COMS W4111: Introduction to Databases
Fall 2023, Section 2

Homework 1, Part 2: Non-Programming

Introduction

This notebook contains HW1 Part 2 Nonprogramming. **Only those 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.

Add Student Information

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

name = "Liang Zhao"
uni = "lz2871"
track = "nonprograming Track"

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

Liang Zhao
lz2871
nonprograming Track

Setup

SQL Magic

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

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

You may need to change the password below.

In [ ]:
```
%sql mysql+pymysql://root:Jobapplication2022.@localhost
```

In [ ]:
```
import os
os.getcwd()
```

Out[ ]:
```
'/Users/liangzhao/Downloads/Intro-to-Databases-F23/W4111_Fall_Oct/HomeworkA
ssignments/HW1-Part2-Nonprogramming'
```

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

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

Out[ ]:

1
1

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

Python Libraries

In [ ]:
```
import os

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

You may need to change the password below.

In [ ]:
```
engine = create_engine("mysql+pymysql://root:Jobapplication2022.@localhost")
```

Load Data

We're going to load data into a new database called `lahmans_hw1`. The data is stored as CSV files in the `data/` directory.

In [ ]:
```
%sql DROP SCHEMA IF EXISTS lahmans_hw1
%sql CREATE SCHEMA lahmans_hw1
```

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

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

In [ ]:
```
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 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", ir
```

In [ ]:
```
os.getcwd()
```

Out[ ]:
```
'/Users/liangzhao/Downloads/Intro-to-Databases-F23/W4111_Fall_Oct/HomeworkA
ssignments/HW1-Part2-Nonprogramming'
```

In [ ]:
```
data_dir = "data"
csv_files = [
    "People.csv",
    "Appearances.csv",
    "Batting.csv",
    "Pitching.csv",
    "Teams.csv",
    "Managers.csv",
]
schema = "lahmans_hw1"
```

```
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. In this assignment, we'll fix these issues.

In [ ]:
```
%sql USE lahmans_hw1
```

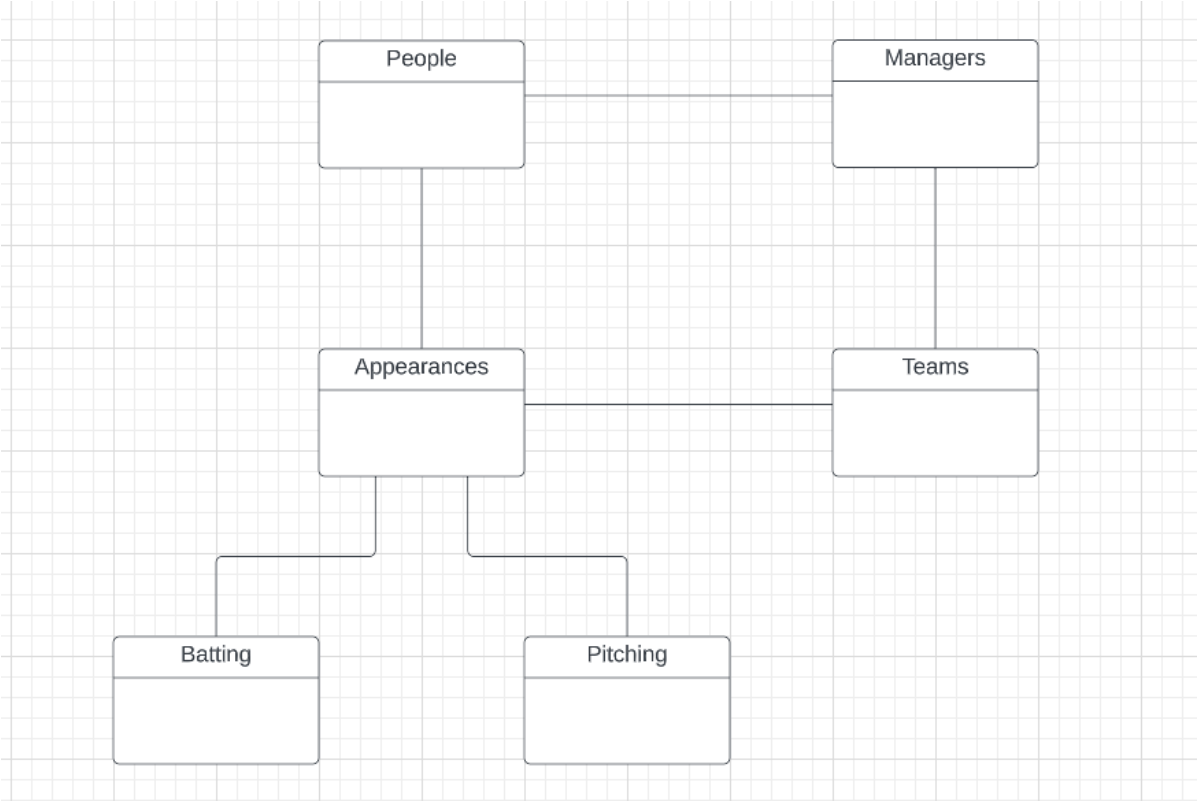
```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

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

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

In [ ]:
```
Image("./lahmans-conceptual.png")
```

Out[ ]:



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
);
```

You are to complete the following tasks:

1. Convert `playerID`, `retroID`, and `bbrefID` to **minimally sized** `CHAR`. 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).
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 [ ]:
```
%%sql

ALTER TABLE People

MODIFY COLUMN playerID CHAR(10),
MODIFY COLUMN birthYear INT,
MODIFY COLUMN birthMonth INT,
MODIFY COLUMN birthDay INT,

MODIFY COLUMN deathYear INT,
MODIFY COLUMN deathMonth INT,
MODIFY COLUMN deathDay INT,

MODIFY COLUMN weight  INT null,
MODIFY COLUMN height  INT null,

Modify column bats ENUM('R','L','B'),
Modify column throws ENUM('R','L','S'),

MODIFY COLUMN retroID CHAR(8),
MODIFY COLUMN bbrefID CHAR(10), # Convert PlayerID, retroID, bbrefID

MODIFY COLUMN    debut        DATE   null,
MODIFY COLUMN    finalGame    DATE   null
```

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

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

In [ ]:
```
%%sql
#Create two new columns, `dateOfBirth` and `dateOfDeath` of type `DATE`.
ALTER TABLE People
ADD COLUMN dateOfBirth DATE,
ADD COLUMN dateOfDeath DATE
```

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

Out[ ]: []

In [ ]:
```
%%sql
#### Update dateOfBirth based on birthYear, birthMonth, and birthDay

UPDATE People
SET dateOfBirth = str_to_date(CONCAT(birthYear,'-',birthMonth,'-',birthDay),
```

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

Out[ ]: []

In [ ]:
```
%%sql
select
    playerid,
    birthDay,
    birthMonth,
    birthYear,
    DATE(CONCAT_WS('-', birthYear, birthMonth, birthDay))
from
    People
limit 10
```

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

Out[ ]:

playerid	birthDay	birthMonth	birthYear	DATE(CONCAT_WS('-', birthYear, birthMonth, birthDay))
aardsda01	27	12	1981	1981-12-27
aaronha01	5	2	1934	1934-02-05
aaronto01	5	8	1939	1939-08-05
aasedo01	8	9	1954	1954-09-08
abadan01	25	8	1972	1972-08-25
abadfe01	17	12	1985	1985-12-17
abadijo01	4	11	1850	1850-11-04
abbated01	15	4	1877	1877-04-15
abbeybe01	11	11	1869	1869-11-11
abbeych01	14	10	1866	1866-10-14

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

```
Appearances.csv Managers.csv    Pitching.csv
Batting.csv     People.csv      Teams.csv
```

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
);
```

You are to complete the following tasks:

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 [ ]:
```
%%sql
ALTER TABLE Managers
MODIFY COLUMN playerID CHAR(10),
MODIFY COLUMN yearID CHAR(4),
MODIFY COLUMN teamID CHAR(4),
MODIFY COLUMN lgID CHAR(3)
```

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

Out[ ]: []

In [ ]:
```
%%sql
ALTER TABLE Managers
ADD COLUMN plyrMgr_temp BOOLEAN;
```

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

Out[ ]: []

In [ ]:

In [ ]:
```
%%sql
# Step 2: Update the temporary column based on existing values
UPDATE Managers
SET plyrMgr_temp = CASE WHEN plyrMgr = 'Y' THEN TRUE ELSE FALSE END;
```

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

Out[ ]: []

In [ ]:
```
%%sql
#  Step 3: Drop the existing plyrMgr column
ALTER TABLE Managers
DROP COLUMN plyrMgr;
```

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

Out[ ]: []

In [ ]:
```
%%sql
# Step 4: Rename the temporary column to plyrMgr
ALTER TABLE Managers
CHANGE COLUMN plyrMgr_temp plyrMgr BOOLEAN;
```

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

Out[ ]: []

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

- `yearID` is char type, giving more flexibility and allowing maximium of 4 digits. However, `year` type is a fixed-length data type that can only store 4-character strings representing eyars in the format `YYYY`. Therefore, if I want to year in a foramt of 3 digits, `Year` type will not have such format.

Therefore, it cannot store years with fewer or more than four digit

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
);
```

You are to complete the following tasks:

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 [ ]:
```
%%sql
ALTER TABLE Appearances
MODIFY COLUMN yearID CHAR(4),
MODIFY COLUMN playerID CHAR(10),
MODIFY COLUMN teamID CHAR(4),
MODIFY COLUMN lgID CHAR(3)
```

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

Out[ ]: []

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
);
```

You are to complete the following tasks:

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 [ ]:
```
%%sql


ALTER TABLE Batting
MODIFY COLUMN yearID CHAR(4),
MODIFY COLUMN playerID CHAR(10),
MODIFY COLUMN teamID CHAR(4),
MODIFY COLUMN lgID CHAR(3)
```

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

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

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
);
```

You are to complete the following tasks:

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 [ ]:
```
%%sql
ALTER TABLE Pitching
MODIFY COLUMN yearID CHAR(4),
MODIFY COLUMN playerID CHAR(10),
MODIFY COLUMN teamID CHAR(4),
MODIFY COLUMN lgID CHAR(3)
```

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

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

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
);
```

You are to complete the following tasks:

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 [ ]:
```
%%sql

ALTER TABLE Teams
MODIFY COLUMN yearID CHAR(4),
MODIFY COLUMN franchID CHAR(4),
MODIFY COLUMN divID CHAR(10),
MODIFY COLUMN teamID CHAR(4),
MODIFY COLUMN lgID CHAR(3)
```

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

Out[ ]: []

Primary Keys

Now we need to add primary keys to our tables. In the following cells, write and execute SQL statements that show the column/combination of columns that is a valid primary key for each of the 6 tables.

Recall the properties of primary keys and think about how you could represent them using queries. Note that you aren't simply selecting columns. You need to show **why** they can be a primary key.

In [ ]:
```
%%sql
USE lahmans_hw1
```

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

Out[ ]: []

In [ ]:
```
%%sql
#1. For People
select count(*) as row_counntall,
       count(distinct playerID) as id_count
from people
```

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

Out[ ]:

row_counntall	id_count
20370	20370

In [ ]:
```
%%sql
# `playerID, yearID, inseason` is the primary key
## 2. For Managers
```

```
select count(*) as countall,
       count(distinct concat(playerID, yearID, teamID, inseason)) as player_
from managers
```

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

Out[ ]:

countall	player_year_inseason_ID_count
3684	3684

In [ ]:
```
%%sql
#3. Primary candidate key for Appearances
select playerid, teamID, yearID, count(*) as count from appearances
group by playerID, teamID, yearID
order by count desc
limit 10;



select count(*) as countall,
       count(distinct concat(playerid, teamID, yearID)) as player_year_teamI
from Appearances;
```

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

Out[ ]:

countall	player_year_teamID_count
110422	110422

In [ ]:
```
%%sql
# 4. Batting primary key
select count(*) as countall,
count(distinct concat(playerID, yearID,stint)) as player_year_stint_ID
from batting
```

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

Out[ ]:

countall	player_year_stint_ID
110493	110493

In [ ]:
```
%%sql
# 5. Pitching Primary Key
select
       count(*) as countall,
       count(distinct concat(playerID, yearID ,stint)) as player_year_stint_
from pitching
```

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

Out[ ]:

countall	player_year_stint_ID
49430	49430

In [ ]:
```
%%sql
#6. primary key for team:
select
       count(*) as countall,
       count(distinct concat(yearID ,teamID)) as player_year_team_ID
from teams;
```

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

Out[ ]:

countall	player_year_team_ID
2985	2985

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

In [ ]:
```
%%sql
#1. People
alter table People
add primary key (playerID)
```

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

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

In [ ]:
```
%%sql
select
    playerID, yearID, inseason, count(*) as count
from Managers
group by  playerID, yearID, inseason
order by count desc
limit 10;
```

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

Out[ ]:

playerID	yearID	inseason	count
wrighha01	1871	1	1
woodji01	1871	1	1
paborch01	1871	1	1
lennobi01	1871	1	1
deaneha01	1871	2	1
fergubo01	1871	1	1
mcbridi01	1871	1	1
hastisc01	1871	1	1
pikeli01	1871	1	1
cravebi01	1871	2	1

In [ ]:
```
%%sql
# 2. Managers
```

```
alter table Managers
add primary key (playerID, yearID, teamID, inseason)
```

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

Out[ ]: []

In [ ]:
```
%%sql
# 3. Appearances
alter table Appearances
add primary key (playerID, yearID, teamID)
```

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

Out[ ]: []

In [ ]:
```
%%sql
# 4. Batting

alter table Batting
add primary key (playerID, yearID , teamID, stint)
```

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

Out[ ]: []

In [ ]:
```
%%sql
# 5. Pitching
alter table Pitching
add primary key (playerID, yearID , teamID, stint)
```

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

Out[ ]: []

In [ ]:
```
%%sql
# 6. Teams
alter table Teams
add primary key (yearID, teamID);
```

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

Out[ ]: []

Foreign Keys

Let's add foreign keys. **The conceptual ER diagram above should indicate to you which tables are related by foreign keys.** In the following cells, write and execute SQL statements that show the column/combination of columns that is a valid foreign key for each of the 6 relationships.

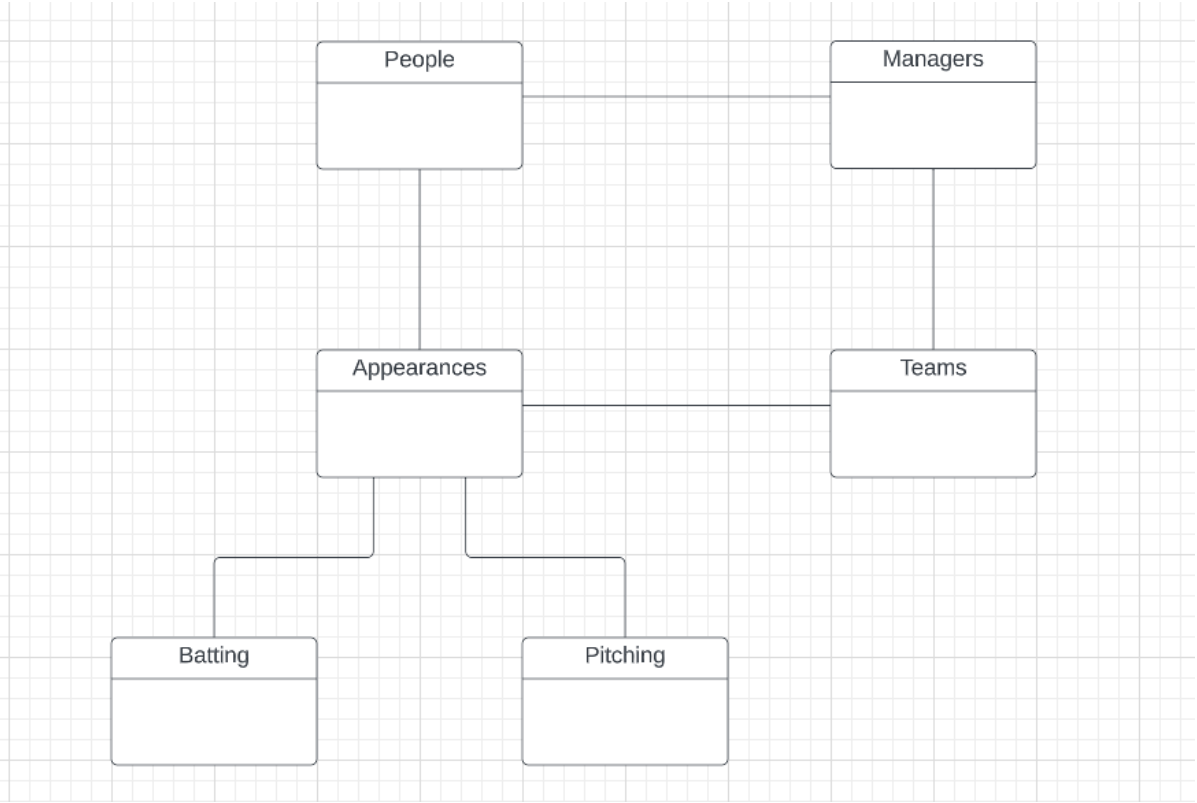
Recall the properties of foreign keys and think about how you could represent them using queries. Note that you aren't simply selecting columns. You need to show **why** they can be a foreign key.

A foregin key is a relational datase constraint that make the likn between 2 tables. It ensures that the values in a specific column in 1 table correspond to the values in a priamry key in another table.

A foreign key establishes a relationship between tables, allowing one table to refer to the records in another table.

In [ ]:
```
Image("./lahmans-conceptual.png")
```

Out[ ]:



In [ ]:
```
%%sql
# 1. Appearances primary key  check with foregin key
select * from appearances
where playerID not in (
    select playerID from people
)
```

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

Out[ ]:

yearID	teamID	lgID	playerID	G_all	GS	G_batting	G_defense	G_p	G_c	G_1b	G_2b	G

In [ ]:
```
%%sql
select * from teams limit 1;
```

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

Out[ ]:

yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin	W
1871	None	BS1	BNA	None	3	31	None	20	10	None	None	N	

In [ ]:

```
%%sql
#2. Primary candidate key check for Appearances
select playerID, teamID, yearID, count(*) as count from appearances
group by playerID, teamID, yearID
order by count desc
limit 10;
```

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

Out[ ]:

playerID	teamID	yearID	count
aardsda01	ATL	2015	1
aardsda01	BOS	2008	1
aardsda01	CHA	2007	1
aardsda01	CHN	2006	1
aardsda01	NYA	2012	1
aardsda01	NYN	2013	1
aardsda01	SEA	2009	1
aardsda01	SEA	2010	1
aardsda01	SFN	2004	1
aaronha01	ATL	1966	1

In [ ]:

```
%%sql
## 3. Managers primary key check with foregin key
select * from Managers where playerID not in (
    select playerID from People)
```

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

Out[ ]:

playerID	yearID	teamID	lgID	inseason	G	W	L	rank	plyrMgr

In [ ]:

```
%%sql
# 4. Team primary key  check with foregin key
select * from Teams
where TeamID not in (
    select TeamID from Managers)
and yearID not in (
    select yearID from Managers
)
```

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

Out[ ]:

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

In [ ]:
```
%%sql
#another way to check the foregin and primary key for Appearance (pitching a
#2. Primary candidate key check for Appearances
select playerID, teamID, yearID,  stint, count(*) as count from pitching
group by playerID, teamID, yearID,stint
order by count desc
limit 10;
%%sql
#2. Primary candidate key check for Appearances
select playerID, teamID, yearID,  stint, count(*) as count from batting
group by playerID, teamID, yearID,stint
order by count desc
limit 10;
```

```
 * mysql+pymysql://root:***@localhost
10 rows affected.
(pymysql.err.ProgrammingError) (1064, "You have an error in your SQL synta
x; check the manual that corresponds to your MySQL server version for the r
ight syntax to use near '%%sql\n#2. Primary candidate key check for Appeara
nces\nselect playerID, teamID, y' at line 1")
[SQL: %%%%sql
#2. Primary candidate key check for Appearances
select playerID, teamID, yearID,  stint, count(*) as count from batting
group by playerID, teamID, yearID,stint
order by count desc
limit 10;]
(Background on this error at: https://sqlalche.me/e/14/f405)
```

In [ ]:
```
%%sql
# 5. Batting primary key  check with foregin key
select * from Batting where playerID not in (
    select playerID from Appearances)
and yearID not in (
    select yearID from Appearances)
and teamID not in (
    select teamID from Appearances
)
```

```
 * 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

In [ ]:
```
%%sql
#6. Pitching primary key check with foregin key
select * from Pitching where playerID not in (
    select playerID from Appearances)
and yearID not in (
```

```
    select yearID from Appearances)
and teamID not in (
    select teamID from Appearances
)
```

```
 * 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

In [ ]:
```
%%sql
#check team primary key with managers
select * from Managers where teamID not in (
    select teamID from Teams)
```

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

Out[ ]:

playerID	yearID	teamID	lgID	inseason	G	W	L	rank	plyrMgr

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

In [ ]:
```
%%sql
USE lahmans_hw1;
```

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

Out[ ]: []

In [ ]:
```
%%sql
# 1. Add foregin key
# Managers & People : `PlayerID``
ALTER TABLE Managers
add constraint Managers__fk
foreign key (playerID) references People (playerID);


# 2. Add Foreign key
# Manger and Teams: `yearID`,`teamID`
alter table Managers
    add constraint Managers_Teams__fk
        foreign key (`yearID`,`teamID`) references Teams (`yearID`,`teamID`)

# 3. Add foregin key
#  appearances & team : `playerID`

ALTER TABLE Appearances
add constraint Appearances_Teams_fk
foreign key (`yearID`,`teamID`) references Teams (`yearID`,`teamID`);

#4. Add foregin key
#   People &  Appearances : `PlayerID`
ALTER TABLE Appearances
add constraint Appearances__fk
foreign key (playerID) references People (playerID);
```

```
#5. Add foregin key !!!!!!!! ORDER MATTERS (you have to follow the )
#   Pitching &  Appearances : `playerID`,`yearID`,`teamID`
alter table Pitching
    add constraint Pitching_Appearance__fk
        foreign key (`playerID`,`yearID`,`teamID`) references Appearances (`

#6. Add foregin key !!!!!!!! ORDER MATTERS (you have to follow the )
#   Batting &  Appearances : `playerID`,`yearID`,`teamID`
alter table Batting
    add constraint Batting_Appearance__fk
        foreign key (`playerID`,`yearID`,`teamID`) references Appearances (`
```

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

Out[ ]: []

''' Notes:

1. creating 2 sepreate FK

(yearID)-> teams(yearID) (teamID)->teams(teamID)

a more specific constraint on the relationship (a valid tuple will need to have both the same yearID and the same teamID as the table it is related to).

(yearID,teamID)->teams(yearID,teamID)

2. `Order matters` : NOT valid if ((yearID,teamID)->teams(teamID,yearID)) this wont work.

3. A foregin key can only reference primary key of another table. For exp:

alter table Pitching add constraint Pitching_Appearance__fk foreign key (`playerID` , `yearID` , `teamID`) references Appearances (`playerID` , `yearID` , `teamID`); solution: you either ceate `new index columns` for reference or make sure the `table you are refering to have all the necessary columns` '''

SQL Queries

On-Base Percentage and Slugging

The formula for `onBasePercentage` is

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

Note that `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 [ ]:
```
%sql USE lahmans_hw1
```

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

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

In [ ]:
```
%%sql

SELECT
    b.playerID,
    p.nameFirst,
    p.nameLast,
    b.yearID,
    b.stint,
    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
FROM
    Batting AS b
JOIN
    People AS p on p.playerID = b.playerID

order by
    onBasePercentage DESC,
    p.nameLast ASC


LIMIT 10
```

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

Out[ ]:

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`. Similarly, a person in `People` was a manager if their `playerID` appears in `Managers`. Note that 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 the below codes, the desried out is names, playerID, the total sum of playergames and managergames order by their sums.
- First I would retrive the manger and player data by using left join those 3 tables on the playerID column. As a result, this combines from both tables based on matching "playerID" values and allows me to perform sum on manager and palyers.

- Next, we use groupby to calculate all the statistics and ensure 1 record for each individual. The order by clause sortes the results based on the sum of palyer games and manager games in descending order (careerPlayerGames + careerManagerGames).

--- In essence, these two left joins allow you to create a combined result set that includes data from all three tables, `People`, `Appearances`, and `Managers`, for individuals who may have been both players and managers. `The left joins ensure that you retrieve all records from the People table (which represents individuals) and supplement that data with information from the Appearances and Managers tables where applicable`. If there is no corresponding data in the latter two tables for a given playerID, the relevant columns will contain NULL values in the result.

In [ ]:
```
%%sql

SELECT
    p.playerID,
    p.nameFirst,
    p.nameLast,
    ifnull(SUM(a.G_all),0) as careerPlayerGames,
    ifnull(SUM(m.G),0) as careerManagerGames
FROM
    People AS p
LEFT JOIN
    Appearances AS a
ON
    p.playerID = a.playerID
LEFT JOIN
    Managers AS m
ON
    p.playerID = m.playerID

GROUP BY
    p.playerID,
    p.nameFirst,
    p.nameLast
ORDER BY
    careerPlayerGames + careerManagerGames DESC
limit 10;
```

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

Out[ ]:

playerID	nameFirst	nameLast	careerPlayerGames	careerManagerGames
torrejo01	Joe	Torre	64061	77814
mcgrajo01	John	McGraw	39780	85842
mackco01	Connie	Mack	38372	85305
bakerdu01	Dusty	Baker	48936	70376
dykesji01	Jimmy	Dykes	50226	65164
ansonca01	Cap	Anson	53004	61776
durocle01	Leo	Durocher	42562	67302
pinielo01	Lou	Piniella	40181	63648
clarkfr01	Fred	Clarke	42674	59409
robinfr02	Frank	Robinson	44928	49324

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 [ ]:

```
%%sql
SELECT
    p.playerID,
    p.nameFirst,
    p.nameLast,
    IFNULL(SUM(a.G_all), 0) AS careerPlayerGames,
    IFNULL(SUM(m.G), 0) AS careerManagerGames
FROM
    People AS p
LEFT JOIN
    Appearances AS a
ON
    p.playerID = a.playerID
LEFT JOIN
    Managers AS m
ON
    p.playerID = m.playerID
GROUP BY
    p.playerID,
    p.nameFirst,
    p.nameLast
HAVING
    careerManagerGames = 0
ORDER BY
    careerPlayerGames + careerManagerGames desc
limit 10;
```

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

Out[ ]:

playerID	nameFirst	nameLast	careerPlayerGames	careerManagerGames
yastrca01	Carl	Yastrzemski	3308	0
aaronha01	Hank	Aaron	3298	0
henderi01	Rickey	Henderson	3081	0
murraed02	Eddie	Murray	3026	0
musiast01	Stan	Musial	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