

Out[35]:
```
[]
```

## 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 [36]:
```
%%sql

# 1
ALTER TABLE Managers
MODIFY COLUMN playerID CHAR(9);

ALTER TABLE Managers
MODIFY COLUMN teamID CHAR(3);

ALTER TABLE Managers
MODIFY COLUMN lgID CHAR(2);


# 2
ALTER TABLE Managers
MODIFY COLUMN yearID CHAR(4);


# 3
ALTER TABLE Managers
ADD plyrMgrBool BOOLEAN;

UPDATE Managers
```

```
SET plyrMgrBool = TRUE
WHERE plyrMgr = "Y";

UPDATE Managers
SET plyrMgrBool = FALSE
WHERE plyrMgr = "N";

ALTER TABLE Managers
DROP COLUMN plyrMgr;

ALTER TABLE Managers
RENAME COLUMN plyrMgrBool TO plyrMgr;
```

```
 * mysql+pymysql://root:***@localhost
3684 rows affected.
3684 rows affected.
3684 rows affected.
3684 rows affected.
0 rows affected.
645 rows affected.
3039 rows affected.
0 rows affected.
0 rows affected.
```

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

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

Our database contains years earlier than 1901, however, the `YEAR` can only hold years from 1901 to 2155.

## 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,
    G_dh      double null,
    G_ph      double null,
    G_pr      double null
);
```

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

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

In [37]:
```
%%sql

# 1
ALTER TABLE Appearances
```

```
MODIFY COLUMN yearID CHAR(4);


# 2
ALTER TABLE Appearances
MODIFY COLUMN teamID CHAR(3);

ALTER TABLE Appearances
MODIFY COLUMN lgID CHAR(2);

ALTER TABLE Appearances
MODIFY COLUMN playerID CHAR(9);
```

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

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

## 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,
    H        bigint null,
    `2B`     bigint null,
    `3B`     bigint null,
    HR       bigint null,
    RBI      double null,
    SB       double null,
    CS       double null,
    BB       bigint null,
    SO       double null,
    IBB      double null,
    HBP      double 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 [38]:
```
%%sql

# 1
ALTER TABLE Batting
MODIFY COLUMN playerID CHAR(9);

ALTER TABLE Batting
MODIFY COLUMN teamID CHAR(3);

ALTER TABLE Batting
MODIFY COLUMN lgID CHAR(2);
```

```
# 2
ALTER TABLE Batting
MODIFY COLUMN yearID CHAR(4);
```

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

Out[38]:
```
[]
```

## 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 [39]:
```
%%sql

# 1
ALTER TABLE Pitching
MODIFY COLUMN playerID CHAR(9);

ALTER TABLE Pitching
MODIFY COLUMN teamID CHAR(3);

ALTER TABLE Pitching
MODIFY COLUMN lgID CHAR(2);
```

```
# 2
ALTER TABLE Pitching
MODIFY COLUMN yearID CHAR(4);
```

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

Out[39]:
```
[]
```

## 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,
    teamIDlahman45 text   null,
    teamIDretro    text   null
);
```

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

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

In [40]:
```
%%sql

# 1
ALTER TABLE Teams
MODIFY yearID CHAR(4);


# 2
ALTER TABLE Teams
MODIFY COLUMN lgID CHAR(2);

ALTER TABLE Teams
MODIFY COLUMN teamID CHAR(3);

ALTER TABLE Teams
MODIFY COLUMN franchID CHAR(3);

ALTER TABLE Teams
MODIFY COLUMN divID CHAR(1);
```

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

Out[40]:
```
[]
```

## 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 [41]:
```
%%sql

WITH isAnyNullTbl AS (
    SELECT COUNT(*) AS isAnyNull
    FROM Appearances
    WHERE
        playerID IS NULL
        OR yearID IS NULL
        OR teamID IS NULL
),

isAllUniqueTbl AS (

    SELECT COUNT(*) = COUNT(DISTINCT playerID, yearID, teamID) AS isAllUnique
    FROM Appearances
)
SELECT * FROM isAnyNullTbl JOIN isAllUniqueTbl;
```

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

Out[41]:

| isAnyNull | isAllUnique |
|---|---|
| 0 | 1 |

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

In [42]:

```
%%sql

ALTER TABLE People
ADD PRIMARY KEY (playerID);

ALTER TABLE Managers
ADD PRIMARY KEY (playerID, yearID, inseason);

ALTER TABLE Appearances
ADD PRIMARY KEY (playerID, yearID, teamID);

ALTER TABLE Batting
ADD PRIMARY KEY (playerID, yearID, stint);

ALTER TABLE Pitching
ADD PRIMARY KEY (playerID, yearID, stint);

ALTER TABLE Teams
ADD PRIMARY KEY (teamID, yearID);
```

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

Out[42]:

```
[]
```

## 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 [43]:

```
%%sql

SELECT COUNT(*) AS numMismatch
FROM Appearances
WHERE playerID NOT IN (
    SELECT playerID
    FROM People
);
```

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

Out[43]:

| numMismatch |
|---|
| 0 |

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

In [44]:

```
%%sql

ALTER TABLE Managers
ADD FOREIGN KEY (playerID) REFERENCES People(playerID);

ALTER TABLE Appearances
ADD FOREIGN KEY (playerID) REFERENCES People(playerID);
```

```
ALTER TABLE Managers
ADD FOREIGN KEY (teamID, yearID) REFERENCES Teams(teamID, yearID);

ALTER TABLE Appearances
ADD FOREIGN KEY (teamID, yearID) REFERENCES Teams(teamID, yearID);

ALTER TABLE Batting
ADD FOREIGN KEY (playerID, yearID, teamID) REFERENCES Appearances(playerID, yearID, teamID);

ALTER TABLE Pitching
ADD FOREIGN KEY (playerID, yearID, teamID) REFERENCES Appearances(playerID, yearID, teamID);
```

```
 * mysql+pymysql://root:***@localhost
3684 rows affected.
110422 rows affected.
3684 rows affected.
110422 rows affected.
110493 rows affected.
49430 rows affected.
```

Out[44]:
```
[]
```

---

## SQL Queries

### On-Base Percentage and Slugging

- The formula for `onBasePercentage` is

$$\frac{(H - 2B - 3B - HR) + 2 \times 2B + 3 \times 3B + 4 \times HR)}{AB} \tag{1}$$

- `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 [12]:
```
%%sql

SELECT
    playerID,
    nameFirst,
    nameLast,
    yearID,
    stint,
    H,
    AB,
    G,
    (
        (H - `2B` - `3B` - HR + 2*`2B` + 3*`3B` + 4*HR) / `AB`
    ) AS onBasePercentage
FROM
    Batting
JOIN People USING (playerID)
ORDER BY
    onBasePercentage DESC, nameLast
LIMIT 10;
```

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

Out[12]:

| 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 [30]:

```
%%sql

WITH
playersTbl AS (
    SELECT playerID, SUM(G_all) AS careerPlayerGames
    FROM Appearances
    GROUP BY playerID
),
managersTbl AS (
    SELECT playerID, SUM(G) AS careerManagerGames
    FROM Managers
    GROUP BY playerID
)
SELECT
    playerID,
    nameFirst,
    nameLast,
    IFNULL(careerPlayerGames, 0) AS careerPlayerGames,
    IFNULL(careerManagerGames, 0) AS careerManagerGames
FROM
    People
    LEFT JOIN playersTbl USING (playerID)
    LEFT JOIN managersTbl USING (playerID)
ORDER BY
    (IFNULL(careerPlayerGames, 0) + IFNULL(careerManagerGames, 0)) DESC

LIMIT 10;
```

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

Out[30]:

| 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 [27]:

```
%%sql

WITH
playersTbl AS (
    SELECT playerID, SUM(G_all) AS careerPlayerGames
    FROM Appearances
    GROUP BY playerID
),
managersTbl AS (
    SELECT playerID, SUM(G) AS careerManagerGames
    FROM Managers
    GROUP BY playerID
)
SELECT
    playerID,
    nameFirst,
    nameLast,
    IFNULL(careerPlayerGames, 0) AS careerPlayerGames,
    IFNULL(careerManagerGames, 0) AS careerManagerGames
FROM
    People
    LEFT JOIN playersTbl USING (playerID)
    LEFT JOIN managersTbl USING (playerID)
WHERE IFNULL(careerManagerGames, 0) = 0 #Exclude
ORDER BY
    (IFNULL(careerPlayerGames, 0) + IFNULL(careerManagerGames, 0)) DESC
LIMIT 10;
```

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

Out[27]:

| 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 |