Cloud Engineering Intern

Requirements –

- This assignment can be done in AWS or in GCP with equivalent resources (Storage, Cloud SQL, Cloud Functions, etc).
- Deployment of all the cloud resources (S3, Lambda, Database) must be automated using any Infrastructure as Code tool (CDK, Pulumi, Cloudformation, Terraform, etc) of your choice.
- API can be hosted on any platform of your choice.
- The database can be deployed in a public subnet for this assignment.
- Please don't store any public endpoint details or credentials on a public GitHub repo.
- Candidates should send a video of the working functionality
- The API code should be written in NodeJS or typescript and with proper linting.
- API endpoint should be protected by basic authentication, API key, or OAuth
- The lambda to API calls should happen in batches of 100. So, for a CSV with 1000 records, 10 API calls should be made

1. **Develop a data API that gets the JSON input and stores the data in a database hosted on a cloud provider. Following is an example of the input -**

```
{
 "id": "12345"
 "name": "John",
 "surname": "Doe",
 "dob": "11/11/2011",
 "gender": "male"
}
```

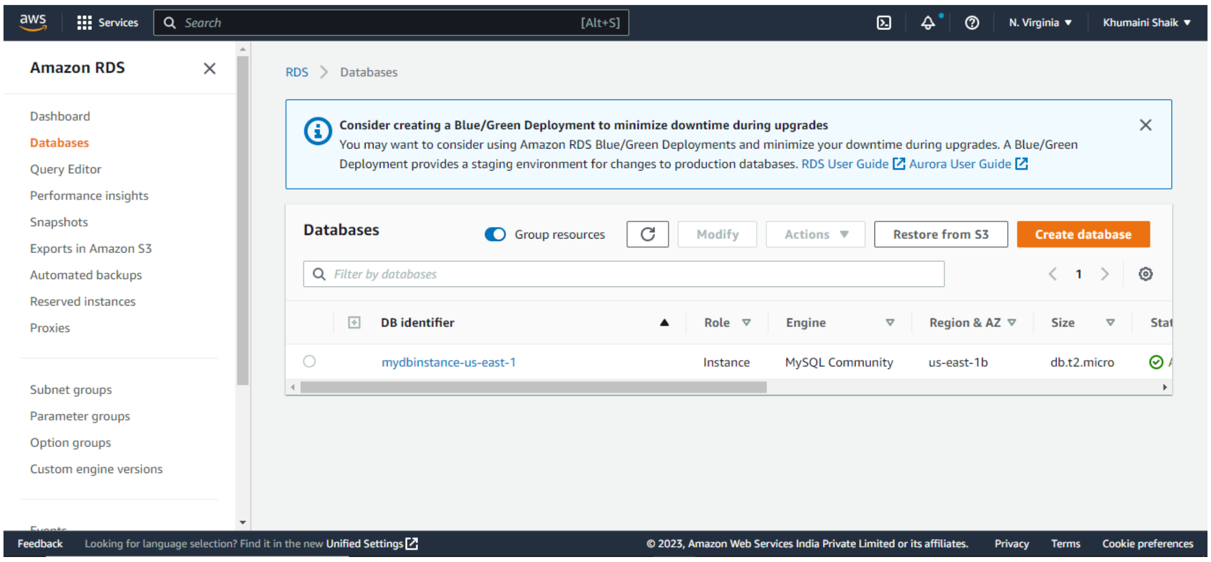
CloudFormation Template for launching AWS RDS MySQL:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: This stack creates an RDS MySql Instance
Parameters:
  MasterUsername:
    Description: Database administration name.
    Type: String
    Default: rdsroot
  MasterUserPassword:
    NoEcho: 'true'
    Description: Database administration password.
    Type: String
    MinLength: '8'
    AllowedPattern: "[a-zA-Z0-9!?]*"
    ConstraintDescription: Must only contain upper and lowercase letters and numbers
  TcpPort:
    Description: Enter RDS Listening TCP Port number.
    Type: Number
    Default: '3306'
  DatabaseName:
    Type: String
Resources:
  MyRDSDBInstance:
    Type: AWS::RDS::DBInstance
    DeletionPolicy: Snapshot
    Properties:
      AllocatedStorage: "20"
      DBName: !Ref DatabaseName
      DBInstanceClass: db.t2.micro
      DBInstanceIdentifier: !Join ["-",["MyDbInstance",!Ref "AWS::Region"]]
      Engine: MySQL
      MasterUsername: !Ref MasterUsername
      MasterUserPassword: !Ref MasterUserPassword
      Port:
        Ref: TcpPort
      PubliclyAccessible: True
      StorageEncrypted: 'false'
      StorageType: gp2
      AvailabilityZone: !Select [1,!GetAZs ""]
      Tags:
```

- Key: Owner

  Value: Abdul Khumaini

Database Created:



aws | Services | Search [Alt+S] | N. Virginia | Khumaini Shaik

Amazon RDS

Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies

Subnet groups
Parameter groups
Option groups
Custom engine versions

RDS > Databases

Consider creating a Blue/Green Deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide

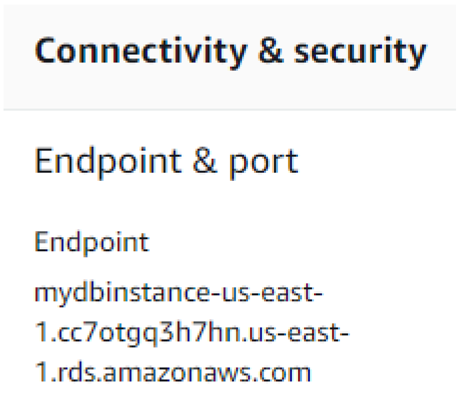
Databases | Group resources | Modify | Actions | Restore from S3 | Create database

Filter by databases

DB identifier	Role	Engine	Region & AZ	Size	Stat
mydbinstance-us-east-1	Instance	MySQL Community	us-east-1b	db.t2.micro	✓

Feedback Looking for language selection? Find it in the new Unified Settings © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Database Endpoint:



Connectivity & security

Endpoint & port

Endpoint

mydbinstance-us-east-1.cc7otgq3h7hn.us-east-1.rds.amazonaws.com

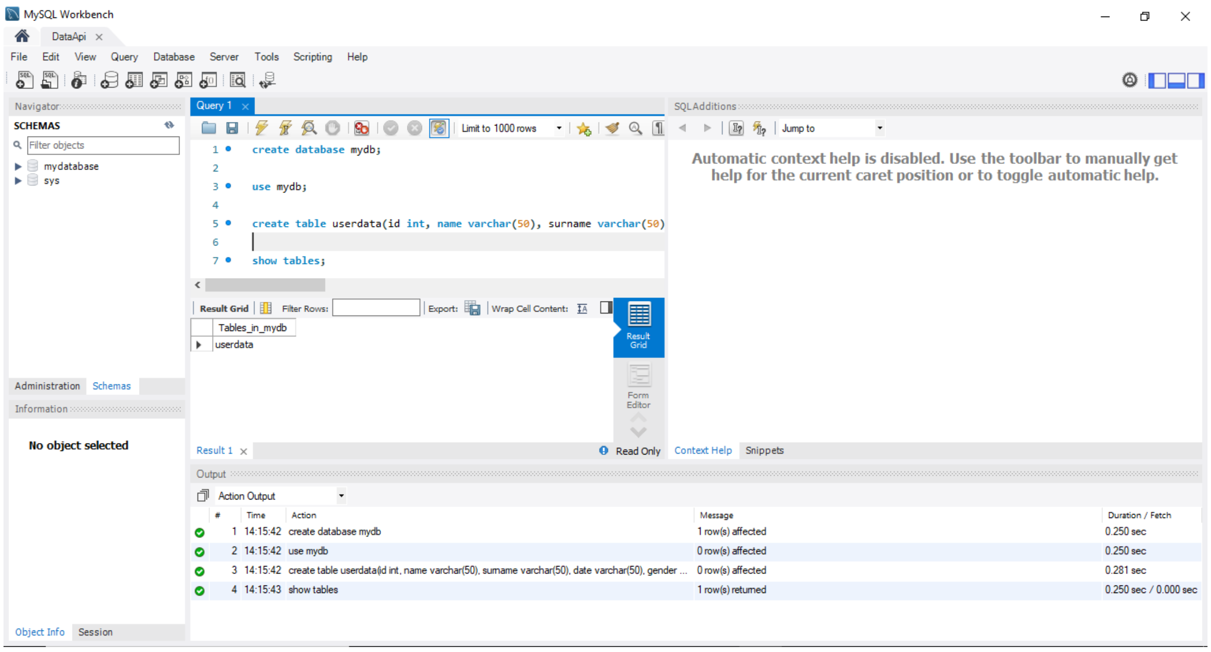
Connection created through MySQL Workbench and required sql queries are executed:

create database mydb;

use mydb;

create table userdata(id int, name varchar(50), surname varchar(50), date varchar(50), gender varchar(10));

show tables;



```
create database mydb;

use mydb;

create table userdata(id int, name varchar(50), surname varchar(50)

show tables;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
Tables_in_mydb
userdata
```

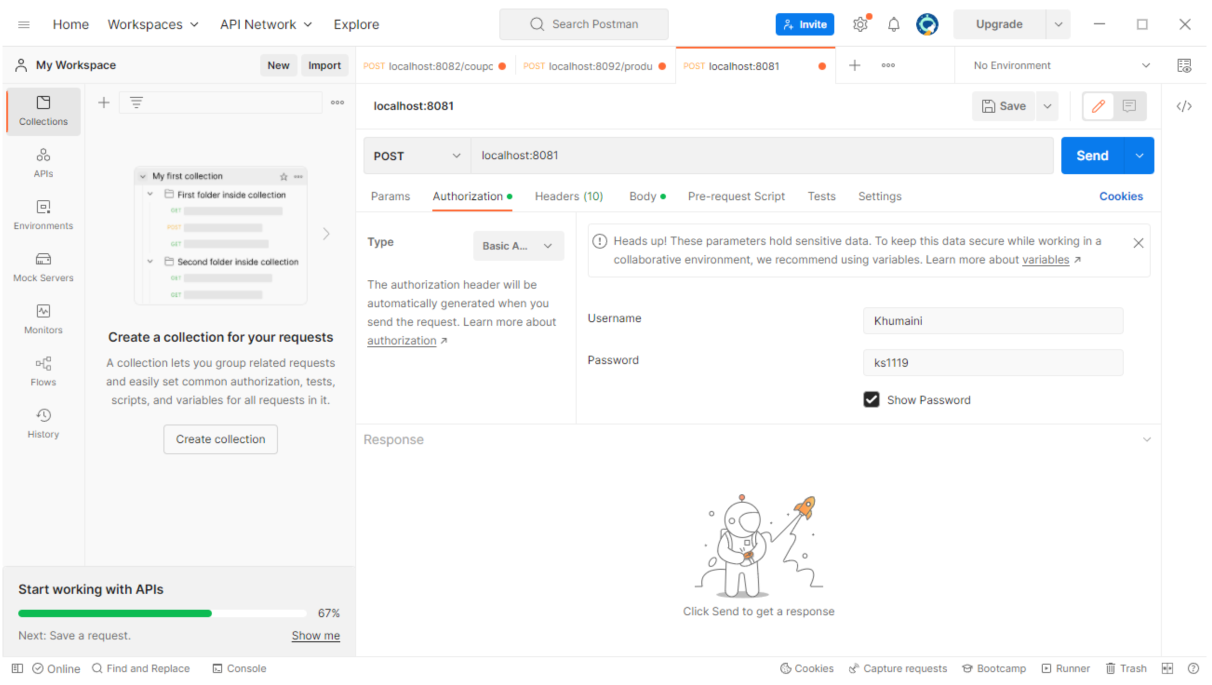
```
#  Time      Action                                                                              Message              Duration / Fetch
1  14:15:42  create database mydb                                                                1 row(s) affected    0.250 sec
2  14:15:42  use mydb                                                                            0 row(s) affected    0.250 sec
3  14:15:42  create table userdata(id int, name varchar(50), surname varchar(50), date varchar(50), gender ...  0 row(s) affected    0.281 sec
4  14:15:43  show tables                                                                         1 row(s) returned    0.250 sec / 0.000 sec
```

API to get JSON data and store it in AWS RDS, API is enabled with basic authentication to protect against attacks:

```
const express = require('express');

const app = express();

const mysql = require('mysql');

const basicAuth = require('basic-auth');


async function dbcon(req,res){

    try{

        var connection = mysql.createConnection({

            host: 'mydbinstance-us-east-1.cc7otgq3h7hn.us-east-1.rds.amazonaws.com',

            user: 'admin',
            password: 'admin1234',
            database: 'mydb',
            port:3306
        });
        connection.connect((err) => {
            if (err) {
                console.error('Database connection failed: ' + err.stack);
                return;
            }
            console.log('connected!!');
            var que = `insert into userdata 
values(${req.body.id},'${req.body.name}','${req.body.surname}','${req.body.dob}','${req.bo
dy.gender}')`;
            connection.query(que,(err,res)=>{
                if(err){
                    console.error(err);
                }
                console.log('1 record inserted');
            })
        });
    }
    catch(error){
        console.error(error);
    }
}

var auth = function (req, res, next) {
  var user = basicAuth(req);
  if (!user || !user.name || !user.pass) {
    res.set('WWW-Authenticate', 'Basic realm=Unauthorized');
    res.sendStatus(401);
```

```
   return;
  }
  if (user.name === 'Khumaini' && user.pass === 'ks1119') {
   next();
  } else {
   res.set('WWW-Authenticate', 'Basic realm=Unauthorized');
   res.sendStatus(401);
   return;
  }
}

app.use(express.json());
app.post('/', auth, (req, res) => {
   res.write('{authenticated:true}\n');
   dbcon(req,res);
   res.end('1 record inserted');
});
app.listen(8081, () => console.log("Server started at 127.0.0.1://8081"));
```

Authentication:



POST localhost:8082/coupc · POST localhost:8092/produ · POST localhost:8081

No Environment

localhost:8081

POST localhost:8081

Type: Basic A...

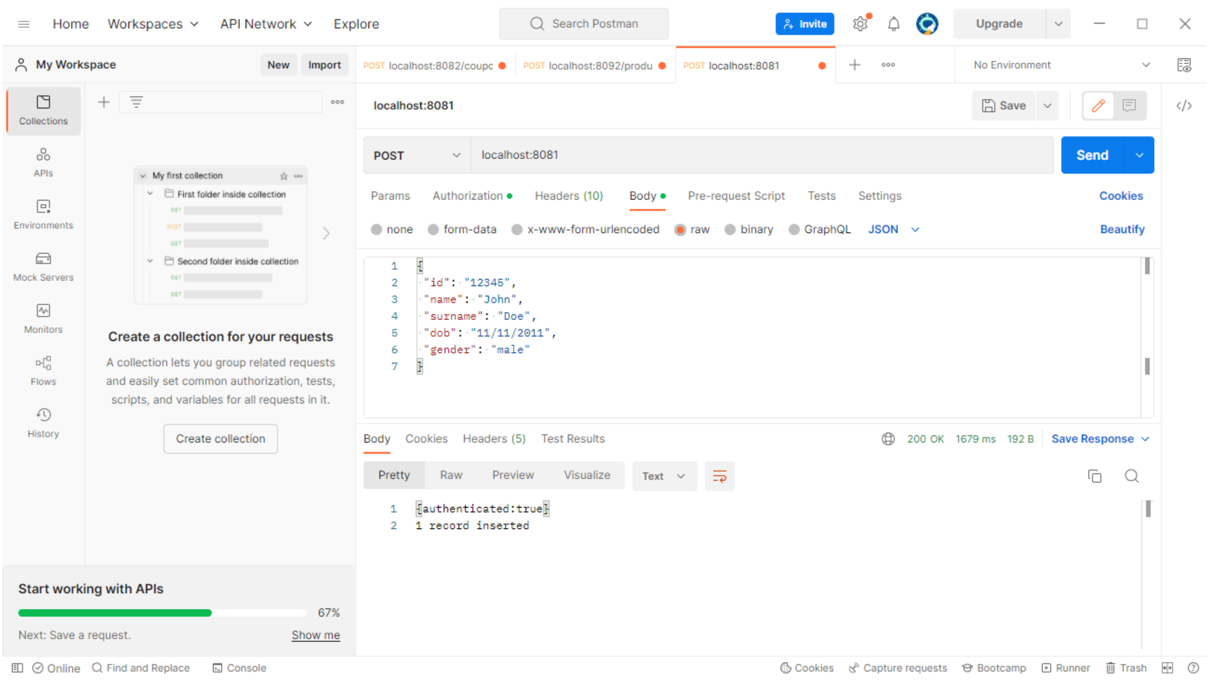
The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Username	Khumaini
Password	ks1119

☑ Show Password

API testing:



localhost:8081

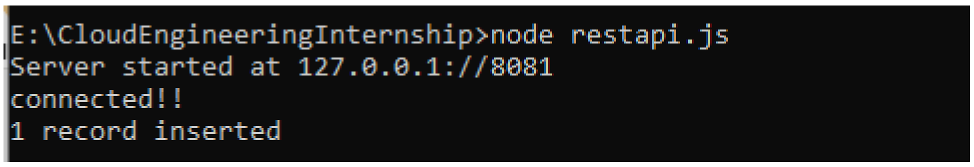
POST localhost:8081

```
{
"id": "12345",
"name": "John",
"surname": "Doe",
"dob": "11/11/2011",
"gender": "male"
}
```

200 OK 1679 ms 192 B

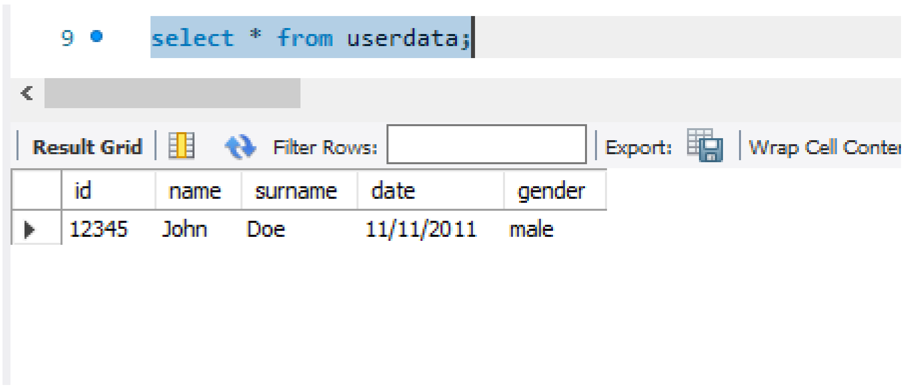
```
{authenticated:true}
1 record inserted
```

Server Logs:



```
E:\CloudEngineeringInternship>node restapi.js
Server started at 127.0.0.1://8081
connected!!
1 record inserted
```

Database:



```
9 select * from userdata;
```

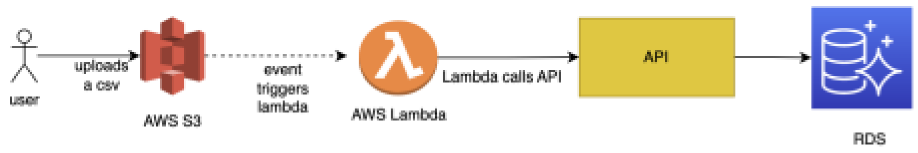
id	name	surname	date	gender
12345	John	Doe	11/11/2011	male

2. **Create an S3 bucket on which a user can manually upload CSV files from the AWS console. Following is the sample of the CSV -**

```
id,name,surname,dob,gender

12357,John,Doe,11/11/2011,male

12356,Doe,Jane,01/01/2001,female

12354,Jes,Dam,11/12/2013,male
```

- Once the file is uploaded, it should create an event that should trigger a lambda function.
- The lambda function will read the uploaded file, parse the content of that file and post the data API.

Following is the high-level sequence diagram –



CloudFormation Template to launch S3, Lambda and to configure the EventNotification (Trigger) between them whenever a new object is created:

```
AWSTemplateFormatVersion: '2010-09-09'

Description: This stack creates a lambda for s3 trigger


Parameters:

  BucketName:

    Type: String

    Default: 557bucket1


Resources:

  Bucket:

    Type: AWS::S3::Bucket

    Properties:

      BucketName: !Ref BucketName

      NotificationConfiguration:

        LambdaConfigurations:

          - Event: 's3:ObjectCreated:*'
          Filter:
            S3Key:
              Rules:
                - Name: suffix
                  Value: .csv
          Function: !GetAtt Lambda.Arn

Lambda:
  Type: AWS::Lambda::Function
  Properties:
    FunctionName: 'mynodelambda1'
    Handler: index.handler
    Role: arn:aws:iam::573351626142:role/service-role/mynodelambda-role-cd8ibrmh
    Code:
      S3Bucket: '558bucket1'
      S3Key: 'code/lambdards.zip'
    Runtime: "nodejs16.x"
    Timeout: 300
    MemorySize: 512
    TracingConfig:
      Mode: Active

S3InvokeLamdaPermission:
  Type: AWS::Lambda::Permission
  Properties:
    Action: lambda:InvokeFunction
    FunctionName: !Ref Lambda
    Principal: s3.amazonaws.com
    SourceArn: !Sub arn:aws:s3:::${BucketName}
```

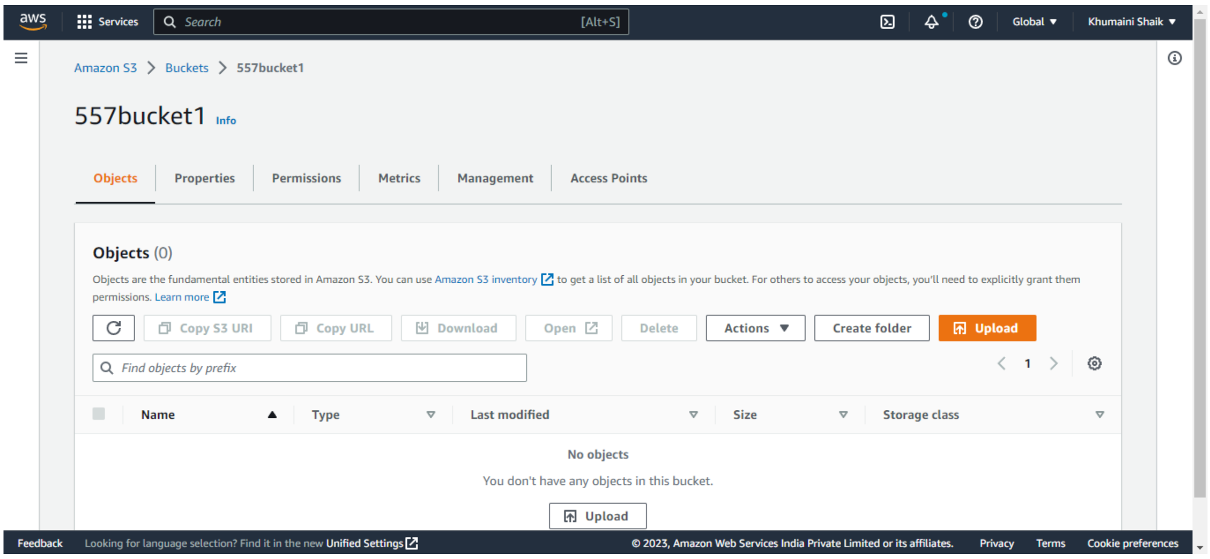
Lambda API to get called when a csv object is created in S3 Bucket persistent data in RDS (created in problem-1):

[Lambda API calls are batched in 100 to write 100 records from csv to the database if no of records of csv are greater than 1000]

```
const aws = require('aws-sdk');
const s3 = new aws.S3({
    apiVersion: '2006-03-01',
    region: 'us-east-1',
    accessKeyId: #accesskey,
    secretAccessKey: '#secretKey
});

const mysql = require('mysql');
const readline = require('readline');
let data = [];

const connection = mysql.createConnection({
    host: 'mydbinstance-us-east-1.cc7otgq3h7hn.us-east-1.rds.amazonaws.com',
    user: 'admin',
    password: 'admin1234',
    database: 'mydb',
    port:3306
});

exports.handler = async (event,context,callback) => {
    // TODO implement
    var bucketName = event["Records"][0]["s3"]["bucket"]["name"];
    var bucketObject = event["Records"][0]["s3"]["object"]["key"];
    const bucketParams = {
        Bucket: `${bucketName}`,
        Key:`${bucketObject}`
    };
    console.log(bucketParams);
    try {
        var tbname = bucketObject.split('.')[0];
        var crque = `create table ${tbname} (id int, name varchar(50),surname varchar(50), dob
varchar(50), gender varchar(10))`;
        console.log(crque);
        connection.query(crque, (err, res) => {
            if (err) {
                console.error(err);
            }
            console.log('table created');
        });

        const s3ReadStream = s3.getObject(bucketParams).createReadStream();

        const rl = readline.createInterface({
            input: s3ReadStream,
            terminal: false
        });

        var myReadPromise = new Promise((resolve, reject) => {
            var c = 0;

            rl.on('line', (line) => {
                c++;
                line = line.split(',');
                data.push(line);
                if (c % 100 == 0) {
                    var i = (c-100);
                    while (i < data.length) {
```

```
                    var que = `insert into ${tbname}
values(${data[i][0]},'${data[i][1]}','${data[i][2]}','${data[i][3]}','${data[i][4]}')`;
                    console.log(que);
                    connection.query(que, (err, res) => {
                        if (err) {
                            console.error(err);
                        }
                    });
                    i+=1;
                }
                console.log((i - 1) + ' records inserted');
            }

        });
        rl.on('error', () => {
            console.log('error');
        });
        rl.on('close',()=>{
            var i=1;
           if(c<100){
               while (i < data.length) {
                    var que = `insert into ${tbname}
values(${data[i][0]},'${data[i][1]}','${data[i][2]}','${data[i][3]}','${data[i][4]}')`;
                    console.log(que);
                    connection.query(que, (err, res) => {
                        if (err) {
                            console.error(err);
                        }
                    });
                    i+=1;
                }
                console.log((i - 1) + ' records inserted');
            }
          });
        });
        await myReadPromise;
    } catch (err) {
        console.log(err);
    }

};
```

Uploading a .csv file to the bucket created:



Amazon S3 > Buckets > 557bucket1

557bucket1 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

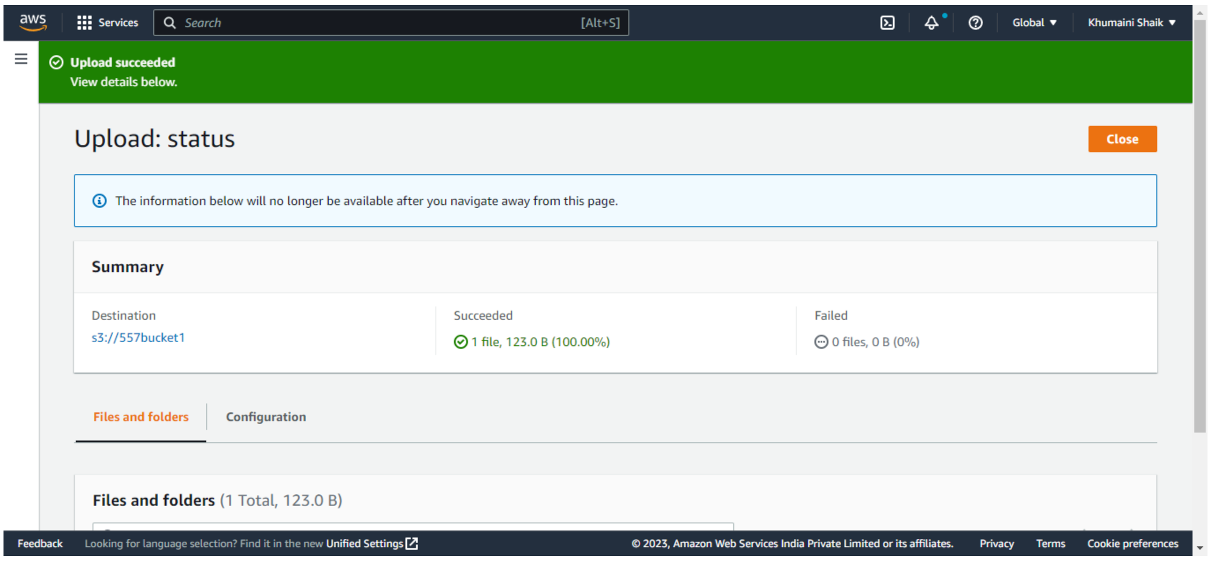
Find objects by prefix

Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.

Upload



Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

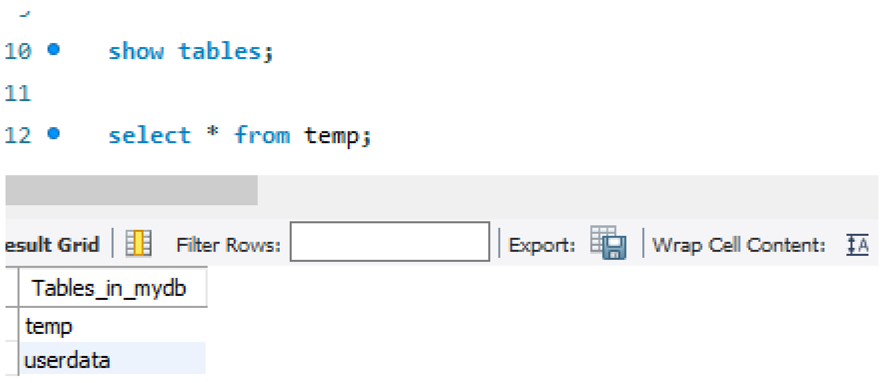
Summary

Destination	Succeeded	Failed
s3://557bucket1	1 file, 123.0 B (100.00%)	0 files, 0 B (0%)

Files and folders | Configuration

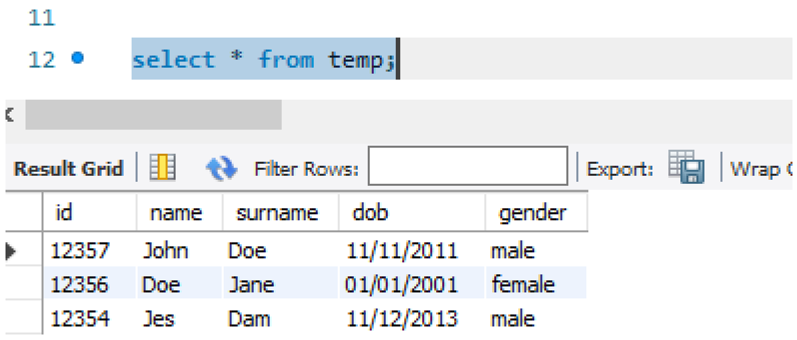
Files and folders (1 Total, 123.0 B)

Verifying the database:



```sql
show tables;

select * from temp;
```

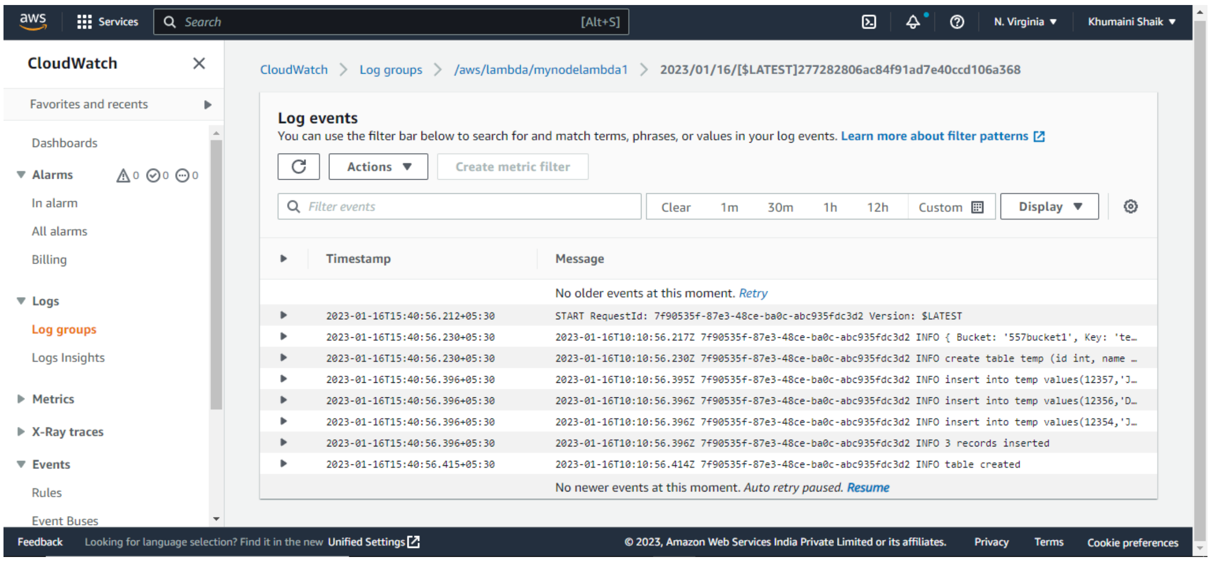
Tables_in_mydb
temp
userdata



```sql
select * from temp;
```

id	name	surname	dob	gender
12357	John	Doe	11/11/2011	male
12356	Doe	Jane	01/01/2001	female
12354	Jes	Dam	11/12/2013	male

Verifying the CloudWatch Logs to view the output of the Lambda function:



CloudWatch > Log groups > /aws/lambda/mynodelambda1 > 2023/01/16/[$LATEST]277282806ac84f91ad7e40ccd106a368

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                        Message
                                 No older events at this moment. Retry
2023-01-16T15:40:56.212+05:30    START RequestId: 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 Version: $LATEST
2023-01-16T15:40:56.230+05:30    2023-01-16T10:10:56.217Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO { Bucket: '557bucket1', Key: 'te…
2023-01-16T15:40:56.230+05:30    2023-01-16T10:10:56.230Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO create table temp (id int, name …
2023-01-16T15:40:56.396+05:30    2023-01-16T10:10:56.395Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO insert into temp values(12357,'J…
2023-01-16T15:40:56.396+05:30    2023-01-16T10:10:56.396Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO insert into temp values(12356,'D…
2023-01-16T15:40:56.396+05:30    2023-01-16T10:10:56.396Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO insert into temp values(12354,'J…
2023-01-16T15:40:56.396+05:30    2023-01-16T10:10:56.396Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO 3 records inserted
2023-01-16T15:40:56.415+05:30    2023-01-16T10:10:56.414Z 7f90535f-87e3-48ce-ba0c-abc935fdc3d2 INFO table created
                                 No newer events at this moment. Auto retry paused. Resume
```