Homework 2: Database Applications

(10 points) Donald Ferguson

Fall 2019 COMS W4111

Due 10/18/19 at 11:59pm

Please post all clarification questions about this homework on the class Piazza under the “hw2” folder. You may post your question privately to the instructors if you wish.

This is an individual assignment. Although you may discuss the questions with other students, you should not discuss the answers with other students. All code and written answers must be entirely your own.

Late policy: You may use late days on this assignment. However, you MUST include as a comment in your submission on courseworks how many late days you are using. If you don’t tell us, or have used all your late days, 10% per late day will be deducted from the homework grade.

This is a long assignment that builds conceptually. Please start it early. There will be less hand-holding than with homework 1.

1 General Instructions

Please see download the lahman2019raw.sql file from github here

Please pull the template from github here

This homework has the following objectives:

Develop initial experience with database centric web applications, specifically microservices that surface data through a REST API.

- Understand and apply advanced SQL query capabilities, e.g.
  - Joins.
  - Subqueries.
  - Group By.
  - Views.
  - Accessing relational database metadata.

The use of REST APIs to extend the web from simple page viewing to a web of links, data, and APIs, and understanding how to surface data sources and applications via the web.

The lectures provided conceptual overviews of these concepts. The homework is intended to reinforce the concepts covered in lecture.

Solving the HW requires:

- A written portion that should be submitted as a pdf
- A SQL portion that should be completed in mySQL workbench and saved as a .sql file
- Implementing a RESTful API in python to understand how to surface data from the web. Development should be done in Pycharm.

It is in your best interest to begin with the written, then the SQL, then the programming portion as the concepts build on one another. As always you must thoroughly test your code to receive full credit on this assignment. You will include the output of tests in a .txt file and/or submit screenshots of output.

2 Written Homework

The following must be *typed* and submitted under Written Homework 2 on Courseworks. They are the kinds of questions that might be asked at a job interview about the topics we have recently covered in lecture. Most of it can be found through googling, but try to explain it in your own words.

[hint] Look at the links in the homework description above if you're stuck

1. What is the difference between a view and a table in sql?
2. How does a GROUP BY statement in SQL work?
3. What is HTTP and how do HTTP requests work?
4. What are the characteristics of a RESTful API?
5. Given the two tables 'Birth Info' and 'Astrological Info', what is the result of the following queries in table form.

Table 1: Birth_Info

name	**birth_month**
Don	September

Meghna	June
Aly	January
Ara	September
Kirit	May

Table 2: Astrological_Info

month	**astrological_sign**
January	Aquarius
September	Virgo
June	Gemini

a. SELECT name, birth_month, astrological sign FROM birth_info LEFT JOIN astrological_info ON birth_info.birth_month = astrological_info.month

b. SELECT name, birth_month, astrological sign FROM birth_info OUTER JOIN astrological_info ON birth_info.birth_month = astrological_info.month

c. SELECT name, birth_month, astrological sign FROM birth_info INNER JOIN astrological_info ON birth_info.birth_month = astrological_info.month

d. SELECT name, birth_month, astrological sign FROM birth_info RIGHT JOIN astrological_info ON birth_info.birth_month = astrological_info.month

6. What do these HTTP response codes mean:

Code	**Meaning**
200	OK
201	
400	

401	
300	
403	
404	
500	

3 SQL Queries

3.1 Understanding the Schema

This portion of the homework assignment is based on the Lahman baseball statistics database. As you know from the first homework assignment, The database contains pitching, hitting, and fielding statistics for Major League Baseball from 1871 through 2019. It includes data from the two current leagues (American and National), four other "major" leagues (American Association, Union Association, Players League, and Federal League), and the National Association of 1871-1875.

The database is comprised of the following main tables:

`People - Player names, date of birth (DOB), and biographical info`

`Batting - batting statistics`

`Pitching - pitching statistics`

`Fielding - fielding statistics`

It is supplemented by these tables:

`AllStarFull - All-Star appearance`

`HallofFame - Hall of Fame voting data`

`Managers - managerial statistics`

`Teams - yearly stats and standings`

`BattingPost - post-season batting statistics`

`PitchingPost - post-season pitching statistics`

`TeamFranchises - franchise information`

`FieldingOF - outfield position data`

`FieldingPost- post-season fielding data`

`ManagersHalf - split season data for managers`

`TeamsHalf - split season data for teams`

`Salaries - player salary data`

`SeriesPost - post-season series information`

`AwardsManagers - awards won by managers`

`AwardsPlayers - awards won by players`

`AwardsShareManagers - award voting for manager awards`

`AwardsSharePlayers - award voting for player awards`

`Appearances - details on the positions a player appeared at`

`Schools - list of colleges that players attended`

`CollegePlaying - list of players and the colleges they attended`

For more detailed information, see the docs online.

Importing an entire schema from a dump file (instead of one csv at a time) will be covered in lecture on Friday 10/4. The dump file can be found on github. To put it simply, once mySQL workbench is open, navigate to the administration tab. Click data import on the left, and then navigate to the self-contained dump file and import. You can drop the original Lahman2019 database we used for homework 1.

3.2 Writing Queries

There are 6 queries in the homework assignment. They are intended to become progressively more difficult. Focus on understanding how to:

a. Select
b. Insert

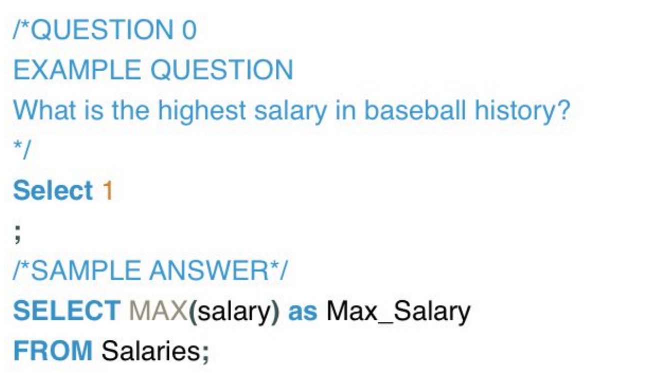
c. Update
d. Delete
e. Limit
f. Offset
g. Join
h. Create views
i. Order by

...in SQL. These concepts are very foundational and essential to doing well in industry. Pay special attention to the query that necessitates a *compound key*. A compound key is composed of two independent columns, such as birthDay and birthMonth and joining them together with an underscore. For example, “23” and “January” would become “23_January”. These are just example columns and not present in the lahman2019 schema.

We have created a skeleton file, hw2.sql, to help you get started. In this file, you will find a line of code under each question. Your job is to fill out the section of code in a way that returns what the question requires.

For example, consider Question 0:
"What is the highest salary recorded in baseball history?"

In the file we would provide the following, and you would be responsible for the sample answer portion:



```
/*QUESTION 0
EXAMPLE QUESTION
What is the highest salary in baseball history?
*/
Select 1
;
/*SAMPLE ANSWER*/
SELECT MAX(salary) as Max_Salary
FROM Salaries;
```

Some questions require creating a view or table. Here is a SQL reference: https://www.w3schools.com/sql/sql_ref_keywords.asp Feel free to google and use stack overflow as well!

4 RESTful API (Programming)

4.1 Required

The questions up until now were intended to prepare you for this portion. Students should start by pulling the project template from github for HW2. We will break down the contents of each file in a moment.

The conceptual breakdown of the homework is as follows:

4.1.1 Using Flask

The majority of the code is provided, essentially when you click "run" in pycharm, flask connects to a local host where you can send HTTP requests. The methods we write are intended to decipher the string and send back results. For further understanding, click here.

4.1.2 Using Postman to connect to the local host

For this project, we will use postman to test our code. Download it here. Think about postman as a browser that will use the local host address that flask opens for a user to interact with our database.

4.1.3 Adding methods to data_table_adaptor.py and RDBDataTable.py

Further instructions in the files.

4.1.4 Default @app.routes to show how paths come in and are handled in python in app.py

All of the following code should occur in app.py. You should process GET, POST, PUT, and DELETE requests by converting the input string into a where clause, sending it to SQL to obtain the response, and then returning the result on postman. The responses should be in JSON format.

1. Path that lists the databases, 'GET' a list of the databases in your sql environment
   `@application.route("/api/databases", methods=["GET"])`

   Example: GET on the path `'api/databases'`
   Response: ['lahman2019','sys']

2. Path that lists the tables in a database, 'GET' a list of the tables in a database
   `@application.route("/api/databases/<dbname>", methods=["GET"])`

Example: GET on the path ‘`api/databases`’
Response: [‘People’,’Batting’,....*]

Note that in this example not all tables are included for brevity but you would be expected to return all tables.

3. Path using primary key, ‘Get, update or delete’ a resource based on primary key.
@app.route(**'/api/<dbname>/<resource>/<primary_key>'**, methods=[**'GET'**, **'PUT'**, **'DELETE'**])

   Example: GET on the path ‘`api/lahman2019/People/willite01`’

   Should have a SQL query that looks like:

   SELECT * FROM lahman2019.People WHERE playerid = ‘willite01’

   And would have a response of the following format:

   [{"playerID": "willite01", “birthYear”:”1918”, “birthDay”:”30”, ”birthCountry”:”USA”, ”birthState”:”CA”, ”birthCity”:”San Diego”, … }]

   This is a PRIMARY key and therefore should always return one row of data. Note that in this example not all fields are included for brevity but you would be expected to return all columns.

   DELETE and PUT should all perform their appropriate action and return the proper message. DELETE would indicate how many rows were successfully deleted.

   PUT should support partial features, updating a part of the entry.

   **This path should also support fields*

   @app.route(**'/api/<dbname>/<resource>/<primary_key>?fields=f1,f2,f3'**, methods=[**'GET'**]])

   Example 2: GET on ‘api/lahman2019/People/willite01?fields=nameLast’

   Should have a SQL query that looks like:

   SELECT nameLast FROM lahman2019.People WHERE playerid = ‘willite01’

And would have a response of the following format:

[{“nameLast”:”Williams”}]

4. Select or insert for a table. Select should support query parameters and fields for selections

   ```
   @app.route('/api/<dbname>/<table_name>', methods=['GET', 'POST'])
   ```

   Example: POST on ‘/api/lahman2019/People/

   ..in the body of postman pass a dictionary of values (type them directly):

   {‘nameLast’ : ‘Hudson’,

   ‘nameFirst’: ‘Aly’,

   ‘PlayerID’: ‘alh2202’}

   Should have a SQL query that looks like:

   INSERT INTO lahman2019.People (‘nameLast’, ‘nameFirst’, ‘PlayerID’)

   VALUES (‘Hudson’, ‘Aly’,’alh2202’);

   And would have a response of the following format:

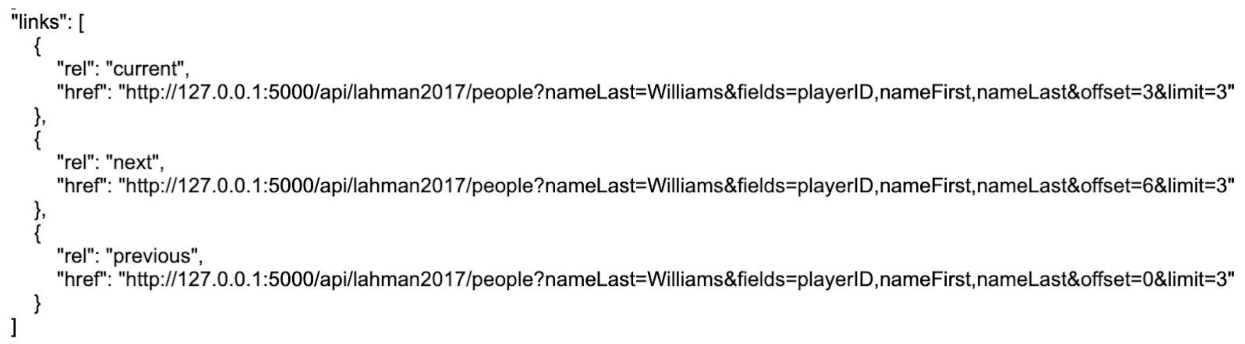
   “HTTP: 200 Entry successfully inserted” (or failure with respective code)

Use the methods for hw1 to our advantage. A clean solution code for the necessary hw1 methods is included in the hw2 template on github. Don’t forget to test all of your code in the test directory.

4.2 Extra Credit - Links and Pagination

Limit the number of responses returned on a single get request. Add a dictionary entry to all of the above requests called “links” that will navigate to the current, next, and previous page (think online shopping!)

[HINT] limit can be parsed like fields



```
"links": [
    {
        "rel": "current",
        "href": "http://127.0.0.1:5000/api/lahman2017/people?nameLast=Williams&fields=playerID,nameFirst,nameLast&offset=3&limit=3"
    },
    {
        "rel": "next",
        "href": "http://127.0.0.1:5000/api/lahman2017/people?nameLast=Williams&fields=playerID,nameFirst,nameLast&offset=6&limit=3"
    },
    {
        "rel": "previous",
        "href": "http://127.0.0.1:5000/api/lahman2017/people?nameLast=Williams&fields=playerID,nameFirst,nameLast&offset=0&limit=3"
    }
]
```

5 Submission

Submission format is in 3 parts, all on courseworks

- Written
    - pdf file named ‘[uni]_HW2.pdf’
- SQL
    - .sql file named ‘[uni]_HW2.sql’
- Code
    - A zip file named ‘[uni]_HW2F19’ with the HW2 Template with fully functional code, thorough testing, a README, and output. Either .txt or screenshots.
    -

- We will use Piazza for clarifications and discussions.