UQ College

CRICOS Provider No: 04039H | RTO: 32454 | www.uqcollege.uq.edu.au

INFORMATION TECHNOLOGY
ASSESSMENT 4: TERM 2 PROJECT

Program Name	Standard Foundation Program/Tertiary Preparation Program
Cohort	February 2024
Course Name	IT
Course Code	INFS1101E
Assessment Name	Assessment 4: Term 2 (Database-Python) Project
Assessment Weighting	15%
Assessment Instructions	See the following pages for full instructions.
Due	11.59pm on Friday 1of Week 20 (19th July 2024)

Statement of Original Authorship

Your work will be submitted through Turnitin (plagiarism detection software).
By submitting your assessment item, you declare that:

- this is your own original work, and that no part of this assignment has been copied from any other source or person except where due acknowledgement is made.
- no part of the work has been previously submitted for assessment in this or any other institution except where explicitly acknowledged.

I understand that should these statements be found to be false; I will be subject to disciplinary action in accordance with UQ College's academic integrity and misconduct policy and procedure which is available on the UQ College website.

Marking Summary

Task 1:	
Marks	/10
Task 2:	
Marks	/15
Task 3:	
Marks	/5
Task 4:	
Marks	/15
Task 5:	
Marks	/10
Task 6:	
Marks	/20
Task 7:	
Marks	/5
Task 8:	
Marks	/15
Task 9:	
Marks	/5
TOTAL MARKS:	**/100**

Database-Python Project: An application for school club registration

You are tasked with **building a computer application that handles club registration for school students**. This application is used by school administrators for registering students for some sessions offered by some clubs. There are several clubs that offer extracurricular activities. For example, Yoga club, Drama club, Music club, Sports club, Debate club, Dance club, and Outdoor club.

The club registration application is called **"Rego" and is written in Python.** The entry-point source code is stored in a file named "rego.py". Rego runs in Terminal.

You will also be **required to enter values in this registration system**. These values will be **fictitious**, meaning you will need to make them up. Each task mirrors what you have done in class (I.e. the Supermarket example, for which we build the Cashier application), and you can refer to your class notes to help guide you to do the tasks in this project/assessment successfully.

The submission instructions (including video presentation) are provided in the last section.

Task 1 (10/100 marks): Design a database for the Rego application

For description about entity and relationship sets as well as their attributes, refer to the description for Tasks 4, 5 and 6 about clubs, students, and registrations respectively.

The database design is represented as **ER and schema diagrams**, which are saved in a PDF file named "er_and_schema_diagrams.pdf". The database for the Rego application is called "regodb", which is created under the username **postgres** (whose password is **123**).

Task 2 (15/100 marks): Write SQL code to create all necessary tables for the regodb.

The code is saved in a file named "create_tables.sql". This code implements the database design in Task 1.

Task 3 (5/100 marks): Write Python code to display and handle menu selection

Once launched, Rego displays a set of **five activity menus**, and **asks for a menu selection** from the user. If a valid menu is selected, Rego will execute the corresponding procedure. After this procedure finishes, Rego will display the five activity menus again. This cycle (i.e., menu selection, menu execution, menu selection, and so on) is repeated until the user enters the number 0, by which Rego terminates.

Rego provides five menus for its user to select from. Each menu is represented by a positive integer, namely 1, 2, 3, 4, and 5. If a number outside the range of 0, 1, …, 5 is entered, then Rego will simply ignore it and display the menu message again, continuing the menu loop. Note that the five menus are explained in Tasks 4, 5, 6, 7 and 8.

Write code to implement this functionality in a file called "rego.py", which also serves as the entry-point for Rego (such that Rego can be launched by typing the command: "python rego.py").

Functionality breakdowns:

a) Menu message
b) Menu selection input
c) Menu selection
d) Menu looping
e) Termination by the number 0
f) Unknown menu number handling

Sample outputs:

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)?
```

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 1
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<
Enter the club ID? 0
BACK to the Menu!

########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 2
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 0
BACK to the Menu!

########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)?
########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 7
Unknown menu selection: BACK to the Menu!
```

```
########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
```

Task 4 (15/100 marks): Write Python and SQL code to accept then insert the records of 3 clubs into the regodb.

Once selected (**by entering number 1**), this selection asks for four types of information, namely

a) ID: This is a club identification (ID), represented as an **integer.**
b) Name: This is a club name, represented as a **string**.
c) Quantity: This refers to the number of sessions available in a club. One session can only be taken by one student. The quantity is represented as an **integer**.
d) Price: This refers to the price of each session (i.e., the unit price). This is represented as a **numerical value with two decimal places**. Assume there is a unit price per session for casual memberships.

After the above four kinds of information are entered, this club information is inserted into the corresponding table in the regodb. The information about the 3 clubs is fictitious.

This menu handles multiple club record additions. That is, by repeatedly asking the set of information, one after another in a loop. To terminate this input loop, the number 0 should be entered as ID.

Write code to implement this functionality for Menu 1 in a file called "club.py".

Functionality breakdowns:

a) Handling multiple club additions through the club input loop.
b) Club information inputs. This includes handling erroneous non-number inputs for Quantity and Price, for example: if Quantity is entered as "10x", which cannot be converted to a decimal number. In such a non-number input, the current entry is ignored (and discarded), and the club input loop continues.
c) Club record insertion into the corresponding table in the regodb.

Sample outputs:

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

################################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 1
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 1
Enter the club name? basketballclub
Enter the club quantity? 100
Enter the club unit price? 30.50
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 2
Enter the club name? footballclub
Enter the club quantity? 250
Enter the club unit price? 25.75
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 3
Enter the club name? errorquantity
Enter the club quantity? 10x
Invalid club quantity, skipping this club entry
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 4
Enter the club name? errorprice
Enter the club quantity? 10
Enter the club unit price? 50
Invalid club unit price, skipping this club entry
>>>>>>> Entering club info (To EXIT, enter ID= 0) <<<<<<<<<<
Enter the club ID? 0
BACK to the Menu!
```

Task 5 (10/100 marks): Write Python and SQL code to accept then insert the records of 5 students into the regodb.

This menu is similar to Menu number 1 (Task 4). Once selected (**by entering number 2**), this selection asks for four types of information, namely

a) ID: This is a student identification (ID), represented as an **integer**.
b) Name: This is a student name, represented as a **string**.
c) DOB: This refers to the date of birth (DOB) of a student. The DOB is represented as a **string** in the following format: yyyy/mm/dd, for example, "2021/12/31"
d) Suburb: This refers to the suburb where the student lives. The suburb is represented as a **string**.

After the above four kinds of information are entered, this student information is inserted into the corresponding table in the regodb. The information about 5 students is fictitious.

This menu handles multiple student record additions. That is, by repeatedly asking the set of information, one after another in a loop. To terminate this input loop, the number 0 should be entered as ID.

Write code to implement this functionality for Menu 2 in a file called "student.py".

Functionality breakdowns

a) Handling multiple student additions through the student input loop.
b) Student information inputs. This includes handling erroneous date format and values for DOB. For

example: if DOB is entered as “2000/100/100”, which does not adhere to the required format. In such a wrong format, the current student input is ignored (and discarded), and the student input loop continues.

c) Student record insertion into the corresponding table in the regodb.

Sample outputs

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 2
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 701
Enter the student name? alex
Enter the student DOB: dd/mm/yyyy? 01/01/2000
Enter the student suburb? toowong
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 702
Enter the student name? josh
Enter the student DOB: dd/mm/yyyy? 05/05/2005
Enter the student suburb? milton
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 703
Enter the student name? wrongdob
Enter the student DOB: dd/mm/yyyy? 2000/100/100
Invalid DOB format, skipping this student entry
>>>>>> Entering student info (To EXIT, enter ID= 0) <<<<<<<<
Enter the student ID? 0
BACK to the Menu!
```

Task 6 (20/100 marks): Write Python and SQL code to accept then insert registration records into the regodb

Once selected (**by entering number 3**), Menu 3 asks for three types of information, namely

a) sID: This refers to the student ID, represented as an **integer**.

b) cID: This refers to the club ID, represented as an **integer**.

c) Quantity: This refers to the number of sessions of this club that the student registers for. The quantity is represented as an **integer**.

This menu handles multiple registrations of one student. That is, by first asking the student ID once, then repeatedly asking about the club ID (cID) along with the Quantity, one after another in a loop. To terminate this input loop, the number 0 should be entered as cID.

After the above three kinds of information are entered, this registration information is inserted into the corresponding table in the regodb. Once registration has finished, Rego outputs a file named "receipt.txt” that contains the registration receipt.

The information about registration is fictitious (not real). You must make up registration records such that they contain the following facts:

a) There are 2 students who have not registered for any club.

b) There is 1 club that has no students (there is no student who registers for any session of this club).

Write code to implement the functionality for Menu 3 in a file called "registration.py".

Functionality breakdowns:

a) Handling multiple club registrations for one student through the registration input loop.
b) Registration information inputs. This includes handling non-existent student and club IDs, and erroneous non-number quantity. For example: if Quantity is entered as "5x", which cannot be converted to an integer. For such error cases, the current input is ignored (and discarded) then the registration loop continues.
c) Registration record insertion into the corresponding table in the regodb.
d) The quantity (number of sessions) updates whenever necessary in the corresponding table. For example, a club initially has 100 sessions available in the table. After one student registers for 10 sessions of that club, then the number of available sessions becomes 90 *(100 –10)* in the table.
e) Writing the registration receipt into a file called "receipt.txt" that contains the following information:
    i) School name
    ii) Student name
    iii) Registration date and time
    iv) A list of registered sessions along with their clubs and unit prices
    v) The total price of all registered sessions

Sample outputs:

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

##############################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 3
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 701
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 1
Enter the club quantity? 10
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 2
Enter the club quantity? 5
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 0
BACK to the Menu!
```

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

########################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 3
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 999
Your ID is not registered!
>>>> Entering a student ID (To EXIT, enter student ID=0) <<<
Enter your student ID? 702
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 555
Unknown club ID!
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 2
Enter the club quantity? hundred
Invalid club quantity, skipping this entry
>>>>>>> Entering an item (To EXIT, enter club ID=0) <<<<<<<
Enter the club ID? 0
BACK to the Menu!
```

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more receipt.txt
########## Receipt 2 ##############
School XYZ
20/11/2022 at 17:52:05

Item 1: swimmingclub
15 x 20.99 AUD =         314.84999999999997 AUD
Item 2: footballclub
25 x 25.75 AUD =         643.75 AUD
---------------------------------
TOTAL            958.5999999999999 AUD
######### Thank you, Sara #########
```

Note: On Windows, the command to display the content of a text file is "type" (instead of "more). Therefore, to display the receipt, we type in Terminal: **type receipt.txt**

Task 7 (5/100 marks): Write Python and SQL code to write the records into CSV files

Once selected (**by entering number 4**), this selection outputs three CSV files, namely

a) "clubs.csv": This file contains the current **records about clubs**. The club attributes (ID, name, quantity, price) become the CSV column names (written in the first line). Note that the quantity column contains the current number of available sessions.
b) "students.csv": This file contains the current **records about students**. The student attributes (ID, name, DOB, and suburb) become the CSV column names (written in the first line).
c) "registrations.csv": The file contains the current **record about registrations**. The registration attributes (for example, sID and cID-to-quantity-mapping) become the CSV column names (written in the first line).

Write code to implement this functionality for Menu 4 in a file called "writer.py".

Functionality breakdowns:

a) Writing the club records into a CSV file
b) Writing the student records into a CSV file
c) Writing the registration records into a CSV file

Sample outputs:

```
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ python rego.py

######################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 4
writing clubs.csv...
writing students.csv...
writing registrations.csv...

######################################################################
The Rego app: MENU
[1] Add clubs
[2] Add students
[3] Register
[4] Print CSV
[5] Interactive Analysis

Menu choice (To EXIT, enter 0)? 0
The Rego app is exiting... BYE!
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more students.csv
ID,name,dob,suburb
701,alex,01/01/2000,toowong
702,josh,05/05/2005,milton
703,sara,07/07/2007,stlucia
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more clubs.csv
ID,name,quantity,price
1,basketballclub,80,30.5
2,footballclub,215,25.75
3,swimmingclub,115,20.99
(uqcit) tor@l7480:~/uqcit/programming/project/rego-s1234567$ more registrations.csv
ID,sID,cID_qty_dict
1,701,{'1': 10, '2': 5}
2,703,{'2': 25, '3': 15}
3,702,{}
```

Task 8 (15/100 marks): Write Python and SQL code to implement the interactive analysis menu.

Once selected (**by entering number 5**), this selection provides an interactive analysis. It is interactive in that the user can enter a limited number of question labels in the prompt. Rego will then display the corresponding answers. The analysis is about students, clubs, and registrations.

Write code to implement this functionality for Menu 5 in a file called "analysis.py".

The interactive analysis (Menu 5) handles the following queries, which can be triggered by entering their alphabet labels (instead of their complete wording).

a) The IDs, names, DOBs, and suburbs of **students with no registration**.
b) The ID and names of **clubs with no students**.
c) The ID, name, DOB, and suburb of the **student with the biggest spending.**
d) The ID and name of clubs in **descending order based on the number** of registered sessions.
e) At least one of **your very own analysis questions**. Label your questions e, f, g, and so on.

Task 9 (5/100 marks): Extract (dump) the regodb database into a script file and an archive file.

The database dump should contain the latest snapshot of your regodb, which is expected to contain the complete records of clubs, students, and registrations as required.

The extraction should be carried out with the following two commands (each will output a file), where rego-s1234567 refers to your folder path (see the Submission Instructions on the last page).

a) For **script** file extraction
- **On Windows**: pg_dump -U postgres -f rego-s1234567\regodb_dump.sql regodb
- **On MacOS/Unix**: pg_dump -U postgres regodb > rego-s1234567/regodb_dump.sql

b) For **archive** file extraction
- **On Windows**: pg_dump -U postgres -Fc -f rego-s1234567\regodb_dump.archive regodb
- **On MacOS/Unix**: pg_dump -U postgres -Fc regodb > rego-s1234567/regodb_dump.archive

Submission instructions:

a) Create a folder called “rego-s1234567, where **s1234567 is your student number.**
b) Create a **presentation video to demonstrate all functionalities** of your Rego application. The video should be created using Zoom (or other video editing applications) and saved in a file called **video.mp4** in the folder “rego-s1234567” . If the video file size is relatively big, you may need to submit your video file with Kaltura by following the instruction guide in https://web.library.uq.edu.au/library-services/it/learnuq-blackboard-help/kaltura-media/submit-video-or-audio-assignment-kaltura
c) Put **all necessary files** in the folder “rego-s1234567”. There are **at least 15 files** as follows:
    1) er_and_schema_diagrams.pdf (Task 1)
    2) create_tables.sql (Task 2)
    3) rego.py (Task 3)
    4) club.py (Task 4)
    5) student.py (Task 5)
    6) registration.py (Task 6)
    7) receipt.txt (Task 6)
    8) writer.py (Task 7)
    9) clubs.csv (Task 7)
    10) students.csv (Task 7)
    11) registrations.csv (Task 7)
    12) analysis.py (Task 8)
    13) regodb_dump.sql (Task 9)
    14) regodb_dump.archive (Task 9)
    15) video.mp4 (if possible, otherwise submit with Kaltura, see Item b above)
d) Zip the folder “rego-s1234567” into a file called “**rego-s1234567.zip**”.
e) **Submit “rego-s1234567.zip”** through a dedicated submission portal in Blackboard.