HW2: SQL

In this HW, you are going to be creating relational tables (with your own rows of data), and writing queries that will make use of the data.

The HW is an 'extension' of HW1, in a way - you can reuse the entities you created in HW1, OR, you can use a different set of entities if you like [there is no req'mt that you MUST use your HW1's tables].

ALL the SQL knowledge/commands you need to answer the questions have been covered in class! You do NOT need to learn more commands or techniques (eg. use of 'triggers') etc. on your own in order to do this HW set.

To run SQL code, you can use one of the three ways mentioned in the lecture notes [a locally installed DB, or a remote server-based DB via an online shell running in a browser page, or a cloud DB] to do the problems - including using https://livesql.oracle.com/, https://bit.io, SQLite Browser, Oracle XE, sqlfiddle, etc.

There are 6 questions below, each is worth 1 point.

Description [created by our own Vinutha, who created HW1 as well :)]

Q1. Find the sponsor who has sponsored the highest amount in YouTube. Display the sponsor’s name, phone number and the total amount sponsored.

Q2. Find the ratio of likes to views of each video belonging to any of the channels owned by users having the word “Marvel Entertainment” in them. Display the Video Title, channel name and the ratio in the ascending order of the title.

Q3. Find unique user/s with the total number of paid subscribers greater than 100 for their channel/s created on 01.01.2023. Display the username, email, channel name and the subscriber count.

Q4. Find the average sentiment score for each keyword category. Display the keyword name along with average score such that the highest score is displayed first.

Q5. Find the minimum and maximum age of viewers who watched the most commented on video on Taylor Swift’s channel. Display the video title, minimum age and the maximum age.

Q6. Find all the content creators living in the US who have consistently posted at least 1 video each week of the last month. Display their username, channel/s they

own and their total subscriber count.

You can hardcode the subjects just for submission purposes, but your query should work for ANY such table! Using comments in the code, YOU NEED TO EXPLAIN IN YOUR OWN WORDS, WHAT THE QUERY DOES (how it works); don't just say things like "Now I'm using a WHERE condition", instead explain what it's for (why you're using it).

Bonus(1(+1) point(s)). For Q2, you will get 1 extra point if you can reformulate the query in a very different way! If you do this, submit a separate text file with the code, eg. Q2_v2.sql. If you come up with YET ANOTHER very different way, you can get 1 more bonus point (submit yet another file, eg. Q2_v3.sql). **'Very different' means just that** - the approaches do have to be totally distinct, eg. you can't use NOT to invert an existing solution, or use IN() instead of OR, etc. Sooo... is this actually possible [to do it in two, or three, different ways]? Yes! Or, as they say in Minne-so-ttta - "yooo betcha!"

What you need to submit are text files with the SQL commands that you come up with, one file for each question (Q1.sql, Q2.sql.. Q6.sql). PLEASE MENTION AT THE TOP OF EACH FILE, which database (eg. livesql, Oracle XE, SQLite..) you used for that question! Your grader(s) will execute the SQL commands from the text files you submit, using the same software you used, to see if they produce the expected results. Please be sure to also include your table creation commands and the row insertion ones - in other words, **make sure your query is self-contained and therefore runnable "out of the box"** [so that your grader would not need to create a table before running your code].

To reiterate, everything you need is in the slides (the material we went through in class) - you do not need to look up more commands online! Look at each relational operator, each command, each function, each keyword that we covered, and ask yourself how it could be of use in constructing your query. Also, you can 'build up' each query by first creating a subquery (or several), then combining the parts - in other words, **'divide and conquer' is a very useful strategy for creating non-trivial SQL queries!**

You can post questions (and answer others' :)) on Piazza, under 'hw2'. **ENJOY!**
