Cumulative Project: Gold Medal Metrics

Create the SQL queries to power an Olympics analytics web app.

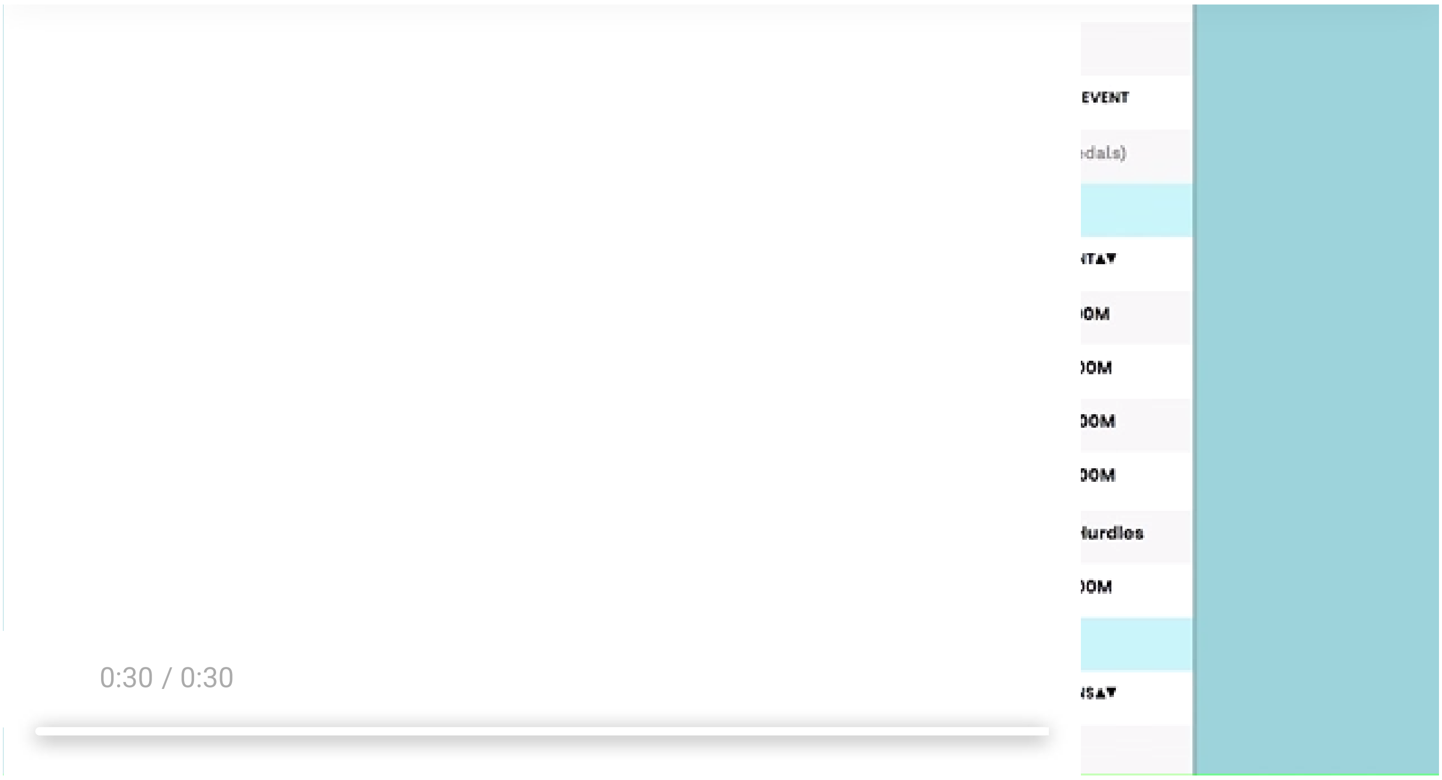
Gold Medal Metrics

Project Overview

In this project you will be writing all the SQL statements for an Olympic metrics reporting web application called Gold Medal Metrics.

Gold Medal Metrics allows users to:

- View countries in a list with their population, GDP, and number of Olympic gold medals.
- Sort the list of countries by any of these attributes, as well as alphabetically by name.
- View a detailed description of a country, with statistics on their Olympic wins.
- View a list of every Olympic win a country has with the year, season, winner name, city, and event.
- Sort the list of Olympic wins by any of these attributes.



How To Begin

To start, download the starting code for this project here. After downloading the zip folder, double click it to uncompress it and access the contents of this project.

To view the webpage, run `npm install` and then `npm run webpack` to compile the front-end files, and follow that with `open index.html` from the root directory of this project. To start your server, run `node server.js`. Refresh your browser to collect the information from the server. Every time you change **server.js**, you will have to restart your server before the changes will take effect. To do this press “control + c” in the bash terminal where your server is running (or close the terminal) to shut it down and then re-run `node server.js` to start it again. While your server is running, you will not be able to run commands in the bash terminal, so open a new terminal if you want to run other commands.

Implementation Details

To complete this project, you will need to write a series of JavaScript functions that

Next

Get Help

are stubbed out in **sql.js** with comments about the query each should return. Below we list the different functions and the expected returned query.

Gold Medal Metric Functions

createCountryTable

Returns the SQL command that will create a table, named `Country` with the following columns:

- `name` a required text field.
- `code` a required text field.
- `gdp` an integer.
- `population` an integer.

createGoldMedalTable

Returns the SQL command that will create a table, named `GoldMedal` with the following columns:

- `id` an integer that will function as the primary key.
- `year` a required integer.
- `city` a required text field.
- `season` a required text field.
- `name` a required text field.
- `country` a required text field.
- `gender` a required text field.
- `sport` a required text field.
- `discipline` a required text field.
- `event` a required text field.

Next Get Help

Takes an argument, the name of a country. Returns the SQL command that will retrieve the number of gold medals that country has won in all Olympic games, aliased to the name `count`.

mostSummerWins

Takes an argument, the name of a country. Returns the SQL command that will retrieve the `year` that the county has won the most gold medals (only in Summer games), and how many medals were won, aliased to the name `count`.

mostWinterWins

Takes an argument, the name of a country. Returns the SQL command that will retrieve the `year` that the county has won the most gold medals (only in Winter games), and how many medals were won, aliased to the name `count`.

bestYear

Takes an argument, the name of a country. Returns the SQL command that will retrieve the `year` that country won the most Olympic gold medals, and how many medals were won, aliased to the name `count`.

bestDiscipline

Takes an argument, the name of a country. Returns the SQL command that will retrieve the `discipline` in which that country won the most Olympic gold medals, and how many medals were won, aliased to the name `count`.

bestSport

Takes an argument, the name of a country. Returns the SQL command that will retrieve the `sport` in which that country won the most Olympic gold medals, and how many medals were won, aliased to the name `count`.

bestEvent

Takes an argument, the name of a country. Returns the SQL command that will

Next Get Help

how many medals were won, aliased to the name `count`.

numberMenMedalists

Takes an argument, the name of a country. Returns the SQL command that will retrieve the number of men who have won Olympic gold medals for that country, aliased to the name `count`.

numberWomenMedalists

Takes an argument, the name of a country. Returns the SQL command that will retrieve the number of women who have won Olympic gold medals for that country, aliased to the name `count`.

mostMedaledAthlete

Takes an argument, the name of a country. returns the sql command that will retrieve the `name` of the athlete who won Olympic gold medals for that country, aliased to the name `count`.

orderedMedals

Takes three arguments, the name of the country and, optionally, a `field` to sort the results by and a boolean representing the `direction` the sort should go in. This function should return a SQL query that returns all fields for every Olympic gold medal won by the given country. When the `field` argument is present, the function should return a SQL query that orders the results by that field – ascending if the `direction` is `true` and descending if the `direction` is `false`.

Bonus: orderedSports

Takes three arguments, the name of the country and, optionally, a `field` to sort the results by and a boolean representing the `direction` the sort should go in. This function should return a SQL query that retrieves all the sports that country has received a Gold Medal in, additionally the query returned should return the number of times the given country received a medal in that sport, aliased to the name `count`

Next Get Help

Olympic gold medals were in that sport, aliased to the name ‘percent’. When the `field` argument is present, the function should return a SQL query that orders the results by that field – ascending if the `direction` is `true` and descending if the `direction` is `false`.

Testing

A testing suite has been provided for you, checking for all essential functionality and edge cases.

To run these tests, first, open the root project directory in your terminal. Then run `npm install` to install all necessary testing dependencies (if you haven’t already). Finally, run `npm test`. You will see a list of tests that ran with information about whether or not each test passed. After this list, you will see more specific output about why each failing test failed.

As you implement functionality, run the tests to ensure you are creating correctly named variables and functions that return the proper values. The tests will additionally help you identify edge cases that you may not have anticipated when first writing the functions.

Next Get Help