```
"use strict";
//Thomas Kelly - 16323455
const express = require("express");
const cors = require("cors");
const app = express();
const https = require("https");
var AWS = require("aws-sdk");
var fs = require('fs');

const port = 3000;
app.use(cors())


/*Project Outline
I outline below the functionality of my program!
1. Firstly, incoming url requests are dealt with using the app.get functions
2. For each requested function, the 'handle' function is called with a different 'choice' integer parameter to indicate what function should be performed
3. For /create, the dynamodb create table function is called. After this, a custom 'populateTable' function is called.
4. The populateTable function accesses the S3 bucket and pulls the moviedata.json file. The table is then populated with this movie data.
5. This function returns a promise which resolves after the table finishes populating, notifying the client when it is successfully finsihed.
6. The query part utilises the inbuilt docClient.query function along with a promise to query the database and then return query results to the client, as specified in the assignment outline.
7. The /destroy request results in usage of dynamodb.deleteTable function to delete the table.
8. I also decided to add an extra 'Top Rated' function. This uses the scan function to scan movies from since the year 2000 who have a rating over 8.5. The first ten are returned to the client.
*/

//DEAL WITH GET REQUESTS ========================================
app.get("/", (req, res) => res.send("")) //default condition
app.get("/create", (req, res) => handle(req,res,0))
app.get("/query/:movie/:year", (req, res) => handle(req,res,1))
app.get("/destroy", (req, res) => handle(req,res,2))
app.get("/top", (req, res) => handle(req,res,3))
//================================================================

let creds = new AWS.Credentials('AKIA2SHF3LWC6TKFAPMF', 'PJMGGJZufFrx3EiM5FC+Ok181TMUMEH59+1n4SiR'); //set AWS CREDENTIALS


//======================REQUEST HANDLER FUNCTION ========================================
function handle(req,res,function_choice,movie){
    let index =0;
    AWS.config.update({
        region: "eu-west-1",
        credentials: creds
      });


    var dynamodb = new AWS.DynamoDB({
        httpOptions: {
            agent: new https.Agent(
                {
```

```
50                      rejectUnauthorized: true,
51                      keepAlive: true,
52                      secureProtocol: "TLSv1_method"
53                  })
54          }
55      });
56
57  //============================CREATE================================================
58      if(function_choice == 0){ // deal with a CREATE REQUEST
59          console.log("Creating database...");
60          var params = { //outline table params
61              TableName : "Movies",
62              KeySchema: [
63                  { AttributeName: "year", KeyType: "HASH" },   //partition key
64                  { AttributeName: "title", KeyType: "RANGE"}   //sort key
65              ],
66              AttributeDefinitions: [
67                  { AttributeName: "year", AttributeType: "N" },
68                  { AttributeName: "title", AttributeType: "S" }
69              ],
70              BillingMode: "PAY_PER_REQUEST"
71          };
72
73          //Create table function
74          dynamodb.createTable(params, function(err, data) {
75              if (err) {
76                  console.error("Unable to create table.", JSON.stringify(err, null,
    2));
77              } else {
78                  console.log("> Created table.");
79
80                  //Call a function to populate the table
81                  //But wait to allow the creation of the table to finished with SLEEP
82                  //If the table was populated without sleep, most movies would not add
    for some reason.
83                  const sleep = (milliseconds) => {
84                      let prom =  new Promise(resolve => setTimeout(resolve,
    milliseconds));
85                      return prom;
86                    }
87                    console.log("Please wait while the table is populated...");
88
89                    sleep(1000).then(() => {
90                      populateTable().then((data) => { //populate table returns a
    promise resolution once the table has been fully populated
91                          res.send(data); //upon resolution, tell the client creation
    and loading of data has completed
92                      }).catch(()=> {
93                          res.send("> Error creating table.\n> Try deleting and trying
    again.");
94                      });
95                    }
96                  ).catch(() => {
97                      console.log("error");
98                  })
99              }
100
101         });
102
```

```
103 //============================QUERY=================================================
    ========
104     } else if (function_choice == 1){ // deal with query
105         var docClient = new AWS.DynamoDB.DocumentClient();
106         let year = parseInt(req.params.year); //get year
107         let movie = req.params.movie; //get movie title
108         let set=0,
109             result="> Query Results\n> Movies from " + year + " beginning with '" +
    movie + "'\n_____________________________\n";
110 
111         movie = movie.replace(/%20/g, " "); //remove %20 space indications inserted
    during transmission
112         console.log("> Searching database for: " + movie + " from " + year)
113 
114         //Specify the Query Parameters
115         var params = {
116             TableName : "Movies",
117             KeyConditionExpression: "#yr = :yyyy and begins_with(title, :ss) ", //
    this is the search condition (if year = 'year' and title begins with('substring'))
118             ExpressionAttributeNames:{
119                 "#yr": "year"
120             },
121             ExpressionAttributeValues: {
122                 ":yyyy": year,
123                 ":ss": movie
124             }
125         };
126 
127         let prom = new Promise ( (resolve,rej) => { //promise to send response to
    query
128             docClient.query(params, function(err, data) {
129                 if (err) {
130                     console.log("Unable to query. Error:", JSON.stringify(err, null,
    2));
131                     res.send("> There was a query error. Consider reloading the
    database.");
132                 } else { //the query is successfuly
133                     data.Items.forEach(function(item) { //for each item that matches
    criteria
134                         index = index + 1;
135                         set = 1;
136                         result = result.concat("\nTitle: ",item.title, "\nMovie Rank:
    ", item.info.rank,
137                                 "\nMovie Rating: ",item.info.rating, "\nDirector: ",
138                                 item.info.directors[0],
    "\n_____________________________\n"); //build a resulting string to return to client
139                         //build a result string to store all movies found in the
    query!
140                         if(index == data.Items.length){ //when we have found all
    movies that meet the query specifications, resolve the promise
141                             resolve(result); //resolve the promise
142                         } else if (data.Items.length == 0){
143                             reject("> There are no movies in the database matching
    that criteria."); //reject the promise.
144                         }
145 
146                     });
147                 }
148                 if(set == 0){
```

```
149                     console.log("That movie is not in the database."); //only print this after the query is completed. i.e. block it from running asynchronously
150                     res.send("> There are no movies in the database matching that criteria.");
151                 }
152                 set = 0;
153             });
154 
155         });
156 
157         prom.then((result) => { //promise resolves
158             console.log("Promise resolved. Sending response.\n", result);
159             res.send(result); //now send the response
160         });
161 
162 
163  //=============================================destroy=================================
=====
164     } else if(function_choice==2){ //otherwise it'll be a DESTROY REQUEST
165         console.log("Destroying database...");
166 
167         //specify table to destroy
168         var params = {
169             TableName : "Movies"
170         };
171 
172         dynamodb.deleteTable(params, function(err, data) {
173         if (err) {
174             console.error("Unable to delete table. Error JSON:", JSON.stringify(err, null, 2));
175         } else {
176             console.log("Table description JSON:", JSON.stringify(data, null, 2), "\n> Table has been deleted.");
177             res.send("> Database deleted")
178         }
179     });
180 
181 //=========================================SCAN: Top Rated Movie Extra Functionality=====================================
182     } else {
183         console.log("Pulling top rated movies");
184         var docClient = new AWS.DynamoDB.DocumentClient();
185         let result = "Top Ten\n", set=0;
186         let i=0, block= 0;
187 
188         //Specify the Query Parameters
189         var params = {
190             TableName: "Movies",
191             ProjectionExpression: "#yr, title, info.rating",
192             FilterExpression: "#yr between :start_yr and :end_yr and info.rating > :r1",
193             ExpressionAttributeNames: {
194                 "#yr": "year",
195             },
196             ExpressionAttributeValues: {
197                  ":start_yr": 2000,
198                  ":end_yr": 2014,
199                  ":r1": 8.5
200             }
201         };
```

```
202
203         let prom = new Promise ( (resolve,reject) => { //promise to send response to
    query
204             docClient.scan(params, onScan);
205
206             function onScan(err, data) {
207                 if (err) {
208                     console.error("Unable to scan the table. Error JSON:",
    JSON.stringify(err, null, 2));
209                 } else {
210                     // print all the movies
211                     console.log("Scan succeeded.");
212                     data.Items.forEach(function(movie) {
213                     i++;
214                     console.log(i, ". ", movie.title, "- rating:",
    movie.info.rating);
215                             result = result.concat(i, ". ", movie.title, " - rating:
    ", movie.info.rating, "\n");
216                     });
217
218                     if(i == 10 && block==1){ //when we have found all movies that
    meet the query specifications, resolve the promise
219                         resolve(result); //resolve the promise
220                     } else if (data.Items.length == 0){
221                         reject("> There are no movies in the database matching that
    criteria."); //reject the promise.
222                     }
223                     if (typeof data.LastEvaluatedKey != "undefined") {
224                         console.log("Scanning for more...");
225                         params.ExclusiveStartKey = data.LastEvaluatedKey;
226                         block=1;
227                         docClient.scan(params, onScan);
228                     }
229
230                 }
231             }
232
233         });
234
235
236         prom.then((result) => { //promise resolves
237             console.log("Promise resolved. Sending response.\n", result);
238             res.send(result); //now send the response
239         });
240
241     }
242
243 }
244 //=====================================================================
245
246 //FUNCTION TO POPULATE TABLE ==============================================
247 function  populateTable(){
248     //HERE WE NEED TO ACCESS THE BUCKET AND PULL THE JSON FILE
249     const s3 = new AWS.S3();
250     let file,ii=0;
251
252     //Access the bucket specifying parameters
253     return new Promise ((resolve, reject)=>{  //promise to send data to client once
    table has been filled
254     s3.getObject({
```

```
            Bucket: 'csu44000assignment2',
            Key: 'moviedata.json'
        },
        function(error, data){
            if(error != null){ //ERROR LOADING BUCKET DATA
                console.log("failed" + error);
            } else { //SUCESSFULLY LOADED BUCKET DATA
                console.log("> Loaded " + data.Body.length + " bytes ");
                file = data;
                console.log("Data:\n");
                var docClient = new AWS.DynamoDB.DocumentClient();

                var allMovies = JSON.parse(data.Body);

                    allMovies.forEach(function(movie) {
                        var params = {
                            TableName: "Movies",
                            Item: {
                                "year":  movie.year,
                                "title": movie.title,
                                "info": movie.info
                            }
                        };

                        docClient.put(params, function(err, data) { //Load these items into the Dynamodb

                            if (err) {
                                console.error("X", ii);
                            } else {
                                console.log(">", ii, ". ", movie.title, " | ", movie.year);
                                }
                            //check if we reached the last one

                            ii++;
                            if(ii == 4609){ //once the table finishes filling up
                                console.log("|| Finished loading table ||");
                                resolve("> Database created & successfully loaded");
                            }
                        });
                    });
                }
            });
        }
    )
}


app.listen(port, ()=> console.log("Server listening on port 3000"))
```