CS 251 Outlab9 - Web Scraping, DB and Networking

Instructions :

- Do this lab only in cs251 docker container. All the dependencies needed for this lab are already installed.
- Follow submission and output format strictly otherwise 0 marks will be awarded.
- Untar the resources (**outlab9_resources.tar.gz**) using
  tar -xvf outlab9_resources.tar.gz
- Any changes made to PS will be marked in this color.
- Command to be run on the terminal are marked in this **color** and the output to be printed on terminal is marked in this color.
- Use piazza to clarify your doubts.

Q1. Moodle Announcements Scraper [18 points]

(python, bs4, requests, html5lib)

you to keep track of the announcements being made on moodle.

Create a python program **q1.py** which when run like
$ python3 q1.py "<ldap_username>" "<ldap_password>" "<TA_name>"
does the following :

- Establishes a login session in moodle using the **<ldap_username>** and **<ldap_password>**. Use your own credentials for testing. This is probably the hardest part of this question. Stackoverflow will help you.
- Find the link for the course **CS 251-2022-1** on your start page. Consider the possibility of the user being registered to multiple CS 251 courses (like the TAs) but you have to fetch only **CS 251-2022-1** course . Also consider the possibility of the user to have selected any of Card, List or Summary views. You can assume that you'll find this course for sure with the **In progress** filter which will also be the default filter for the user.
- In the course page find the link for the **Announcements** forum.
- Iterate over all announcements and find the announcements made by **<TA_name>**. You have to scrape the subject of the topic (announcement) and the timestamp.
- Create a file **announcements.txt** in the same directory as your python program. Output **timestamp; subject** corresponding to <TA_name> in each line of the file sorted by timestamps (ascending order).

$ python3 q1.py "190050013" "password" "Ekatpure Rutuja Dattatray Dattatray"
Exact moodle username of the TA will be passed (notice the repetition of last name in this case). The output for this is in the **announcements2.txt** file of the resources.

A tester file **q1_tester.sh** is included in the resources. Pass your ldap username and password to this script for testing. **For security reasons your code should not exceed 80 lines.**

Q2. Remote Procedure Call [8 points]

(python, xmlrpc)

Reference : https://docs.python.org/3/library/xmlrpc.html

Module **magic.py** included in the resources implements the function **getMagicNumber(int)** which takes in a number and returns a randomized value.
You have to create two python programs **q2_client.py** and **q2_server.py**.
q2_server.py when run like `` `python3 q2_server.py` `` followed by **q2_client.py** when run like `` `python3 q2_client.py <num>` `` on separate terminals do the following :

- The client invokes the procedure **getMagicNumber(<num>)** with the command line argument **<num>** as an argument to the function on the server.

stop running the server. **kill()** should also be one of the registered functions on the server side which stops the server. After this the client also exits.

The secret recipe to compute magic numbers is known only to the server (which can import the magic module and use it). Client has no access to the magic module and computes **getMagicNumber()** by RPC to the server.
During testing you can assume **magic.py** to be in the same directory as **q2_server.py** and the file **q2_client.py** can be anywhere (does not have access to **magic.py**).
Run the server on **port 8080** of localhost. And **silence the logs** on the server side.
The final output of both client and server should be same number that lies between **<num>** and **<num>+20**
A testing script **q2_tester.py** is included in the resources with a basic test case. Server is always run before the client. Remember that **magic.py** can be changed while grading. Only function signature and that it prints the number before returning it is guaranteed to be the same.

Q3. IPL forever ! [19 points]

(python, sqlite3)

References :
https://pynative.com/python-sqlite/
https://www.javatpoint.com/sqlite-tutorial (keys, conditions, joins and aggregate functions)
https://www.sqlitetutorial.net/sqlite-subquery/ (being able to use subqueries is very essential for this outlab)

going on.

Task A [2 points]

Find the file **q3a.py** in the resources. It creates a SQLite DB file named **ipl.db.** Then creates tables as per schema (except BALL_BY_BALL) and loads data from csv files into the tables. You can assume that **q3a.py** and all files for following questions will be in the same directory as of the csv files.

q3a.py does not create the table **BALL_BY_BALL** (corresponding to **ball_by_ball.csv**). So your task is to add the create statement in the **# Begin .. # End** block of the file. Make sure to encode all the **primary key** and **foreign key** constraints into the table (refer to schema for more details). Bold fields are primary keys and underlined fields are foreign keys in the schema.pdf.
Submit **q3a.py** after the changes.
Now running the script using `**python3 q3a.py**` should create **ipl.db** with all the tables having data in them.
DO NOT proceed without doing this part. We will use the **ipl.db** created in this task for all further tasks in this lab.
We will inspect the state of **ipl.db** for evaluation of this part.

Task B [4 points]

average runs per match to exactly two decimal places and also format it to print 2 decimal places only. Note that ordering happens before rounding off. Output format is as follows (note that this is the not the final output and the output does not include any header)

```
Punjab Cricket Association IS Bindra Stadium,132.50
Punjab Cricket Association Stadium,Mohali,122.75
Maharashtra Cricket Association Stadium,120.00
Rajiv Gandhi International Stadium Uppal,118.50
```

q3a.py followed by **q3b.py** will be run for evaluating this part.

Task C [4 points]

Create a python script **q3c.py** which when executed finds the **player_id**, **name** and **total runs scored** by a player in the tournament and sorts them in **descending** order based on total runs (break ties by sorting in ascending order of player name). Print only the first 20 rows i.e. **top 20 run scorers** only.
Note : Extra runs don't count towards the batsmen's total.

Sample output (not the final output and does not include any header) :

```
139,LA Pomersbach,217

338,MC Juneja,197
```

Reference : https://stackoverflow.com/questions/14581259/using-case-with-aggregate-function-group-by-clauses (CASE statement inside aggregate function will help you in this question a lot)

Create a python script **q3d.py** which makes a new table named **POINTS_TABLE** which has first two columns similar to the current **TEAM** table (Note that in **POINTS_TABLE** you need not add any foreign key relationship and you can just copy **TEAM** table) but adds two more columns to this table namely **points** and **nrr**.

1. The **points** column is of type **INT** and should be initialized with value **0**.
2. The **nrr** column is of type **REAL** and should also be initialized with value **0**.

Now using the **MATCH** table update this **POINTS_TABLE** keeping in mind the following rules:

1. The team that **wins** a match gets 2 points and the losing team gets 0 points.
2. If the match is a **tie**, ignore the match_winner and award 1 point to each team. **nrr** doesn't change in this case.
3. If a match gets **abandoned** (given by **NULL** match winner in **MATCH**) then each team gets 1 point.
4. Net run rate is calculated as follows (this is not used in actual practice):

d. Note that a win from **null margin** is equivalent to 0 margin of runs and 0 wickets left

The final output should print the **POINTS_TABLE** in sorted order based on **points** and then on the basis of **nrr** in **descending** orders for both. The **POINTS_TABLE** can be stored without worrying about sorting it (will not be checked). Print nrr by rounding off to exactly 2 decimal places and also format it to print 2 decimal places.

Sample output (not the final output and does not include any header) :

```
12,Rising Pune Supergiants,10,3.45

5,Rajasthan Royals,7,0.60

2,Royal Challengers Bangalore,7,-7.00

13,Gujarat Lions,6,1.70
```

Grading will be done for this task by checking the output after running **q3a.py** & **q3d.py.**

Submission Instructions

The following is what your submission directory must look like. Please follow the submission format strictly. 0 marks will be awarded if the submission format is not correct.

tar -cvzf <roll_no>_outlab9.tar.gz <roll_no>_outlab9

```
<roll_no>_outlab9
├── q1.py
├── q2_client.py
├── q2_server.py
├── q3a.py
├── q3b.py
├── q3c.py
├── q3d.py
└── references.txt
```