**Web Application Revision Practice 2 [RVHS/2023/Prelim/P2/Task 4]**

Name your Jupyter Notebook as:

`Task4_<your name>_<centre number>_<index number>.ipynb`

In a parallel universe, ABC Secondary School wishes to digitalise the enrolment process for its Secondary 1 students. The school will need to create a suitable database to store records of student applicants' details, results, and guardian contacts for the process. The database will have three tables: a table to store data about the students who opt to the school as 1st choice; a table about the results obtained by the students and a table to store guardian email contact of the students. All fields cannot be left empty.

`guardians`:

- `guardian_id` – unique guardian identification, for example, `900001`
- `guardian_name` – name of guardian
- `guardian_email` – contact email of guardian.

`students`:

- `student_id` – unique student identification, for example, `10001`
- `stu_name` – name of student
- `gender` – gender of student; can only accept `male` or `female`
- `primary_sch` – primary school which the student is from
- `guardian_id` – the unique identification reference for guardian; different students may be taken care of by the same guardian.

`results`:

- `student_id` – unique student identification, for example, `10001`
- `subject` – subject name which student has taken
- `al` – achievement level of the student's subject between level 1 to 8; can only accept `1` to `8`.

For each of the sub-tasks 4.1 to 4.3, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

In [1] :

```
#Task 4.1
Program code
```

Output:

**Task 4.1**

Write a Python program that uses SQL code to create the database `ENROLMENT` with the three tables given. Define the primary and foreign keys for each table. Include the checks needed to constrain the values. [7]

**Task 4.2**

Sample data sets for the different tables are created for testing the database and the programs using it. The csv files `GUARDIAN.csv`, `STUDENT.csv` and `RESULT.csv` store the comma-separated values for each of the tables in the database.

Write a Python program to read in the data from each file and then store each item of the data in the correct place in the database. [6]

**Task 4.3**

The contact emails of guardians are subjected to frequent changes and thus need a program to facilitate updates.

Write a Python program that uses SQL code to update the contact email according to guardian_id. Run the program to update `guardian_id 900005` email contact to `see@see.com` [2]

Save your Jupyter Notebook for Task 4.

**Task 4.4**

The overall achievement level (AL) of a student is calculated by summing the achievement levels obtained for the 4 subjects by the student.

You can assume that the data sets are complete with all the students submitted having AL of all 4 subjects captured in the database.

Write a Python program and the necessary files to create a web application. The web application will return a HTML page to display a summary table with the following header and its associated information obtained and calculated from the enrolment database query using SQL code.

- `Primary School` – primary school name
- `No. of Students` – number of students that are from the primary school
- `Mean Achievement Level` – average overall achievement level obtained by the students

The table should look as follows:

| **Primary School** | **No. of Students** | **Mean Achievement Level** |
|---|---|---|
| Boon Day Park Primary | 4 | 9.25 |
| … | … | … |

The table will display the value rows in descending alphabetical order according to primary school name. Mean achievement level will be in 2 decimal places. The table shown on the webpage must have visible border.

Consider the following in the SQL for the query:

- multiply an `INTEGER` value by `1.0` to cast it into `REAL` value
- `ROUND(real number, 2)` to limit SQL `REAL` aggregated values to 2 decimal places.

Save your program code as
`Task4_4_<your name>_<centre number>_<index number>.py`

with any additional files/subfolders as needed in a folder named
`Task4_4_<your name>_<centre number>_<index number>`

Run the web application and save the returned page as
`Task4_4_<your name>_<centre number>_<index number>.html` [7]

**Task 4.5**

Students whose overall AL is lower than or equal to a cutoff point will be enrolled into the school.

In the web application on the same web page as Task 4.4, add a text input for entering the cutoff point with an appropriate label and a submit button.

On submitting the cutoff point, the web application will return a page with a table containing student information of student applicants who will enrol into the school. The table will have the following headers and the associated information for each student record – **student name**, **gender**, **primary school**, **overall AL**, **guardian name** and **guardian email**.

Update the Python program and create the necessary files such that the web application will return a HTML page to display the student information table obtained and calculated from the enrolment database query using **SQL code**.

The table will display the value rows in descending alphabetical order according to primary school name, followed by overall AL in ascending order. The table shown on the webpage must have visible border.

Run the web application with cutoff value of 11 and save the returned pages as:
`Task4_5_1_<your name>_<centre number>_<index number>.html`
`Task4_5_2_<your name>_<centre number>_<index number>.html` [8]