```
var express = require("express");
var cors = require("cors");
var AWS = require("aws-sdk");

require("dotenv").config();
const app = express();
app.use(cors());
const port = 8080;

var s3 = new AWS.S3({
  accessKeyId: process.env.AWS_ID,
  secretAccessKey: process.env.AWS_KEY,
});
const bucketparams = {
  Bucket: "csu44000assignment220",
  Key: "moviedata.json",
};

AWS.config.update({
  region: "eu-west-1",
  endpoint: "https://dynamodb.eu-west-1.amazonaws.com",
  accessKeyId: process.env.AWS_ID,
  secretAccessKey: process.env.AWS_KEY,
});
const tableName = "Movies";
var dynamodb = new AWS.DynamoDB();
var docClient = new AWS.DynamoDB.DocumentClient();

var tableCreateParams = {
  TableName: tableName,
  KeySchema: [
    { AttributeName: "yr", KeyType: "HASH" },
    { AttributeName: "rating", KeyType: "RANGE" },
  ],
  AttributeDefinitions: [
    { AttributeName: "yr", AttributeType: "N" },
    { AttributeName: "rating", AttributeType: "N" },
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 5,
  },
};

async function fetchMovieData() {
  try {
    let data = await s3.getObject(bucketparams).promise();
    return JSON.parse(data.Body);
  } catch (err) {
    console.log("s3 error: " + err);
    return false;
  }
}

async function createTable() {
  try {
    await dynamodb.createTable(tableCreateParams).promise();
    return true;
  } catch (err) {
```

```
    console.log("ddb table creation error: ");
    console.log(err);
    return false;
  }
}

async function populateTable(moviedata) {
  try {
    moviedata.forEach(async function (movie) {
      let movieRating = 11;
      if (movie.info.rating) movieRating = movie.info.rating;
      let doc = {
        TableName: tableName,
        Item: {
          yr: movie.year,
          rating: movieRating,
          title: movie.title.toLowerCase(),
        },
      };
      try {
        await docClient.put(doc).promise();
      } catch (err) {
        console.log("error adding doc to db:");
        console.log(doc);
      }
    });
    return true;
  } catch (err) {
    console.log(err);
    return false;
  }
}

var created = true;
app.post("/create", async function (req, res) {
  if (created == false) {
    let moviedata = await fetchMovieData();
    if (moviedata == false) {
      res.status(500);
      res.json({ success: false, error: "failed to fetch movie data from s3"
});
      return;
    }
    console.log("movie data fetched from s3");
    let creation = await createTable();
    if (creation == false) {
      res.status(500);
      res.json({ success: false, error: "failed to create ddb table" });
      return;
    }
    console.log("table created");
    let initialised = false;
    while (!initialised) {
      let description = await dynamodb
        .describeTable({ TableName: tableName })
        .promise();
      if (description.Table.TableStatus == "ACTIVE") initialised = true;
    }
    console.log("table initialised");
    let populated = await populateTable(moviedata);
```

```
    console.log("table populated");
    created = true;
  }
  res.status(200);
  res.json({ success: true });
});

app.get("/query", async function (req, res) {
  let queryParams = {
    ExpressionAttributeValues: {
      ":y": { N: req.query.year },
      ":r": { N: req.query.rating },
      ":t": { S: req.query.name.toLowerCase() },
    },
    KeyConditionExpression: "yr = :y and rating >= :r",
    FilterExpression: "contains (title, :t)",
    TableName: tableName,
  };

  try {
    let matching = [];
    let result = await dynamodb.query(queryParams).promise();
    result.Items.forEach(function (movie) {
      matching.push({
        year: movie.yr.N,
        rating: movie.rating.N == 11 ? "-" : movie.rating.N,
        title: movie.title.S,
      });
    });
    res.status(200);
    res.json({ result: matching });
  } catch (err) {
    res.status(500);
    res.json({ result: [], error: err });
  }
});

app.delete("/destroy", async function (req, res) {
  dynamodb.deleteTable({ TableName: tableName }, function (err, data) {
    if (err) {
      res.status(500);
      res.json({ destroyed: false, error: err });
    } else {
      console.log("table deleted");
      created = false;
      res.status(200);
      res.json({ destroyed: true });
    }
  });
});

app.listen(port, function () {
  console.log(`listening on port ${port}`);
});

```