INSERT, UPDATE, and DELETE

The purpose of this exercise is to practice inserting, updating, and deleting rows in database tables using Structured Query Language (SQL).

Learning objectives

After completing this exercise, you'll understand:

- How to insert data using the `INSERT` statement.
- How to update data using the `UPDATE` statement.
- How to delete data using the `DELETE` statement.

Evaluation criteria and functional requirements

- All of the queries run as expected.
- The unit tests pass as expected.
- Code is clean, concise, and readable.

To complete this exercise, you need to write SQL statements in the files that are in the `Exercises` folder. You'll use the `MovieDB` database for all these exercises.

In each file, there's a commented out description of the change you must make to the database. Below it, write one or more `INSERT`, `UPDATE`, or `DELETE` statements to make the requested change. The value immediately after the description is the number of rows expected to be affected by your SQL.

Getting started

1. If you haven't done so already, create the `MovieDB` database. The script for this is available in yesterday'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 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

- The `INSERT` statement adds rows of data (records) to a database table.
- The `UPDATE` statement updates existing data in a table.
- The `DELETE` statement deletes data from a table.
- IMPORTANT: Be sure to include a `WHERE` clause when you're updating or deleting data from a table unless you intend to update or delete *all* rows in the table.
- Using transactions allows you to quickly get a database back into the state it was in before you ran statements. Consider doing this as you work through the exercises to avoid having to restore your

database. *If you do, be sure to remove the transaction-related statements before running the unit tests.*
- If you need to restore the database, use the same script that created the `MovieDB` database to restore it back to the original state.