Project 1: Culinary Tour

CmpE 321: Introduction to Database Systems, Spring 2022

Due: **April 4, Monday 23:59**

Please ask the project related questions in the Moodle forum!

1 Project Description

Everything starts with the foundation of the tire company by Michelin brothers in a small French town called Clermont-Ferrand in 1889. The number of cars in the country were less than 3000 at that time. They decided to publish a free small red guide which provides some useful information for travelers. In 1920, the guide was rebranded as MICHELIN Guide [1] with the price of seven francs. The new guide contained hotel list for Paris, restaurant lists for different categories and so on. Afterwards, restaurants were visited and reviewed anonymously by a special team in Michelin. Then, this process evolved into giving stars. At first, only one star was given. Two and three stars were added 5 years later. This ranking system survived until today. Here is the meaning of the stars.

- 1 star: High quality cooking, worth a stop!
- 2 star: Excellent cooking, worth a detour!
- 3 star: Exceptional cuisine, worth a special journey!

In our project, there are some gourmets who are willing to travel only to visit the 3 star Michelin restaurants around the world. A culinary tour organization firm wants to expand the business by bringing together the 3 star Michelin restaurants and the gourmets. While sharing some statistics and special lists is important for user engagement and advertisement, finding the restaurants according to the expectations of the users is a must. To this end, they have collected some data and created a database, but they need someone to manage these requirements by automatically retrieving the necessary information from the database. Now it's your turn! Write queries to fulfill the requirements!

[1] You can visit https://guide.michelin.com/th/en for further information.

2 Data

The culinary tour organizer company has a database that is useful for the organization. They present the database as an SQLite database file which is named as ***Culinary_Tour.db***. The database consists of 4 tables: Restaurant, Gourmet, Destination, Cuisine. Detailed information about the tables is provided below.

- **Restaurant**
  - Restaurant_ID INTEGER,
  - Name TEXT,
  - Chef TEXT,
  - Country_Region INTEGER,
  - Cuisine_Type INTEGER,
  - Awarded_Since TEXT,
  - Average_Price INTEGER,
  - PRIMARY KEY(Restaurant_ID),
  - FOREIGN KEY(Country_Region) REFERENCES Destination(Destination_ID),
  - FOREIGN KEY(Cuisine_Type) REFERENCES Cuisine(Cuisine_ID)
- **Gourmet**
  - Gourmet_ID INTEGER,
  - Favorite_Cuisine INTEGER,
  - Favorite_Restaurant INTEGER,
  - Travel_Destination INTEGER,
  - Average_Budget INTEGER,
  - PRIMARY KEY(Gourmet_ID),
  - FOREIGN KEY(Travel_Destination) REFERENCES Destination(Destination_ID),
  - FOREIGN KEY(Favorite_Cuisine) REFERENCES Cuisine(Cuisine_ID),
  - FOREIGN KEY(Favorite_Restaurant) REFERENCES Restaurant(Restaurant_ID)
- **Destination**
  - Destination_ID INTEGER,
  - Destination_Name TEXT,
  - PRIMARY KEY(Destination_ID)
- **Cuisine**
  - Cuisine_ID INTEGER,
  - Type TEXT,
  - PRIMARY KEY(Cuisine_ID)

3 Queries

1. Find the number of restaurants and display as *Restaurant_Count*.
2. List *Name*, *Chef*, *Awarded_Since* of the restaurants which are awarded before 2000 (exclusive). Display *Awarded_Since* in YYYY format. Sort the results by *Awarded_Since* in ascending order.
3. List all the fields of the restaurants with the minimum *Average_Price*.
4. List *Destination_Name* and *Type* of the restaurants with the maximum *Average_Price*.
5. Find the total price of going to all restaurants based on *Average_Price*.
6. List *Country_Region*, *Destination_Name* together with the number of restaurants (display as *Restaurant_Count*) in each *Country_Region*. Sort the results by *Restaurant_Count* in descending order.
7. List *Country_Region*, *Destination_Name* together with the number of distinct cuisine types (display as *Cuisine_Count*) in each *Country_Region*. Sort the results by *Cuisine_Count* in ascending order.
8. Find *Name*, *Chef*, *Destination_Name* of restaurants that meet the requirements of the gourmet with *Gourmet_ID* 1 regarding *Travel_Destination*, *Average_Budget* (*Average_Budget* should be greater than or equal to *Average_Price*) and *Favorite_Cuisine*.
9. Find *Name*, *Chef*, *Destination_Name* of restaurants that meet some requirements of the gourmet with *Gourmet_ID* 2. *Travel_Destination* match is a must. *Average_Budget* match (*Average_Budget* should be greater than or equal to *Average_Price*) or *Favorite_Cuisine* match is sufficient.
10. Insert gourmet whose *Favorite_Cuisine* is 15, *Favorite_Restaurant* is 119, *Travel_Destination* is 3, and *Average_Budget* is 260.
11. List *Name*, *Chef*, *Destination_Name* of the restaurants which are selected as at least one gourmet's *Favorite_Restaurant*.
12. Return all the fields of all restaurants and all the fields of gourmets with matching *Favorite_Restaurant*. If a restaurant is more than one gourmets' favorite, there should be multiple rows for that restaurant. If a restaurant is not a favorite restaurant for any gourmet, the column values related to gourmet should be NULL.
13. List all the fields of the cuisines together with the number of restaurants in this *Cuisine_Type* as *Restaurant_Count* and the number of distinct *Country_Region* for this *Cuisine_Type* as *Distinct_Country_Region_Count*. Sort by *Type* in ascending order.

14. List *Destination_Name* and the number of gourmets in each *Destination_Name* as *Gourmet_Count*. Include only travel destinations which are in more than 1 gourmet's plans.

15. List *Gourmet_ID* and *Feasibility* which displays TRUE if there is at least one restaurant that meets the requirements (*Travel_Destination*, *Favorite_Cuisine*, and *Average_Budget*) of the gourmet, and displays FALSE otherwise.

The output of each query is given in ***output<query_index>.txt*** (e.g. *output1.txt, output2.txt,...,output15.txt*).

4 Submission Details

- This project can be implemented either individually or as a team of two people. You are free to change teams in the upcoming projects.
- Write each query to a file named ***q<query_index>.sql*** (e.g. *q1.sql, q2.sql,...,q15.sql*).
- Include a comment in each file that explains the reasoning that led you to the query in detail.
- The .sql files that do not contain explanatory comments will receive 0.
- Put all .sql files into a folder.
- Name the folder as <***StudentId1***>_<***StudentId2***> if you are working as a group.
- Name the folder as <***StudentId1***> if you are working as an individual.
- Zip the folder for submission and name the .zip file with same name given to the folder.
- Submit .zip file through **Moodle** until the deadline.
- Each group should submit **one** .zip file.
- Do not include any other files to your .zip file.
- **Any other submission method and late submissions are not allowed.**
- 10 points will be deducted in case of non-compliance to any of the naming and folder conventions explained above.
- Do not inject your observations into your queries to skip some essential steps! Do not obtain values manually from the database and inject into the queries! This kind of queries will receive 0. There will be additional sanctions too since violating this condition will be treated as a dishonest behaviour.