COMS W4111-003 (Fall 2022) Introduction to Databases

</span>

Homework 2: Non-Programming and Non-Programming

Note: Please replace the information below with your last name, first name and UNI.

Jin, Jason, hj2602

Setup Environment

In [1]:

```
%load_ext sql
```

In [2]:

```
%sql mysql+pymysql://root:1168622jin@localhost
```

In [3]:

```
%sql select * from db_book.classroom
```

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

Out[3]:

building	room_number	capacity
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Introduction

Structure

This homework has four sections:

- PART A: Written questions on concepts covered in class.
- PART B: Common problems for both the programming and non-programmiong tracks.

Because of delays in progress and lectures, I am not defining track specific questions for HW 2.

Submission

Please refer Ed Discussion Announcement for the submission instructions.

This assignment is due October 30, 11:59 pm EDT

Collaboration

- You may use any information you get in TA or Prof. Ferguson's office hours, from lectures or from recitations.
- You may use information that you find on the web.
- You are NOT allowed to collaborate with other students outside of office hours.

Part A: Written

Place your answers in the Markdhown cells following each question. Your answers should be succinct. We will deduct points for long or rambling answers.

W1

Question:

Codd's 3rd Rule states: "Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in fully relational DBMS for representing missing information and inapplicable information in a systematic way, independent of data type."

Briefly explain the meaning of this rule.

The example database from the book has a table `takes` with a column `grade.` The value is `NULL` if a student took the course but did not get a grade. Why would using the string "NA" instead of `NULL` cause problems for some queries on this table?

Answer:

Null values are not equivalent to zeros or blank strings in SQL because they are used as placeholders waiting for the values to be assigned. So to answer the question, if the string "NA" was used instead of NULL, SQL would use "NA" as a value assigned which is not what we want. For instance, if we processed a SUM function, then SQL will skip the null values automaically and return the sum of the values. However since the value is assigned as "NA" here it would not work because "NA" is not a valid value for a grade.

W2

Question:

Codd's 4th Rule states: "The data base description is represented at the logical level in the same way as ordinary data, so that authorized users can apply the same relational language to its interrogation as they apply to the regular data."

- What is the schema that contains the database description information for MySQL?

- Give three examples of information about database structure that is in the schema.

Answer:

A logical schema contains the database description information for MySQL and three examples of information are attributes(description), primary keys and foreign keys.

W3

Question:

- What is the primary reason for creating indexes?
- Why is creating very many indexes potentially a problem? What is the negative affect of creating unecessary indexes?

Answer:

The primary reason is to quickly locate the data without having to search through every row in a database table every time the table needs to be accessed. A potential problem with creating many indexes is related to data storage. Indexes that are created will pile up taking more space and time to accomplish maintenance tasks. Also unnecessary indexes will create unorganized indexing which will lead to wasted disk space and cache. So it is important to free up the space by deleting the unnecessary indexes after making appropriate use of them.

W4

Question:

- What is the primary reason for creating indexes?
- Why is creating very many indexes potentially a problem? What is the negative affect of creating unecessary indexes?

Answer:

The primary reason is to quickly locate the data without having to search through every row in a database table every time the table needs to be accessed. A potential problem with creating many indexes is related to data storage. Indexes that are created will pile up taking more space and time to accomplish maintenance tasks. Also unnecessary indexes will create unorganized indexing which will lead to wasted disk space and cache. So it is important to free up the space by deleting the unnecessary indexes after making appropriate use of them.

W5

Question:

- In SQL, what is the main difference between a primary key and a unique key?

Answer:

Primary key does not accept NULL values but unique key accepts NULL values. So a table can have only one primary key and multiple unique keys in some instances.

W6

Question:

- Views are a valuable concept in relational databases. What are three distinct reasons for/benefits of creating views?

Answer:

One, creating views help hide the complexity of data, so if there are some sensitive information related column names, views can help hide those information. Two, since insertion, update and deletion are not possible on views it helps the users not to manipulate the data providing access restriction. Three, by using join we can simplify multiple tables into one and even use that view to create another useful view.

W7

Question:

- Explain the concept of a *domain?* for table column values.
- Consider Columbia Course numbers, e.g. W4111, E1006, C1001. What is the domain for course numbers not just `CHAR(5).`

Answer:

Domain is a set of values that the column can take. For Columbia Course numbers, the domain would be better if it was varchar(5) instead of char(5) because char(5) takes in a value with specific length of 5 while varchar(5) takes in a value that can hold either numeric or character or both with lengths shorter than 5. This way it gives more flexibility to number the courses and name the course types in variety of different ways.

W8

Question:

- List two examples of `integrity constraints` that apply to a single table, and one example that applies to multiple tables.

Answer:

Let's say that we are querying a database of all the banks in the United States. If we were to apply an integrity constraint on a single table for a designated bank, an example would be having a must condition for a checking account to have a balance over $5000 (for one bank). An example that applies to multiple tables would be a customer must have non-null phone number for all the banks in the United States.

So the prior integrity constraint can apply to a single table that is associated with a particular bank whereas the latter integrity constraint must apply to multiple tables as phone number is a required entry for all possible banks that the customer has an account with.

W9

Question:

Consider the table `time_slot` from the sample database associated with the recommended text book.

- The data type for the column `day` is `char(1).` Given the data types MySQL supports, what is a better data type?
- What is a scenario that would motivate creating an index on the column `day`?

Answer:

We can use ENUM and declare it in DDL as such: day ENUM('M', 'T', 'W', 'T', 'F') NOT NULL

The advantage of declaring ENUM is that the values are already indexed so 1 would be mapped to 'm', 2 is mapped to 'T' and so forth. This way it is easier to access the values using the indexes and also distinguish ambiguity between 'T'uesday and 'T'hursday for instance.

W10

Question:

Consider the table `course` from the sample database associated with the recommended text book.

- There is a design problem with the column `course_id`. What is the problem and how would you fix it?

Answer:

The course_id here from DDL shows that it is a primary key. However, course_id is not really an unique value because when we look at the table named "teaches" we can see that the same course_id can have different sections even though they share the same ID number and course_id number. So even though two courses can share the same name and ID number, two courses are different in that they are taught in different sections with different students enrolled in each section.

Part B: Common Tasks

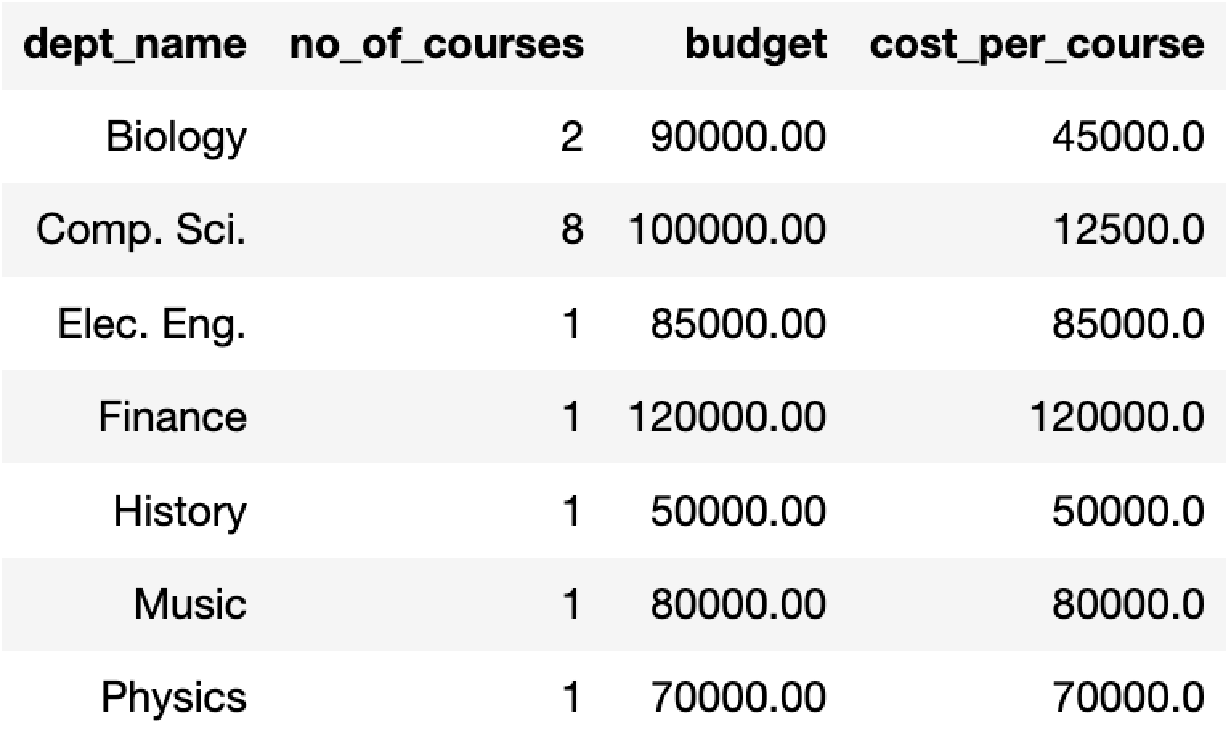
- You will use the example datatabase from the book associated with the class and the Lahman baseball data you loaded from HW 0 to answer these questions.

- Execute the SQL you write as answers in the answer cells.

C 1

Question:

Write a query that produces the following table. You must match column names and formatting of values.



dept_name	no_of_courses	budget	cost_per_course
Biology	2	90000.00	45000.0
Comp. Sci.	8	100000.00	12500.0
Elec. Eng.	1	85000.00	85000.0
Finance	1	120000.00	120000.0
History	1	50000.00	50000.0
Music	1	80000.00	80000.0
Physics	1	70000.00	70000.0

Answer:

```
In [4]: %sql use db_book;
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
Out[4]: []
```

```
In [6]: %%sql

drop view if exists question_3;
create view question_3 as
select dept_name, count(*) as no_of_courses
from instructor, teaches
where instructor.ID = teaches.ID group by dept_name;

select dept_name, no_of_courses, budget, (budget/no_of_courses) as cost_per_course
from question_3
join department
using(dept_name);
```

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

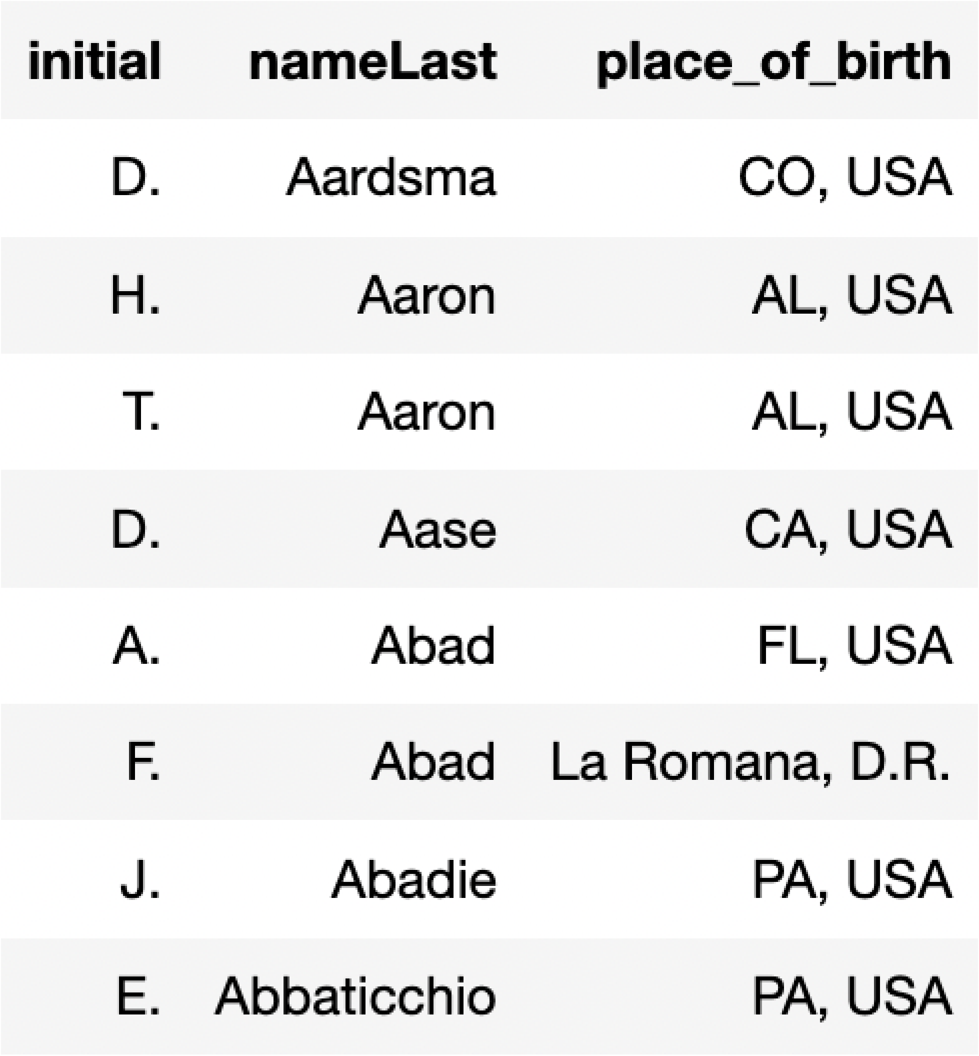
Out[6]:

dept_name	no_of_courses	budget	cost_per_course
Biology	2	90000.00	45000.000000
Comp. Sci.	8	100000.00	12500.000000
Elec. Eng.	1	85000.00	85000.000000
Finance	1	120000.00	120000.000000
History	1	50000.00	50000.000000
Music	1	80000.00	80000.000000
Physics	1	70000.00	70000.000000

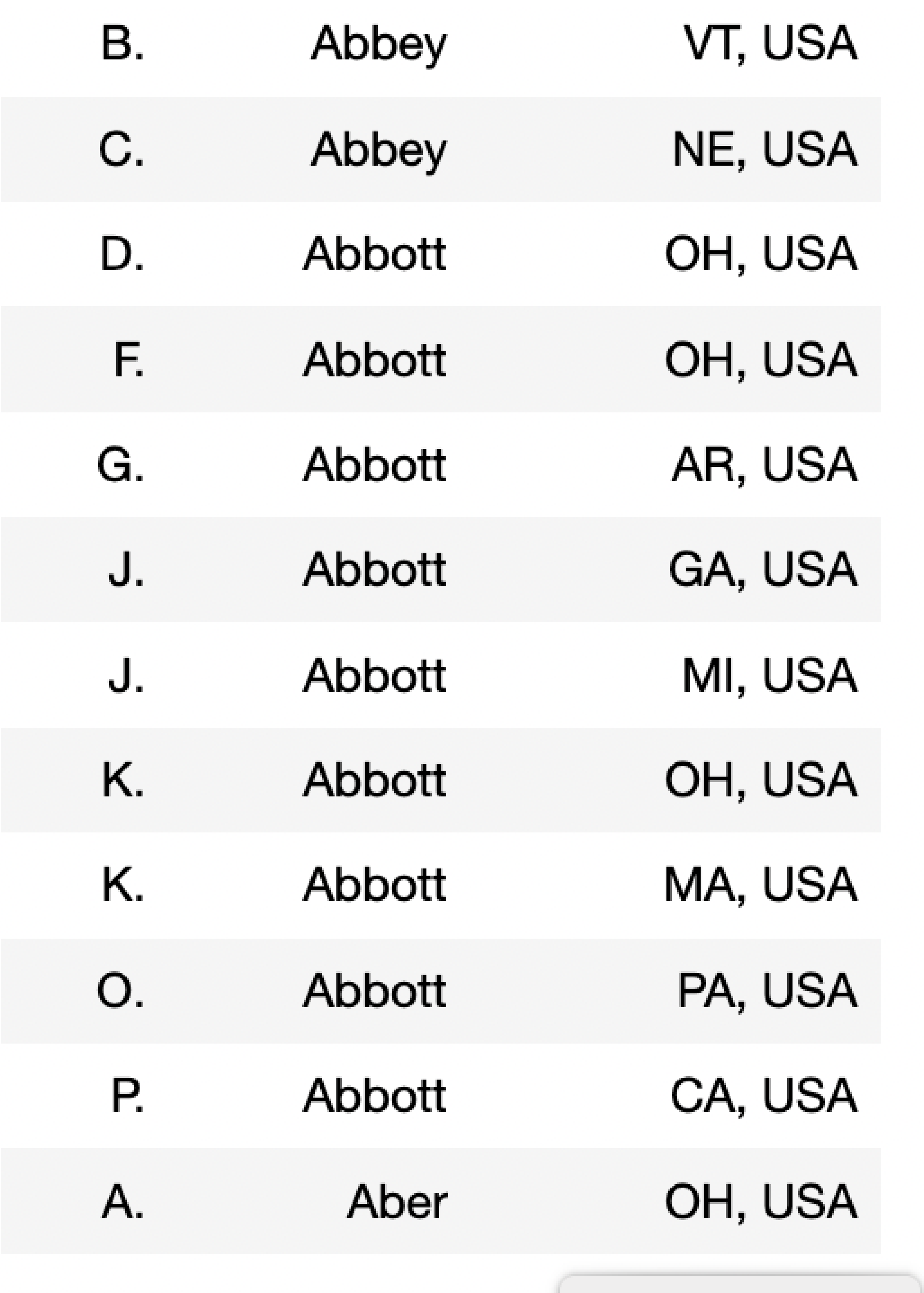
C 2

Question

- Use the `people` table for Lahman Baseball data for this query.
- Write a query that produces a result with the following columns:
    - `first_initial` is the first first letter of `nameFirst` followed by `.`
    - `nameLast`
    - `place_of_birth` is the `birthCity`, a comma, and the `birthCountry.`
- You can just run your queries for the first 10 people (fyi, the example table below have more than the first 10 people)



initial	nameLast	place_of_birth
D.	Aardsma	CO, USA
H.	Aaron	AL, USA
T.	Aaron	AL, USA
D.	Aase	CA, USA
A.	Abad	FL, USA
F.	Abad	La Romana, D.R.
J.	Abadie	PA, USA
E.	Abbaticchio	PA, USA



B.	Abbey	VT, USA
C.	Abbey	NE, USA
D.	Abbott	OH, USA
F.	Abbott	OH, USA
G.	Abbott	AR, USA
J.	Abbott	GA, USA
J.	Abbott	MI, USA
K.	Abbott	OH, USA
K.	Abbott	MA, USA
O.	Abbott	PA, USA
P.	Abbott	CA, USA
A.	Aber	OH, USA

Answer

In [7]:

```
%sql use lahmansbaseballdb;
```

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

Out[7]:

```
[]
```

In [12]:

```
%%sql

select concat(substring(nameFirst, 1, 1), ".") as initial,
        nameLast,
        concat(birthState, ",", birthCountry) as place_of_birth
        from people limit 10;
```

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

Out[12]:

initial	nameLast	place_of_birth
D.	Aardsma	CO,USA
H.	Aaron	AL,USA
T.	Aaron	AL,USA
D.	Aase	CA,USA
A.	Abad	FL,USA
F.	Abad	La Romana,D.R.
J.	Abadie	PA,USA
E.	Abbaticchio	PA,USA
B.	Abbey	VT,USA
C.	Abbey	NE,USA

C3

Question

- Use the tables `people, appearances, batting` from the Lahman's Baseball data to answer this question.

- Produce a table of the form:
  - `playerID`
  - `nameLast`
  - `nameFirst`
  - `career_teams` is a semi-colon separated list of the team
  - `career_games` is the sum of `G_all` from `appearances`
  - `total_abs` is the sum of `AB` from `batting`
  - `total_hits` is the sum of `h` from `batting`
  - `batting_avg` is `total_hits/total_abs` is `total_abs` is not 0, and is `` `NULL `` otherwise.

- Show the first 10 rows like below.

playerID	nameLast	nameFirst	career_teams	career_games	total_abs	total_hits	batting_avg
aardsda01	Aardsma	David	ATL;BOS;CHA;CHN;NYA;NYN;SEA;SFN	331	4	0	0.0000
aaronha01	Aaron	Hank	ATL;ML1;ML4	3298	12364	3771	0.3050
aaronto01	Aaron	Tommie	ATL;ML1	437	944	216	0.2288
aasedo01	Aase	Don	BAL;BOS;CAL;LAN;NYN	448	5	0	0.0000
abadan01	Abad	Andy	BOS;CIN;OAK	15	21	2	0.0952
abadfe01	Abad	Fernando	BOS;HOU;MIN;OAK;SFN;WAS	384	9	1	0.1111
abadijo01	Abadie	John	BR2;PH3	12	49	11	0.2245
abbated01	Abbaticchio	Ed	BSN;PHI;PIT	857	3044	772	0.2536
abbeybe01	Abbey	Bert	BRO;CHN;WAS	79	225	38	0.1689
abbeych01	Abbey	Charlie	WAS	452	1756	493	0.2808

Answer 1

In [9]:

```
%%sql

select playerID,
       nameLast,
       nameFirst,
       group_concat(distinct teamID order by teamID separator ';') as career_teams,
       sum(G) as career_games,
       sum(AB) as total_abs,
       sum(H) as total_hits,
       (sum(H) / sum(AB)) as batting_avg
       from people join batting using(playerID) group by playerID limit 10;
```

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

Out[9]:

playerID	nameLast	nameFirst	career_teams	career_games	total_abs	total_l
aardsda01	Aardsma	David	ATL;BOS;CHA;CHN;NYA;NYN;SEA;SFN	331	4	
aaronha01	Aaron	Hank	ATL;ML1;ML4	3298	12364	3
aaronto01	Aaron	Tommie	ATL;ML1	437	944	
aasedo01	Aase	Don	BAL;BOS;CAL;LAN;NYN	448	5	
abadan01	Abad	Andy	BOS;CIN;OAK	15	21	
abadfe01	Abad	Fernando	BOS;HOU;MIN;OAK;SFN;WAS	384	9	
abadijo01	Abadie	John	BR2;PH3	12	49	
abbated01	Abbaticchio	Ed	BSN;PHI;PIT	855	3044	
abbeybe01	Abbey	Bert	BRO;CHN;WAS	79	225	
abbeych01	Abbey	Charlie	WAS	452	1756	

Answer 2

Demonstrate that you computed `batting_avg` correctly by returning the the first 10 rows with a null batting average like below.

playerID	nameLast	nameFirst	career_teams	career_games	total_abs	total_hits	batting_avg
abbotgl01	Abbott	Glenn	DET;OAK;SEA	248	0	0	None
abreubr01	Abreu	Bryan	HOU	7	0	0	None
abreuju01	Abreu	Juan	HOU	7	0	0	None
achteaj01	Achter	A. J.	LAA;MIN	45	0	0	None
acrema01	Acre	Mark	OAK	114	0	0	None
adamja01	Adam	Jason	KCA;TOR	54	0	0	None
adamsch01	Adams	Chance	NYA	16	0	0	None
adamswi02	Adams	Willie	OAK	25	0	0	None
adenhni01	Adenhart	Nick	LAA	4	0	0	None
adkinst01	Adkins	Steve	NYA	5	0	0	None

In [10]:

```
%%sql

drop view if exists question_2;
create view question_2 as
select playerID,
        nameLast,
        nameFirst,
        group_concat(distinct teamID order by teamID separator ';') as career_teams,
        sum(G) as career_games,
        sum(AB) as total_abs,
        sum(H) as total_hits,
        (sum(H) / sum(AB)) as batting_avg
        from people join batting using(playerID) group by playerID;

select * from question_2 where total_abs = 0 and total_hits = 0 limit 10;
```

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

Out[10]:

playerID	nameLast	nameFirst	career_teams	career_games	total_abs	total_hits	batting_avg
abbotgl01	Abbott	Glenn	DET;OAK;SEA	248	0	0	None
abreubr01	Abreu	Bryan	HOU	7	0	0	None
abreuju01	Abreu	Juan	HOU	7	0	0	None
achteaj01	Achter	A. J.	LAA;MIN	45	0	0	None
acrema01	Acre	Mark	OAK	114	0	0	None
adamja01	Adam	Jason	KCA;TOR	54	0	0	None
adamsch01	Adams	Chance	NYA	16	0	0	None
adamswi02	Adams	Willie	OAK	25	0	0	None
adenhni01	Adenhart	Nick	LAA	4	0	0	None
adkinst01	Adkins	Steve	NYA	5	0	0	None

C4

question

- A person (from `people`) was a player in MLB if their `playerID` appears in `appearances.`
- A person (from `managers`) was a manager if their `playerID` appears in `managers.`
- Produce the following table from `halloffame` for people in `halloffame` that were not managers or players.
- My first 10 rows look like below.

playerid	nameLast	nameFirst	category
bulkemo99	Bulkeley	Morgan	Pioneer/Executive
johnsba99	Johnson	Ban	Pioneer/Executive
cartwal99	Cartwright	Alexander	Pioneer/Executive
chadwhe99	Chadwick	Henry	Pioneer/Executive
landike99	Landis	Kenesaw	Pioneer/Executive
connoto99	Connolly	Tommy	Umpire
klembi99	Klem	Bill	Umpire
frickfo99	Frick	Ford	Pioneer/Executive
weissge99	Weiss	George	Pioneer/Executive
gibsojo99	Gibson	Josh	Player

- Your query should produce all rows.

Answers

In [13]:

```
%%sql

drop view if exists revised;
create view revised as
(
(select playerID, category from halloffame
          left join appearances using (playerID)
 where halloffame.playerID <> appearances.playerID or appearances.playerID is NULL)

union

(select playerID, category from halloffame
          left join managers using (playerID)
 where halloffame.playerID <> managers.playerID or managers.playerID is NULL)
);

select distinct playerID, nameLast, nameFirst, revised.category
from people right join revised using(playerID)
where people.playerID is not NULL limit 15;
```

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

```
Out[13]:  playerID    nameLast  nameFirst           category
         bulkemo99    Bulkeley     Morgan  Pioneer/Executive
         johnsba99     Johnson        Ban  Pioneer/Executive
          cartwal99  Cartwright  Alexander  Pioneer/Executive
         chadwhe99    Chadwick      Henry  Pioneer/Executive
         mccarjo99    McCarthy        Joe            Manager
         landike99      Landis    Kenesaw  Pioneer/Executive
         barroed99      Barrow         Ed  Pioneer/Executive
         connoto99    Connolly      Tommy             Umpire
          klembi99        Klem       Bill             Umpire
         frickfo99       Frick       Ford  Pioneer/Executive
         weissge99       Weiss     George  Pioneer/Executive
         gibsojo99      Gibson       Josh             Player
         harriwi99    Harridge       Will  Pioneer/Executive
         leonabu99     Leonard       Buck             Player
         evansbi99       Evans      Billy             Umpire
```

The query above processes the same result but the player ID is ordered differently. For example, playerID = mccarjo99, nameLast = McCarthy, nameFirst = Joe category = Manager in the middle of the table does not appear in managers but appears in halloffame.

```
In [ ]:
```