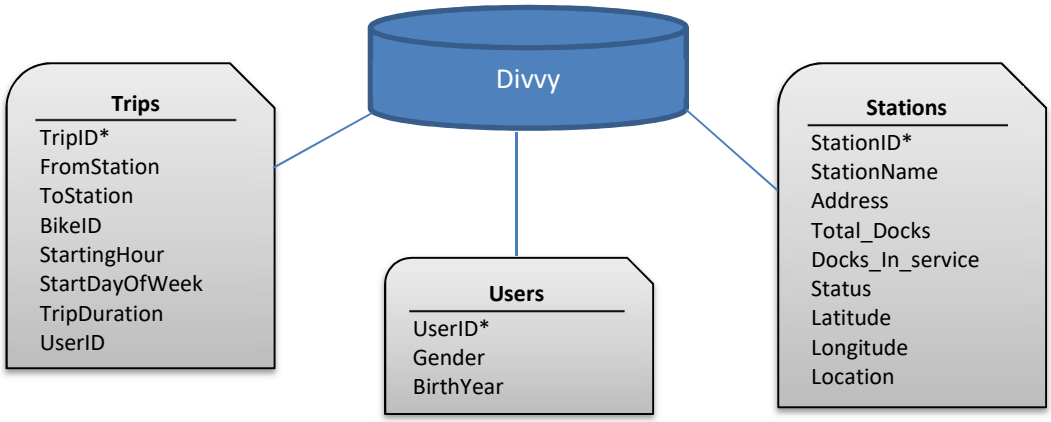
Project #05: Divvy Database App

Complete By: Thursday November 21st @ noon
Assignment: C# console application using ADO.NET
Policy: Individual work only, late work *is* accepted (up to 24 hours late for a penalty of 10%)
Submission: via Gradescope

Overview

The project is to develop a console application using C# and ADO.NET that retrieves data from a Divvy database residing in Azure:



Use the same login credential as the crimes database:

server: uiccs341.database.windows.net
authentication type: SQL Login
user: student
pwd: cs341!uic

Much like HW #11, you'll be putting the SQL queries to retrieve various data from the database into separate .sql files. In this case you'll be retrieving data about Divvy trips much like you did against the csv file of trips in Project 4.

Assignment

For each of the following problems, write a query which retrieves the data containing the answer in Project05##.sql, where ## is the number of the question, such as 01.

1. Retrieve the number of trips recorded in the database as NumTrips.

2. Retrieve the number of unique bikeIDs as NumBikes.

3. For each BikeID, retrieve the bikeID, number of trips taken with that bike as NumTrips and the total amount of time that bike was checked out as TimeCheckedOut. Restrict the results to the top 10 by total time checked out in order.

4. For each StationID, retrieve the ID, number of trips taken from that station as NumTripsFrom and to that station as NumTripsTo (as separate values). Order the results in descending order by the total number of trips both from and to the station.

5. For each customer list the number of trips they have taken. Include the customer ID and the number of trips as NumTrips, ordering the results from most to fewest trips. If multiple users have taken the same number of trips, order the results by UserID in ascending order. Restrict the results to the 10 users who have taken the most trips.

6. For each age group (year) among customers, list the average ride duration for trips customers of that age group took. The table should include the age group as Age, and Average trip duration for users in that age group as AverageTripDurationPerAgeGroup. Sort the results from longest trips to shortest, trimming the table to the top 10 entries.

7. For each hour of the day, list how many bikes were checked out during that time as NumTrips. Order the results starting from 0 (midnight) up through 23 (11 pm).

8. For each hour of the day, list the percentage of bikes checked out during that hour relative to the other hours of the day AS Percentage. Order the results starting from 0 (midnight) up through 23 (11 pm).

9. The station contains a location as latitude and longitude. Compute for each trip the distance covered by that trip, using the following equation to approximate: sqrt((69 miles * difference in latitude)^2 + (52 miles * difference in longitude)^2). For this computation, use the SQRT function and SQUARE function in SQL. For the 10 longest trips, return the trip ID, starting station ID, ending station ID, and distance travelled as Distance.

10. Compute for each trip the average speed of the bicyclist, by taking the distance travelled computed in the previous question (which is in miles) and divide by the length in hours (the length is stored in seconds). For the 10 fastest trips, return the trip ID, bike ID, and the speed as mph.

Requirements

You will be required to use an SQL **join** when retrieving data from more than one table. Join tables by putting commas between them in the FROM clause, then including the condition that the primary key matches the foreign key in the WHERE clause.

Given the importance of database applications, there are numerous tools that generate SQL for you (e.g. from the database schema). Such tools cannot be used --- you are required to write your SQL queries yourself, from scratch.

Programming environment: Codio

You are free to work in whatever programming environment you prefer. By default, you can login to Codio and begin work immediately using the project "project06". This environment is similar to the one from HW11, containing SQL files for you to write your code and buttons to test your answers.

Electronic Submission and Grading

The grade reported by Gradescope will be a tentative one. After the due date, submissions will be re-evaluated against a new database to see that the correct answers are still being computed on a new set of data.

When you are ready to submit your program for grading, login to Gradescope and upload all your "Project05##.sql" source files, either as a zip or individually. You have unlimited submissions, and Gradescope keeps a complete history of all submissions you have made. By default, Gradescope records the score of your last submission, but if that score is lower, you can click on "Submission history", select an earlier score, and click "Activate" to select it. The activated submission will be the score that gets recorded, and the submission we grade. If you submit on-time and late, we'll grade the last submission (the late one) unless you activate an earlier submission.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 10%. After 24 hours, no submissions will be accepted.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them (e.g. your "iClicker teammates"), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your

program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml .