COMS W4111-002, V02 (Spring 2022) Introduction to Databases

</span>

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.

1. 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.

1. 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:

(1)Interpretations of NULL value: missing the data, no value in a cell, inappropriate information, unknown data or not applicable.

(2)Benefits of using NULL: when using -1 for representation, it might cause problems regarding problems with numbers For example, now we have a database containing the data of people with birthday and age, some people don't know exactly what is the date of their birth and exact ages, which are unknown info. If we use -1 represent people's unknown age, when we try to select people aged under 20 will also select the person whose age is "-1". It needs more requirements for selecting and searching. So when using NULL, unless the data itself is called "NULL", or "NULL" could always represent the missing or unknown data, which is more fault tolerant.

Question 2: Keys

Briefly explain the following concepts:

- Primary Key
- Candidate Key

- Super Key
- Alternate Key
- Composite Key
- Unique Key
- Foreign Key

Answer:

(1) Primary Key: A primary key is a special relational database table column designated to uniquely identify each table record.

(2) Candidate Key: A candidate key is a specific type of field in a relational database that can identify each unique record independently of any other data.

(3) Super Key: A super key is a set of one or more attributes (columns), which can uniquely identify a row in a table.

(4) Alternate Key: An alternate key is candidate key which is not the Primary key.

(5) Composite Key: A composite key, in the context of relational databases, is a combination of two or more columns in a table that can be used to uniquely identify each row in the table.

(6) Unique Key: A unique key is a group of one or more than one fields or columns of a table which uniquely identify database record.

(7) Foreign Key: A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables.

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:

Closure property: the relational algebra consists operator that takes one or more relations as input and produces another relation as output. Relations are closed under the algebra, just as numbers are closed under arithmetic operations. That the output is another relation is the so called closure property of the relational algebra.

Importance: It allows expressions to be nested, just as in arithmetic. Since the output from every operation is the same kind of thing as the input, the output from one operation can become the input to another.

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:

Concept: Two SQL queries are semantically equivalent if they produce the same results given any valid input relations.

Derive JOIN: we can select the columns or attributes that satisfy the requirements in each table in nested form, and use project operation to retrieve column -specific data in the selected columns(selected from table), which could finish the task as "JOIN". For example, For example,

```
project B1, B2,...,Bn

    from (select A1, A2,...,An

        from r1, r2, ...rm where P)

            where Q
```

This clause could equal the JOIN operation.

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:

(1) Simple attribute: simple attributes are atomic values, which cannot be divided further.

(2) Composite attribute: composite attribute is an attribute where the values of that attribute can be further subdivided into meaningful sub-parts, or are made of more than one simple attribute.

(3) Derived attribute: derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.

(4) Single-value attribute: single-valued attribute is an attribute that can have only a single value.

(5) Multi-value attribute: multi-value attributes may contain more than one values.

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 = "Xcz990208!"
```

In [2]:

```
database_url = "mysql+pymysql://" + \
    database_user_id + ":" + database_pwd + "@localhost"
database_url
```

Out[2]:

```
'mysql+pymysql://root:Xcz990208!@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 drop schema if exists S22_W4111_HW2
%sql create schema if not exists S22_W4111_HW2
%sql select 1;
```

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

Out[11]:

	0
0	Towny,Cavet,tcavet0@blinklist.com,1/9/1971,+62...
1	Port,Gaylor,pgaylor1@blogger.com,3/15/1939,+86...
2	Georgetta,Haddon,ghaddon2@symantec.com,9/19/19...
3	Wylma,Lanney,wlanney3@list-manage.com,2/21/201...
4	Mignonne,Georgeson,mgeorgeson4@123-reg.co.uk,8...
5	Cchaddie,Cossins,ccossins5@chronoengine.com,3/...
6	Andie,Matyushonok,amatyushonok6@ask.com,4/24/1...
7	Skippie,Zuenelli,szuenelli7@merriam-webster.co...
8	Averyl,Barajas,abarajas8@fastcompany.com,6/19/...
9	Olia,Habens,ohabens9@quantcast.com,2/28/1922,+...

In [12]:

```
df.to_sql(
    "people_info", con=sqla_engine, if_exists="replace", index=False,
    schema="S22_W4111_HW2")
```

In [13]:

```
%sql use S22_W4111_HW2
%sql select 1;
```

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

Out[13]:

1
1

- Test loading the data.

In [14]:

```
%sql select * from people_info
```

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

Out[14]:

0

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 [15]:

```
%sql describe people_info;
```

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

Out[15]:

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 [16]:

```
%%sql
```

```
drop table if exists people_info_clean;
create table people_info_clean as (select
                                  (select substring_index(substring_index(people_in
                                  ,(select substring_index(substring_index(people_i
                                  ,(select substring_index(substring_index(people_i
                                  ,(select substring_index(substring_index(people_i
                                  ,(select substring_index(substring_index(people_i
                                  from people_info);
```

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

Out[16]: []

In [17]:

```
#
# Show the data.
#
%sql select * from people_info_clean
```

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

Out[17]:

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 [18]:

```
#
# Show the schema (architecture and structure).
#
%sql describe people_info_clean;
```

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

Out[18]:

Field	Type	Null	Key	Default	Extra
first_name	longtext	YES		None	
last_name	longtext	YES		None	
email	longtext	YES		None	
date_of_birth	longtext	YES		None	
telephone_no	longtext	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 [19]:

```
# Your code
```

In [20]:

```
%sql create schema if not exists S22_W4111_HW2;
```

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

Out[20]:

```
[]
```

In [21]:

```
%sql drop table if exists S22_W4111_HW2.people;
%sql drop table if exists S22_W4111_HW2.appearances;
%sql drop table if exists S22_W4111_HW2.batting;
```

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

Out[21]:

```
[]
```

In [22]:

```
%%sql
create table if not exists S22_W4111_HW2.people
(
        playerID text null,
        birthYear int null,
        birthMonth int null,
        birthDay int null,
        birthCountry text null,
        birthState text null,
        birthCity text null,
        deathYear float null,
        deathMonth float null,
        deathDay float null,
        deathCountry text null,
        deathState text null,
        deathCity text null,
        nameFirst text null,
        nameLast text null,
        nameGiven text null,
        weight float null,
        height float null,
        bats text null,
        throws text null,
        debut text null,
        finalGame text null,
        retroID text null,
        bbrefID text null
);
```

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

Out[22]:

```
[]
```

In [23]:

```
%%sql
create table if not exists S22_W4111_HW2.appearances
(
        yearID text null,
        teamID text null,
        lgID text null,
        playerID text null,
        G_all  float null,
        GS float null,
        G_batting float null,
        G_defense float null,
        G_p float null,
        G_c float null,
        G_1b float null,
        G_2b float null,
        G_3b float null,
        G_ss float null,
        G_lf float null,
        G_cf float null,
        G_rf float null,
        G_of float null,
        G_dh float null,
        G_ph float null,
        G_pr float null
);
```

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

Out[23]:

```
[]
```

In [24]:

```
%%sql
create table if not exists S22_W4111_HW2.batting
(
        playerID text null,
        yearID text null,
        stint text null,
        teamID text null,
        lgID text null,
        G float null,
        AB float null,
        R float null,
        H float null,
        2B float null,
        3B float null,
        HR float null,
        RBI float null,
        SB float null,
        CS float null,
        BB float null,
        SO float null,
        IBB float null,
        HBP float null,
        SH float null,
        SF float null,
        GIDP float null
);
```

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

Out[24]:

```
[]
```

In [25]:

```
%sql SET GLOBAL local_infile = ’ON’;
```

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

Out[25]:

```
[]
```

In [26]:

```
import pymysql
con = pymysql.connect(host=”localhost”,
                      user=”root”,
                      password=”Xcz990208!”,
                      autocommit=True,
                      local_infile=1)
```

In [27]:

```
sql_people = ”””
LOAD DATA LOCAL INFILE
’C://Users//94822//Desktop//Intro_to_databases_4111//HW2//s22_w4111_hw2_all//People.cs
INTO TABLE S22_W4111_HW2.people
    FIELDS TERMINATED BY ’,’
    ENCLOSED BY ’”’
    LINES TERMINATED BY ’\n’
    IGNORE 1 LINES;
”””

sql_appearances = ”””
LOAD DATA LOCAL INFILE
’C://Users//94822//Desktop//Intro_to_databases_4111//HW2//s22_w4111_hw2_all//Appearanc
INTO TABLE S22_W4111_HW2.appearances
    FIELDS TERMINATED BY ’,’
    ENCLOSED BY ’”’
    LINES TERMINATED BY ’\n’
    IGNORE 1 LINES;
”””

sql_batting = ”””
LOAD DATA LOCAL INFILE
’C://Users//94822//Desktop//Intro_to_databases_4111//HW2//s22_w4111_hw2_all//Batting.c
INTO TABLE S22_W4111_HW2.batting
    FIELDS TERMINATED BY ’,’
    ENCLOSED BY ’”’
    LINES TERMINATED BY ’\n’
    IGNORE 1 LINES;
”””
```

In [28]:

```
cur = con.cursor()
cur.execute(sql_people)
cur.execute(sql_appearances)
cur.execute(sql_batting)
cur.close()
```

In [29]:

```
# Some tests
```

In [30]:

```
%sql select * from S22_W4111_HW2.people limit 10;
```

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

Out[30]:

playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deatl
aardsda01	1981	12	27	USA	CO	Denver	0.0	
aaronha01	1934	2	5	USA	AL	Mobile	2021.0	
aaronto01	1939	8	5	USA	AL	Mobile	1984.0	
aasedo01	1954	9	8	USA	CA	Orange	0.0	
abadan01	1972	8	25	USA	FL	Palm Beach	0.0	
abadfe01	1985	12	17	D.R.	La Romana	La Romana	0.0	
abadijo01	1850	11	4	USA	PA	Philadelphia	1905.0	
abbated01	1877	4	15	USA	PA	Latrobe	1957.0	
abbeybe01	1869	11	11	USA	VT	Essex	1962.0	
abbeych01	1866	10	14	USA	NE	Falls City	1926.0	

In [31]:

```
%sql select * from S22_W4111_HW2.appearances limit 10;
```

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

Out[31]:

yearID	teamID	lgID	playerID	G_all	GS	G_batting	G_defense	G_p	G_c	G_1b	G_2b	G_3b
1871	TRO	NA	abercda01	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
1871	RC1	NA	addybo01	25.0	25.0	25.0	25.0	0.0	0.0	0.0	22.0	0.0
1871	CL1	NA	allisar01	29.0	29.0	29.0	29.0	0.0	0.0	0.0	2.0	0.0
1871	WS3	NA	allisdo01	27.0	27.0	27.0	27.0	0.0	27.0	0.0	0.0	0.0
1871	RC1	NA	ansonca01	25.0	25.0	25.0	25.0	0.0	5.0	1.0	2.0	20.0
1871	FW1	NA	armstbo01	12.0	12.0	12.0	12.0	0.0	0.0	0.0	0.0	0.0

yearID	teamID	lgID	playerID	G_all	GS	G_batting	G_defense	G_p	G_c	G_1b	G_2b	G_3b
1871	RC1	NA	barkeal01	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
1871	BS1	NA	barnero01	31.0	31.0	31.0	31.0	0.0	0.0	0.0	16.0	0.0
1871	FW1	NA	barrebi01	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0
1871	BS1	NA	barrofr01	18.0	17.0	18.0	18.0	0.0	0.0	0.0	1.0	0.0

In [32]:

```
%sql select * from S22_W4111_HW2.batting limit 10;
```

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

Out[32]:

playerID	yearID	stint	teamID	lgID	G	AB	R	H	2B	3B	HR	RBI	SB	CS	BB
abercda01	1871	1	TRO	NA	1.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
addybo01	1871	1	RC1	NA	25.0	118.0	30.0	32.0	6.0	0.0	0.0	13.0	8.0	1.0	4.0
allisar01	1871	1	CL1	NA	29.0	137.0	28.0	40.0	4.0	5.0	0.0	19.0	3.0	1.0	2.0
allisdo01	1871	1	WS3	NA	27.0	133.0	28.0	44.0	10.0	2.0	2.0	27.0	1.0	1.0	0.0
ansonca01	1871	1	RC1	NA	25.0	120.0	29.0	39.0	11.0	3.0	0.0	16.0	6.0	2.0	2.0
armstbo01	1871	1	FW1	NA	12.0	49.0	9.0	11.0	2.0	1.0	0.0	5.0	0.0	1.0	0.0
barkeal01	1871	1	RC1	NA	1.0	4.0	0.0	1.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0
barnero01	1871	1	BS1	NA	31.0	157.0	66.0	63.0	10.0	9.0	0.0	34.0	11.0	6.0	13.0
barrebi01	1871	1	FW1	NA	1.0	5.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
barrofr01	1871	1	BS1	NA	18.0	86.0	13.0	13.0	2.0	1.0	0.0	11.0	1.0	0.0	0.0

In [33]:

```
%sql describe S22_W4111_HW2.people;
```

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

Out[33]:

Field	Type	Null	Key	Default	Extra
playerID	text	YES		None	
birthYear	int	YES		None	
birthMonth	int	YES		None	
birthDay	int	YES		None	
birthCountry	text	YES		None	
birthState	text	YES		None	
birthCity	text	YES		None	
deathYear	float	YES		None	
deathMonth	float	YES		None	
deathDay	float	YES		None	
deathCountry	text	YES		None	

Field	Type	Null	Key	Default	Extra
deathState	text	YES		None	
deathCity	text	YES		None	
nameFirst	text	YES		None	
nameLast	text	YES		None	
nameGiven	text	YES		None	
weight	float	YES		None	
height	float	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 [34]:

```
%sql describe S22_W4111_HW2.appearances;
```

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

Out[34]:

Field	Type	Null	Key	Default	Extra
yearID	text	YES		None	
teamID	text	YES		None	
lgID	text	YES		None	
playerID	text	YES		None	
G_all	float	YES		None	
GS	float	YES		None	
G_batting	float	YES		None	
G_defense	float	YES		None	
G_p	float	YES		None	
G_c	float	YES		None	
G_1b	float	YES		None	
G_2b	float	YES		None	
G_3b	float	YES		None	
G_ss	float	YES		None	
G_lf	float	YES		None	
G_cf	float	YES		None	
G_rf	float	YES		None	
G_of	float	YES		None	

Field	Type	Null	Key	Default	Extra
G_dh	float	YES		None	
G_ph	float	YES		None	
G_pr	float	YES		None	

In [35]:

```
%sql describe S22_W4111_HW2.batting;
```

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

Out[35]:

Field	Type	Null	Key	Default	Extra
playerID	text	YES		None	
yearID	text	YES		None	
stint	text	YES		None	
teamID	text	YES		None	
lgID	text	YES		None	
G	float	YES		None	
AB	float	YES		None	
R	float	YES		None	
H	float	YES		None	
2B	float	YES		None	
3B	float	YES		None	
HR	float	YES		None	
RBI	float	YES		None	
SB	float	YES		None	
CS	float	YES		None	
BB	float	YES		None	
SO	float	YES		None	
IBB	float	YES		None	
HBP	float	YES		None	
SH	float	YES		None	
SF	float	YES		None	
GIDP	float	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 $\frac{sum(H)}{sum(AB)}$
    - `on_base_percentage` , which is defined as $\frac{(sum(H)+sum(BB))}{(sum(AB)+sum(BB))}$

- 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 [36]:

```
# To simplify the people to be selected
%sql with people_basic as (select playerID,nameLast,nameFirst from s22_w4111_hw2.peop
```

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

Out[36]:

playerID	nameLast	nameFirst
aardsda01	Aardsma	David
aaronha01	Aaron	Hank
aaronto01	Aaron	Tommie
aasedo01	Aase	Don
abadan01	Abad	Andy
abadfe01	Abad	Fernando
abadijo01	Abadie	John
abbated01	Abbaticchio	Ed
abbeybe01	Abbey	Bert
abbeych01	Abbey	Charlie

In [37]:

```
%%sql
with people_basic as
        (select playerID, nameLast, nameFirst from s22_w4111_hw2.people),
appearance_career as
        (select playerID, sum(G_all) as total_g from s22_w4111_hw2.appearances grou
batting_sum as
        (select playerID,
    sum(G) as total_G,
        sum(AB) as total_AB,
        sum(2B) as total_2B,
        sum(3B) as total_3B,
        sum(HR) as total_HR,
        sum(RBI) as total_RBI,
        sum(BB) as total_BB,
        round(sum(H)/if(sum(AB)=0, Null, sum(AB)),3) as batting_average,
        round(sum(H)+sum(BB)/if(sum(AB)=0, Null, sum(AB))+if(sum(BB)=0, Null, sum
                from s22_w4111_hw2.batting group by playerID
        ) select * from
                (people_basic join appearance_career using(playerID))
        join batting_sum using(playerID) order by on_base_percentage DESC limit 20
```

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

Out[37]:

playerID	nameLast	nameFirst	total_g	total_G	total_AB	total_2B	total_3B	total_HR	total_RB
rosepe01	Rose	Pete	3562.0	3562.0	14053.0	746.0	135.0	160.0	1314.
bondsba01	Bonds	Barry	2986.0	2986.0	9847.0	601.0	77.0	762.0	1996.
cobbty01	Cobb	Ty	3034.0	3035.0	11436.0	724.0	295.0	117.0	1944.
yastrca01	Yastrzemski	Carl	3308.0	3308.0	11988.0	646.0	59.0	452.0	1844.
henderi01	Henderson	Rickey	3081.0	3081.0	10961.0	510.0	66.0	297.0	1115.
musiast01	Musial	Stan	3026.0	3026.0	10972.0	725.0	177.0	475.0	1951.
aaronha01	Aaron	Hank	3298.0	3298.0	12364.0	624.0	98.0	755.0	2297.
ruthba01	Ruth	Babe	2504.0	2503.0	8398.0	506.0	136.0	714.0	2217.
speaktr01	Speaker	Tris	2792.0	2789.0	10195.0	792.0	222.0	117.0	1529.
collied01	Collins	Eddie	2825.0	2826.0	9949.0	438.0	187.0	47.0	1300.
mayswi01	Mays	Willie	2992.0	2992.0	10881.0	523.0	140.0	660.0	1903.
willite01	Williams	Ted	2292.0	2292.0	7706.0	525.0	71.0	521.0	1839.
murraed02	Murray	Eddie	3026.0	3026.0	11336.0	560.0	35.0	504.0	1917.
ottme01	Ott	Mel	2730.0	2730.0	9456.0	488.0	72.0	511.0	1860.
pujolal01	Pujols	Albert	2862.0	2862.0	10839.0	669.0	16.0	662.0	2100.
jeterde01	Jeter	Derek	2747.0	2747.0	11195.0	544.0	66.0	260.0	1311.
rodrial01	Rodriguez	Alex	2784.0	2784.0	10566.0	548.0	31.0	696.0	2086.
boggswa01	Boggs	Wade	2440.0	2440.0	9180.0	578.0	61.0	118.0	1014.
ansonca01	Anson	Cap	2524.0	2524.0	10281.0	582.0	142.0	97.0	2075.
molitpa01	Molitor	Paul	2683.0	2683.0	10835.0	605.0	114.0	234.0	1307.

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 [38]:

```
%%sql
use s22_w4111_hw2;
select count(playerID) as diff_player
    from people
         where playerID not in (select distinct playerID from appearances);
```

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

Out[38]:

diff_player
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 [39]:

```
%%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[39]:

```
[]
```

- Run your query here.

In [40]:

```
%%sql
use s22_w4111_hw2;
select count(playerID) as diff_player
```

```
    from people_fast
         where playerID not in (select distinct playerID from appearances_fast);
```

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

Out[40]:

diff_player
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 [100…

```
# Insert statement.
```

In [101…

```
%%sql
use s22_w4111_hw2;
INSERT INTO people(playerID, nameFirst, nameLast) values('fergusdo', 'Fergusdo', 'D
```

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

Out[101…

```
[]
```

In [ ]:

```
# Update statement.
```

In [102…

```
%%sql
select birthYear, birthMonth, birthDay from people where people.playerID = 'willite
```

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

Out[102…

birthYear	birthMonth	birthDay
1918	8	30

In [103…

```
%%sql
update people
set birthYear = '1918',
    birthMonth = '8',
    birthDay = '30'
where playerID = 'fergusdo';
```

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

Out[103…

```
[]
```

In [104…

```
%%sql
select birthYear, birthMonth, birthDay from people where people.playerID = 'fergusd
```

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

Out[104…

birthYear	birthMonth	birthDay
1918	8	30

In [ ]:

```
# Select statement showing row.
```

In [107…

```
%%sql
select playerID, birthYear, birthMonth, birthDay, nameLast, nameFirst
    from people
        where playerID = 'fergusdo';
```

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

Out[107…

playerID	birthYear	birthMonth	birthDay	nameLast	nameFirst
fergusdo	1918	8	30	Donald	Fergusdo

In [ ]:

```
# Delete the created row.
```

In [108…

```
%%sql
DELETE from people where playerID = 'fergusdo';
```

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

Out[108…

```
[]
```

Question 10: There is No Question 10

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

In [ ]: