COMS W4111: Introduction to Databases
Spring 2023, Sections 002

Homework 1, Part 2
Introduction to Core Concepts, ER Modeling, Relational Algebra, SQL

* </i>*

Introduction and Overview

HW Objectives

- *HW 1 part 1 covered general topics and knowledge from the class material. Part 2 has practical exercises.*
- *The notebook contains core practical exercises for both tracks (Programming, Non-Programming). All students complete this section.*
- *There are not track specific assignments for this HW.*

Submission Instructions

Complete all the tests in this notebook and submit only this notebook as a PDF to GradeScope. To convert the jupyter notebook into a pdf you can use either of the following methods:

- *File --> Print Preview --> Print --> Save to PDF*
- *File --> Download As HTML --> Print --> Save to PDF*

Due date: February 12, 11:59 PM EDT on GradeScope

It is recommended that you put the screenshots into the same folder as this notebook so you do not have to alter the path to include your images.

Please read all the instructions thoroughly!

Guidelines

*You may not work with or collaborate with anyone in any way to complete the homework. You may speak with the professor and TAs. You may ask **private** questions on Ed if you need clarification.*

You may use lecture slides, the textbook slides, the textbook or public information on the web to help you answer your questions. You may not "cut and past" information. Your answer must be in your own words and demonstrate the you understand the concept. If you use information for sources other than lectures, lecture slides, textbook slides or the textbook, you MUST provide a URL to the source you used.

Add Student Information

1. *Replace my name with your full name.*
2. *Replace my UNI with your UNI.*
3. *Replace "Cool Track" with either "Programming" or "Non-programming."*

In [ ]:
```
# Print your name, uni, and track below

name = "Yixuan Wang"
uni = "yw3928"
track = "Programming Track"

print(name)
print(uni)
print(track)
```

```
Yixuan Wang
yw3928
Programming Track
```

Testing Environment

Run the following cells to ensure that your environment is set up.

You may need to change passwords.

General Packages

In [ ]:
```
import json
```

In [ ]:
```
import csv
```

In [ ]:
```
import pandas
```

In [ ]:
```
import os
```

pymysql

In [ ]:
```
import pymysql
```

In [ ]:
```
#
# Run this cell but change your user ID and password.
#
pymysql_conn = pymysql.connect(
    user="root",
    password="12345678",
    host="localhost",
    port=3306,
    autocommit=True,
    cursorclass=pymysql.cursors.DictCursor
)
```

In [ ]:
```
cursor = pymysql_conn.cursor()
sql = "show databases"
res = cursor.execute(sql)
databases = cursor.fetchall()
```

In [ ]:
```
#
# Your list of databases will be different.
# You are fine as long as you do not get an error.
#
databases
```

Out[ ]:
```
[{'Database': 'db_book'},
 {'Database': 'information_schema'},
 {'Database': 'mysql'},
 {'Database': 'performance_schema'},
 {'Database': 'sys'}]
```

ipython-SQL

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

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [ ]:
```
#
# Remember to change your user ID and password.
#
%sql mysql+pymysql://root:12345678@localhost
```

Out[ ]:
```
'Connected: root@None'
```

In [ ]:
```
%sql select * from db_book.student where ID=12345
```

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

Out[ ]:

ID	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32

SQLAlchemy

In [ ]:
```
from sqlalchemy import create_engine
```

In [ ]:
```
#
# Remember to change your user ID and password.
#
sql_url = "mysql+pymysql://root:12345678@localhost"
```

In [ ]:
```
engine = create_engine(sql_url)
```

In [ ]:
```
sql = "select * from db_book.student"
df = pandas.read_sql(sql, con=engine)
```

In [ ]:
```
df
```

Out[ ]:

	ID	name	dept_name	tot_cred
0	00128	Zhang	Comp. Sci.	102.0
1	12345	Shankar	Comp. Sci.	32.0
2	19991	Brandt	History	80.0
3	23121	Chavez	Finance	110.0
4	44553	Peltier	Physics	56.0
5	45678	Levy	Physics	46.0
6	54321	Williams	Comp. Sci.	54.0
7	55739	Sanchez	Music	38.0
8	70557	Snow	Physics	0.0
9	76543	Brown	Comp. Sci.	58.0
10	76653	Aoi	Elec. Eng.	60.0
11	98765	Bourikas	Elec. Eng.	98.0
12	98988	Tanaka	Biology	120.0

Common Exercises

Loading Data

- *If you are running the notebook in the folder that you cloned/downloaded, there are files in the `data` directory.*

In [ ]:
```
!ls data
```

```
Appearances.csv                          course_feed.json
Batting.csv                              course_info.json
Managers.csv                             departments.csv
People.csv                               evalkit_eval_courses_instructors.json
Pitching.csv                             evalkit_eval_instructors.json
Teams.csv                                instructors.json
characters-groups.csv                    scenes.csv
characters.csv                           tmp.py
```

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

- *Use `%sql` to create a databases schema `lahmanshw1`.*

In [ ]:
```
#
# Answer
#
%sql create schema lahmanshw1
```

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

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

- *The class lecture showed how to load a CSV file in Pandas and save to a database.*
- *Load and save the CSV files. You should implement the function and then call in the following cells.*

In [ ]:
```
def load_and_save_csv(data_dir:str, file_name:str, schema:str, table_name:str=N
    """

    :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, th
        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)

    #
    # TODO: Remove answer and have your code goes here.
    #
    df = pandas.read_csv(full_file_name)
    df.to_sql(table_name, con=engine, schema=schema, if_exists='replace', index
```

In [ ]:
```
# for f in csv_files:
#     load_and_save_csv(data_dir, f, "lahmanshw1")
#     print("Saved file:", f)
#read the csv file in reverse order
for f in csv_files[::-1]:
    load_and_save_csv(data_dir, f, "lahmanshw1")
    print("Saved file:", f)
```

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

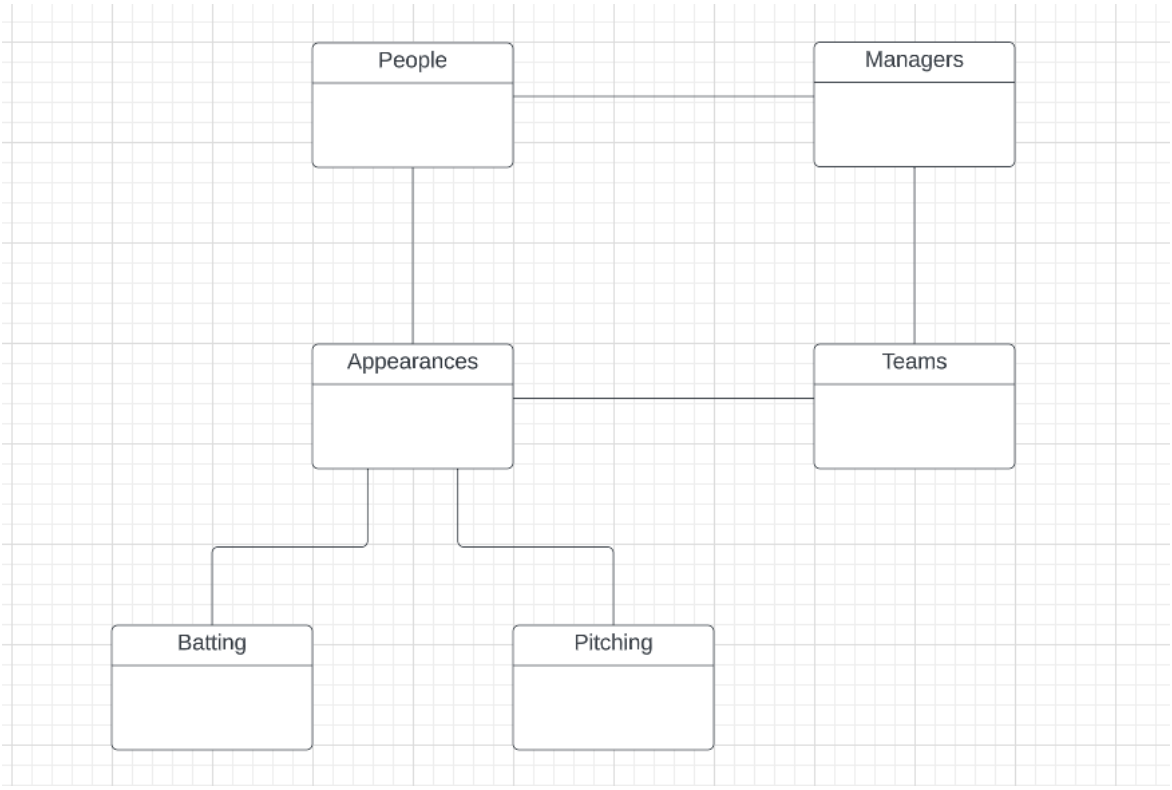
In [ ]:
```
#
# The following should get the create table statements for the tables
# you created above.
#
# This code is here just because I was bored.
#
tables = %sql show tables from lahmanshw1
table_names = [t[0] for t in tables]
table_names

all_tables = ""

%sql use lahmanshw1

for t in table_names:
    sql = "show create table " + t
    tmp = %sql $sql
    tmp = tmp[0][1]
    all_tables += "\n\n" + tmp
```

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

In [ ]:
```
#
# If you want to see the schema printed as text, just uncomment the following a
#
#print(all_tables)
```

Schema and Data Cleanup

- *There is a section below for each of the tables you created/loaded.*
- *There is a set of instructions in each section.*
- *You are going to "clean up" the tables, primarily focusing on keys.*
- *A conceptual ER diagram for the schema is:*



Conceptual ER Diagram

People

- *The `people` table scheme is below.*

```
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
);
```

- *You are to implement the following tasks:*
  1. *Determine reasonable lengths for `text` columns and convert the columns to `varchar`.*
  2. *Convert the columns that are `double` to `int`.*
  3. *Add columns `dateOfBirth` and `dateOfDeath` of type `Date`. Set the values of the new columns based on the `birthYear, birthMonth, birthDay, deathYear, deathMonth, deathDay` column values.*
  4. *Change `bats` and `throws` to `ENUM` types.*
  5. *Convert the column type for `debug` and `finalGame` to date and set the values correctly.*

- *You implement the tasks changing the schema by executing `ALTER TABLE` statements.*

- *Changing or setting values usually requires you to execute `UPDATE` statements.*
- *You need to execute you statements in the cells below. You may add additional cells.*

In [ ]:
```
%sql select max(length(playerid)) from people
%sql select max(length(birthCountry)) from people
%sql select max(length(birthState)) from people
%sql select max(length(birthCity)) from people
%sql select max(length(deathCountry)) from people
%sql select max(length(deathState)) from people
%sql select max(length(deathCity)) from people
%sql select max(length(nameFirst)) from people
%sql select max(length(nameLast)) from people
%sql select max(length(nameGiven)) from people
```

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

Out[ ]:

max(length(nameGiven))
43

In [ ]:
```
%sql ALTER TABLE People MODIFY playerID varchar(9);
%sql ALTER TABLE People MODIFY nameFirst varchar(15);

%sql ALTER TABLE People MODIFY nameLast varchar(15);
%sql ALTER TABLE People MODIFY nameGiven varchar(50);
%sql ALTER TABLE People MODIFY birthCity varchar(30);
%sql ALTER TABLE People MODIFY birthState varchar(25);
%sql ALTER TABLE People MODIFY birthCountry varchar(15);
%sql ALTER TABLE People MODIFY deathCity varchar(30);
%sql ALTER TABLE People MODIFY deathState varchar(20);
%sql ALTER TABLE People MODIFY deathCountry varchar(15);

%sql ALTER TABLE People MODIFY birthYear int;
%sql ALTER TABLE People MODIFY birthMonth int;
%sql ALTER TABLE People MODIFY birthDay int;
%sql ALTER TABLE People MODIFY deathYear int;
%sql ALTER TABLE People MODIFY deathMonth int;
%sql ALTER TABLE People MODIFY deathDay int;
%sql ALTER TABLE People MODIFY weight int;
```

```
%sql ALTER TABLE People MODIFY height int;

%sql ALTER TABLE People ADD dateOfBirth DATE;
%sql ALTER TABLE People ADD dateOfDeath DATE;
%sql UPDATE People SET dateOfBirth = CONCAT_WS('-', birthYear, birthMonth, birt
%sql UPDATE People SET dateOfDeath = CONCAT_WS('-', deathYear, deathMonth, deat

%sql ALTER TABLE People MODIFY bats ENUM('R', 'L', 'B');
%sql ALTER TABLE People MODIFY throws ENUM('R', 'L', 'S') null;

%sql ALTER TABLE People MODIFY debut DATE;
%sql ALTER TABLE People MODIFY finalGame DATE;
```

```
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
0 rows affected.
 * mysql+pymysql://root:***@localhost
0 rows affected.
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (1292, "Incorrect date value: '1868' for column
'dateOfBirth' at row 86")
[SQL: UPDATE People SET dateOfBirth = CONCAT_WS('-', birthYear, birthMonth, bi
rthDay);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (1292, "Incorrect date value: '' for column 'da
teOfDeath' at row 1")
[SQL: UPDATE People SET dateOfDeath = CONCAT_WS('-', deathYear, deathMonth, de
athDay);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
 * mysql+pymysql://root:***@localhost
20370 rows affected.
```

Out[ ]: []

Managers

- The schema for the managers table is

```
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
);
```

- You are to implement the following tasks:
  - Convert `playerID, teamID, lgID` to `varchar` with reasonable sizes.
  - Convert `yearID` to `char(4)`. I will explain why we are not using the data type `Year`.
  - Convert `plyrMgr` to a `BOOLEAN`.
- Some of the tasks may require both `ALTER TABLE` and `UPDATE`.

In [ ]:
```
%sql ALTER TABLE Managers MODIFY playerID varchar(9);
%sql ALTER TABLE Managers MODIFY teamID varchar(3);
%sql ALTER TABLE Managers MODIFY lgID varchar(2);

%sql ALTER TABLE Managers MODIFY yearID char(4);

%sql UPDATE Managers SET plyrMgr = IF(plyrMgr = 'Y', 1, 0);
%sql ALTER TABLE Managers MODIFY plyrMgr BOOLEAN;
```

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

Out[ ]: []

Appearances

- *The schema for the appearances table is*

```
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
);
```

- *Do not worry about the columns that are numeric (`double, bigint`).*
- *Tasks:*
  - *Convert `yearID` to `char(4)`*
  - *Convert the `text` columns to reasonably sized `varchar`.*

In [ ]:
```
%sql ALTER TABLE Appearances MODIFY yearID char(4);
%sql ALTER TABLE Appearances MODIFY teamID varchar(3);
%sql ALTER TABLE Appearances MODIFY lgID varchar(2);
%sql ALTER TABLE Appearances MODIFY playerID varchar(9);
```

```
 * mysql+pymysql://root:***@localhost
110423 rows affected.
 * mysql+pymysql://root:***@localhost
110423 rows affected.
 * mysql+pymysql://root:***@localhost
110423 rows affected.
 * mysql+pymysql://root:***@localhost
110423 rows affected.
```

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

Batting

- *The Batting table is*

```
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
);
```

- *You only need to fix the definitions* `playerID, teamID, yearID` *and* `lgID`.

In [ ]:
```
%sql ALTER TABLE Batting MODIFY playerID varchar(9);
%sql ALTER TABLE Batting MODIFY teamID varchar(3);
%sql ALTER TABLE Batting MODIFY lgID varchar(2);
%sql ALTER TABLE Batting MODIFY yearID char(4);
```

```
 * mysql+pymysql://root:***@localhost
110495 rows affected.
 * mysql+pymysql://root:***@localhost
110495 rows affected.
 * mysql+pymysql://root:***@localhost
110495 rows affected.
 * mysql+pymysql://root:***@localhost
110495 rows affected.
```

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

In [ ]:
```

```

In [ ]:
```

```

Pitching

- *The Pitching table is:*

```
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
);
```

- *You only need to fix the definitions* `playerID, teamID, yearID` *and* `lgID`.

In [ ]:
```
%sql ALTER TABLE Pitching MODIFY playerID varchar(9);
%sql ALTER TABLE Pitching MODIFY teamID varchar(3);
%sql ALTER TABLE Pitching MODIFY lgID varchar(2);
%sql ALTER TABLE Pitching MODIFY yearID char(4);
```

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

Out[ ]: `[]`

In [ ]:

In [ ]:

Teams

- *The Teams table is:*

```
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
);
```

- *You need to make the following changes:*
  - *Convert* `yearID, teamID, lgID, franchID, divId` *to reasonable types.*
  - *Convert* `DivWin, WCWin, LGWin, WSWin` *to* `boolean`.

In [ ]:
```
%sql ALTER TABLE Teams MODIFY yearID char(4);
%sql ALTER TABLE Teams MODIFY teamID varchar(3);
%sql ALTER TABLE Teams MODIFY lgID varchar(2);
%sql ALTER TABLE Teams MODIFY franchID varchar(3);
%sql ALTER TABLE Teams MODIFY divID varchar(5);
%sql UPDATE Teams SET DivWin = IF(DivWin = '',0,1);
%sql UPDATE Teams SET WCWin = IF(WCWin = '',0,1);
%sql UPDATE Teams SET LgWin = IF(LgWin = '',0,1);
%sql UPDATE Teams SET WSWin = IF(WSWin = '',0,1);
%sql ALTER TABLE Teams MODIFY DivWin boolean;
%sql ALTER TABLE Teams MODIFY WCWin boolean;
%sql ALTER TABLE Teams MODIFY LgWin boolean;
%sql ALTER TABLE Teams MODIFY WSWin boolean;
```

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

Out[ ]: []

Keys

Primary Keys

- *In the following cells, write and SQL statements that demonstrates the combination of columns that is a valid primary key for each of the 6 tables.*

In [ ]:
```
%%sql
use lahmanshw1;
select count(*) as all_count, count(distinct playerID) as playerID_count from
```

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

Out[ ]:

all_count	playerID_count
20370	20370

In [ ]:
```
%%sql
select count(*) as all_count, count(distinct yearID, teamID, inseason) as ID_co
```

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

Out[ ]:

all_count	ID_count
3684	3684

In [ ]:
```
%%sql
select count(*) as all_count, count(distinct yearID, teamID, playerID) as ID_co
```

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

Out[ ]:

all_count	ID_count
110423	110423

In [ ]:
```
%%sql
select count(*) as all_count, count(distinct teamID,playerID, stint, yearID) as
```

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

Out[ ]:

all_count	ID_count
110495	110495

In [ ]:
```
%%sql
select count(*) as all_count, count(distinct playerID, yearID, stint) as ID_cou
```

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

Out[ ]:

all_count	ID_count
49430	49430

In [ ]:
```
%%sql
select count(*) as all_count, count(distinct teamID, yearID) as ID_count from T
```

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

Out[ ]:

all_count	ID_count
2985	2985

- *Write and execute SQL* `ALTER TABLE` *staments to add the primary keys to the tables.*

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

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

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

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

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

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

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

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

Out[ ]: []

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

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

Out[ ]: []

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

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

Out[ ]: []

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

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

Out[ ]: []

- *You will need to write queries that determine which columns form the foreign keys in the relationships. Write and execute your queries below.*

In [ ]:
```
# %sql SELECT * FROM Managers WHERE teamID NOT IN (SELECT teamID FROM Teams);
%sql select * from managers where (yearID, teamID) not in (select yearID, teaml
```

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

Out[ ]:

playerID	yearID	teamID	lgID	inseason	G	W	L	rank	plyrMgr

In [ ]:
```
%sql select * from batting where (playerID) not in (select playerID from appeal
```

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

Out[ ]:

playerID	yearID	stint	teamID	lgID	G	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	IBB	H

In [ ]:
```
%sql select * from Pitching where playerID not in (select playerID from appeara
```

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

Out[ ]:

playerID	yearID	stint	teamID	lgID	W	L	G	GS	CG	SHO	SV	IPouts	H	ER	HR	BB	SO

In [ ]:
```
%sql select * from teams where (teamID, yearID) not in (select teamID, yearID f
```

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

Out[ ]:

yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin	WSWin

In [ ]:
```
%sql select * from managers where (yearID, teamID) not in (select yearID, teaml
```

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

Out[ ]:

playerID	yearID	teamID	lgID	inseason	G	W	L	rank	plyrMgr

- *Write and execute the `ALTER TABLE` statements to create the foreign keys.*
- ***NOTE:** There may be some minor issues with missing or incorrect data. You can delete a few rows if necessary.*

In [ ]:
```
#add foreign keys to managers referecing appearances
%sql ALTER TABLE Managers ADD CONSTRAINT fk_managers_appearances FOREIGN KEY (y
```

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

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

In [ ]:
```
#add foreign keys to batting referecing appearances
%sql ALTER TABLE Batting ADD CONSTRAINT fk_batting_appearances FOREIGN KEY (pla
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (1822, "Failed to add the foreign key constrain
t. Missing index for constraint 'fk_batting_appearances' in the referenced tab
le 'Appearances'")
[SQL: ALTER TABLE Batting ADD CONSTRAINT fk_batting_appearances FOREIGN KEY (p
layerID) REFERENCES Appearances(playerID);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
```

In [ ]:
```
#add foreign keys to pitching referecing appearances
%sql ALTER TABLE Pitching ADD CONSTRAINT fk_pitching_appearances FOREIGN KEY (p
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (1822, "Failed to add the foreign key constrain
t. Missing index for constraint 'fk_pitching_appearances' in the referenced ta
ble 'Appearances'")
[SQL: ALTER TABLE Pitching ADD CONSTRAINT fk_pitching_appearances FOREIGN KEY
(playerID) REFERENCES Appearances(playerID);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
```

In [ ]:
```
#add foreign keys to teams referecing appearances
%sql ALTER TABLE Teams ADD CONSTRAINT fk_teams_appearances FOREIGN KEY (teamID,
```

```
 * mysql+pymysql://root:***@localhost
(pymysql.err.OperationalError) (1822, "Failed to add the foreign key constrain
t. Missing index for constraint 'fk_teams_appearances' in the referenced table
'Appearances'")
[SQL: ALTER TABLE Teams ADD CONSTRAINT fk_teams_appearances FOREIGN KEY (teamI
D, yearID) REFERENCES Appearances(teamID, yearID);]
(Background on this error at: https://sqlalche.me/e/14/e3q8)
```

SQL Queries

On-Base Percentage and Slugging

- *Use the `Batting` table and `People` table.*
- *The formula for `onBasePercentage` is:*

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

- *Write a query that returns a table of the form*

  (playerID, nameLast, nameFirst, h, ab, G, onBasePercentage)
- *Test your query with* `playerID` `willite01`.

In [ ]:
```
%%sql
select b.playerID, p.nameLast, p.nameFirst, b.h, b.ab, b.G,
(b.h - b.2B - b.3B - b.HR + 2 * b.2B + 3 * b.3B + 4 * b.HR)/b.AB as onBasePerce
from lahmanshw1.Batting b join lahmanshw1.People p on b.playerID = p.playerID
where p.playerID = 'willite01';
```

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

Out[ ]:

playerID	nameLast	nameFirst	h	ab	G	onBasePercentage
willite01	Williams	Ted	185	565	149	0.6088
willite01	Williams	Ted	193	561	144	0.5936
willite01	Williams	Ted	185	456	143	0.7346
willite01	Williams	Ted	186	522	150	0.6475
willite01	Williams	Ted	176	514	150	0.6673
willite01	Williams	Ted	181	528	156	0.6345
willite01	Williams	Ted	188	509	137	0.6149
willite01	Williams	Ted	194	566	155	0.6502
willite01	Williams	Ted	106	334	89	0.6467
willite01	Williams	Ted	169	531	148	0.5556
willite01	Williams	Ted	4	10	6	0.9000
willite01	Williams	Ted	37	91	37	0.9011
willite01	Williams	Ted	133	386	117	0.6347
willite01	Williams	Ted	114	320	98	0.7031
willite01	Williams	Ted	138	400	136	0.6050
willite01	Williams	Ted	163	420	132	0.7310
willite01	Williams	Ted	135	411	129	0.5839
willite01	Williams	Ted	69	272	103	0.4191
willite01	Williams	Ted	98	310	113	0.6452

In [ ]:
```
```

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.`*
- *Write a query that returns a table of the form*

  `(playerID, nameLast, nameFirst, career_player_games, career_manager_games)`
- *`career_player_games` is the sum of `Appearances.G_all`. The value should be `0` if the person was never a player.*
- *`career_manager_games` is the sum of `Managers.G`. The value should be `0` if the person was never a manager.*
- *Test your query with players born in California with `nameLast` "Williams."*

In [ ]:
```
%%sql
select p.playerID, p.nameLast, p.nameFirst,
       (CASE WHEN p.playerID in (select playerID from Appearances)
       THEN (select sum(G_all) from Appearances where playerID = p.playerID) El
       (CASE WHEN p.playerID in (select playerID from Managers)
         THEN (select sum(G) from Managers where playerID = p.playerID) ELSE 0
from lahmanshw1.People p
where p.nameLast = 'Williams' and p.birthState = 'CA';
```

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

Out[ ]:

playerID	nameLast	nameFirst	career_player_games	career_manager_games
willibe01	Williams	Bernie	102	0
willido02	Williams	Don	3	0
williji03	Williams	Jimy	14	1700
willike02	Williams	Ken	451	0
willima04	Williams	Matt	1866	324
willimi02	Williams	Mitch	619	0
williri02	Williams	Rinaldo	4	0
williri03	Williams	Rick	48	0
willish01	Williams	Shad	14	0
willite01	Williams	Ted	2292	637
willitr01	Williams	Trevor	129	0

In [ ]: