SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT233

End-of-Course Assessment - July Semester 2022

Data Programming

INSTRUCTIONS TO STUDENTS:

1. This End-of-Course Assessment paper comprises **EIGHT (8)** pages (including the cover page).

2. You are to include the following particulars in your submission: Course Code, Title of the ECA, SUSS PI No., Your Name, and Submission Date.

3. Late submission will be subjected to the marks deduction scheme. Please refer to the Student Handbook for details.

IMPORTANT NOTE

ECA Submission Deadline: Sunday, 13 November 2022 12:00 pm

ECA Submission Guidelines

Please follow the submission instructions stated below:

This ECA carries 70% of the course marks and is a compulsory component. It is to be done individually and not collaboratively with other students.

Submission

You are to submit the ECA assignment in exactly the same manner as your tutor-marked assignments (TMA), i.e. using Canvas. Submission in any other manner like hardcopy or any other means will not be accepted.

Electronic transmission is not immediate. It is possible that the network traffic may be particularly heavy on the cut-off date and connections to the system cannot be guaranteed. Hence, you are advised to submit your assignment the day before the cut-off date in order to make sure that the submission is accepted and in good time.

Once you have submitted your ECA assignment, the status is displayed on the computer screen. You will only receive a successful assignment submission message if you had applied for the e-mail notification option.

ECA Marks Deduction Scheme

Please note the following:

*a) Submission Cut-off Time – Unless otherwise advised, the cut-off time for ECA submission will be at **12:00 noon** on the day of the deadline. All submission timings will be based on the time recorded by Canvas.*

*b) Start Time for Deduction – Students are given a grace period of 12hours. Hence calculation of late submissions of ECAs will begin at **00:00 hrs** the following day (this applies even if it is a holiday or weekend) after the deadline.*

*c) How the Scheme Works – From 00:00 hrs the following day after the deadline, **10 marks** will be deducted for each **24-hour block**. Submissions that are subject to more than 50 marks deduction will be assigned **zero mark**. For examples on how the scheme works, please refer to Section 5.2 Para 1.7.3 of the Student Handbook.*

Any extra files, missing appendices or corrections received after the cut-off date will also not be considered in the grading of your ECA assignment.

Plagiarism and Collusion

Plagiarism and collusion are forms of cheating and are not acceptable in any form of a student's work, including this ECA assignment. You can avoid plagiarism by giving appropriate references when you use some other people's ideas, words or pictures (including diagrams). Refer to the American Psychological Association (APA) Manual if you need reminding about quoting and referencing. You can avoid collusion by ensuring that your submission is based on your own individual effort.

The electronic submission of your ECA assignment will be screened through a plagiarism detecting software. For more information about plagiarism and cheating, you should refer to the Student Handbook. SUSS takes a tough stance against plagiarism and collusion. Serious cases will normally result in the student being referred to SUSS's Student Disciplinary Group. For other cases, significant marking penalties or expulsion from the course will be imposed.

Answer all questions. (Total 100 marks).

Question 1

Objectives:

- Understand dataset with Data Scientist mindset.
- Exposure to real-world dataset analysis.
- Understand and design computation logic and routines in Python.
- Assess use of Pandas and Dataframes to perform extract, load, transformation and calculation operations.
- Assess the Design and use of Database method to perform create and load operations.
- Conduct visualization in an appropriate way.
- Structure code in appropriate methods (functions), looping and conditions.

There are 3 datasets:

1. meetings.csv
2. locations.csv
3. granted_permissions.csv

The above csv files contain the data of a simple meeting access control system, which indicates who has access to specific meetings based on granted locations.

meetings.csv

Column	Description
id	The meeting ID
location_id	• The location ID of the meeting venue • Foreignly reference a location in the *locations.csv*

locations.csv

Column	Description
id	The location ID of the meeting venue
name	The location name
access_path	The path from the root location to the location itself. Please refer to the detailed description below for **Figure 1**.

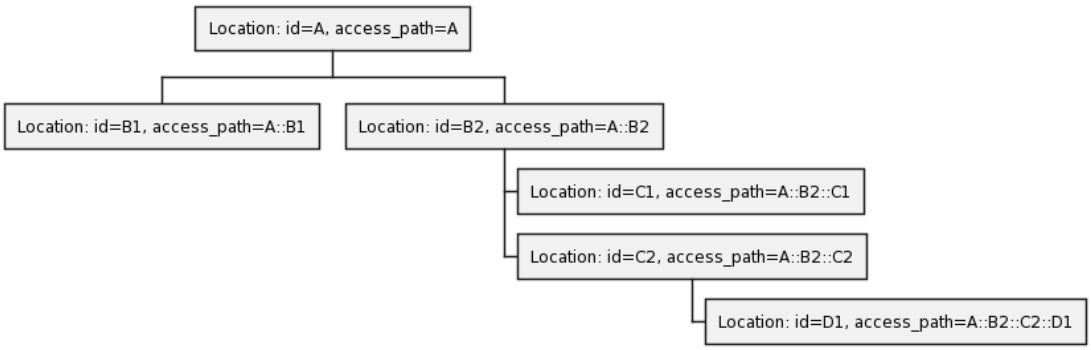

Figure Q1: Location diagram example

Locations are organized in a tree diagram as depicted in the above diagram (**Figure Q1**). The example diagram contains 6 locations. Considering the location with *id=C2*, its access_path is *A::B2::C2* which contains all *C2*'s ancestor IDs. In other words, an *access_path* of a location stores the path from the root location to the location itself.

granted_permissions.csv

Column	Description
id	The granted permission ID
user_id	Whom this permission is granted to
access_path	The *access_path* of the location where the permission is granted at for the *user_id*

Consider the *granted_permission* record *(id=1, user_id=user_1, access_path=A::B2)* and the above location diagram (**Figure Q1**), it means the *user_1* is **allowed** to access all meetings associated to the location: *id=B2* and all meetings associated to **ALL** *B2* location's **descendants**: *C1*, *C2* and *D1*.

Question 1a

Design the function namely ***get_location_ancestors*** which takes in a location ID as its input and returns the set of **ancestors** of the input location ID from the locations.csv. Call the function with the input location ID = location_1380 and display the result. Use **dataframe** and apply its operations to perform this task.

(8 marks)

Question 1b

Design the function namely ***get_location_descendants*** which takes in a location ID as its input and returns the set of **descendants** of the input location ID from the locations.csv. Call the function **THREE (3)** times (1 call for each location ID) and display the results (the return of the function and the number of records) for the following input locations:

1. location_32
2. location_216
3. location_1380

(10 marks)

Question 1c

(c) Following this reference (https://plotly.com/python/tree-plots/)visualize all locations with the tree layout.
Take note to replace *G = Graph.Tree(nr_vertices, 2)* with the following:
G = Graph()
G.add_vertices(...)
G.add_edges(...)

Refer to the "Create a graph from scratch" (https://igraph.org/python/tutorial/0.9.8/tutorial.html#creating-a-graph-from-scratch) section for the instructions of add_vertices and add_edges usages.

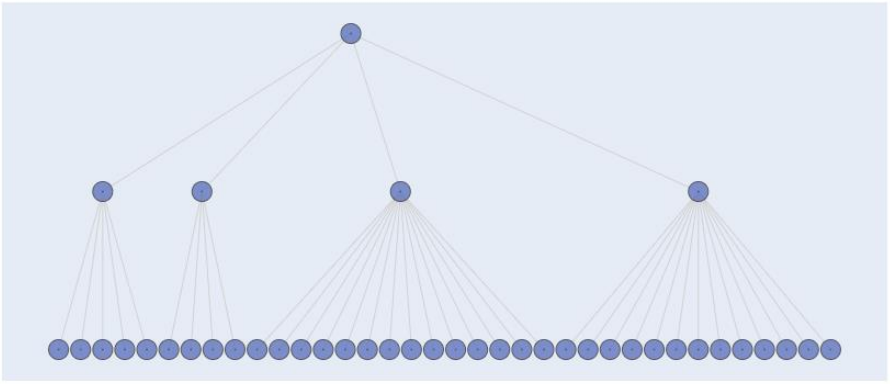

Figure 2: Tree graph

(10 marks)

Question 1d

A person who has the granted access permission at a location also has access permissions at all descendant locations of the direct granted location. Develop (construct) a new dataframe which contains all granted access permissions at the direct granted locations and their descendant locations for all user IDs granted in the *granted_permissions*.csv.

The output dataframe contains these fields:

id	The granted location ID
name	The granted location name
access_path	The access_path of the granted location
user_id	Whom the location is granted to

(10 marks)

Question 1e

The granted permission provides read access to meetings. If a person has the granted access permission at a location (outputted by the **Q1(d)** dataframe), the person could read meetings associated with the location. Design the function *read_meetings(user_id)* which returns the meeting IDs that the *user_id* could read. Call the function with *user_id=user_0000000009* and display the result.

(7 marks)

Question 1f

Apply and compose queries by using the **SQLite3** library to answer the questions:
(i) Q1(a)
(ii) Q1(b)
(iii) Q1(d)
(iv) Q1(e)

(15 marks)

Question 2

Objectives:

- Manipulate dataset with data scientist mindset.
- Exposure to real-world dataset analysis.
- Design computation logic and routines in Python.
- Structure code in appropriate methods (functions), looping and conditions.
- Design methods to extract and parse information from the internet.
- Assess use of Pandas and Dataframes to perform extract, load, transformation and calculation operations.
- Conduct visualization in an appropriate way.

Question 2a

Scrape and analyze the list of **operational** Singapore MRT stations from the URL: https://en.wikipedia.org/w/index.php?title=List_of_Singapore_MRT_stations&oldid=1094758210

Store the scrapped MRT stations into a **dataframe** with the following fields:

Field	**Description**
alpha_numeric_codes	The comma separated string of alphanumeric codes • Some stations have more than 1 code, e.g. Jurong East station has the value for this field as "*NS1, EW24*"
name	The English station name
opening	The opening date
abbreviation	The abbreviation of the station
mrt_line	The MRL line, for example, North South Line (NSL), etc.

Note that only operational stations are required to be scrapped. For example, stations with future opening dates or N/A alpha-numeric code(s) are excluded.

(15 marks)

Question 2b

On a **SINGLE** diagram, design and visualize the number of MRT stations per MRT line.

(5 marks)

Question 2c

Design and construct a new dataframe to store pairs of stations which are immediately next to each other. For example, Orchard and Somerset are immediately next to each other and this relationship is represented by two rows (Orchard, Somerset) and (Somerset, Orchard) in the dataframe.

The dataframe contains the following columns:

Column	**Description**
from	The abbreviation of the from station
to	The abbreviation of the to station
mrt_line	The MRL the edge belongs to

Note that some pairs can be manually added into the dataframe to capture the connection in the real world. However, majority of pairs can be extracted from the scrapped dataframe.

(10 marks)

Question 2d

Design a function *get_shortest_travel_path(from_station, to_station)* which returns one shortest traveling path (measuring by the number of stations) from the *from_station* to the *to_station*. *from_station* and *to_station* are abbreviations. Note that *from_station* and *to_station* can be from different MRT lines.

Apply the function by calling *get_shortest_travel_path('WDL', 'CGA')* and display the result. Note that there is a shortest traveling path between Woodlands and Changi Airport.

(10 marks)

----- END OF ECA PAPER -----