Project Group 53: A Traveler's Guide to California

Group Members (Name | email | Github username):
- Nimish Jayakar | nimishmj@seas.upenn.edu | Jediknight1224
- Yuqiao Xue | joexue@seas.upenn.edu | brinrs
- Jack Greff | jgreff@haverford.edu | jackgreff
- Isabella Salathe | isalathe@haverford.edu | irsalathe

Credentials:
Host: database-1.cf808uyewuwd.us-east-1.rds.amazonaws.com
Username: admin
Password: gpjU35xkgnUfGYP
Database name: project

Queries:

1. Join the business table and reviews table in the Yelp dataset on business_id. This query will give us a comprehensive list of all reviews in the dataset corresponding to each business. This is a basic query that can be used in many of the below questions. All Reviews should be in California as well.

```
SELECT b.business_id, b.name, b.state, r.stars AS review_stars,
        CASE
            WHEN r.stars >= 4 THEN 'Positive'
            WHEN r.stars >= 3 THEN 'Neutral'
            ELSE 'Negative'
        END AS review_sentiment
FROM Business b
JOIN Review r ON b.business_id = r.business_id
WHERE b.state = "CA"
```

2. Select text reviews
   From the Yelp dataset for businesses that are restaurants from the business table Where that are between 3 and 5 stars and the reviews are marked as useful by others and sorted reviews that are recent and by useful attributes in descending order. The top useful reviews each day for the last 10 days.

```
With Businesses as (
SELECT b.business_id, b.name, b.state, r.stars AS review_stars,
FROM Business b
JOIN Review r ON b.business_id = r.business_id
WHERE b.state = "CA" ),
SELECT r.text
FROM Review r
WHERE business_id IN Businesses.business_ID and r.stars => 3 and r.stars <= 5 and
r.useful >= 1
SORT By r.date DESC, .useful DESC
LIMIT 10
```

3. **find positive-rated businesses in the restaurant category, and evaluate their user engagement through number of tips. It can help identify popular and well-reviewed businesses within a specific category, potentially aiding decision-making processes for consumers**

```
WITH BusinessReviews AS (
  SELECT
    b.business_id,
    b.name,
    r.stars AS review_stars,
    CASE
      WHEN r.stars >= 4 THEN 'Positive'
      WHEN r.stars == 3 THEN 'Neutral'
      ELSE 'Negative'
    END AS review_sentiment
  FROM Business b JOIN Review r ON b.business_id = r.business_id
),
BusinessTips AS (
  SELECT b.business_id, COUNT(*) AS total_tips
  FROM Business b
  JOIN Tip t ON b.business_id = t.business_id
  GROUP BY b.business_id
)
SELECT
  br.business_id,
  br.name,
  br.review_stars,
  br.review_sentiment,
  COALESCE(bt.total_tips, 0) AS total_tips
FROM BusinessReviews br
LEFT JOIN BusinessTips bt ON br.business_id = bt.business_id
WHERE br.review_sentiment = 'Positive'
AND br.business_id IN (
    SELECT business_id FROM Business WHERE categories LIKE CONCAT('%',
${Restaurant}, '%'))
ORDER BY br.review_stars DESC, total_tips DESC;
```

4. **Trending Businesses - Join the Review and CheckIn tables to identify businesses with an increase in check-ins or reviews over a recent time period. We could further refine this query by category (i.e. restaurants) from Business**

```
SELECT b.business_id, b.name, b.categories,
COUNT(DISTINCT r.review_id) AS total_reviews,
COUNT(DISTINCT c.checkin_date) AS total_checkins
FROM Business b
JOIN Review r ON b.business_id = r.business_id
JOIN Checkin c ON b.business_id = c.business_id
WHERE STR_TO_DATE(r.date, '%Y-%m-%d %H:%i:%s') >= DATE_SUB(CURDATE(),
INTERVAL 1 MONTH)
OR STR_TO_DATE(c.checkin_date, '%Y-%m-%d %H:%i:%s') >=
DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY b.business_id, b.name, b.categories
ORDER BY total_reviews DESC, total_checkins DESC
```

5. Identify businesses that have been reviewed by users who have also reviewed other businesses sorted by recent reviews. This provides insight about reviews from users who are most active.

```
WITH user_with_reviews as (
        SELECT r1.user_ID
        FROM Reviewer r1
        GROUP BY user_id
        HAVING COUNT(review_id) > ${num_reveiws}
)
SELECT DISTINCT business_id
FROM Review review
WHERE rev.user_id IN user_with_reviews
```

6. **Recommended Business within a specific category. - Join the "business", "review", and "tips tables". This query will provide a list of businesses in a specified category and city with a high average rating, a substantial number of reviews, and useful tips.**

```
SELECT b.name
          b.address,
          b.city,
          b.state,
          b.stars AS business_rating,
          b.review_count,
          AVG(r.stars) AS average_review_rating,
            COUNT(distinct t.tip_id) AS total_tips,
            MAX(t.compliment_count) AS highest_tip_compliments,
FROM Business b
INNER JOIN Review r ON b.business_id = r.business_id
LEFT JOIN Tip t ON b.business_id = t.business_id
WHERE b.city = 'CityName'
        AND b.categories LIKE CONCAT('%’, ${Category}, ‘%')
        AND b.is_open = 1
GROUP BY b.business_id
HAVING business_rating >= 4.0 AND review_count > 50
ORDER BY business_rating DESC, average_review_rating DESC, total_tips DESC
LIMIT 10
```

7. Select the population density within ‘x’ range in lat or log values or postal codes
   From the population dataset joined over the Yelp dataset
   Where a user selects a particular business or city. We calculate the max and min values using the equation to find distances between two coordinates(same as finding the length of an arc on a circle) before we pass the values to the queries.

   ```
   SELECT c.population, c.density
   FROM Cities c
   JOIN Business b ON c.city = b.city
   WHERE b.postal_code = ${postal_code}
   OR (b.latitude BETWEEN ${maxLat} AND ${minLat}
   AND b.longitude BETWEEN ${maxLog} AND ${minLog})
   ```

8. Join User, Review, and Tip tables. Determine which users are most influential based on their review counts, the usefulness of reviews, and tip counts to highlight top contributors in a community

   ```
   SELECT u.user_id, u.name,
   u.review_count AS total_review_count,
   SUM(u.useful) AS total_useful_reviews,
   COUNT(t.tip_id) AS total_tips
   FROM User u
   LEFT JOIN Review r ON u.user_id = r.user_id
   LEFT JOIN Tip t ON u.user_id = t.user_id
   GROUP BY u.user_id, u.name
   ORDER BY total_review_count DESC, total_useful_reviews DESC, total_tips DESC
   ```

9. Join Business, Tip table. Lists businesses in a category, including tips, and how many times these tips were given, along with the overall rating of business. Basic tip retrieval query, variation of which can be used for different parts of our application to display

reviews and tips information.

```
SELECT b.name, b.address, b.stars, t.text AS tip_text, t.compliment_count
FROM Business b
Join Tip t ON b.business_id = t.business_id
WHERE b.categories LIKE CONCAT('%’, ${category}, ‘%')
ORDER BY t.compliment_count DESC, b.stars DESC
LIMIT 10
```

10. **Find top ‘x’ businesses for each postal code that have been reviewed by users who have also reviewed atleast ‘min_review’ number of times. The window function over postal code should find rank in order of stars and useful columns, which we can then use to select the top ‘x’ ranks from**

```
WITH UserReviewCounts AS (
   SELECT user_id, COUNT(*) AS total_reviews
   FROM Review
   GROUP BY user_id
   HAVINGCOUNT(*) >= ${min_review}
),
TopReviews AS (
   SELECT r.business_id,
          r.review_id,
          r.user_id,
          r.stars,
          r.useful,
          b.postal_code
          ROW_NUMBER() OVER (PARTITION BY b.postal_code ORDER BY r.stars
DESC, r.useful DESC) AS review_rank
   FROM Review r JOIN Business b ON r.business_id = b.business_id
   JOIN UserReviewCounts urc ON r.user_id = urc.user_id

)
SELECT b.name AS business_name,
         tr.postal_code,
         tr.review_id,
         tr.stars,
         tr.useful
FROM Business b
JOIN TopReviews tr ON b.business_id = tr.business_id
WHERE tr.review_rank <= ${x}
ORDER BY tr.postal_code, tr.useful DESC;
```