SFU SIMON FRASER UNIVERSITY

CMPT 354 Module 3 Assignment

Due: March 26, 2021 @ 11:59 PM

Weighting: 8%

1. Overview

The purpose of this assignment is to test your ability to use and apply SQL concepts to complete tasks in a real-world scenario. Specifically, this assessment will examine your ability to use SQL Data Manipulation Language to return specific subsets of information which exist in a database.

This assignment can be completed individually.

2. Submission

All submissions must be made through an electronic marking tool called Gradescope, which will also be used for providing feedback (enroll with the entry code M68KZM). You **must** record all your answers in the spaces provided in this document. Altering the format or layout of this document in anyway will attract penalties. You may however add landscape images in the submission boxes without changing the orientation of the page.

3. Marking

The Module 3 assignment counts for 8% of course mark.

4. Task

For this assignment you will be presented with the simplified schema of an event management application. The goal of the application is to track both the events attended by users and relationships between users and other users. The system is then able to use this data to effectively market recommended events to users based on the events their friends have attended. You will be required to write 10 SQL queries which answer higher level questions about the data in this database. (Note: Your queries must compile using a MySQL DBMS). A sample database of this system has been provided here which will allow you to test your queries.

Assignment Specification

Events Inc. is a small start-up company which provides its users with an event tracking and recommendation platform for various local community activities. A simplified version of their database schema has been provided below including foreign key constraints.

Relational Schema

User [id, fName, mInitial, lName, age, phone, email, nationality, significantOther]

Event [title, date, description, location, sponsor]

Attends [id, title, date, travelMethod]

Friends [requestor, requestee, startDate]

Foreign Keys

User.significantOther references User.id

Attends.{title, date} references Event.{title, date}

Attends.id references User.id

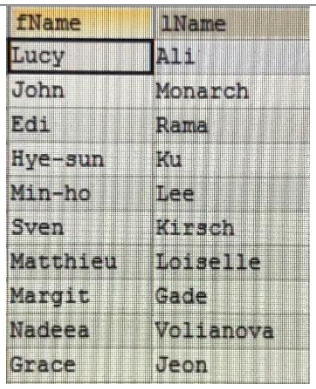
Friends.requestor references User.id

Friends.requestee references User.id

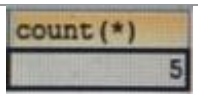
For this assignment you will be required to write SQL queries to answer to complete the following tasks. Please use the submission boxes provided to record your answers. For queries with a returning relation of more than 10 tuples, you can use the **LIMIT 10** clause to only capture the first 10 tuples of the table.

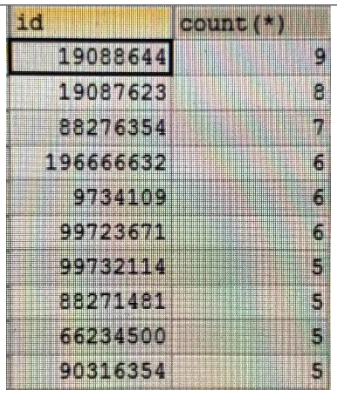
Example	
Task	Return the first name and last name of all users.
Explanation	This query should return a table with two columns, one for first name and one for last name.
SQL Solution	SELECT fName, lName FROM User LIMIT 10;
Output Screenshot	

fName	lName
Eduard	Khil
Mikhail	Mishustin
Lucy	Ali
John	Monarch
Ursula	Smith
Marcus	Jacobs
Nevena	Ivanovic
Leo	Montgomery
Edi	Rama
Jamie	Sleeman

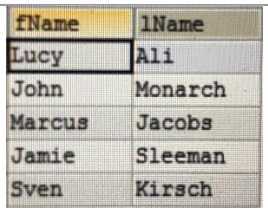
Query 1	
Task	Return the first name and last name of all users with an “@uq.edu.au” email address.
Explanation	This query should return a table with two columns, one for first name and one for last name.
SQL Solution	SELECT fName, lName FROM `user` WHERE email LIKE "%@uq.edu.au" LIMIT 10;
Output Screenshot	

fName	lName
Lucy	Ali
John	Monarch
Edi	Rama
Hye-sun	Ku
Min-ho	Lee
Sven	Kirsch
Matthieu	Loiselle
Margit	Gade
Nadeea	Volianova
Grace	Jeon

Query 2	
Task	Return the number of Korean users who are between 20 and 60 years old.
Explanation	This query should return a table with one column that has a single numerical tuple. The age condition is inclusive meaning Korean users who are 20 or 60 years old should also be included in the total.
SQL Solution	SELECT COUNT(*) FROM `user` WHERE `nationality` = "Korean" AND `age` BETWEEN 20 AND 60;
Output Screenshot	

Query 3	
Task	Return a list with the number of events each user has attended in descending order of the number of events.
Explanation	This query should return a table with two columns, one for user id and one for the number of events attended by that user. Users who have not attended any events can be ignored.
SQL Solution	SELECT id, COUNT(*) FROM `attends` GROUP BY id ORDER BY COUNT(*) DESC LIMIT 10;
Output Screenshot	

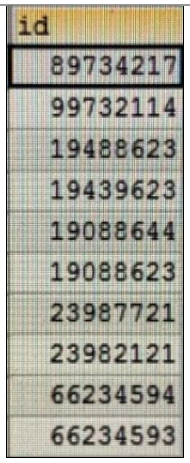
id	count(*)
19088644	9
19087623	8
88276354	7
196666632	6
9734109	6
99723671	6
99732114	5
88271481	5
66234500	5
90316354	5

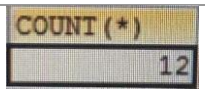
Query 4	
Task	Return the names of all users who have initiated (are the requester) of more than 10 friendships.
Explanation	This query should return a table with two columns, one for first name and one for last name.
SQL Solution	SELECT fName, lName FROM `user` WHERE id IN (SELECT `requestor` FROM `friends` GROUP BY `requestor` HAVING COUNT(`requestor`) > 10);
Output Screenshot	

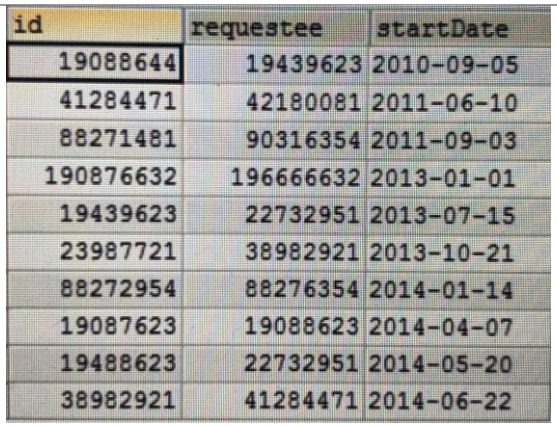
fName	lName
Lucy	Ali
John	Monarch
Marcus	Jacobs
Jamie	Sleeman
Sven	Kirsch

Query 5	
Task	Output the full name of each user along with the full name of their significant other. If the user does not have a significant other, those details should be left null.
Explanation	This query should return a table with four columns. The first two should be the first name and last name of a user and the next two should be the first name and last name of that user's significant other (or null if they do not have one).
SQL Solution	SELECT u1.`fName`, u1.`lName`, u2.`fName`, u2.`lName` FROM `user` u1 LEFT JOIN `user` u2 ON u1.`significantOther` = u2.`id` AND u1.`significantOther` IS NOT NULL LIMIT 10;
Output Screenshot	

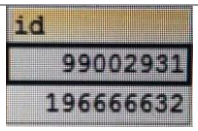
fName	lName	fName	lName
Eduard	Khil	Nadeea	Volianova
Mikhail	Mishustin	Sofia	Rotaru
Lucy	Ali	(NULL)	(NULL)
John	Monarch	(NULL)	(NULL)
Ursula	Smith	Leo	Montgomery
Marcus	Jacobs	Nevena	Ivanovic
Nevena	Ivanovic	Marcus	Jacobs
Leo	Montgomery	Ursula	Smith
Edi	Rama	(NULL)	(NULL)
Jamie	Sleeman	(NULL)	(NULL)

Query 6	
Task	Return a distinct list of users who either have a significant other or have attended 3 events by taking the Bus **Note: You must use UNION in your solution.**
Explanation	This query should return a table with a single column containing ids of users who meet either of the two conditions described above.
SQL Solution	SELECT id FROM `user` WHERE `significantOther` IS NOT NULL UNION SELECT id FROM `user` u WHERE(SELECT COUNT(*) FROM `attends` a WHERE u.id = a.id AND a.`travelMethod` = 'Bus') = 3 LIMIT 10;
Output Screenshot	 id 89734217 99732114 19488623 19439623 19088644 19088623 23987721 23982121 66234594 66234593

Query 7	
Task	Find the total number of users where the nationality of their significant other has at least three people. That is to say, the system has recorded at least three users of that nationality including the significant other.
Explanation	This query should return a table containing a single column which has a single numerical tuple.
SQL Solution	SELECT COUNT(*) FROM `user` u1, `user` u2 WHERE u1.`significantOther` = u2.`id` AND 3 <= (SELECT COUNT(*) FROM `user` u3 WHERE u1.`nationality` = u3.`nationality`);
Output Screenshot	

Query 8	
Task	Find the oldest friendship started (requested) by each user.
Explanation	This query should return a table containing three columns, the first being the id of a user, the second column being the id of that user's oldest requested friendship (i.e., the startdate of the requested friendship is the earliest) and the third column is the starting date of the friendship. Users without friends can be ignored.
SQL Solution	SELECT `requestor`, `requestee`, MIN(`startDate`) FROM `friends` GROUP BY `requestor` LIMIT 10;
Output Screenshot	

id	requestee	startDate
19088644	19439623	2010-09-05
41284471	42180081	2011-06-10
88271481	90316354	2011-09-03
190876632	196666632	2013-01-01
19439623	22732951	2013-07-15
23987721	38982921	2013-10-21
88272954	88276354	2014-01-14
19087623	19088623	2014-04-07
19488623	22732951	2014-05-20
38982921	41284471	2014-06-22

Query 9	
Task	Return a list of all the users who have attended at least all the events that “Grace Jeon” has.
Explanation	This query should return a table with a single column of user ids.
SQL Solution	SELECT id FROM `user` u0 WHERE NOT EXISTS (SELECT a.title, a.`date` FROM `attends` a INNER JOIN `user` u ON a.`id` = u.`id` AND u.`fName` = 'Grace' AND u.`lName` = 'Jeon' AND a.`title` NOT IN(SELECT title FROM `attends` a1 WHERE u0.`id` = a1.`id`) AND a.`date` NOT IN(SELECT `date` FROM `attends` a1 WHERE u0.`id` = a1.`id`));
Output Screenshot	

Query 10	
Task	Return the title and date of the event which had the most participants. **Note: You must use VIEW in your solution.**
Explanation	As above.
SQL Solution	CREATE VIEW participant AS SELECT e.`title`, e.`date` FROM `event` e, `attends` a WHERE e.`title` = a.`title`; SELECT * FROM participant GROUP BY `title`, `date` ORDER BY COUNT(*) DESC LIMIT 1;
Output Screenshot