Assignment 5

This assignment is meant to cover questions from across the course. Additionally, certain questions ask you to write Python code, which may contain SQL.

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-5-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-5-results.pdf`.
4. Submit both `module-5-assignment.sql` and `module-5-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

5.1 Total spend by all owners

Using `owner_spend_date`, calculate the total amount spent across all owners. Return this total spend amount.

5.2 Big Spends

Using the table `owner_spend_date`, write a query that finds some specific shopping days. For owners who spent more than $10,000 in 2017, return the amount of their maximum spend in that year. Return the columns `card_no` and `spend`. Order the results by spend, *descending*. Exclude card number 3.

5.3 Filtering `department_date`

Using the table `department_date`, write a query that returns all columns for rows that meet the following conditions:

- The department number is not 1 or 2
- The spend on the day for the department is between $5,000 and $7,500
- The month is May, June, July, or August

Order your results by spend, decreasing.

5.4 Busy months and departments

Using the tables `date_hour` and `department_date`, write a query that returns department spend during the busiest months. Use `date_hour` to determine the months with the four highest spends. Then use `department_date` to return the department spend in departments during these months. Your query should return the year, the month, the total store spend (from `date_hour`), the department, and the department spend. Arrange your results ascending for year and month and descending for department spend. Limit your results to departments with over $200,000 spend in the month.

5.5 Big Transaction Zips

Write a query that answers the following question: for zip codes that have at least 100 owners, what are the top five zip codes in terms of spend per transaction? Return the zip code, the number of owners in that zip code, the average amount spent per owner, and the average spend per transaction.

5.6 Small Transaction Zips

Repeat 5.5 but return the zip codes with the lowest spend per transaction. It's up to you if you'd like to exclude blank zips in the output.

5.7 Zip Summary

Write a query against the `owners` table that returns zip code, number of active owners, number of inactive owners, and the fraction of owners who are active. Restrict your results to zip codes that have at least 50 owners. Order your results by the number of owners in the zip code.

Note: in order to get `/` to give you real-number division, cast your numerator or denominator

in the "fraction active" column to a `REAL`.

Python Exercises

These next exercises as you to write Python code. The questions build on each other, so you should be able to write these in a notebook. You do not need to include results screenshots for these exercises.

5.8 Create a Table

Using the Python library `sqlite3`, write Python code that opens a connection to a database called `owner_prod.db`. Create a cursor to that database. Use the cursor to create a table called `owner_products` in the database with the following columns and data types:

- owner: integer
- upc: integer
- description: text
- dept_name: text
- spend: numeric
- items: integer
- trans: integer

5.9 Populate a Table

This assignment includes a zip file called `owner_products.zip`, which has inside it `owner_products.txt`. Extract this file into the directory that holds your code if you'd like to run the below code. This tab-delimited text file has the same columns as the table you created in **5.8**. The following lines of Python will read this file into a list of lists.

```
owner_prod = []

with open("owner_products.txt",'r') as infile :
    next(infile)

    for line in infile :
        line = line.strip().split("\t")
        owner_prod.append(line)
```

Beginning with this code (and assuming 5.8 has run correctly), write Python code that will insert the list of lists `owner_prod` into your database table and commit your results.

5.10

In Python, execute a query against your new table that returns `description` (which is the description of the product), `dept_name`, and `total_spend` for every product in a department that has "groc" as a substring in the department name. The total spend should be the spend across all owners for that product. Order the results by `total_spend` *descending*. It is strongly recommended to develop the query in DB Browser before moving to Python so that you can ensure your query is correct.

Have your code iterate over the first 10 rows of the results and print them to the screen. Then close the DB.