**Spring 2024**  **Project 1- MySQL**

Given below is the schema for the data. There is a total of 12 tables, thus 12 CSV files, each corresponding to a relational table.

**user** (email, password, name, date\_of\_birth, address, type)

primary key(email)

**celebrity** (email, website, kind)

primary key(email)

**blurt** (blurtid, email, text, location, time)

primary key(blurtid,email)

foreign key(email) references user(email)

**hobby** (email, hobby)

primary key(email,hobby)

foreign key(email) references user(email))

**follow** (follower,followee)

primary key(follower,followee)

foreign key(follower) references user(email)

foreign key(followee) references user(email))

**vendor** (id, name)

primary key(id)

**vendor\_ambassador** (vendorid, email)

primary key(vendorid)

foreign key(email) references user(email)

foreign key(vendorid) references vendor(id))

**topic** (id, description)

primary key(id)

**vendor\_topics** (vendorid,topicid)

primary key(vendorid, topicid)

foreign key(vendorid) references vendor(id)

foreign key(topicid) references topic(id))

**blurt\_analysis** (email,blurtid,topicid,confidence,sentiment)

primary key(email, blurtid, topicid)

foreign key(email,blurtid) references blurt(email,blurtid)

foreign key(topicid) references topic(id)

constraint confidence >= 0 and confidence <=10

constraint sentiment >= -5 and sentiment <=5

**advertisement** (id, content, vendorid)

primary key(id)

foreign key(vendorid) references vendor(id))

**user\_ad** (email,adid)

primary key(email,adid)

foreign key(email) references user(email)

foreign key(adid) references advertisement(id))

**The design model:**

Users can post their thoughts in form of short messages that we call “blurts”. When signing up, users need to provide their email and a password of their choice. In addition, they need to enter some basic information – name, date of birth, address, email ID and hobbies. Once signed up, they can (besides blurting) “follow” other users. To “follow” a user means subscribing to his/her “blurts”. Users are categorized into “regular” users and “celebrities”. A celebrity has an associated website url and an attribute called “kind” indicating whether he is a politician, actor, singer, etc. Each blurt by a user (regular or celebrity) is assigned an id. Blurt ids are serial and unique to the “blurts” by a given user; the first blurt by a given user would have blurt id 1 and ids are incremented for each successive blurt by the user. Note that blurt ids are unique only to a user, so blurts by two different users may have the same blurt id. Besides an id, each blurt also has its text, timestamp, and user location as additional attributes. The system should have a pre-defined notion of “topics” that are simply subjects that people may blurt about. Examples of topics might include music, pollution, disease, disaster, sports, weather, etc. A topic has a unique id and description (the name of the topic). Each blurt by a user is analyzed to associate with it zero or more topics. Related blurt-topic pairs are stored in blurt\_analysis table. To account for the possible ambiguity arising from the choice of words or language used by a user, an association with a topic has a corresponding confidence level (an integer ranging from 1 – 10 indicating the strength of the association). For example consider the following blurt: “I absolutely hate the rainy weather, can’t go out, listening to the Beatles, just love them” is analyzed to be associated with two topics, weather and music (Beatles). For each topic, the associated sentiment is evaluated and quantified as an integral value ranging between -5 and 5, with higher values indicating a more positive sentiment. Considering the example blurt used above, the topic weather would have an associated sentiment of -5 (hate) while for music the corresponding value is 4 (love). **Note: You don’t need to implement the value constraint as MySQL doesn’t support it.** A vendor has interest in one or more topics and is interested in tracking all users who are blurting about a topic of interest. A vendor may also have a celebrity as its brand ambassador. Vendors create advertisements that have an associated unique id and a textual content. These advertisements are stored in the system and are available to be shown to the regular set of users (that is, not to the celebrities, just to the other “regular” users). Careful matching is done based upon a historical analysis of all blurts by a user. Based upon the analysis, a user may be shown zero or more advertisements.

**Form SQL Queries**

For the following statements, you are required to form SQL queries and execute them using the GUI client. Then export the result using the name "Query x.csv", x being the label of each query. The filenames of your result has to follow the instructions exactly or you may get a deduction in your credit.

1) Print every hobby, along with the total number of users that included that hobby in their profile.

2) Print the name and address of all celebrities who are not movie stars.

3) Print a list of every vendor and the number of different advertisements they’re running, in descending order.

4) Print email, password, name, date\_of\_birth, kind of all celebrities who are Movie Stars and are following someone.

5) Print the name and email of every celebrity along with their total number of followers if the amount is over 55. The contents should be in descending order based on the number of followers.6) Find all pair of users (A,B) such that A and B have not blurted on a common topic, but user A is following user B. The query should print the names of A and B in that order.

7) Print every location, along with the total amount of blurts, where users made more than 40 collective blurts. The query should print the name of the location and total amount of user blurts in descending order of the amount of blurts.

8) For each vendor, print the email of its brand ambassador along with the number of users the brand ambassador is following. Your query should print the vendor name, ambassador email, and number of users the ambassador is following. The output should be in descending order according to the numbers of users being followed.

9) Print the following: vendor id, vendor name, vendor ambassador’s email, and the total number of users that have been shown the ad(s) from the vendor. Do this for every vendor.