Assignment 1: SQL Queries

This assignment asks that you work through a set of database query tasks - first by explaining a set of pre-written queries, then by writing some of your own.

If you have questions about any part of the assignment, please drop by during office hours or contact me via Slack or email. You are welcome to assist your peers within the constraints of the Academic Honesty policy (that is, don't write code for each other).

Submission format

In your home folder on `cs1.ncf.edu`, create an `assignment_1` folder. Please save the files for your response in that folder. You may submit the Canvas assignment empty or with a zip file of your work, but I will be grading the version that exists on CS1 as of the deadline.

- Other students cannot read the files in your home folder.
- For suggestions on working on the server and transferring files, see this page.

Explaining queries

Write your answers to problems 1-4 in a Markdown file at `~/assignment_1/explanations.md`.

- Markdown is the text format used by Jupyter notebooks and many other applications. If you're not familiar with Markdown, this guide may be helpful.
- I suggest using VSCode to edit your files.

Example

In the following example, we have an SQL query for some imaginary database. In the block comment below the query, I've written a natural-language explanation of what it does.

```
-- imaginary contacts db
SELECT
   name,
   dob as "Date of Birth"
FROM contact_info
JOIN birthdays ON contact_info.person_id=birthdays.person_id
ORDER BY name;

/*
This query returns a table of each person's name and date of birth.

Steps:
- Find the rows in the `contact_info` and `birthdays` tables that have the same `person_id`.
- Combine each matching pair of rows.
- Select the `name` and `dob` columns of each combination.
- Rename `dob` to `"Date of Birth"`.
- Sort the results alphabetically by name.
*/
```

You can use this example as a template for problems 1-3.

Problem 1

```
-- homelessness db
SELECT *
FROM bookings
WHERE TO_CHAR(dob, 'YYYY') IN ('1990', '1991', '1992');
```

- [ ] Write your own explanation of this query.

Use natural language (as in the example), pseudocode, or any other method that demonstrates an understanding of the query.

Problem 2

```
-- mini_example db
SELECT name,
    employment.job_title,
    employment.employer,
    employment.salary,
    averages.average_salary
FROM employment
    LEFT JOIN person on person.user_id = employment.user_id
    LEFT JOIN (
        SELECT job_title,
            AVG(salary) AS average_salary
        FROM employment
        GROUP BY job_title
    ) averages ON employment.job_title = averages.job_title;
```

- ☐ Write your own explanation of this query.
- ☐ If we changed the left joins to inner joins (i.e., deleted the `LEFT` keywords), what would be different about the result?

Problem 3

In our second lecture, we attempted to *find the oldest person arrested for each charge*. What follows is a correct solution to this exercise.

```
-- homelessness db
SELECT DISTINCT
    info.dob,
    info.charge,
    info.name
FROM (
    SELECT
        charge,
        MIN(dob) as dob
    FROM
        bookings
        JOIN charges ON charges.bookingnumber=bookings.bookingnumber
    GROUP BY charge
) oldest JOIN (
    SELECT
        charge,
        name,
        dob
    FROM
        booking_dates JOIN charges ON booking_dates.bookingnumber=charges.bookingnumber
) info
ON oldest.charge=info.charge AND oldest.dob=info.dob
ORDER BY charge,name
LIMIT 20;
```

- ☐ Write your own explanation of this query.
    - I suggest explaining each of the two subqueries separately and then explaining how they are joined.

Problem 4

PostgreSQL will not run the following query.

```
-- homelessness db
SELECT
    soid,
    address,
    MIN(dob)
FROM bookings
GROUP BY address;
```

- ☐ Why doesn't this work? Explain briefly.
- ☐ How does the Problem 3 query avoid this issue?

Writing queries

For the problems 5-8, you will need to **write your own SQL queries** for the `homelessness` database.

Please save each answer in a separate SQL file. At the bottom of each file, in a `/* block comment*/`, **write a brief explanation** of the query.

Run each query on the server and **save the output** as a CSV file.

- For example, run the command `psql -a -d homelessness -f problem_7_a.sql -o problem_7_a.csv --csv` for Problem 7a.

Problem 5

Selection

- [ ] Find every row in the `booking_dates` table where the booking date is on or after January 1, 1990. Sort the results by booking date.

Problem 6

Text search

- [ ] (a) Find every row in the `arrestees` in which the person's first name is "Mary".
- [ ] (b) List every **unique** first name in the `arrestees` table that starts with "Mary".
    - e.g. "Mary Ann", "Mary Lu", etc.
    - Hint: consider using the `SPLIT_PART` function.

Problem 7

Aggregation and formatting

- [ ] (a) Using `booking_dates`, find the age of each person at the date of their first arrest. Display the age in years, months and days.
    - For information on handling dates in Postgres, see the documentation here.
- [ ] (b) Using `charges` and `booking_dates`, find the average age of the people arrested for each charge at the date of arrest.
- [ ] (c) Find the difference in age between the oldest and youngest age arrested for each charge. Display each difference as a number of years, months and days.
    - You can base your work on the query in Problem 3.
- [ ] (d) Using `booking_dates`, `COUNT` the number of people living at each street address. Ensure that each person is only counted once; use the SOID to identify each person.

Problem 8

Exploration

- [ ] (a) How do we know if someone is homeless? This question is exploratory. Show and explain any exploratory queries you use, and cite any external sources of information (e.g. maps, articles). Develop and state a criterion for inferring homelessness.
    - Save your response in `problem_8_a.md`.
- [ ] (b) Write a query that returns the columns of `bookings` with an additional `homleessness` column (true or false).
    - Save your query in `problem_8_b.sql` and its output in `problem_8_b.csv`.

Before submitting

Check that running `ls ~/assignment_1` on CS1 shows the following files:

```
explanations.md

problem_5.sql

problem_6_a.sql
problem_6_b.sql

problem_7_a.sql
problem_7_b.sql
problem_7_c.sql
problem_7_d.sql

problem_8_a.md
problem_8_b.sql

problem_5.csv
problem_6_a.csv
problem_6_b.csv
problem_7_a.csv
problem_7_b.csv
problem_7_c.csv
problem_7_d.csv
problem_8_b.csv
```