Intro to databases exercise

The purpose of this exercise is to practice the important skill of analyzing data in databases using Structured Query Language (SQL).

Learning objectives

After completing this exercise, you'll understand:

- How to write `SELECT` statements.
- How to filter data using `WHERE` clauses.
- How to execute mathematical expressions and string concatenation in SQL statements.
- How to filter data for `NULL` values.

Evaluation criteria and functional requirements

- All of the queries run as expected.
- The number of results returned from your query equals the number of results specified in each question.
- The unit tests pass as expected.
- Code is clean, concise, and readable.

To complete this exercise, you need to write SQL queries in the files that are in the `Exercises` folder. You'll use the `UnitedStates` database as a source for all queries.

In each file, there's a commented out problem statement. Below it, write the query needed to solve the problem. The value immediately after the problem statement is the expected number of rows that must be returned by the query.

Getting started

1. If you haven't done so already, create the `UnitedStates` database. The script for this is available in today's lecture code.
2. Open the `Exercises` folder. Each file is numbered in suggested order of completion, but you can do them in any order you wish.
3. Launch pgAdmin and open each numbered exercise file one-by-one. Write and run the query for the individual exercise. Save the file before moving onto the next exercise.
4. The unit tests project `intro-to-databases-exercise` is in the same directory as this README. You can open it in IntelliJ and run the tests as you did in earlier exercises.

> Note: Make sure to save your changes to the SQL file before running the unit tests.

Tips and tricks

- `SELECT` statements specify the columns of a table that you want to return from a query. While the values in the `SELECT` statement are usually directly mapped to a column name, they can also be aliased using the `AS` keyword.
- `WHERE` clauses filter results. Some operators you can use for filtering out data include:

    - `=`, `<>`, `!=`, `>`, `>=`, `<`, `<=`
    - `IN(values)`, `NOT IN(values)`
    - `BETWEEN value AND value`
    - `IS NULL`, `IS NOT NULL`
    - `LIKE`, `ILIKE` (with wildcard characters)
- Multiple filter conditions can be combined using `AND` and `OR`.
- The `DISTINCT` clause removes duplicate values from the results.
- The PostgreSQL documentation includes a tutorial for querying database tables, as well as documentation related to the `SELECT` statement.