COMS W4111-003 (Fall 2022)
Introduction to Databases

Homework 2: Non-Programming and Non-Programming

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

Tan, Lili, lt2846

Setup Environment

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

In [2]:
```
%sql mysql+pymysql://root:dbuserbdbuser@localhost
```

In [3]:
```
%sql select * from COMS4111.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:

If there is no data existing, NULL values are assigned to it. NULL values do not represent spaces, blanks or a zero value. It is a distinct representation of missing information.

A NULL can be interpreted as one the following – data is missing, data is not known, or data is not applicable. NA can only represent data is not applicable but not datais missing or not known. For a specific query example, when we use `IS NULL` to detect NULL values in the table, it can not recognize string "NA" as NULL.

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:

INFORMATION_SCHEMA is where all the other databases' information is stored for MySQL.

name of a database or table, the data type of a column, integrity constraints

More Specific Example: TABLES(Table information); COLUMNS(Columns in each table); REFERENTIAL_CONSTRAINTS(Foreign key information); TABLE_CONSTRAINTS(Which tables have constraints)

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:

Indexes are used to locate data fast without having to look through every row in a database table each time a database table is queried.

The more indexes we create, the slower our inserts and deletes will go, and the more competition pages will have for precious memory space.

Creating unnecessary indexes reduce performance of writes and take up space.

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:

Indexes are used to locate data fast without having to look through every row in a database table each time a database table is queried.

The more indexes we create, the slower our inserts and deletes will go, and the more competition pages will have for precious memory space.

Creating unnecessary indexes reduce performance of writes and take up space.

W5

Question:

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

Answer:

Primary key will not accept NULL values whereas Unique key can accept NULL values.

W6

Question:

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

Answer:

1. Views can join and simplify multiple tables into a single virtual table.
2. Views take very little space to store.
3. Views enables us to hide the WHERE clause or other columns to which we do not want the user to have access.

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:

A domain is a set of values that a column can take in a database table. In other words, it is the set of all possible values that can be stored in a column.

CHECK(REGEXP_LIKE(course_number, '^[A-Z][1-9][0-9]{3}$'))

W8

Question:

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

Answer:

1. NOT NULL constraint 2. UNIQUE constraint
2. FOREIGN KEY constraint

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:

ENUM('M', 'T', 'W', 'R', 'F')

The column `day` is queried frequently OR A referential integrity constraint exists on the column `day`.

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 column `course_id` is the primary key of table `course`, but `course_id` composes two parts: abbreviation of department name AND course number.

We can seperate `course_id` column into two columns which one is abbreviation of department and one is course number. Then, we can assgin these two columns as our primary key of table `course`.

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 COMS4111;

WITH dept_total_num AS (
    SELECT
        dept_name,
        COUNT(title) AS no_of_courses
    FROM section
    LEFT JOIN
```

```
        course
        ON section.course_id = course.course_id
    GROUP BY dept_name
    ORDER BY dept_name
)

SELECT
    department.dept_name,
    no_of_courses,
    budget,
    ROUND(budget/no_of_courses, 1) AS cost_per_course
FROM dept_total_num
INNER JOIN
    department
    ON dept_total_num.dept_name = department.dept_name
```

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

Out[4]:

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

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

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 [5]:

```
%%sql

USE lahmansbaseballdb;

SELECT
    CONCAT(LEFT(nameFirst, 1), '.') AS initial,
```

```
    nameLast,
    CONCAT(birthState, ', ', birthCountry) AS place_of_birth
FROM people
LIMIT 10;
```

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

Out[5]:

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 [6]:

```
%%sql

USE lahmansbaseballdb;

SELECT
    people.playerID,
    nameLast,
    nameFirst,
    GROUP_CONCAT(DISTINCT teamID SEPARATOR ';') AS career_teams,
    SUM(G_all) AS career_games,
    total_abs,
    total_hits,
    batting_avg
FROM people

INNER JOIN (SELECT
```

```
                playerID,
                SUM(AB) AS total_abs,
                SUM(h) AS total_hits,
                IF(SUM(AB)=0, NULL, SUM(h)/SUM(AB)) AS batting_avg
            FROM batting
            GROUP BY playerID) AS b
    ON b.playerID = people.playerID

LEFT JOIN appearances
    ON people.playerID = appearances.playerID
GROUP BY playerID
LIMIT 10;
```

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

Out[6]:

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 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 [7]:

```
%%sql

SELECT
    people.playerID,
    nameLast,
    nameFirst,
    GROUP_CONCAT(DISTINCT teamID SEPARATOR ';') AS career_teams,
    SUM(G_all) AS career_games,
    total_abs,
    total_hits,
    batting_avg
FROM people

INNER JOIN (SELECT
                playerID,
                SUM(AB) AS total_abs,
                SUM(h) AS total_hits,
                IF(SUM(AB)=0, NULL, SUM(h)/SUM(AB)) AS batting_avg
            FROM batting
```

```
            GROUP BY playerID) AS b
    ON b.playerID = people.playerID

LEFT JOIN appearances
    ON people.playerID = appearances.playerID
GROUP BY playerID
HAVING batting_avg IS NULL
LIMIT 10;
```

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

Out[7]:

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 [8]:
```
%%sql

WITH hp_combine AS(
    SELECT
        halloffame.playerID,
        nameLast,
        nameFirst,
        category
    FROM halloffame
    LEFT JOIN people
        ON halloffame.playerID = people.playerID
)

SELECT *
FROM hp_combine
WHERE playerID NOT IN
    (SELECT playerID
     FROM appearances
     UNION
     SELECT playerID
     FROM managers
    )
```

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

Out[8]:

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
harriwi99	Harridge	Will	Pioneer/Executive

leonabu99	Leonard	Buck	Player
evansbi99	Evans	Billy	Umpire
bellco99	Bell	Cool Papa	Player
johnsju99	Johnson	Judy	Player
charlos99	Charleston	Oscar	Player
hubbaca99	Hubbard	Cal	Umpire
dihigma99	Dihigo	Martin	Player
lloydpo99	Lloyd	Pop	Player
macphla99	MacPhail	Larry	Pioneer/Executive
gileswa99	Giles	Warren	Pioneer/Executive
yawketo99	Yawkey	Tom	Pioneer/Executive
fosteru99	Foster	Rube	Manager
chandha99	Chandler	Happy	Pioneer/Executive
dandrra99	Dandridge	Ray	Player
barlial99	Barlick	Al	Umpire
veeckbi99	Veeck	Bill	Pioneer/Executive
mcgowbi99	McGowan	Bill	Umpire
dayle99	Day	Leon	Player
hulbewi99	Hulbert	William	Pioneer/Executive
fostebi99	Foster	Bill	Player
wellswi99	Wells	Willie	Player
macphle99	MacPhail	Lee	Pioneer/Executive
roganbu99	Rogan	Bullet	Player
chylane99	Chylak	Nestor	Umpire
willijo99	Williams	Smokey Joe	Player
steartu99	Stearnes	Turkey	Player
smithhi99	Smith	Hilton	Player

brownra99	Brown	Ray	Player
coopean99	Cooper	Andy	Player
grantfr99	Grant	Frank	Player
hillpe99	Hill	Pete	Player
mackebi99	Mackey	Biz	Player
manleef99	Manley	Effa	Pioneer/Executive
mendejo99	Mendez	Jose	Player
pompeal99	Pompez	Alex	Pioneer/Executive
poseycu99	Posey	Cum	Pioneer/Executive
santolo99	Santop	Louis	Player
suttlmu99	Suttles	Mule	Player
taylobe99	Taylor	Ben	Player
torricr99	Torriente	Cristobal	Player
whiteso99	White	Sol	Pioneer/Executive
wilkijl99	Wilkinson	J. L.	Pioneer/Executive
wilsoju99	Wilson	Jud	Player
dreyfba99	Dreyfuss	Barney	Pioneer/Executive
kuhnbo99	Kuhn	Bowie	Pioneer/Executive
omallwa99	O'Malley	Walter	Pioneer/Executive
harvedo99	Harvey	Doug	Umpire
gillipa99	Gillick	Pat	Pioneer/Executive
ruperja99	Ruppert	Jacob	Pioneer/Executive
seligbu99	Selig	Bud	Pioneer/Executive
schurjo99	Schuerholz	John	Pioneer/Executive