Assignment 3

Instructions

1. Use the database `SQL_Wedge.db` for this assignment, just like last week.
2. Complete each of the queries described below. You will write your queries in DB Browser. When you've completed a query and feel confident it's returning the correct results, paste it into a file called `module-1-assignment.sql`. Feel free to use comments (`--` or `/* */`) to add commentary, or ask me questions.
3. Additionally, take a screenshot of the visible portion of the results set in DB browser. Paste these screenshots into a document called `module-1-results.pdf`.
4. Submit both `module-1-assignment.sql` and `module-1-results.pdf` into the assignment on Moodle.
5. For feedback you will receive a copy of your `.sql` file. Each query will be marked "complete" or "incorrect" and the `.sql` file will be returned to you. Part of the learning process is trying to figure out *why* your query was marked "incorrect", but feel free to ask in class or office hours.

Assignment 3

3.1 Create temporary owner, year, month table

Using the table `owner_spend_date`, write a query that creates a temporary table `owner_year_month` that has the following columns: `card_no`, `year`, `month`, `spend`, and `items`. Those last two columns should be the sum of the columns of the same name in the original table.

For your "results" screenshot, just take a screenshot of the successful query message.

3.2 Top 5 Months

Using your temporary table, write a query that returns year, month, and total spend for the five month-year combos with the highest spend.

3.3 Average Spend by Month for 55405

Again using your temporary table (and the `owners` table), write a query that returns the average spend by month (across all years) for owners who live in the 55405 zip code. Use a subquery to get the appropriate set of card_no values. Order the results by month and give your average spend value a sensible column name. Round average spend to two decimal places.

3.4 Sales by Zip Code

Write a query that returns zip code and total sales for the three zip codes with the highest total sales.

3.5 Average Spend by Month for Top 3 Zips

Note: this query is, I think, more challenging than the above.

Repeat **3.3** but now include four columns in your output. You will again include month, but now include one column of average sales for each of the zip codes you found from the previous query. You can begin with the query from **3.3**, but now call the results `avg_spend_55405`. Join two CTEs to append the columns holding the average sales for the two zips that are not 55405. Use the `avg_spend_XXXXX` naming convention.

3.6 A New Temporary Table

Rewrite your query for **3.1**, adding a column called `total_spend` which holds the total spend, across all years and months, for that owner. Use a CTE to calculate the total sales and `JOIN` that to your previous query. Add a line at the beginning that deletes the temporary table if it exists.

In the below code block, I have included a query of your table. If your table has the right shape, it will return a result. Paste that result into your "results" file.

```
-- Run this after rebuilding your temporary table.
SELECT COUNT(DISTINCT(card_no)) AS owners,
        COUNT(DISTINCT(year)) AS years,
        COUNT(DISTINCT(month)) AS months,
        ROUND(AVG(spend),2) AS avg_spend,
        ROUND(AVG(items),1) AS avg_items,
        ROUND(SUM(spend)/SUM(items),2) AS avg_item_price
FROM owner_year_month
```

3.7 Owner Recent Summary

If a business has up-to-date customer information accessible quickly, the business can use that information for customer retention. In this query, we create an "owner at a glance" view that tells us a bit about the owner overall and their recent behavior.

Using the `owner_spend_date` table, create a view with total amount spent by owner, the average spend per transaction, number of dates they have shopped, the number of transactions they have, and the date of their last visit. Give these columns sensible names. Call your view `vw_owner_recent`.

After you've built your view, run the query below and paste your results into the "results" file.

```
SELECT COUNT(DISTINCT card_no) AS owners,
       ROUND(SUM(total_spend)/1000,1) AS spend_k
FROM vw_owner_recent
WHERE 5 < total_trans AND
        total_trans < 25 AND
      SUBSTR(last_visit,1,4) IN ('2016','2017')
```

3.8 Adding Detail to View and Building a Table

In this query you will create a table in your database. This table, called `owner_recent`, will be built off your view and will add an additional column. This new column (called `last_spend`) will be the amount spent on the date of that last visit. Adding this information uses a clever trick that is common in SQL work. You'll essentially be joining `owner_spend_date` to *itself*, but using the view as an intermediary.

The approach is straightforward: select all of the columns out of the view and the column you need out of `owner_spend_date`, joining two fields (`card_no` and the date fields).

Once you've built your table, run the two queries below. Verify that the results look sensible. Answer these two questions in a comment in your code file. (Reminder `/* */` allow you to comment blocks of code or text.)

1. What is the time difference between the two versions of the query?
2. Why do you think this difference exists?

```
-- Select a row from the table
SELECT *
FROM owner_recent
WHERE card_no = "18736";

-- Select a row from the view
SELECT *
FROM vw_owner_recent
WHERE card_no = "18736";
```

3.9 Owner Filtering

For this query, imagine the marketing manager wants you to help identify owners who are high value, but have lapsed. The data set runs from 2010-01-01 through 2017-01-31. This is a bit over 360 weeks.

Write a query that retuns the columns from `owner_recent` for owners who meet the following criteria:

1. Their last spend was less than half their average spend.
2. Their total spend was at least $5,000.
3. They have at least 270 shopping dates.
4. Their last visit was at least 60 days before 2017-01-31.
5. Their last spend was greater than $10

Order your results by the drop in spend (from largest drop to smalles) and total spend.

3.10 Non-close Owner Filtering

Repeat the above analysis, but with a slightly different set of criteria. Our goal is to identify people who do not live in the Wedge's zip code or adjacent to the Wedge. Recall that our zip code classificaiton is the following:

- `wedge` : 55405
- `adjacent` : 55442, 55416, 55408, 55404, 55403
- `other` : any other zip codes.

Write a query that retuns the columns from `owner_recent` for owners who meet the following criteria:

1. The have non-null, non-blank zips and they do not live in the Wedge or adjacent zip codes.
2. Their last spend was less than half their average spend.
3. Their total spend was at least $5,000.
4. They have at least 100 shopping dates.
5. Their last visit was at least 60 days before 2017-01-31.
6. Their last visit was over $10

Include the owner's zip code in your query results. Order your results by the drop in spend (from largest drop to smalles) and total spend.