FACULTY OF COMPUTER SCIENCE

ASSIGNMENT 1: PART C

In
The Class of

CSCI5710: SERVERLESS DATA PROCESSING

by

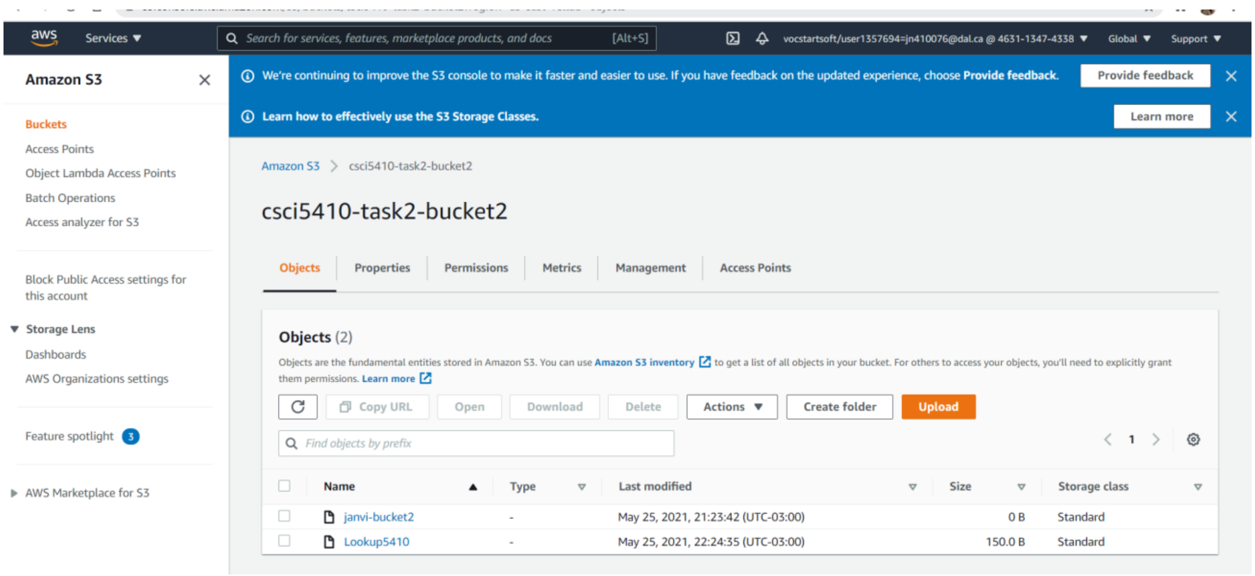
Janvi Patel [B00863421]

Submitted to
Prof. Saurabh Dey
Department of Computer Science
Dalhousie university.

Date: 25th May 2021

Part C. AWS RDS database service experiment

(i) screenshots of the S3 bucket with Lookup table:



We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**.

Learn how to effectively use the S3 Storage Classes.

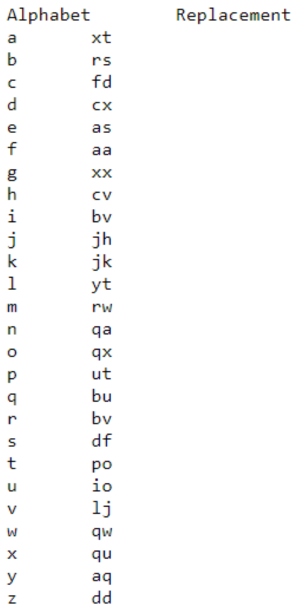
csci5410-task2-bucket2

Objects (2)

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**

□	Name ▲	Type	Last modified	Size	Storage class
□	janvi-bucket2	-	May 25, 2021, 21:23:42 (UTC-03:00)	0 B	Standard
□	Lookup5410	-	May 25, 2021, 22:24:35 (UTC-03:00)	150.0 B	Standard

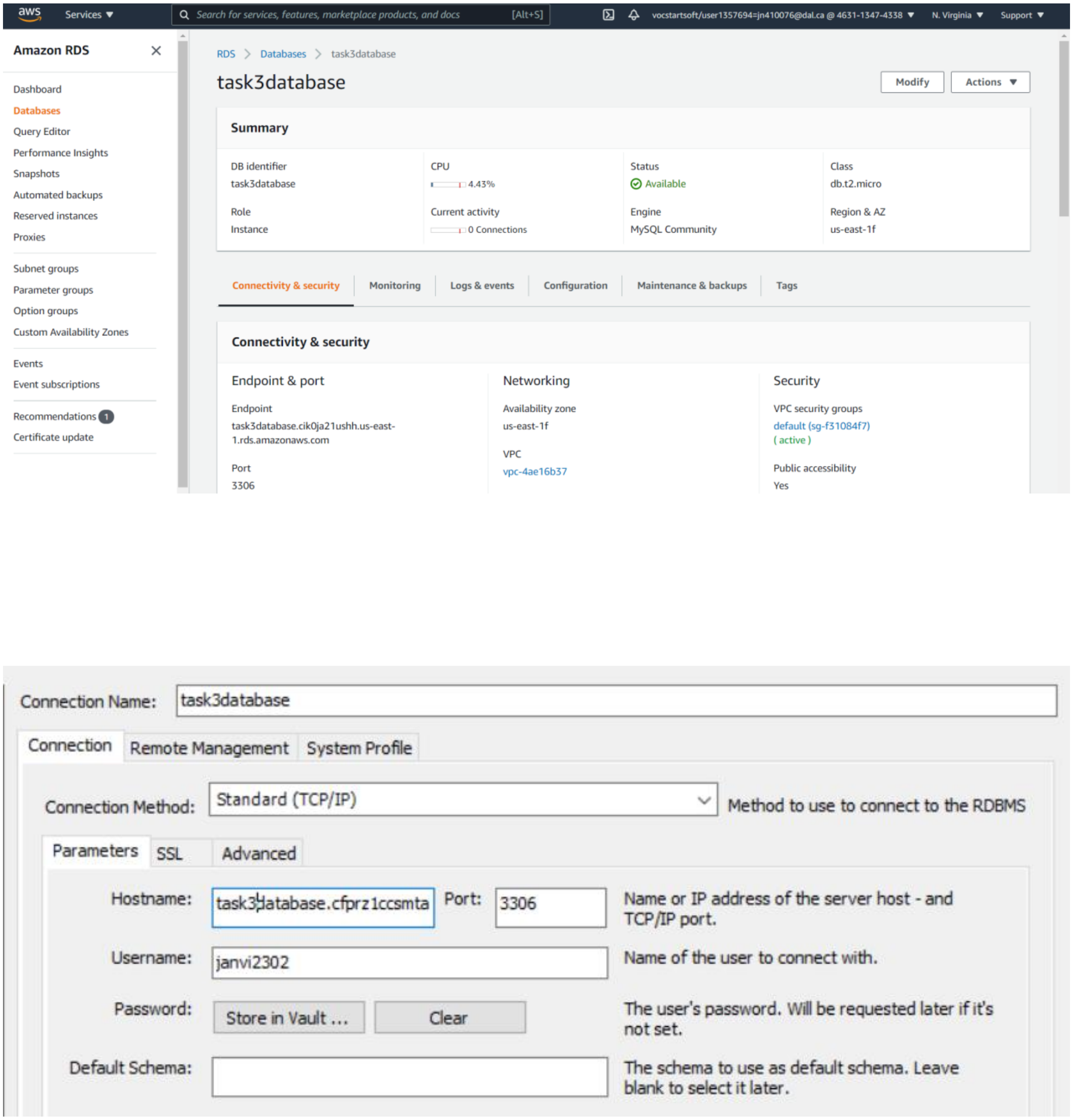
Figure 1: AWS S3 with Lookup Table file



```
Alphabet        Replacement
a       xt
b       rs
c       fd
d       cx
e       as
f       aa
g       xx
h       cv
i       bv
j       jh
k       jk
l       yt
m       rw
n       qa
o       qx
p       ut
q       bu
r       bv
s       df
t       po
u       io
v       lj
w       qw
x       qu
y       aq
z       dd
```

Figure 2: Content from lookup table after making it public

(ii) screenshots of the RDBMS



task3database

Summary

DB identifier: task3database
CPU: 4.43%
Status: Available
Class: db.t2.micro
Role: Instance
Current activity: 0 Connections
Engine: MySQL Community
Region & AZ: us-east-1f

Connectivity & security

Endpoint & port
Endpoint: task3database.cik0ja21ushh.us-east-1.rds.amazonaws.com
Port: 3306

Networking
Availability zone: us-east-1f
VPC: vpc-4ae16b37

Security
VPC security groups: default (sg-f31084f7) (active)
Public accessibility: Yes

Connection Name: task3database

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: task3database.cfprz1ccsmta Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: janvi2302 — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

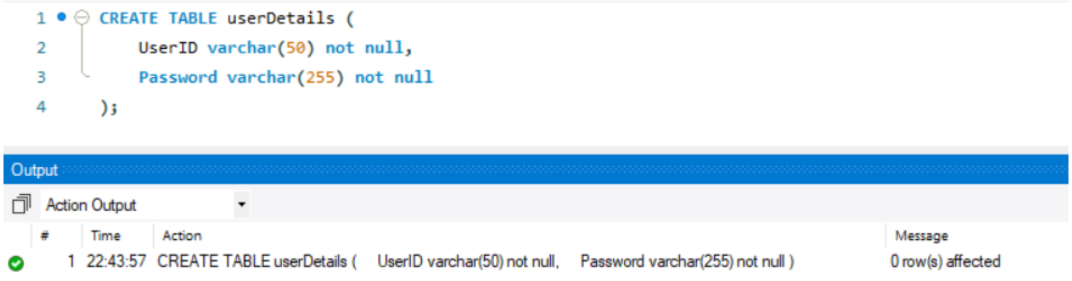
Figure 3: RDBMS creation

(iii) program code:

```
package CSCI5409_Assignment1.AWSproject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.S3Object;

public class Task3
{
            //configuration details for JDBC connection
    private static String driverName = "com.mysql.cj.jdbc.Driver";
    private static String database = "jdbc:mysql://task3database.cfprz1ccsmta.ap-south
1.rds.amazonaws.com:3306/aws_testing?user=janvi&password=janvi123";
    private static String username = "janvi";
    private static String password = "janvi123*";
    private static Connection connection;

    //establish connection with database
    protected static Connection getConnection(String database,String username,String password)
    {
        try
        {
            Class.forName(driverName);
            try
            {
                 connection = DriverManager.getConnection(database, username, password);                    //connecting to database
            }
            catch (SQLException ex)
            {
                  System.out.println("Failed to create the database connection.");
            }
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println("Driver not found.");
        }
        return connection;
    }

    //performs insert, encryption, decryption and select operation
    public static void main(String[] args) throws IOException
   {
            connection = getConnection(database,username,password);
            if (connection != null)
            {
                    java.sql.Statement stmt;
                    try
                    {
                            stmt = connection.createStatement();
        System.out.println("\nConnection established successfully !!");

        //bucket and key name to read from lookup file
        String bucketName = "csci5410-task2-bucket2";
        String          key = "Lookup5410";

        //user name and password to insert into database
        String UserID = "janvi";
        String user_password = "jan";

        System.out.print("\nPassword before encryption:" +user_password);

        //hash maps to store encryption and decryption letters
        HashMap<String, String> LookupEncryption = new HashMap<String, String>();
        HashMap<String, String> LookupDecryption = new HashMap<String, String>();

                    /********* Password Encryption **********/

        //reading from file located at AWS S3
        @SuppressWarnings("deprecation")
                    AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        S3Object object = s3Client.getObject(new GetObjectRequest(bucketName, key));
        InputStream objectData = object.getObjectContent();
        BufferedReader reader = new BufferedReader(new InputStreamReader (objectData));

       //splitting the file content and storing the data into lookup encryption hashmap and look up decryption
hash map
        String readLine = reader.readLine();
        while(readLine!=null)
        {
            String[] ary = readLine.split("            ");
            LookupEncryption.put(ary[0], ary[1]);
            LookupDecryption.put(ary[1], ary[0]);
            readLine = reader.readLine();
        }

        String encryptedPassword = "";

        //generating encrypted password
        for(int i = 0; i<user_password.length(); i++)
        {
            encryptedPassword += LookupEncryption.get(String.valueOf(user_password.charAt(i)));
        }
        System.out.print("\nEncripted Password:" +encryptedPassword);

                    /********* Inserting user details into database **********/
        String sqlquery = " INSERT INTO userDetails(UserID,Password) VALUES ('"+UserID+"',
        '"+encryptedPassword+"');";

         if(stmt.executeUpdate(sqlquery) == 1)
        {
                System.out.print("\nSuccessful insertion into database");
        }

        /********* displaying password for a particular userID**********/
        String sqlselect = "SELECT Password from userDetails where userDetails.UserID = '"+UserID+"';";
        String decryptedPassword = "";
        String passwordDatabase = "";

        //getting password from result set
        ResultSet rs = stmt.executeQuery(sqlselect);
        while(rs.next())
```

```
                    {
                        passwordDatabase =  rs.getString("Password");
                    }

                    //decryption of password
                    for(int i = 0; i<passwordDatabase.length(); i++)
                    {
                        decryptedPassword +=
                        LookupDecryption.get(String.valueOf(passwordDatabase.charAt(i))+String.valueOf(passwordDat
                        abase.charAt(i+1)));
                        i++;
                    }
                    System.out.print("\nSelecting userdetails with userID :" +UserID);
                    System.out.print("\n\tUserID:" +UserID);
                    System.out.print("\n\tDecripted Password:" +decryptedPassword);
                    objectData.close();
                        }
                    catch (SQLException e)
                   {
                               e.printStackTrace();
                    }
               }
          }
}
```

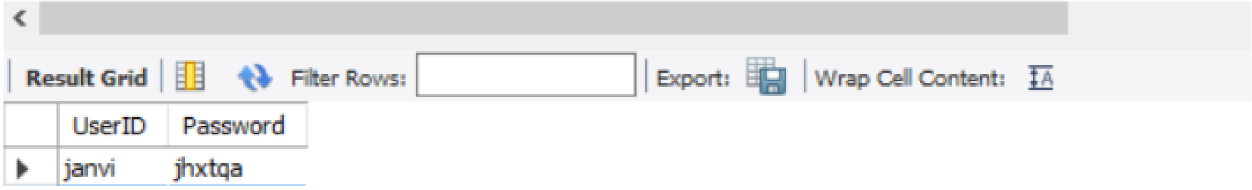
(iv) output – displays password before encryption, after encryption, and after decryption.



```
CREATE TABLE userDetails (
    UserID varchar(50) not null,
    Password varchar(255) not null
);
```

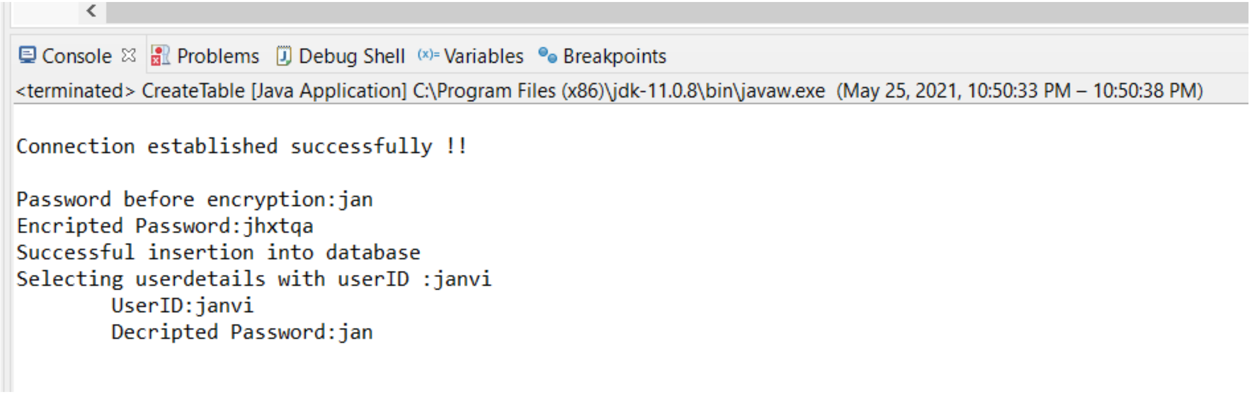
```
#   Time      Action                                                                              Message
1   22:43:57  CREATE TABLE userDetails (   UserID varchar(50) not null,   Password varchar(255) not null )   0 row(s) affected
```

Figure 4: creating datatable



UserID	Password
janvi	jhxtqa

Figure 5: inserting details to datatable



```
<terminated> CreateTable [Java Application] C:\Program Files (x86)\jdk-11.0.8\bin\javaw.exe  (May 25, 2021, 10:50:33 PM – 10:50:38 PM)

Connection established successfully !!

Password before encryption:jan
Encripted Password:jhxtqa
Successful insertion into database
Selecting userdetails with userID :janvi
        UserID:janvi
        Decripted Password:jan
```

Figure 6: password before encryption, after encryption and after decryption

Reference: https://stackoverflow.com/questions/28568635/read-aws-s3-file-to-java-code