```javascript
const AWS = require("aws-sdk");
const express = require("express");
const path = require("path");

const app = express();
const PORT = 3000;

AWS.config.update({
    region: "us-east-1"
});

var dynamodb = new AWS.DynamoDB({
    region: "us-east-1",
    endpoint: 'https://dynamodb.us-east-1.amazonaws.com'
});

var docClient = new AWS.DynamoDB.DocumentClient();
var s3 = new AWS.S3({ region: "us-east-1" });

let publicPath = path.resolve(__dirname, "public");
app.use(express.static(publicPath));

app.get("/", (req, res) => {
    res.sendFile(path.join(__dirname + "/index.html"))
});


// Create the Movies table
app.post('/createDB', async (req, res) => {
    // don't create a table if one exists
    const tableExists = await doesTableExist();
    if(!tableExists){
        console.log("Creating a table in DB")
        var params = {
            TableName: "Movies",
            KeySchema: [
                { AttributeName: "year", KeyType: "HASH" },  //Partition key
                { AttributeName: "title", KeyType: "RANGE" }  //Sort key
            ],
            AttributeDefinitions: [
                { AttributeName: "year", AttributeType: "N" },
                { AttributeName: "title", AttributeType: "S" }
            ],
            ProvisionedThroughput: {
                ReadCapacityUnits: 50,
                WriteCapacityUnits: 50
            }
        };
        dynamodb.createTable(params, function (err, data) {
            if (err) {
                console.error("Unable to create table. Error JSON:",
JSON.stringify(err, null, 2));
            } else {
                console.log("Created table. Table description JSON:",
JSON.stringify(data, null, 2));
            }
        });

        // wait a while for Table to finish getting created
        await sleep(10000);
        // get movie data from s3 bucket and put into the table
        var bucketParams = {
            Bucket: 'csu44000assign2useast20',
            Key: 'moviedata.json'
        }
        var s3 = new AWS.S3();
        s3.getObject(bucketParams, function (err, data) {
            if (err) {
                console.log(err);
            } else {
                var allMovies = JSON.parse(data.Body.toString());
                allMovies.forEach(function (movie) {
                    var params = {
                        TableName: "Movies",
                        Item: {
                            "title": movie.title,
                            "year": movie.year,
                            "director":  movie.info.directors,
                            "rating": movie.info.rating,
                            "rank": movie.info.rank,
                        }
                    };
                    docClient.put(params, function (err, data) {
                        if (err) {
                            console.error("Unable to add movie", movie.title, ". Error JSON:", JSON.stringify(err, null, 2));
                        } else {
                            console.log("PutItem succeeded:", movie.title);
                        }
                    });
                });
                console.log("Table created successfully and filled with movie data.");
            }
        })
    }
    else{
        console.log(`The table 'Movies' already exists.`)
    }
});


// Query - Movies released in a given year, which begin with given string
app.get('/query/:title/:year', (req, res) => {
    console.log("Making query")
    var year = parseInt(req.params.year)
    var title = req.params.title
    var moviesList = [];
    var params = {
        TableName : "Movies",
        ProjectionExpression:"#yr, title, director, rating, #r",
        KeyConditionExpression: "#yr = :yyyy and begins_with (title, :movieTitle)",
        ExpressionAttributeNames:{
            "#yr": "year",
            "#r":"rank"
        },
        ExpressionAttributeValues: {
            ":yyyy": year,
            ":movieTitle": title
        }
    };

    docClient.query(params, function(err, data) {
        if (err) {
            console.log("Unable to query. Error:", JSON.stringify(err, null, 2));
        } else {
            data.Items.forEach(function(item) {
                console.log(item.year + ' ' + item.title);
                moviesList.push(
                    {
                        Title: item.title,
                        Year : item.year,
                        Director: item.director,
                        Rating: item.rating,
                        Rank: item.rank,
                    }
                )
            });
            res.json(moviesList)
            console.log("Query succeeded.");
        }
    });
});


// Delete the Movies table
app.post('/destroyDB', async (req, res) => {
    // only delete table if it exists
    const tableExists = await doesTableExist();
    if(tableExists){
        console.log("Destroying the table.");
        var params = {
            TableName : "Movies",
        };
        dynamodb.deleteTable(params, function(err, data) {
            if (err) {
                console.error("Unable to delete table. Error JSON:",
JSON.stringify(err, null, 2));
            } else {
                console.log("Deleted table. Table description JSON:",
JSON.stringify(data, null, 2));
            }
        });
    }
    else{
        console.log(`The table 'Movies' doesn't exist.`);
    }
});


// helper function to verify if Movie table exists or not
const doesTableExist = async () => {
    const exists = await new Promise(resolve => {
        dynamodb.describeTable({ TableName: "Movies" }, (err, data) => {
            if (err) {
                resolve(false);
            } else {
                resolve(true);
            }
        });
    });
    return exists;
};


// helper function to allow time for table to be created before putting data into it
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}


app.listen(PORT, () => {console.log(`Listening on ${PORT}`)});
```