Coursera: Database Management
Project: Business Recommendation Database
Phase 2: SQL Query for Generating Recommendation Model

Introduction

You have learned how to design a Yelp database. The course project will give you an opportunity to create such a database from scratch and build applications on top of this database. Phase 2 has the same background information with Phase 1.

Prerequisite

You should use the createtable.sql provided by us to create tables and then load the given data into the database by yourself. I recommend use “COPY FROM” to load data in Postgres.

Requirement

You need to implement the following SQL queries. For each query, we provide an example of the schema of the saved query result. **Note that, the examples may not be the complete results of queries.**

1. Write a SQL query to return the total number of businesses for each category. Your query result should be saved in a table called “query1” which has these attributes: category_id, name, count.

	category_id bigint	name text	count bigint
1	1074	Professional Services	6276

2. Write a SQL query to return the average rating per category. Your query result should be saved in a table called “query2” which has these attributes: category_id, name, rating.

	category_id bigint	name text	rating numeric
1	1074	Professional Services	3.7441697058921939
2	251	Landscape Architects	4.0019826517967782

3. Write a SQL query to return the businesses have at least 4000 reviews. Your query result should be saved in a table called “query3” which has these attributes: business_id, title, CountOfRatings.

	business_id text	title text	countofratings bigint
1	2weQS-RnoOBhb1KsHKyoSQ	The Buffet	4534
2	4JNXUYY8wbaaDmk3BPzlWw	Mon Ami Gabi	8570
3	5LNZ67Yw9RD6nf4_UhXOjw	The Cosmopolitan of Las Vegas	4522
4	AV6weBrZFFBfRGCbcRGO4g	Luxor Hotel and Casino Las Vegas	4240

4. Write a SQL query to return all “Chinese” businesses including business_id and title. Your query result should be saved in a table called “query4” which has these attributes, business and title. (They can be in other categories as well, but they are qualified as long as they are in “Chinese” category)

	business_id text	title text
1	00I_YBjgZAvd2ggZr3J0cQ	Shanghai Chinese Restaurant
2	01aNlDhbMObjc9OdAHuNpQ	Rose Garden Chinese Restaurant

5. Write a SQL query to return the average rating per business. Your query result should be saved in a table called “query5” which has these attributes, business_id, title and average.

	business_id text	title text	average numeric
1	-000aQFeK6tqVLndf7xORg	Cool Cat Auto Repair	5.0000000000000000
2	0010xnFOCyJZeMAuTtiv5w	Henna Shoppe	4.8157894736842105

6. Write a SQL query to return the average rating for all “Chinese” businesses. Your query result should be saved in a table called “query6” which has one attribute, average. This average is the average over all review operations rather than the average over the average ratings of all businesses.

	average numeric
1	3.5120886180012006

7. Write a SQL query to return the average rating for all businesses and each of these businesses is both “Chinese” and “Japanese”. Your query result should be saved in a table called “query7” which has one attribute, average.

	average numeric
1	3.5803132623837641

8. Write a SQL query to return the average rating for all businesses and each of these businesses is “Chinese” but not “Japanese”. Your query result should be saved in a table called “query8” which has one attribute, average.

	average numeric
1	3.5038157466001269

9. Find all businesses that are rated by a User such that the userId is equal to ‘CxDOIDnH8gp9KXzpBHJYXw’. Your query result should be saved in a table called “query9” which has these attributes, business_id and rating. **Note that, in this phase, a user may review the same business multiple times.** If this happens, take his average rating of this business. **No duplicate businesses are allowed in the results**.

	business_id text	rating numeric
1	MSA9WlcVCjlektkeDX5keA	3.0000000000000000
2	eNbRmHgNUNyhqpCT_Xsklw	4.0000000000000000

10. Write a SQL query to create a recommendation model using item-based collaborative filtering. Given a userID 'ogAjjUdQWzE_zlAGZWMd0g', you need to recommend new businesses according to the businesses he has rated before. In particular, you need to predict the rating P of a business i that the user Ua didn't rate. In the following recommendation model, P(Ua, i) is the predicted rating of business i for User Ua. L contains all businesses that have been rated by Ua. Sim(i,l) is the similarity between i and l. r is the rating that Ua gave to l.

$$P_{(u_a,i)} = \frac{\sum_{l \in \mathcal{L}} sim(i,l) * r_{u_a,l}}{\sum_{l \in \mathcal{L}} sim(i,l)}$$

Your SQL query should return predicted ratings from all businesses that the given user hasn't rated yet. You only return the businesses whose predicted ratings are >4.33. Your query result should be saved in a table called "query10" which has these attributes, business_id, title.

	business_id text	title text
1	-000aQFeK6tqVLndf7xORg	Cool Cat Auto Repair
2	DO32u1Q6RV2j11W6nzeW6A	Diane Varney - Coldwell Banker Premier Realty

In order to produce the recommendation table, you first need to calculate the similarities between every pair of two businesses. The similarity is equal to the similarity of the average ratings of two businesses (average rating over all users, the average ratings in Query 5). That means, if the average ratings of two businesses are more close, the two businesses are more similar. The similarity score is a fractional value $\in$ [0, 1]. 0 means not similar at all, 1 means very similar. To summarize, the similarity between two businesses, i and l, is calculated using the equation below. Abs() is to take the absolute value.

Sim(i, l) = $1 - \frac{abs(i's\ avgrating - l's\ avgrating)}{5}$

The result of similarity table should look like as follows:

	business_id1 text	business_id2 text	sim numeric
1	-000aQFeK6tqVLndf7xORg	0010xnFOCyJZeMAuTtiv5w	0.96315789473684210000
2	-000aQFeK6tqVLndf7xORg	001jVKJHdhU2z_m9xTgOrg	1.00000000000000000000
3	-000aQFeK6tqVLndf7xORg	002Gv4JE2bAfXkGJVM1IQw	0.74838709677419354000

Note, you don't have to follow the exactly same table schema for the similarity table, as long as you can produce the correct recommendation table.

Above all, your script should be able to generate 10 tables, namely, "query1", "query2", …, "query9", "query10".

Remarks

1. We will use an auto-grading script to test query results using different data. All table names and attribute names must be in lowercase and exactly same with the specification. Otherwise, the script will fail and you will directly lost points for those failed queries.

2. Input data can be downloaded from this link using your ASU email:
https://drive.google.com/file/d/1v3shopwwHJCwQC-Mc6cxiy5dUenI6kVv/view?usp=sharing

The delimiter of all input files is ",". They all have header rows.

3. You should use the following command to save your query result to a table.

CREATE TABLE query0 AS
YOUR SQL STATEMENT

For instance, select the user from the users table which has userID = 'XXX' and store it in query0 and rename the "username" column to "userfullname".

In your SQL script:

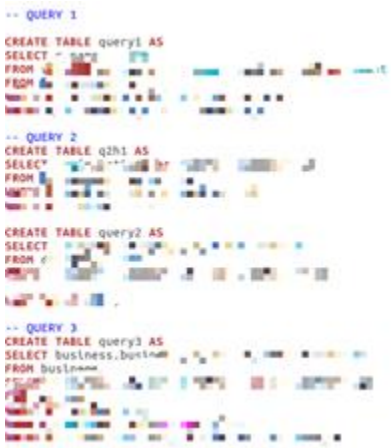
CREATE TABLE query0 AS
SELECT username AS userfullname
FROM users
WHERE users.userid = 'XXX'

4. Do not put "create/select/drop database", or "set system settings or encoding" in your submission SQL script. This may lead to point deductions.

5. The rows in your query result table **don't have to be sorted**.

6. You are free to create any other temp/permanent views, temp/permanent tables to help your queries.

7. Example submission should look like this:



Submission

Submit **a single plain text file** “cse412-project2-MYASUDNumericalID.sql” to Canvas.

Deadline

April 9th 11:59 pm