COMS W4111: Introduction to Databases Spring 2023, Sections 002

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

</span> </center></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 [1]:
```
# Print your name, uni, and track below

name = "Haoqing Wang"
uni = "hw2888"
track = "Programming"

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

```
Haoqing Wang
hw2888
Programming
```

Testing Environment

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

You may need to change passwords.

General Packages

In [2]:
```
import json
```

In [3]:
```
import csv
```

In [4]:
```
import pandas
```

In [5]:
```
import os
```

pymysql

In [6]:
```
import pymysql
```

In [7]:
```
#
# Run this cell but change your user ID and password.
#
pymysql_conn = pymysql.connect(
    user="root",
    password="WHQ21cd1c689742",
    host="localhost",
    port=3306,
    autocommit=True,
    cursorclass=pymysql.cursors.DictCursor
)
```

In [8]:
```
cursor = pymysql_conn.cursor()
sql = "show databases"
res = cursor.execute(sql)
databases = cursor.fetchall()
```

In [9]:
```
#
# Your list of databases will be different.
# You are fine as long as you do not get an error.
#
databases
```

Out[9]:
```
[{'Database': 'db_book'},
 {'Database': 'information_schema'},
 {'Database': 'lahmanshw1'},
 {'Database': 'mysql'},
 {'Database': 'performance_schema'},
 {'Database': 'sys'},
 {'Database': 'W4111_HW1'}]
```

ipython-SQL

In [10]:
```
%load_ext sql
```

In [11]:
```
#
# Remember to change your user ID and password.
#
%sql mysql+pymysql://root:WHQ21cd1c689742@localhost
```

In [12]:
```
%sql select * from db_book.student where ID=12345
```

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

Out[12]:

ID	name	dept_name	tot_cred
12345	Shankar	Comp. Sci.	32

SQLAlchemy

In [13]:
```
from sqlalchemy import create_engine
```

In [14]:
```
#
# Remember to change your user ID and password.
#
sql_url = "mysql+pymysql://root:WHQ21cd1c689742@localhost"
```

In [15]:
```
engine = create_engine(sql_url)
```

In [16]:
```
sql = "select * from db_book.student"
df = pandas.read_sql(sql, con=engine)
```

In [17]:
```
df
```

Out[17]:

	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 [18]:
```
!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 [19]:
```
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 [20]:
```
#
# Answer
#
%sql drop schema lahmanshw1
%sql create schema lahmanshw1
```

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

Out[20]:

```
[]
```

- *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 [21]:

```
def load_and_save_csv(data_dir:str, file_name:str, schema:str, table_name:str=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 o
        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=False)
```

In [22]:
```
for f in csv_files:
    load_and_save_csv(data_dir, f, "lahmanshw1")
    print("Saved file:", f)
```

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

In [23]:
```
#
# 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
```

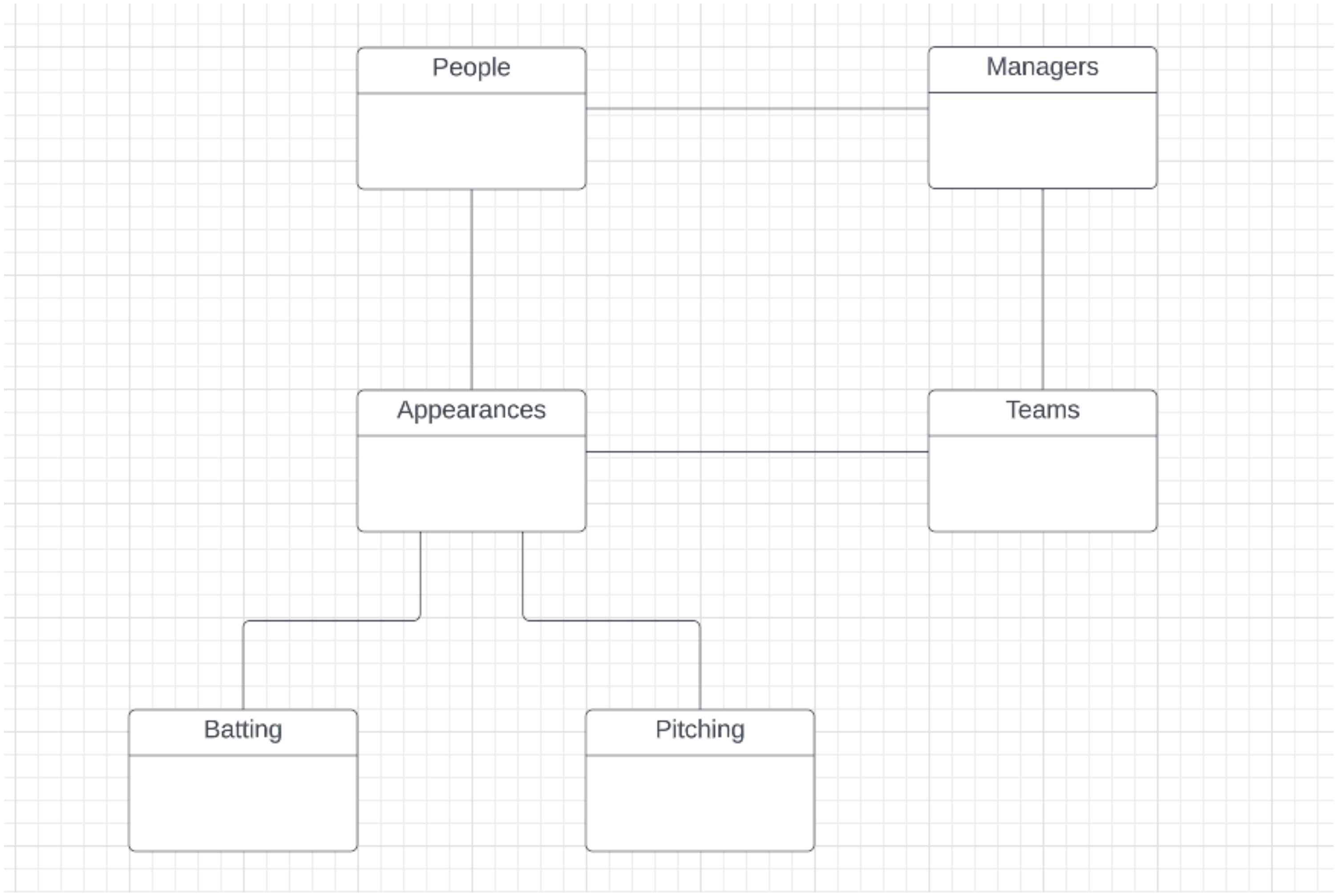
```
 * mysql+pymysql://root:***@localhost
6 rows affected.
 * mysql+pymysql://root:***@localhost
0 rows affected.
 * mysql+pymysql://root:***@localhost
1 rows affected.
 * mysql+pymysql://root:***@localhost
1 rows affected.
 * mysql+pymysql://root:***@localhost
1 rows affected.
 * mysql+pymysql://root:***@localhost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

In [24]:

```
#
# If you want to see the schema printed as text, just uncomment the following and run.
#
#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 [25]:
```
%%sql
alter table People modify playerID varchar(32);
alter table People modify birthCountry varchar(55);
alter table People modify birthState varchar(50);
alter table People modify birthCity varchar(60);
alter table People modify deathCountry varchar(55);
alter table People modify deathState varchar(50);
alter table People modify deathCity varchar(60);
alter table People modify nameFirst varchar(50);
alter table People modify nameLast varchar(50);
alter table People modify nameGiven varchar(50);
alter table People modify bats varchar(2);
alter table People modify throws varchar(2);
alter table People modify debut varchar(10);
alter table People modify finalGame varchar(10);
alter table People modify retroID varchar(32);
alter table People modify bbrefID varchar(32);
```

```
 * 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.
20370 rows affected.
20370 rows affected.
20370 rows affected.
```

Out[25]: `[]`

In [26]:

```
%%sql
alter table People modify birthYear int;
alter table People modify birthMonth int;
alter table People modify birthDay int;
alter table People modify deathYear int;
alter table People modify deathMonth int;
alter table People modify deathDay int;
alter table People modify weight int;
alter table People modify height int;
```

```
 * 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.
```

Out[26]: []

In [27]:
```
%%sql
alter table People add column dateOfBirth date;
alter table People add column dateOfDeath date;
```

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

Out[27]: []

In [28]:
```
%%sql
update People
set dateOfBirth=STR_TO_DATE(CONCAT(birthYear, '-', birthMonth, '-', birthDay), '%Y-%m-%d');

update People
set dateOfDeath=STR_TO_DATE(CONCAT(deathYear, '-', deathMonth, '-', deathDay), '%Y-%m-%d');

select * from People where playerID='aardsda01';
```

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

Out[28]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth	deathDay	deathCountr
aardsda01	1981	12	27	USA	CO	Denver	None	None	None	Non

In [29]:
```
%%sql
ALTER TABLE People
MODIFY COLUMN bats ENUM('L', 'R', 'B', 'T', 'S');

ALTER TABLE People
MODIFY COLUMN throws ENUM('L', 'R', 'B', 'T', 'S');
```

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

Out[29]:
```
[]
```

In [30]:
```
%%sql
ALTER TABLE People MODIFY COLUMN debut date;
ALTER TABLE People MODIFY COLUMN finalGame date;
```

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

Out[30]:
```
[]
```

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 [31]:
```
%%sql
alter table Managers modify column playerID varchar(32);
alter table Managers modify column teamID varchar(16);
alter table Managers modify column lgID varchar(16);
alter table Managers modify column yearID char(4);

update Managers
set plyrMgr =
CASE
    WHEN plyrMgr = 'Y' THEN TRUE
    WHEN plyrMgr = 'N' THEN FALSE
    ELSE plyrMgr
END;

alter table Managers modify column plyrMgr boolean;
```

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

Out[31]:
```
[]
```

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 [32]:
```
%%sql
alter table Appearances modify column yearID char(4);
alter table Appearances modify column playerID varchar(32);
alter table Appearances modify column teamID varchar(16);
alter table Appearances modify column lgID varchar(16);
```

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

Out[32]:
```
[]
```

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 [33]:
```
%%sql
alter table Batting modify column yearID char(4);
alter table Batting modify column playerID varchar(32);
alter table Batting modify column teamID varchar(16);
alter table Batting modify column lgID varchar(16);
```

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

Out[33]:
```
[]
```

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 [34]:

```
%%sql
alter table Pitching modify column yearID char(4);
alter table Pitching modify column playerID varchar(32);
alter table Pitching modify column teamID varchar(16);
alter table Pitching modify column lgID varchar(16);
```

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

Out[34]:

```
[]
```

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 [35]:
```
%%sql
alter table Teams modify column yearID char(4);
alter table Teams modify column teamID varchar(16);
alter table Teams modify column lgID varchar(16);
alter table Teams modify column franchID varchar(16);
alter table Teams modify column divID varchar(16);
```

```
update Teams
set DivWin =
CASE
    WHEN DivWin = 'Y' THEN TRUE
    WHEN DivWin = 'N' THEN FALSE
    ELSE DivWin
END;

update Teams
set WCWin =
CASE
    WHEN WCWin = 'Y' THEN TRUE
    WHEN WCWin = 'N' THEN FALSE
    ELSE WCWin
END;

update Teams
set LGWin =
CASE
    WHEN LGWin = 'Y' THEN TRUE
    WHEN LGWin = 'N' THEN FALSE
    ELSE LGWin
END;

update Teams
set WSWin =
CASE
    WHEN WSWin = 'Y' THEN TRUE
    WHEN WSWin = 'N' THEN FALSE
    ELSE WSWin
END;
alter table Teams modify column DivWin boolean;
alter table Teams modify column WCWin boolean;
alter table Teams modify column LGWin boolean;
alter table Teams modify column WSWin boolean;
```

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

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

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.*

For primary key we just need to check if all rows are unique and all rows have non null values.

In [36]:
```
%%sql
select yearID, teamID, playerID, count(*)
from Appearances
group by yearID, teamID, playerID
having count(*) > 1;

select *
from Appearances
where yearID is null or teamID is null or playerID is null;
```

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

Out[36]:

yearID	teamID	lgID	playerID	G_all	GS	G_batting	G_defense	G_p	G_c	G_1b	G_2b	G_3b	G_ss	G_lf	G_cf	G_rf	G_of	G_c

In [37]:

```
%%sql
select yearID, playerID, stint, count(*)
from Batting
group by yearID, playerID, stint
having count(*) > 1;

select *
from Batting
where yearID is null or playerID is null or stint is null;
```

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

Out[37]:

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

In [38]:

```
%%sql
select yearID, playerID, stint, count(*)
from Pitching
group by yearID, playerID, stint
having count(*) > 1;

select *
from Pitching
where yearID is null or playerID is null or stint is null;
```

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

Out[38]:

playerID	yearID	stint	teamID	lgID	W	L	G	GS	CG	SHO	SV	IPouts	H	ER	HR	BB	SO	BAOpp	ERA	IBB	WP	HBP

In [39]:
```
%%sql
select yearID, playerID, inseason, count(*)
from Managers
group by yearID, playerID, inseason
having count(*) > 1;

select *
from Managers
where yearID is null or playerID is null or inseason is null;
```

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

Out[39]:

playerID	yearID	teamID	lgID	inseason	G	W	L	rank	plyrMgr

In [40]:
```
%%sql
select yearID, teamID, count(*)
from Teams
group by yearID, teamID
having count(*) > 1;

select *
from Teams
where yearID is null or teamID is null;
```

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

Out[40]:

yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LGWin	WSWin	R	AB	H	2B	3B	HR	BB

In [41]:
```
%%sql
SELECT playerID, COUNT(*)
FROM People
GROUP BY playerID
HAVING COUNT(*) > 1;

select *
from People
where playerID is null;
```

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

Out[41]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth	deathDay	deathCountry

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

In [42]:
```
%sql  ALTER TABLE Appearances ADD PRIMARY KEY (yearID, teamID, playerID);
```

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

Out[42]:
```
[]
```

In [43]:
```
%sql  ALTER TABLE Batting ADD PRIMARY KEY (yearID, playerID, stint);
```

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

Out[43]:
```
[]
```

In [44]:
```
%sql ALTER TABLE Managers ADD PRIMARY KEY (yearID, playerID, inseason);
```

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

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

```
In [45]: %sql ALTER TABLE People ADD PRIMARY KEY (playerID);

 * mysql+pymysql://root:***@localhost
0 rows affected.
Out[45]: []
```

```
In [46]: %sql ALTER TABLE Pitching ADD PRIMARY KEY (yearID, playerID, stint);

 * mysql+pymysql://root:***@localhost
0 rows affected.
Out[46]: []
```

```
In [47]: %sql ALTER TABLE Teams ADD PRIMARY KEY (yearID, teamID);

 * mysql+pymysql://root:***@localhost
0 rows affected.
Out[47]: []
```

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

I used left join to check if the number of rows is equal with the original dataset

```
In [73]: %%sql
DELETE FROM Batting
WHERE playerID = 'thompfr01' AND yearID = '1875' AND teamID = 'WS6';
DELETE FROM Batting
WHERE playerID = 'smithbu01' AND yearID = '1911' AND teamID = 'WS1';
DELETE FROM Appearances
WHERE playerID = 'thompan01' AND yearID = '1875' AND teamID = 'WS6';

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

Out[73]:
```
[]
```

In [78]:
```
%%sql
select *
from Appearances
where (playerID) not in (select playerID from People);

select *
from Appearances
where (teamID, yearID) not in (select teamID, yearID from Teams);
```

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

Out[78]:

yearID	teamID	lgID	playerID	G_all	GS	G_batting	G_defense	G_p	G_c	G_1b	G_2b	G_3b	G_ss	G_lf	G_cf	G_rf	G_of	G_c

In [79]:
```
%%sql
select *
from Batting
where (playerID, teamID, yearID) not in (select playerID, teamID, yearID from Appearances);
```

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

Out[79]:

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

In [80]:
```
%%sql
select *
from Pitching
where (playerID, teamID, yearID) not in (select playerID, teamID, yearID from Appearances);
```

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

Out[80]:

playerID	yearID	stint	teamID	lgID	W	L	G	GS	CG	SHO	SV	IPouts	H	ER	HR	BB	SO	BAOpp	ERA	IBB	WP	HBP

In [81]:
```
%%sql
select * from Managers
where (playerID) not in (select playerID from People);

select * from Managers
where (teamID, yearID) not in (select teamID, yearID from Teams);
```

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

Out[81]:

playerID	yearID	teamID	lgID	inseason	G	W	L	rank	plyrMgr

In [61]:
```
%sql SET FOREIGN_KEY_CHECKS=0;
```

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

Out[61]:
```
[]
```

- *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 [82]:
```
%sql ALTER TABLE Appearances ADD FOREIGN KEY (playerID) REFERENCES People(playerID);
%sql ALTER TABLE Appearances ADD FOREIGN KEY (yearID, teamID) REFERENCES Teams(yearID, teamID);
```

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

Out[82]:
```
[]
```

In [83]:
```
%%sql
ALTER TABLE Batting ADD FOREIGN KEY (playerID, yearID, teamID)
    REFERENCES Appearances(playerID, yearID, teamID);
```

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

Out[83]:

```
[]
```

In [84]:

```
%sql ALTER TABLE Managers ADD FOREIGN KEY (playerID) REFERENCES People(playerID);
%sql ALTER TABLE Managers ADD FOREIGN KEY (yearID, teamID) REFERENCES Teams(yearID, teamID);
```

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

Out[84]:

```
[]
```

In [85]:

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

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

Out[85]:

```
[]
```

SQL Queries

On-Base Percentage and Slugging

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

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

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

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

In [66]:
```
%%sql
SELECT Batting.playerID, nameLast, nameFirst, H, AB, G,
       (((H - 2B - 3B - HR) + (2 * 2B) + (3 * 3B) + (4 * HR))/nullif(AB, 0))onBasePercentage
FROM Batting
join People
where People.playerID = Batting.playerID and Batting.playerID = 'willite01'
```

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

Out[66]:

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 [67]:

```
### 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 [101…

```
%%sql
select p.playerID, p.nameLast, p.nameFirst,
       COALESCE(career_player_games, 0)career_player_games,
       COALESCE(career_manager_games, 0)career_manager_games
from People as p
left join
    (select playerID, sum(G) as career_manager_games
         from Managers group by playerID) as manager_table
    on manager_table.playerID = p.playerID
left join
    (select playerID, sum(G_all) as  career_player_games
         from Appearances group by playerID) as player_table
    on player_table.playerID = p.playerID
where p.birthState = 'CA' and p.nameLast = 'Williams'
```

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

Out[101]:

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 [ ]: