**Backend Coding Task**

**Submit task here:** [**https://forms.gle/s66PtRosMJxRtnxx7**](https://forms.gle/s66PtRosMJxRtnxx7)

**Task:**

The *data* folder contains 2 data files of 100 rows

**movies.csv**



**ratings.csv**



You can use any programming language & SQL database for this task.

* Create SQL Tables `***movies`*** & ***`ratings`***, and populate the CSV data into them.

Ans : CREATE TABLE movies (

tconst TEXT PRIMARY KEY,

titleType TEXT,

primaryTitle TEXT,

runtimeMinutes INTEGER

genres TEXT

);

CREATE TABLE ratings (

tconst TEXT PRIMARY KEY,

averageRating DOUBLE,

numVotes INTEGER

FOREIGN KEY (tconst) REFERENCES movies(tconst)

);

-- Importing the data or insertion of data in tables command

.mode csv

.import movies.csv movies

.mode csv

.import ratings.csv ratings

* Create an HTTP server with the following routes
* GET /api/v1/longest-duration-movies

This route returns as JSON the top 10 movies with the longest runTime The output should contain tconst, primaryTitle, runtimeMinutes & genres

* POST /api/v1/new-movie

This route takes JSON as input for new movie and saves it into the database On successful save, it returns “success”

* GET /api/v1/top-rated-movies

This route returns as JSON the movies with an averageRating > 6.0, in sorted order by averageRating

The output should contain tconst, primaryTitle, genre & averageRating.

* GET /api/v1/genre-movies-with-subtotals

Show a list of all movies genre-wise with Subtotals of their numVotes.

The calculation of subtotals should be done in SQL query; not the API code

*Output format* :

* POST /api/v1/update-runtime-minutes

Increment runtimeMinutes of all Movies using only SQL query (not in API code). Increment runtimeMinutes by :

15 if genre = Documentary 30 if genre = Animation

45 for the rest

Ans : const express = require('express');

const sqlite3 = require('sqlite3').verbose();

const app = express();

const port = 3000;

// Create a new SQLite database connection

const db = new sqlite3.Database('your\_database\_name.db');

// Route to get the top 10 movies with the longest runtime

app.get('/api/v1/longest-duration-movies', (req, res) => {

db.all('SELECT tconst, primaryTitle, runtimeMinutes, genres FROM movies ORDER BY runtimeMinutes DESC LIMIT 10', (err, rows) => {

if (err) {

console.error(err);

res.status(500).send('Internal Server Error');

} else {

res.json(rows);

}

});

});

// Route to save a new movie into the database

app.post('/api/v1/new-movie', express.json(), (req, res) => {

const { tconst, primaryTitle, runtimeMinutes, genres } = req.body;

db.run('INSERT INTO movies (tconst, primaryTitle, runtimeMinutes, genres) VALUES (?, ?, ?, ?)', [tconst, primaryTitle, runtimeMinutes, genres], function (err) {

if (err) {

console.error(err);

res.status(500).send('Internal Server Error');

} else {

res.send('success');

}

});

});

// Route to get movies with averageRating > 6.0, sorted by averageRating

app.get('/api/v1/top-rated-movies', (req, res) => {

db.all('SELECT tconst, primaryTitle, genres, averageRating FROM movies JOIN ratings ON movies.tconst = ratings.tconst WHERE averageRating > 6.0 ORDER BY averageRating DESC', (err, rows) => {

if (err) {

console.error(err);

res.status(500).send('Internal Server Error');

} else {

res.json(rows);

}

});

});

// Route to get movies genre-wise with subtotals of numVotes

app.get('/api/v1/genre-movies-with-subtotals', (req, res) => {

db.all('SELECT genres, COUNT(\*) AS totalMovies, SUM(numVotes) AS totalVotes FROM movies JOIN ratings ON movies.tconst = ratings.tconst GROUP BY genres', (err, rows) => {

if (err) {

console.error(err);

res.status(500).send('Internal Server Error');

} else {

res.json(rows);

}

});

});

// Route to increment runtimeMinutes of all movies based on genre using SQL query

app.post('/api/v1/update-runtime-minutes', (req, res) => {

db.run(`

UPDATE movies

SET runtimeMinutes = CASE

WHEN genres = 'Documentary' THEN runtimeMinutes + 15

WHEN genres = 'Animation' THEN runtimeMinutes + 30

ELSE runtimeMinutes + 45

END

`, function (err) {

if (err) {

console.error(err);

res.status(500).send('Internal Server Error');

} else {

res.send('success');

}

});

});

// Start the server

app.listen(port, () => {

console.log(`Server listening on port ${port}`);

});