Aggregate Functions Indicidual Exercises

The purpose of this exercise is to practice using aggregate functions to summarize data using Structured Query Language (SQL).

Learning Objectives

After completing this exercise, students will understand:

- How to order SQL query results using the `ORDER BY` statement.
- How to limit SQL query results using the `LIMIT` statement.
- How to concatenate strings together in SQL.
- How to use aggregate functions to summarize multiple rows of data.
- How to use the `GROUP BY` operator.

Evaluation Criteria & Functional Requirements

- All of the queries run as expected.
- The number of results returned from your query is equal to the number of results specified in each question.
- Code is clean, concise, and readable.
- You *should not* need to use subqueries to produce the expected results.

To complete this exercise, you need to write SQL queries in the `aggregate-functions-exercises.sql` file. Below each commented out question, you'll write the query necessary to answer the question being asked using the world database as the source.

Getting Started

- Open the `aggregate-functions-exercises.sql` file in DB Visualizer.
- If you have not done so already, create the world database. The script for this should be available in yesterday's lecture code.
- In the "Database Connection" properties above the file, select the world database.
- You can run all of the database commands in the file at one time by pressing the command + enter key at the same time.
- You can run a single database command at a time by highlighting the command, and then pressing the command + enter key at the same time.

Tips and Tricks

- The `ORDER BY` statement orders results based on the value in a column. By default, the results are sorted in ascending order. However, if you want to sort in descending order, you can add the `DESC` keyword after the column name. You can also explicitly state that the sort order is ascending using the `ASC` keyword.
- The `LIMIT` statement can be helpful when you want to reduce the size of the results to a specific number. For instance, if you only want to get five of the rows from a result set, you can do so by specifying `LIMIT 5` as the last line of your query. Essentially, the results are truncated when using this

statement. Keep in mind, you should typically combine this with an `ORDER BY` statement if you are looking for the top or bottom results. The PostgreSQL documentation explains why ordering is important in more detail.

- Occasionally, you'll need to combine multiple fields to represent data. This can be achieved using a concatenation operator (`||`). For instance, given a table of employees, if you wanted to get the full name for all of the employees in the table, you could write the following query:

```
SELECT e.first_name || ' ' || e.last_name as employee_name
FROM employees e
```

- Often, you may need to aggregate data when writing SQL queries. PostgreSQL offers several aggregate functions that can be useful for summarizing and grouping data. Several functions that are used quite often include:
  - **`AVG`** returns the average value of a numeric column
  - **`SUM`** returns the total sum of a numeric column
  - **`COUNT`** returns the number of rows matching criteria
  - **`MIN`** returns the smallest value of the selected column
  - **`MAX`** returns the largest value of the selected column
- Sometimes when using aggregate functions, you need to `GROUP BY` other columns to get results. Excluding `GROUP BY` results in a single result being returned from your query. Often, however, this is not the expected behavior. The fields that you specify is the field that duplicate values should be removed for. For instance, if you wanted to get the average population of cities in each state in the USA ordered by the name of the state, you might write a query like the following:

```
SELECT district, AVG(population)
FROM city
WHERE countrycode = 'USA'
GROUP BY district
ORDER BY district;
```
