

COMS W4111-002, V02 (Spring 2022)

Introduction to Databases

Homework 2: Programming and Non-Programming

Due Wednesday, February 23, 2022 at 11:59 PM

Introduction

Overview

This notebook has **2 sections that you must complete**:

1. Written questions testing knowledge of concepts. Answering these questions may require reviewing lecture slides, slides associated with the textbook, and/or online material. Both tracks complete this section.
2. Practical problems involving data modeling, relational algebra and SQL. Both tracks complete this section.

We will separately release the track-specific Programming and Non-Programming parts of HW2.

Submission

You will **submit 2 files** for this assignment.

1. Submit a zip file titled `<your_uni>_hw2_all.zip` to **HW2 All - Zip** on Gradescope.
 - Replace `<your_uni>` with your uni. My submission would be titled `dff9_hw2_all.zip`.
 - The zipped directory you submit should contain the following files:
 - `<your_uni>_hw2_all.ipynb`
 - `Appearances.csv`
 - `Batting.csv`
 - `People.csv`
 - Any image files you choose to embed in your notebook.

- All of these files, except the images you may embed in your notebook, are included in `s22_w4111_hw2_all.zip`, which you downloaded from Courseworks. You will have to rename the notebook file you downloaded to `<your_uni>_hw2_all.ipynb`, as discussed above.

2. Submit a PDF file titled `<your_uni>_hw2_all.pdf` to **HW2 All - PDF** on Gradescope.

- This should be a PDF of your completed HW2 All Python notebook.
- **Tag pages for each problem.** Per course policy, any untagged submission will receive an automatic 0.
- Double check your submission on Gradescope to ensure that the PDF conversion worked and that your pages are appropriately tagged.

Collaboration and Information

- Answering some of the questions may require independent research to find information. We encourage you to try troubleshooting problems independently before reaching out for help.
- You may use any information you get in TA or Prof. Ferguson's office hours, from lectures or from recitations. This includes slides related to the recommended textbook.
- You may use information that you find on the web.
- You are NOT allowed to collaborate with other students outside of office hours.

Written Questions

Question 1: NULL

Briefly explain Codd's 3rd Rule.

- What are some interpretations of a NULL value?
- An alternative to using NULL is some other value for indicating missing data, e.g. using -1 for the value of a weight column. Explain the benefits of NULL relative to other approaches.

Answer:

Some interpretation of a NULL value could be 1. Data is missing. 2. Data is unknown. 3. Data is not applicable

NULL is simple, clear, and a universally understood expression. While other indicators will cause confusion since the specific choice people choose to represent NULL is varying. Use other indicators will generally need more work and cause problems much more easily.

Question 2: Keys

Briefly explain the following concepts:

- Primary Key
- Candidate Key
- Super Key
- Alternate Key
- Composite Key
- Unique Key
- Foreign Key

Answer:

A primary key is a specific choice of a minimal set of attributes that uniquely specify a tuple in a relation. A candidate key is the set of all those attributes that can be selected as a primary key. A super key is used to identify an attribute or set of attributes uniquely within the relation. Super key is a generalized superset of candidate keys. Alternate keys are all those attributes that can be selected as the primary key but not selected as primary keys. A composite key is a set of two or more attributes that are together selected as a composite key. A unique key is a candidate key that is not the primary key of the relation. A foreign key is the primary key of one table that is linked with the field of another table.

Question 3: Algebra

Briefly explain what it means for the relational algebra to be *closed* under the operations in the algebra. What is an important benefit?

Answer:

For relational algebra, because it takes an instance of relations and does operations that work on one or more relations to describe another relation without altering the original relation, it could leave both the operands and the outputs as relations. This is called "the relational algebra to be closed under the operations in the algebra". A benefit is that the output from one operation can turn into the input to another operation, which allows expressions to be nested in the relational algebra.

Question 4: Equivalent Queries

Briefly explain the concept of equivalent queries. Use the concept to explain how it is possible to derive the JOIN operation from other operations (SELECT, PROJECT).

Answer:

Equivalent query is the concept that there is more than one way to write a query in relational algebra. The two queries are not identical, but they are equivalent in the sense that they give the same result on any database.

For the case of JOIN operation and SELECT, PROJECT operations, it the case that although we use different functions, but we can still give the same result as long as working on the same database.

Question 5: More General Attribute Types

The relational model places restrictions on attributes. Many data scenarios have more complex types of attributes. **Briefly** explain the following types of attributes:

- Simple attribute
- Composite attribute
- Derived attribute
- Single-value attribute
- Multi-value attribute

Answer:

Simple attributes are those attributes which can not be divided further, such as age. Composite attributes are those attributes which are composed of many other simple attributes, such as student, which could have attributes like age, class. Derived attributes are those attributes which can be derived from other attributes, such as age, which can be derived from student. Single valued attributes are those attributes which can take only one value for a given entity from an entity set, such as age. Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.

Practical Problems

Setup

- Modify the cells below to setup your environment.
- The change should just be setting the DB user ID and password, replacing my user ID and password with yours for MySQL.

```
In [1]: database_user_id = "root"
        database_pwd = "Edy990127"
```

```
In [2]: database_url = "mysql+pymysql://" + \
        database_user_id + ":" + database_pwd + "@localhost"
        database_url
```

```
Out[2]: 'mysql+pymysql://root:Edy990127@localhost'
```

```
In [3]: %reload_ext sql
```

```
In [4]: %sql $database_url
```

```
Out[4]: 'Connected: root@None'
```

```
In [5]: from sqlalchemy import create_engine
```

```
In [6]: sqla_engine = create_engine(database_url)
```

```
In [7]: #  
# We are going to create a schema and some tables for the HW.  
#  
%sql create schema if not exists S22_W4111_HW2  
%sql select 1;  
  
* mysql+pymysql://root:***@localhost  
1 rows affected.  
* mysql+pymysql://root:***@localhost  
1 rows affected.
```

```
Out[7]: 1  
        1
```

Question 6: Manipulating String and Types

Setup

- Run the following code cells.
- These cells create a table `people_info` and loads the table with a bunch of input strings.

```
In [8]: input_string = [  
    "Towny,Cavet,tcavet0@blinklist.com,1/9/1971,+62 (340) 387-5141",  
    "Port,Gaylor,pgaylor1@blogger.com,3/15/1939,+86 (517) 758-9970",  
    "Georgetta,Haddon,ghaddon2@symantec.com,9/19/1997,+81 (356) 753-5556",  
    "Wylma,Lanney,wlanney3@list-manage.com,2/21/2018,+385 (853) 541-7347",  
    "Mignonne,Georgeson,mgeorgeson4@123-reg.co.uk,8/7/1991,+63 (834) 397-5285",  
    "Cchaddie,Cossins,ccossins5@chronoengine.com,3/12/1911,+242 (313) 943-4080",  
    "Andie,Matyushonok,amatyushonok6@ask.com,4/24/1907,+380 (410) 464-9093",  
    "Skippie,Zuenelli,szuenelli7@merriam-webster.com,3/22/2014,+7 (279) 484-2088",  
    "Averyl,Barajas,abarajas8@fastcompany.com,6/19/1996,+232 (962) 344-7325",  
    "Olia,Habens,ohabens9@quantcast.com,2/28/1922,+98 (935) 300-9359"  
]
```

```
In [9]: import pandas
```

```
In [10]: df = pandas.DataFrame(input_string)
```

```
In [11]: df.to_sql(
         "people_info", con=sqla_engine, if_exists="replace", index=False,
         schema="S22_W4111_HW2")
```

```
In [12]: %sql use S22_W4111_HW2
         %sql select 1;
```

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

```
Out[12]:  1
         1
```

- Test loading the data.

```
In [13]: %sql select * from people_info
```

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

```
Out[13]:
```

	0
Towny,Cavet,tcavet0@blinklist.com,1/9/1971,+62 (340) 387-5141	
Port,Gaylor,pgaylor1@blogger.com,3/15/1939,+86 (517) 758-9970	
Georgetta,Haddon,ghaddon2@symantec.com,9/19/1997,+81 (356) 753-5556	
Wylma,Lanney,wlanney3@list-manage.com,2/21/2018,+385 (853) 541-7347	
Mignonne,Georgeson,mgeorgeson4@123-reg.co.uk,8/7/1991,+63 (834) 397-5285	
Cchaddie,Cossins,ccossins5@chronoengine.com,3/12/1911,+242 (313) 943-4080	
Andie,Matyushonok,amatyushonok6@ask.com,4/24/1907,+380 (410) 464-9093	
Skippie,Zuenelli,szuenelli7@merriam-webster.com,3/22/2014,+7 (279) 484-2088	
Averyl,Barajas,abarajas8@fastcompany.com,6/19/1996,+232 (962) 344-7325	
Olia,Habens,ohabens9@quantcast.com,2/28/1922,+98 (935) 300-9359	

- Can we describe what the table looks like?

```
In [14]: %sql describe people_info;
```

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

```
Out[14]:
```

Field	Type	Null	Key	Default	Extra
0	text	YES		None	

Tasks

- The created table has one column `0` . The values are strings with data separated by `,` . The fields in the string are (in order):
 - `first_name`
 - `last_name`
 - `email`
 - `date_of_birth`
 - `telephone_no` , which is of the form `+CC (XXX)-XXX-XXXX` where `CC` is the country code and the remainder is the number.
- You must process and cleanup the data using **ONLY** SQL statements. The cleanup tasks include:
 - Creating a new table `people_info_clean` with a structure that better represents the data, e.g. columns, column data types, etc.
 - Converting each string and its subfields into the rows of `people_info_clean` .
- You may use as many DDL and DML SQL statements as you need.
- Execute your statements in the cells below and show the output of the execution.
- The last two cells show show the data and schema for the information.

```
In [92]: %sql use S22_W4111_HW2;
%sql drop table if exists S22_W4111_HW2.people_info_clean;
```

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

Out[92]: []

```
In [93]: %%sql
create table if not exists S22_W4111_HW2.people_info_clean
(
first_name varchar(64) null,
last_name varchar(64) null,
email varchar(64) null,
date_of_birth text null,
telephone_no varchar(64) null,
a text null
);
```

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

Out[93]: []

```
In [94]: %%sql
        INSERT INTO people_info_clean(a)
        select `0` from people_info;

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

Out[94]: []

```
In [95]: %%sql
        UPDATE people_info_clean
        SET first_name = SUBSTR(a, 1, LOCATE(',', a) - 1),
        a = SUBSTR(a, LOCATE(',', a) + 1);
        UPDATE people_info_clean
        SET last_name = SUBSTR(a, 1, LOCATE(',', a) - 1),
        a = SUBSTR(a, LOCATE(',', a) + 1);
        UPDATE people_info_clean
        SET email = SUBSTR(a, 1, LOCATE(',', a) - 1),
        a = SUBSTR(a, LOCATE(',', a) + 1);
        UPDATE people_info_clean
        SET date_of_birth = SUBSTR(a, 1, LOCATE(',', a) - 1),
        a = SUBSTR(a, LOCATE(',', a) + 1);
        UPDATE people_info_clean
        SET telephone_no = a;

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

Out[95]: []

```
In [96]: %%sql
        ALTER table people_info_clean DROP a

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

Out[96]: []


```
In [97]: #
# Show the data.
#
%sql select * from people_info_clean

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

```
Out[97]:
```

first_name	last_name	email	date_of_birth	telephone_no
Towny	Cavet	tcavet0@blinklist.com	1/9/1971	+62 (340) 387-5141
Port	Gaylor	pgaylor1@blogger.com	3/15/1939	+86 (517) 758-9970
Georgetta	Haddon	ghaddon2@symantec.com	9/19/1997	+81 (356) 753-5556
Wylma	Lanney	wlanney3@list-manage.com	2/21/2018	+385 (853) 541-7347
Mignonne	Georgeson	mgeorgeson4@123-reg.co.uk	8/7/1991	+63 (834) 397-5285
Cchaddie	Cossins	ccossins5@chronoengine.com	3/12/1911	+242 (313) 943-4080
Andie	Matyushonok	amatyushonok6@ask.com	4/24/1907	+380 (410) 464-9093
Skippie	Zuenelli	szuenelli7@merriam-webster.com	3/22/2014	+7 (279) 484-2088
Averyl	Barajas	abarajas8@fastcompany.com	6/19/1996	+232 (962) 344-7325
Olia	Habens	ohabens9@quantcast.com	2/28/1922	+98 (935) 300-9359

```
In [98]: #
# Show the schema (architecture and structure).
#
%sql describe people_info_clean;

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

```
Out[98]:
```

	Field	Type	Null	Key	Default	Extra
	first_name	varchar(64)	YES		None	
	last_name	varchar(64)	YES		None	
	email	varchar(64)	YES		None	
	date_of_birth	text	YES		None	
	telephone_no	varchar(64)	YES		None	

Question 7: Intermediate SQL and Data Processing

Task 1: Load Data

- Continue to use the schema you created - S22_W4111_HW2 .
- There are three files in the homework folder:
 - People.csv
 - Appearances.csv

- Batting.csv
- Use one of the approaches we have previously used directly in notebooks to load the CSV files into the schema above.
 - You **may not** use external tools like DataGrip.
 - Some examples of techniques are in HW 1 and in the Pandas examples.
- Put your code in the cells provided below. The final cells, which you must run after loading the CSV files, simply display some information.

```
In [20]: # Your code
```

```
In [21]: data_path = "E:\Github\spring-2022-COMS4111\s22_w4111_hw2_all\s22_w4111_hw2_all"
```

```
In [22]: import os.path
```

```
In [23]: people_file_path = os.path.join(data_path, 'People.csv')
people_file_path
```

```
Out[23]: 'E:\\Github\\spring-2022-COMS4111\\s22_w4111_hw2_all\\s22_w4111_hw2_all\\Peopl
e.csv'
```

```
In [24]: people_df = pandas.read_csv(people_file_path)
people_df.to_sql('people', con=sqlla_engine, schema='S22_W4111_HW2', index=False,
```

```
In [25]: Appearances_df = pandas.read_csv('E:\Github\spring-2022-COMS4111\s22_w4111_hw2_all\s22_w4111_hw2_all\Appearances.csv')
Appearances_df.to_sql('appearances', con=sqlla_engine, schema='S22_W4111_HW2', index=False,
```

```
In [26]: Batting_df = pandas.read_csv('E:\Github\spring-2022-COMS4111\s22_w4111_hw2_all\s22_w4111_hw2_all\Batting.csv')
Batting_df.to_sql('batting', con=sqlla_engine, schema='S22_W4111_HW2', index=False,
```

```
In [27]: # Some tests
```

In [28]: %sql select * from people limit 10;

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

Out[28]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathM
aardsda01	1981.0	12.0	27.0	USA	CO	Denver	None	
aaronha01	1934.0	2.0	5.0	USA	AL	Mobile	2021.0	
aaronto01	1939.0	8.0	5.0	USA	AL	Mobile	1984.0	
aasedo01	1954.0	9.0	8.0	USA	CA	Orange	None	
abadan01	1972.0	8.0	25.0	USA	FL	Palm Beach	None	
abadfe01	1985.0	12.0	17.0	D.R.	La Romana	La Romana	None	
abadijo01	1850.0	11.0	4.0	USA	PA	Philadelphia	1905.0	
abbated01	1877.0	4.0	15.0	USA	PA	Latrobe	1957.0	
abbeybe01	1869.0	11.0	11.0	USA	VT	Essex	1962.0	
abbeych01	1866.0	10.0	14.0	USA	NE	Falls City	1926.0	

In [29]: %sql select * from appearances limit 10;

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

Out[29]:

yearID	teamID	lgID	playerID	G_all	GS	G_batting	G_defense	G_p	G_c	G_1b	G_2b	G_3
1871	TRO	None	abercda01	1	1.0	1	1.0	0	0	0	0	
1871	RC1	None	addybo01	25	25.0	25	25.0	0	0	0	22	
1871	CL1	None	allisar01	29	29.0	29	29.0	0	0	0	2	
1871	WS3	None	allisdo01	27	27.0	27	27.0	0	27	0	0	
1871	RC1	None	ansonca01	25	25.0	25	25.0	0	5	1	2	2
1871	FW1	None	armstbo01	12	12.0	12	12.0	0	0	0	0	
1871	RC1	None	barkeal01	1	1.0	1	1.0	0	0	0	0	
1871	BS1	None	barnero01	31	31.0	31	31.0	0	0	0	16	
1871	FW1	None	barrebi01	1	1.0	1	1.0	0	1	0	0	
1871	BS1	None	barrofr01	18	17.0	18	18.0	0	0	0	1	

```
In [30]: %sql select * from batting limit 10;
```

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

Out[30]:

playerID	yearID	stint	teamID	lgID	G	AB	R	H	2B	3B	HR	RBI	SB	CS	BB	SO	
abercda01	1871	1	TRO	None	1	4	0	0	0	0	0	0.0	0.0	0.0	0	0.0	⌵
addybo01	1871	1	RC1	None	25	118	30	32	6	0	0	13.0	8.0	1.0	4	0.0	⌵
allisar01	1871	1	CL1	None	29	137	28	40	4	5	0	19.0	3.0	1.0	2	5.0	⌵
allisdo01	1871	1	WS3	None	27	133	28	44	10	2	2	27.0	1.0	1.0	0	2.0	⌵
ansonca01	1871	1	RC1	None	25	120	29	39	11	3	0	16.0	6.0	2.0	2	1.0	⌵
armstbo01	1871	1	FW1	None	12	49	9	11	2	1	0	5.0	0.0	1.0	0	1.0	⌵
barkeal01	1871	1	RC1	None	1	4	0	1	0	0	0	2.0	0.0	0.0	1	0.0	⌵
barnero01	1871	1	BS1	None	31	157	66	63	10	9	0	34.0	11.0	6.0	13	1.0	⌵
barrebi01	1871	1	FW1	None	1	5	1	1	1	0	0	1.0	0.0	0.0	0	0.0	⌵
barrofr01	1871	1	BS1	None	18	86	13	13	2	1	0	11.0	1.0	0.0	0	0.0	⌵

```
In [31]: %sql describe people;
```

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

Out[31]:

Field	Type	Null	Key	Default	Extra
playerID	text	YES		None	
birthYear	double	YES		None	
birthMonth	double	YES		None	
birthDay	double	YES		None	
birthCountry	text	YES		None	
birthState	text	YES		None	
birthCity	text	YES		None	
deathYear	double	YES		None	
deathMonth	double	YES		None	
deathDay	double	YES		None	
deathCountry	text	YES		None	
deathState	text	YES		None	
deathCity	text	YES		None	
nameFirst	text	YES		None	
nameLast	text	YES		None	
nameGiven	text	YES		None	
weight	double	YES		None	
height	double	YES		None	
bats	text	YES		None	
throws	text	YES		None	
debut	text	YES		None	
finalGame	text	YES		None	
retroID	text	YES		None	
bbrefID	text	YES		None	

In [32]: %sql describe appearances;

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

Out[32]:

Field	Type	Null	Key	Default	Extra
yearID	bigint	YES		None	
teamID	text	YES		None	
lgID	text	YES		None	
playerID	text	YES		None	
G_all	bigint	YES		None	
GS	double	YES		None	
G_batting	bigint	YES		None	
G_defense	double	YES		None	
G_p	bigint	YES		None	
G_c	bigint	YES		None	
G_1b	bigint	YES		None	
G_2b	bigint	YES		None	
G_3b	bigint	YES		None	
G_ss	bigint	YES		None	
G_if	bigint	YES		None	
G_cf	bigint	YES		None	
G_rf	bigint	YES		None	
G_of	bigint	YES		None	
G_dh	double	YES		None	
G_ph	double	YES		None	
G_pr	double	YES		None	

```
In [33]: %sql describe batting;
```

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

```
Out[33]:
```

Field	Type	Null	Key	Default	Extra
playerID	text	YES		None	
yearID	bigint	YES		None	
stint	bigint	YES		None	
teamID	text	YES		None	
lgID	text	YES		None	
G	bigint	YES		None	
AB	bigint	YES		None	
R	bigint	YES		None	
H	bigint	YES		None	
2B	bigint	YES		None	
3B	bigint	YES		None	
HR	bigint	YES		None	
RBI	double	YES		None	
SB	double	YES		None	
CS	double	YES		None	
BB	bigint	YES		None	
SO	double	YES		None	
IBB	double	YES		None	
HBP	double	YES		None	
SH	double	YES		None	
SF	double	YES		None	
GIDP	double	YES		None	

Task 2: Complicated Queries

Note: Performing the query in this task may require changing column values or types.

Query - Career Summary

- Write a query that produces a result of the form:
 - playerID
 - nameLast
 - nameFirst
 - The sum of appearances.G_all for the player over all rows.

- The sum over all rows of the following columns from batting:
 - G
 - AB
 - R
 - AB
 - 2B
 - 3B
 - HR
 - RBI
 - BB
- batting_average , which is defined as $\frac{\text{sum}(H)}{\text{sum}(AB)}$
- on_base_percentage , which is defined as $\frac{(\text{sum}(H) + \text{sum}(BB))}{(\text{sum}(AB) + \text{sum}(BB))}$
- The query should be limited to 20 rows, and sorted by on_base_percentage from highest to lowest.
- batting_average and on_base_percentage should round to three decimal places.


```
In [34]: %%sql
use S22_W4111_HW2;
select * from
(select playerID, nameLast, nameFirst from people) as p
natural join
(select playerID, SUM(G_all) from appearances GROUP BY playerID) as a,
(select playerID, SUM(G), SUM(AB), SUM(R), SUM(H), SUM(2B), SUM(3B), SUM(HR), SUM
ROUND(SUM(H)/SUM(AB),3) as batting_average, ROUND((SUM(H)+SUM(BB))/(SUM(AB)+SUM
from batting GROUP BY playerID) as b
where a.playerID = b.playerID
ORDER BY on_base_percentage desc
LIMIT 20;
```

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

```
Out[34]:
```

playerID	nameLast	nameFirst	SUM(G_all)	playerID_1	SUM(G)	SUM(AB)	SUM(R)	SUM(H)	SUI
sotogr01	Soto	Gregory	60	sotogr01	60	2	0	2	
meansjo01	Means	John	42	meansjo01	42	1	0	1	
alanirj01	Alaniz	R. J.	12	alanirj01	12	1	0	1	
torrejo02	Torres	Jose	66	torrejo02	66	1	1	1	
pindebr01	Pinder	Branden	26	pindebr01	26	1	0	1	
hesslke01	Hessler	Keith	35	hesslke01	35	1	0	1	
barneja01	Barnes	Jacob	200	barneja01	200	0	0	0	
egeco01	Ege	Cody	18	egeco01	18	1	0	1	
garcija03	Garcia	Jason	21	garcija03	21	0	0	0	
schlogu01	Schlosser	Gus	15	schlogu01	15	1	0	1	
hughetr01	Hughes	Travis	24	hughetr01	24	1	0	1	
roachja01	Roach	Jason	2	roachja01	2	2	0	2	
ohmeke01	Ohme	Kevin	2	ohmeke01	2	1	0	1	
bartocl01	Bartosh	Cliff	53	bartocl01	53	1	0	1	
lyonbr01	Lyon	Brandon	572	lyonbr01	572	1	1	1	
campsh01	Camp	Shawn	541	campsh01	541	1	0	1	
colyest01	Colyer	Steve	61	colyest01	61	0	0	0	
sherrge01	Sherrill	George	442	sherrge01	442	0	0	0	
standja01	Standridge	Jason	80	standja01	80	0	0	0	
gideobr01	Gideon	Brett	34	gideobr01	34	1	0	1	

Question 8: "Fun" with Sets

- People represents basic information about people associated with Major League Baseball.

- `Appearances` contains information about people who appeared (played in) MLB games.
- There are some entries in the `People` table that do not appear in `Appearances`.
- Using a **subquery**, write a query that counts the number of people in the `People` table who do not have an entry in `Appearances`.
- Run your query below. Note, your query will be **SLOW**.

```
In [35]: %%sql
use S22_W4111_HW2;
select COUNT(playerID) from people where
not playerID in (select playerID from appearances);

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

```
Out[35]: COUNT(playerID)
         460
```

- Just for the heck of it, run the scripts below and repeat your query. Also, the changes I am making are a good hint on how to solve the problem.

```
In [36]: %%sql

use s22_w4111_hw2;

drop table if exists people_fast;
drop table if exists appearances_fast;

create table people_fast as select * from people;
create table appearances_fast as select * from appearances;

ALTER TABLE `appearances_fast`
CHANGE COLUMN `playerID` `playerID` VARCHAR(16) NULL DEFAULT NULL ,
ADD INDEX `playerID_idx` (playerID) VISIBLE;

ALTER TABLE `people_fast`
CHANGE COLUMN `playerID` `playerID` VARCHAR(16) NULL DEFAULT NULL ,
ADD INDEX `peopleID_idx` (playerID) VISIBLE;

* mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
0 rows affected.
20358 rows affected.
108717 rows affected.
108717 rows affected.
20358 rows affected.
```

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

- Run your query here.

```
In [37]: %%sql
use S22_W4111_HW2;
select COUNT(playerID) from people where
not playerID in (select playerID from appearances);

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

```
Out[37]: COUNT(playerID)
         460
```

Question 9: Don Plays Baseball

- I always wanted to play baseball for the Boston Red Sox, and also play with Ted Williams.
- Ted Williams' playerID is willite01 .
- My playerID would be fergusdo .
- Perform the following tasks using SQL:
 - Insert an entry in people with:
 - playerID = fergusdo
 - nameLast = Ferguson
 - nameFirst = Donald
 - Existence without Ted Williams is meaningless. So, using an Update statement, update the entry in people for fergusdo to have the same birthYear , birthMonth and birthDay as Ted Williams.
- Run a query showing the row in people for fergusdo .
- Delete the row you added.

```
In [38]: # Insert statement
%%sql insert into people (playerID, nameLast, nameFirst) values ("fergusdo", "Fergusdo", "Donald")

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

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

```
In [39]: # Update statement
#
%sql UPDATE people set birthYear = (select * from (select birthYear from people v

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

Out[39]: []

```
In [40]: # Select statement showing row.
#
%sql select * from people where playerID = 'fergusdo';

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

Out[40]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth
fergusdo	1918.0	8.0	30.0	None	None	None	None	None

```
In [41]: # Delete the created row.
#
%sql delete from people where playerID = 'fergusdo';

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

Out[41]: []

Question 10: There is No Question 10

- You all get a free point for putting up with me.