Loughborough University

Coursework Assignment Specification
(100 % of the module assessment)
Submission Deadline: at 9am on 23rd January 2023

Attendance Tracking System

1 Description

1.1 Overview

This coursework is to develop a simple attendance tracking system for the senior tutor of a department. The tutor should be able to check whether students attend lectures, lab sessions and other activities of modules in the current semester.

The university system provides a csv file for each module. You are given a folder that contains some of those files. The table below shows the structure of an example file:

Student Id	**S1.W1 Monday (03-10-2022) 14:00 - 16:00 Lecture CC012**	**S1.W1 Thursday (06-10-2022) 09:00 - 11:00 Computer Lab N001...**	**S1.W1 Thursday (06-10-2022) 11:00 - 13:00 Computer Lab N001...**	**....**
9456	GPS	GPS	Ex	....
1456	GPS	GPS		....
2713	GPS	GPS	X	....
4075	GPS		GPS	....
6789	GPS	GPS		....
...	...	...	..	....

The table includes attendance for every student at every timetabled session for the module. A student record for each session can have a "GPS", "Ex", or "X" value. "GPS" value for presence, "X" value for absence and "Ex" value for authorised absence are used. Sessions that were excluded from attendance monitoring are included here in the record for completeness, but no check-in data are shown (i.e., blank field). Due to the vagaries of the timetabling system, you may see duplicate sessions, or more sessions than you expect, particularly at the start of a semester.

Please note that column headings of the table include the details of each session. For instance, the second column in the table above is a record for the lecture session in Room CC012 at 14:00 on Monday in Week 1 of Semester 1.

1.2 Store Cleaned Data

Your initial task is to write a Python program to create SQLite database tables for the tracking system. The program needs to clean the given attendance records, format them into the appropriate form and store them into the tables.
Save the program as **CW_Preprocessing.ipynb**.

The detail of the task:

- Read data from each CSV file and write them into one separate DataFrame (e.g., dfCOA122)
- Utilize the column headings of the DataFrame and store the details of each session into a new DataFrame (e.g., dfCOA122Sessions).
- Create a copy of the original DataFrames to keep cleaned data (e.g., dfCleanCOA122).
    - Rename columns of the DataFrames so that their names are short and each column has a unique numeric session id.
    - If the record of a student in the DataFrame has only null values, then delete their information in the DataFrame.
    - If any column of the DataFrame has only null values, then delete the whole column.
    - Replace each "Ex" value with null value in the DataFrame and replace "GPS" and "X" values with the Boolean True and False values respectively.
- Use SQLite to create a database and save it as CWDatabase.db.
- Programmatically create SQL tables in the database and store generated DataFrames into those tables. You could use a basic data model, which creates a database table for each DataFrame. Make sure that the tables have the correct data types.

1.3 Student Attendance

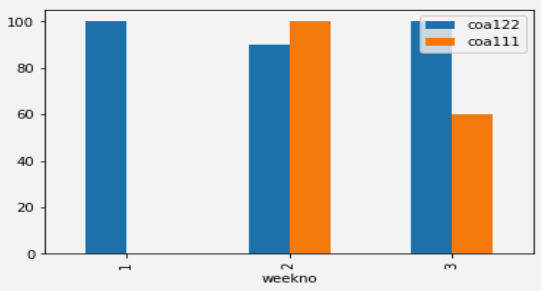
Write a Python program to find out the attendance record of a student. The program must use the attendance records in CWDatabase.db. Save the program as **Student_Att.ipynb**.

Given student ID, your program should produce a list which shows weekly attendance of both the modules, and appropriately visualises the list by using the Matplotlib module.

See an example output below:

The attendance record of Student ID 4523:

Week No	COA122 %	COA111 %
1	100	0
2	90	100
3	100	60



1.4 Module Attendance

Write a Python program to display the attendance of a module in a particular week. Given module code and week number, the program should produce a list which shows attendance of each session in the week. The attendance of each session should be colour coded to highlight poor and good attendance.

Following table is an example output to display

Attendance record for CO122 Week 1:

Time	Room	Type	Attendance %
14:00	CC012	Lecture	85.9
9:00	N001	Lab-1	20.7
11:00	N001	Lab-2	62.5

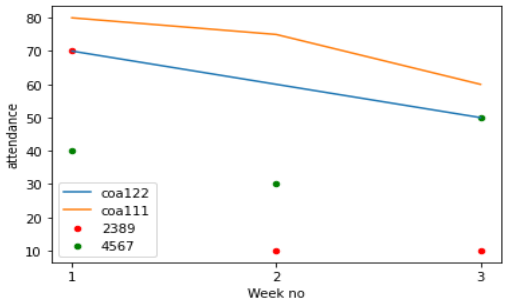
Please note the program must use the records in the database and save the program as **Module_Att.ipynb.**

1.5 Students with Poor Attendance

Write a Python program to use attendance records in the database and find out a list of students with poor attendance. You should come up with the details of the criteria to identify those students. For example, if the average attendance of a student in the current semester is much less than the department average, then student id should be included in the list.
Student IDs must be sorted by their average attendance. The list should include the weekly attendance. The program should highlight the attendance of a student in a week if it is higher than the department average. See the example below.

Student ID	Week1 %	Week2 %	Week3 %	Average %
2389	70	10	10	30
4567	40	30	50	40
…	…	…	…	…

Additionally, the program appropriately visualises the weekly attendance records of each module and student. See the example below.



Save the program as **Poor_Att.ipynb**.

2 What to Submit

Data files: stores all the data (See Section 1.2 for the explanation).

CWDatabase.db: stores attendance records of all students.

Attendance folder: contains csv files which are given on Learn. You are allowed to modify the data to test your program functionality.

Program files:

CW_Preprocessing.ipynb: contains functions to clean the given attendance records and store them into the SQLite tables as described in Section 1.2.

Student_Att.ipynb: contains functions used to ask for a student id the tutor wishes to monitor. Then, after performing the validity checks (i.e., correct ID), carry out the functionality as described in Section 1.3.

Module_Att.ipynb: contains functions used to ask for module code and week number the tutor wishes to display. Then, after performing the validity checks (i.e., acceptable code and valid number),carry out the functionality as described in Section 1.4.

Poor_Att.ipynb: contains functions used to allow the tutor to see attendance records of disengaged students as described in Section 1.5.

menu.ipynb: A python main program which provides the required menu options to the senior tutor for the program functionalities. The menu could be based on the **G**raphical **U**ser **I**nterface (e.g. tkinter module or Jupyter Widgets).
Ensure that all your results are presented in the Jupyter notebook and your codes are executable.

All the files (including sample data files) should be compressed into a zip file and submitted electronically as directed on the module Learn page.

3 Notes on Expectations

You will be marked according to your overall achievement, marked according to the Assessment Matrix that will be provided to you separately from this document. However below follows a qualitative description of some general expectation that may help you understand the general level of expectation associated with this piece of coursework.

Technical mastery of Python Your programs should show mastery of what you have been taught.
Design Your programs should be well structured for the task in hand so that it is as easy as possible for:

- a user to use the program for any likely purpose,
- a programmer to understand the code structure and be able to develop it further,
- a programmer to be able to re-use as much as possible of the code in a related application.

Clarity and Self-Documentation Given the structure of your programs, they should be as easy to read and understand as possible. *Lay your code out* so that it can be listed sensibly on a variety of devices: avoid having any lines longer than 80 characters as these may wrap (to reduce the number of "problem lines" you should use 4 spaces for indentation rather than tabs). *Sensible names* should be chosen for all variables, methods etc. *Documentation strings and/or markdown* cells should be included for each:

Program Fully explain what the program does and how it should be used. Also state who (ID only) wrote it and when.
Optionally, you can provide any special instructions or warnings to the user (or assessor!) or draw attention to any aspects of the program that you are particularly proud of (please don't waste time by writing an excessive amount; max 200 words).

Function State what each function does and explain the roles of its parameters.
In addition, you should include occasional comments in your code; these may be (a) to introduce a new section in the code, or (b) to explain something that is not obvious. Bear in mind that pointless comments make your code harder to read, not easier.

Structural requirements and Restriction

1) Your code must include Class Type wherever it is appropriate to improve reusability and maintainability.
2) Your code must store data into the SQLite Database
3) See a copy of a Conda environment (yml file) on the module Learn page. Submitted code must work in the environment which includes
    a. Python v10.0 or above.
    b. Python standard libraries.
    c. Matplotlib.
    d. Numpy.
    e. Grapviz.
    f. Pandas.