

Assignment - 3

Assignment Date	29 September 2022
Student Name	Bharath Raaj B
Student Roll No	812419104014
Maximum Marks	2 Marks

Question-1:

Create User table with user email, username, roll number, password.

Solution:

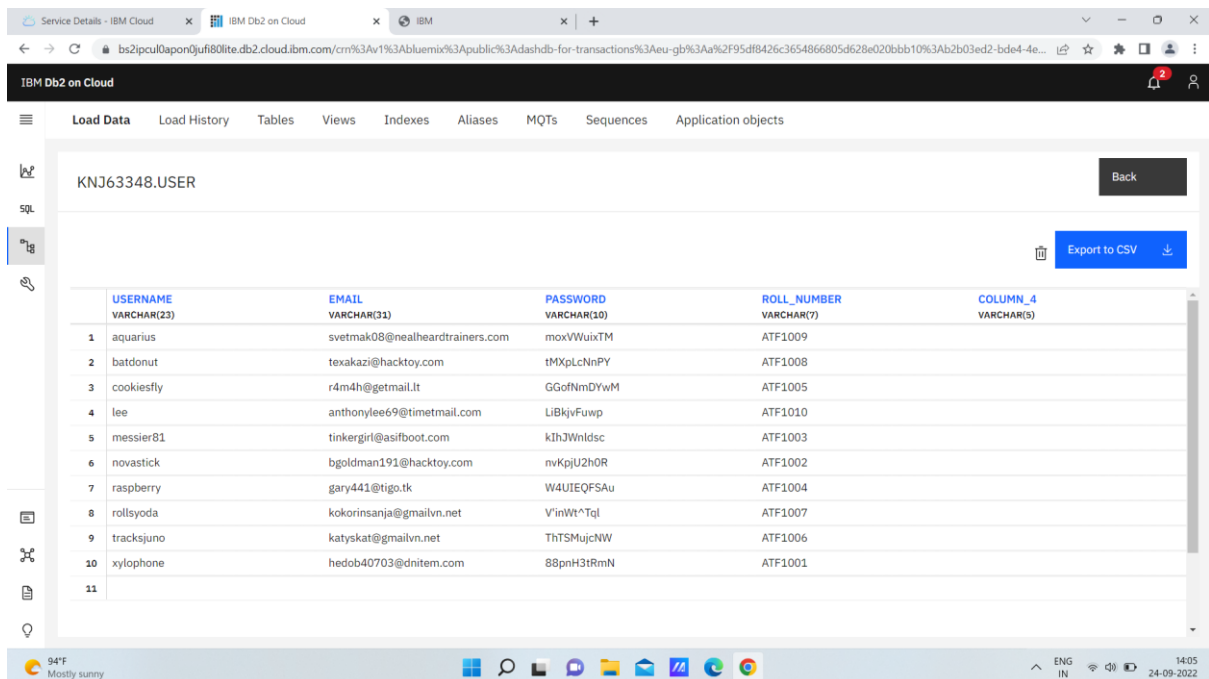
Excel comma file

Username	Email	Password	Roll Number
xylophone	hedob40703@dnitem.com	88puH3RmN	ATF1001
novastick	bgoldman191@hacktoy.com	nvKpjU2h0R	ATF1002
messier81	tinkergirl@asilboot.com	kfhjWulds	ATF1003
raspberry	gary441@tigo.dk	W4UIEQFSa	ATF1004
cookiecstly	r4m4h@getmail.it	GGoFNuDYwM	ATF1005
trackyjuno	katyskat@gmail.vn.net	ThtSMajcNW	ATF1006
rollyoda	kokorimenje@gmail.vn.net	VmWVTrl	ATF1007
batdonut	texakazi@hacktoy.com	tMXpLcNuPY	ATF1008
aquarius	svetmak08@nealheardtrainers.com	mcvWwixTM	ATF1009
lee	anthonylee69@tinetmail.com	LiBkjvFwvp	ATF1010

User table in Db2

Name	Data type	Nullable	Length	Scale
USERNAME	VARCHAR	Y	23	0
EMAIL	VARCHAR	Y	31	0
PASSWORD	VARCHAR	Y	10	0
ROLL_NUMBER	VARCHAR	Y	7	0
COLUMN_4	VARCHAR	Y	5	0

Data in User table



The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The main content area displays the 'KNJ63348.USER' table. A table with 5 columns (USERNAME, EMAIL, PASSWORD, ROLL_NUMBER, COLUMN_4) and 11 rows of data is shown. An 'Export to CSV' button is visible in the top right of the table area.

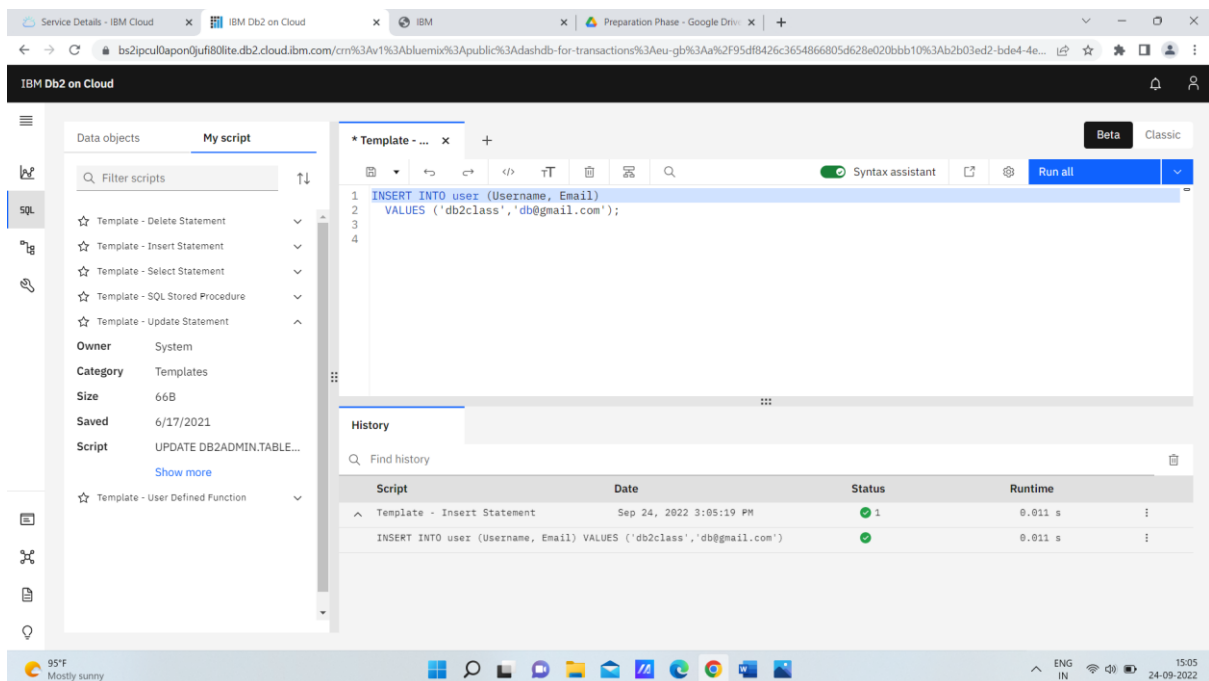
	USERNAME VARCHAR(23)	EMAIL VARCHAR(31)	PASSWORD VARCHAR(10)	ROLL_NUMBER VARCHAR(7)	COLUMN_4 VARCHAR(5)
1	aquarius	svetmak08@nealheardtrainers.com	moxVWuixTM	ATF1009	
2	batdonut	texakazi@hacktoy.com	tMXpLcNnPY	ATF1008	
3	cookiesfly	r4m4h@getmail.lt	GGofNmDYwM	ATF1005	
4	lee	anthonylee69@timetmail.com	LIBkqvFuwp	ATF1010	
5	messier81	tinkergirl@asifboot.com	kIHJWnldsc	ATF1003	
6	novastick	bgoldman191@hacktoy.com	nvKpjU2hOR	ATF1002	
7	raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	
8	rollsodya	kokorinsanja@gmailvn.net	V'inWt^TqI	ATF1007	
9	tracksjuno	katyskat@gmailvn.net	ThTSMujcNW	ATF1006	
10	xylophone	hedob40703@dnitem.com	88pnH3trMn	ATF1001	
11					

Question-2:

Perform INSERT, SELECT, and DELETE Queries in user table

Solution:

Insert Query



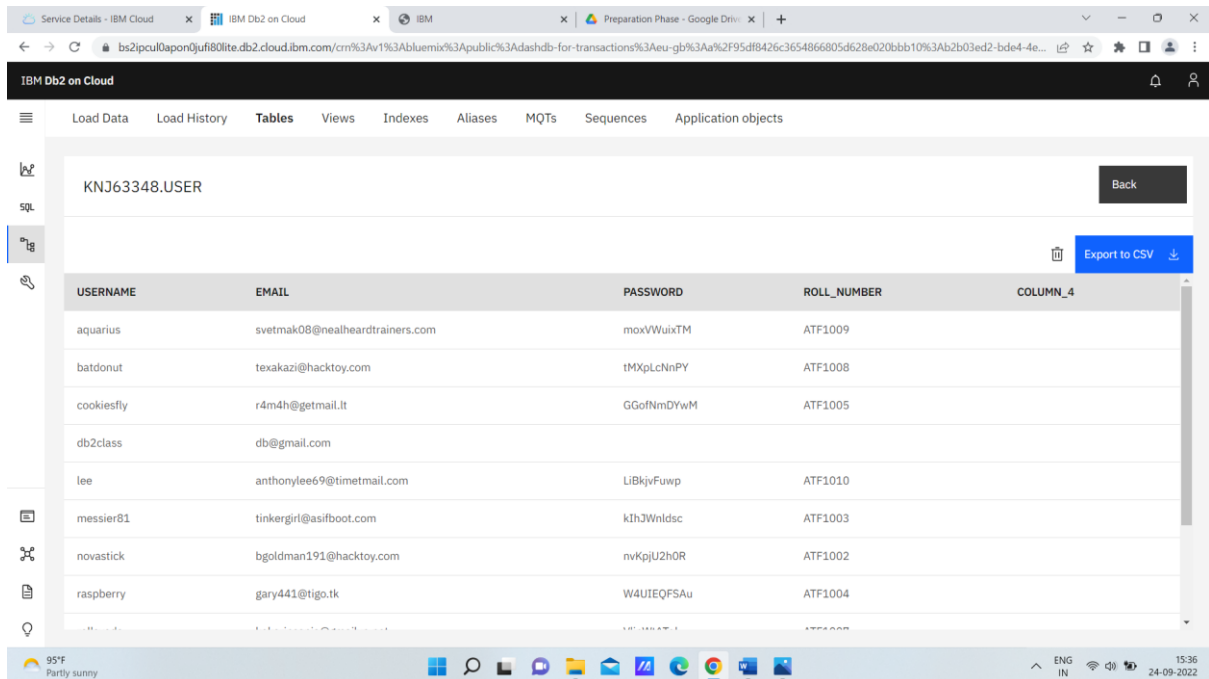
The screenshot shows the IBM Db2 on Cloud console interface. The left sidebar shows 'Data objects' and 'My script'. The main area displays an SQL script with the following content:

```
1 INSERT INTO user (Username, Email)
2 VALUES ('db2class', 'db@gmail.com');
3
4
```

The script is executed, and the 'History' tab shows the execution details:

Script	Date	Status	Runtime
Template - Insert Statement	Sep 24, 2022 3:05:19 PM	1	0.011 s
INSERT INTO user (Username, Email) VALUES ('db2class', 'db@gmail.com')			0.011 s

Insert Query Output



IBM Db2 on Cloud

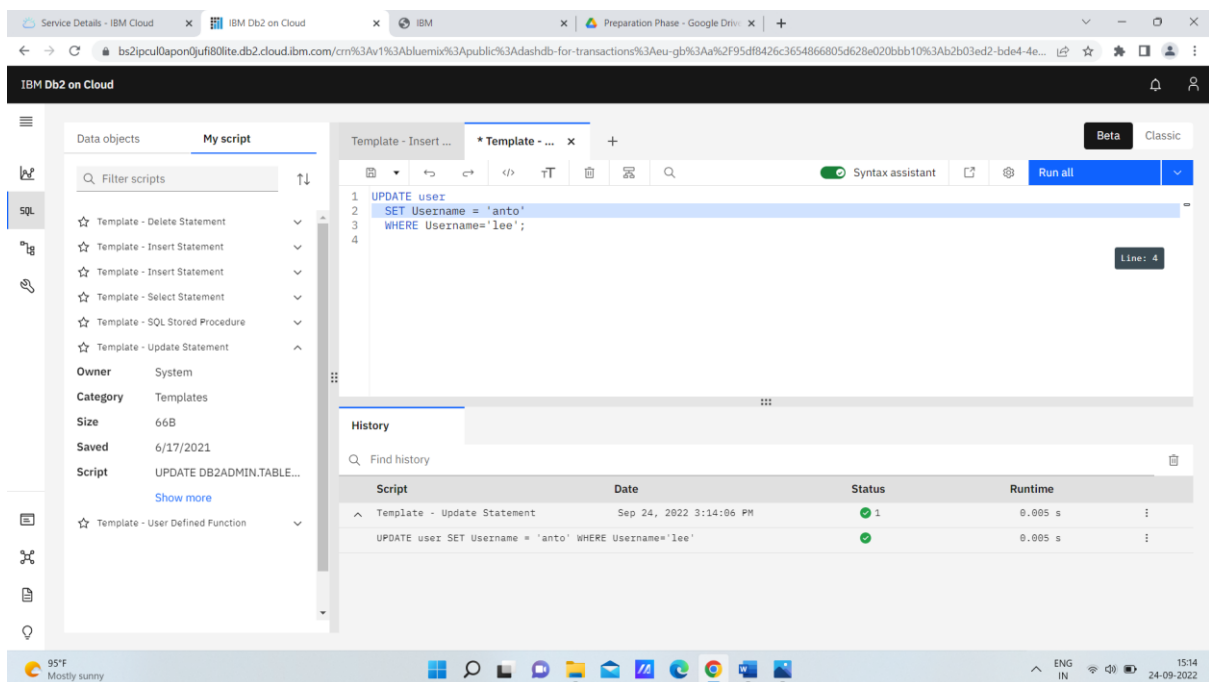
Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

KNJ63348.USER Back

Export to CSV

USERNAME	EMAIL	PASSWORD	ROLL_NUMBER	COLUMN_4
aquarius	svetmak08@nealheardtrainers.com	moxVWuixTM	ATF1009	
batdonut	texakazi@hacktoy.com	tMXpLcNnPY	ATF1008	
cookiesfly	r4m4h@gmail.it	GGofNmDYwM	ATF1005	
db2class	db@gmail.com			
lee	anthonylee69@timetmail.com	LiBkqvFuwp	ATF1010	
messier81	tinkergirl@asifboot.com	kIhJWnldsc	ATF1003	
novastick	bgoldman191@hacktoy.com	nvKpjU2h0R	ATF1002	
raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	

Update Query



IBM Db2 on Cloud

Data objects **My script**

Filter scripts

- Template - Delete Statement
- Template - Insert Statement
- Template - Insert Statement
- Template - Select Statement
- Template - SQL Stored Procedure
- Template - Update Statement

Owner System

Category Templates

Size 66B

Saved 6/17/2021

Script UPDATE DB2ADMIN.TABLE...

Show more

Template - User Defined Function

Template - Insert ...

* Template - ...

Syntax assistant Run all

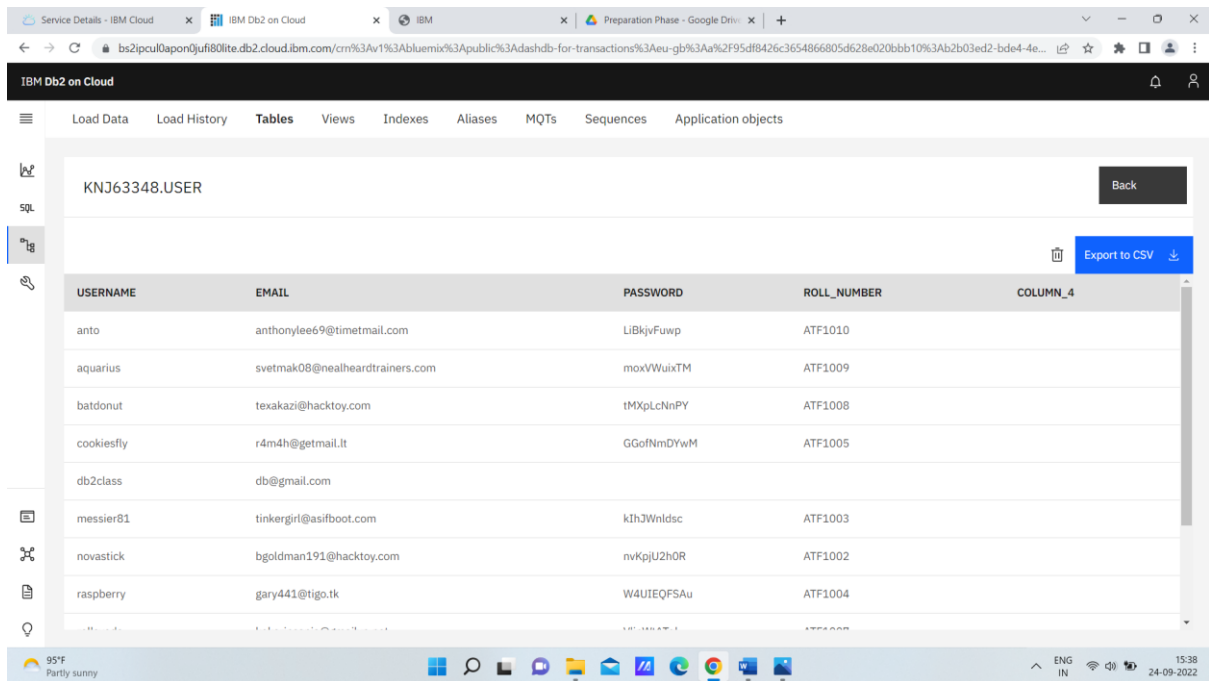
```
1 UPDATE user
2 SET Username = 'anto'
3 WHERE Username='lee';
4
```

History

Find history

Script	Date	Status	Runtime
Template - Update Statement	Sep 24, 2022 3:14:06 PM	1	0.005 s
UPDATE user SET Username = 'anto' WHERE Username='lee'			0.005 s

Update Query Output



IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

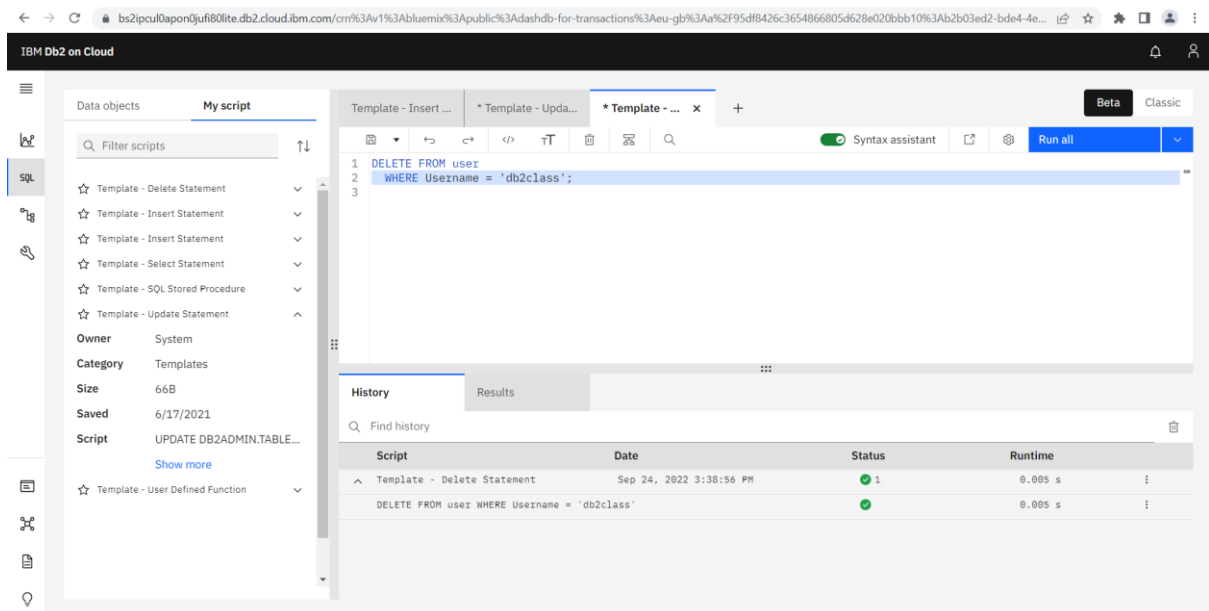
KNJ63348.USER

Back

Export to CSV

USERNAME	EMAIL	PASSWORD	ROLL_NUMBER	COLUMN_4
anto	anthonylee69@timetmail.com	LiBkivFuwp	ATF1010	
aquarius	svetmak08@nealheardtrainers.com	moxVWuixTM	ATF1009	
batdonut	texakazi@hacktoy.com	tMXpLcNnPY	ATF1008	
cookiesfly	r4m4h@gmail.it	GGofNmDYwM	ATF1005	
db2class	db@gmail.com			
messier81	tinkergirl@asifboot.com	kIhJWnldsc	ATF1003	
novastick	bgoldman191@hacktoy.com	nvKpU2h0R	ATF1002	
raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	

Delete Query



IBM Db2 on Cloud

Data objects My script

Filter scripts

Template - Delete Statement

Template - Insert Statement

Template - Insert Statement

Template - Select Statement

Template - SQL Stored Procedure

Template - Update Statement

Owner System

Category Templates

Size 66B

Saved 6/17/2021

Script UPDATE DB2ADMIN.TABLE...

Show more

Template - User Defined Function

Template - Insert ...

* Template - Upda...

* Template - ...

Beta Classic

Syntax assistant

Run all

```
1 DELETE FROM user
2 WHERE Username = 'db2class';
3
```

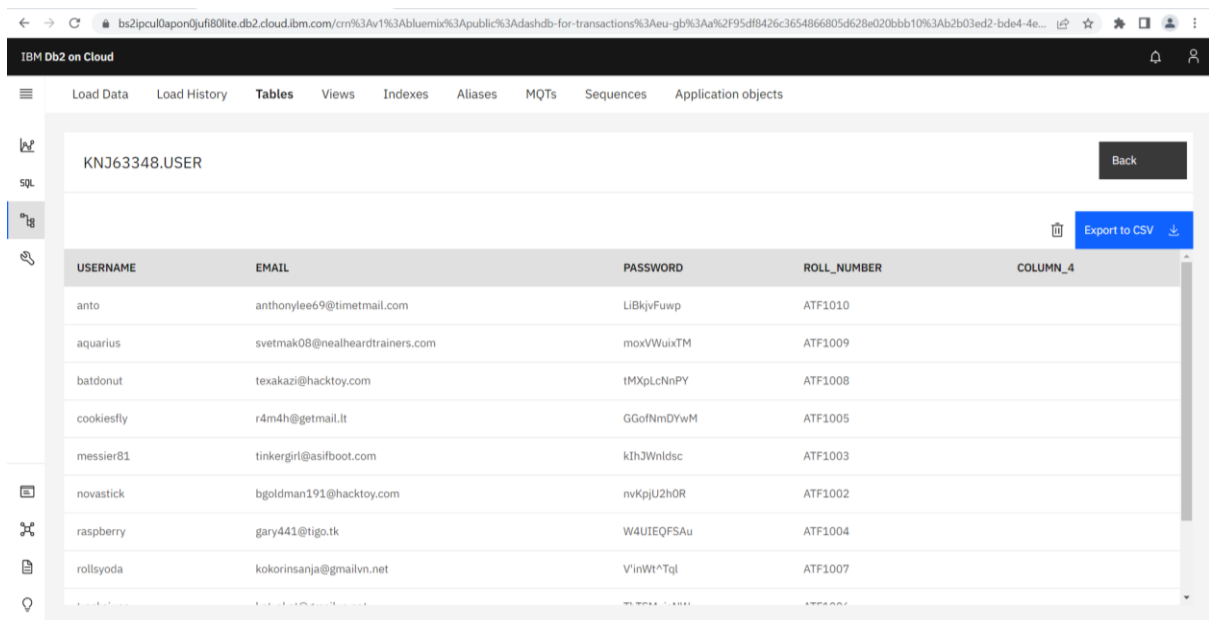
History

Results

Find history

Script	Date	Status	Runtime
Template - Delete Statement	Sep 24, 2022 3:38:56 PM	1	0.005 s
DELETE FROM user WHERE Username = 'db2class'			0.005 s

Delete Query Output



IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

KNJ63348.USER

Back

Export to CSV

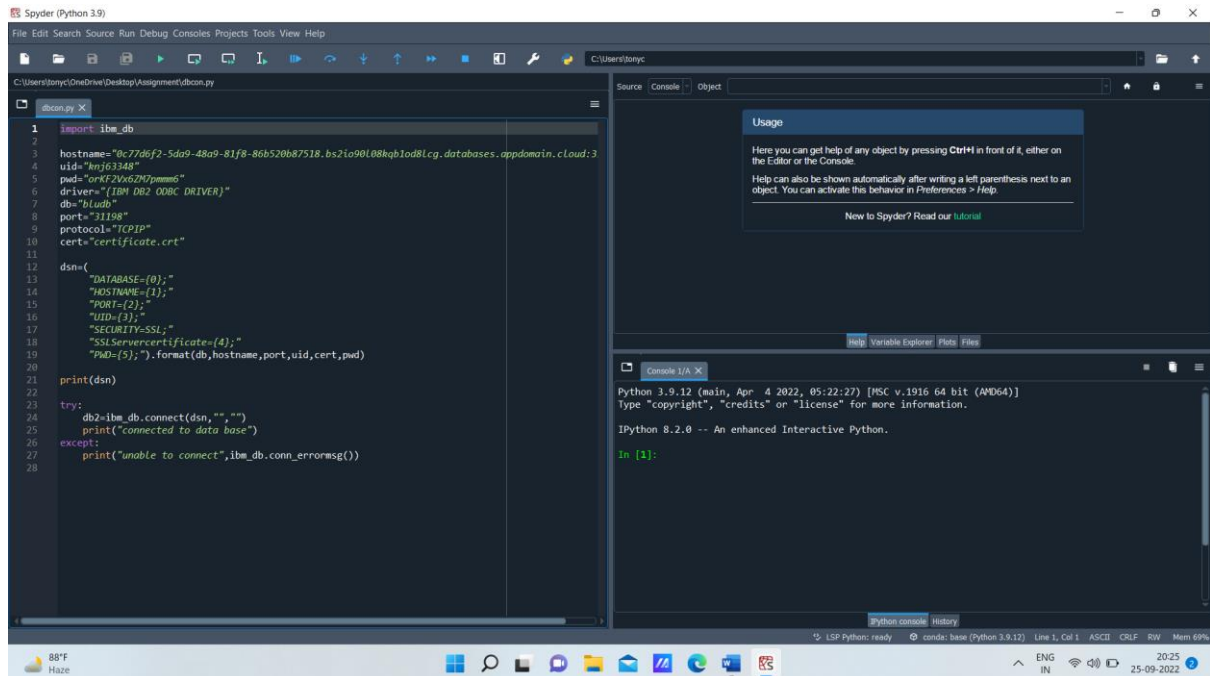
USERNAME	EMAIL	PASSWORD	ROLL_NUMBER	COLUMN_4
anto	anthonylee69@timetmail.com	LiBkivFuwp	ATF1010	
aquarius	svetmak08@nealheardtrainers.com	moxVWuixTM	ATF1009	
batdonut	texakazi@hacktoy.com	tMXpLcNnPY	ATF1008	
cookiesfly	r4m4h@gmail.it	GGofNmDYwM	ATF1005	
messier81	tinkergirl@asifboot.com	kIhJWnldsc	ATF1003	
novastick	bgoldman191@hacktoy.com	nvKpU2h0R	ATF1002	
raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	
rollsoda	kokorinsanja@gmailvn.net	V'inWt^Tql	ATF1007	

Question-3:

Connect Python to DB2

Solution:

Source Code for Python to DB2 Connection



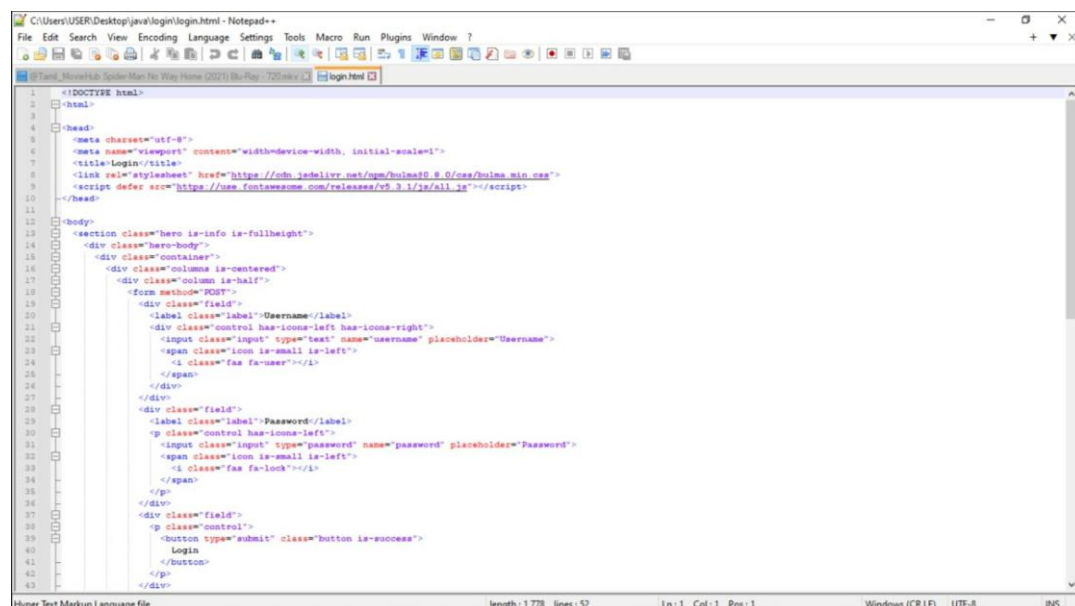
```
1 import ibm_db
2
3 hostname="bc77d6f2-5da9-48a9-81f8-86b520b67518.bs2io90l08qblod8lqg.databases.appdomain.cloud:3
4 uid="my63348"
5 pwd="orKF2Vx6ZNDpmms6"
6 driver="{IBM DB2 ODBC DRIVER}"
7 db="bludb"
8 port="21198"
9 protocol="TCPIP"
10 cert="certificate.crt"
11
12 dsn={
13     "DATABASE={0};"
14     "HOSTNAME={1};"
15     "PORT={2};"
16     "UID={3};"
17     "SECURITY=SSL;"
18     "SSLServerCertificate={4};"
19     "PWD={5};".format(db,hostname,port,uid,cert,pwd)
20
21 print(dsn)
22
23 try:
24     db2=ibm_db.connect(dsn,"")
25     print("connected to data base")
26 except:
27     print("unable to connect",ibm_db.conn_errormsg())
28
```

Question-4:

Create a flask app with login page and welcome page. By default load the registration page once the user enters all the fields store the data in database and navigate to login page authenticate user username and password. If the user is valid show the welcome page

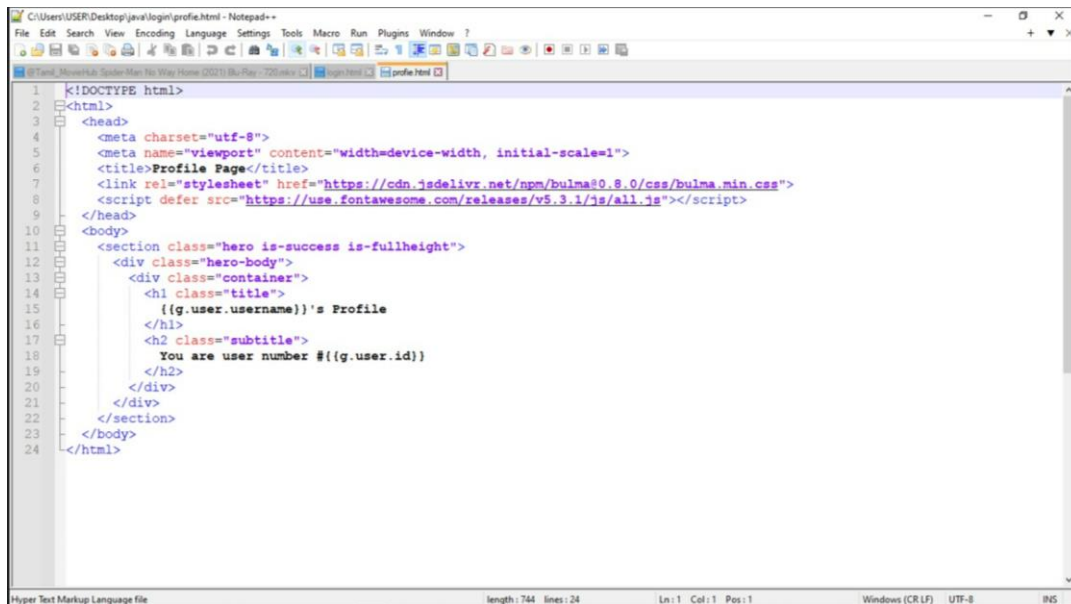
Solution:

Source code for login page



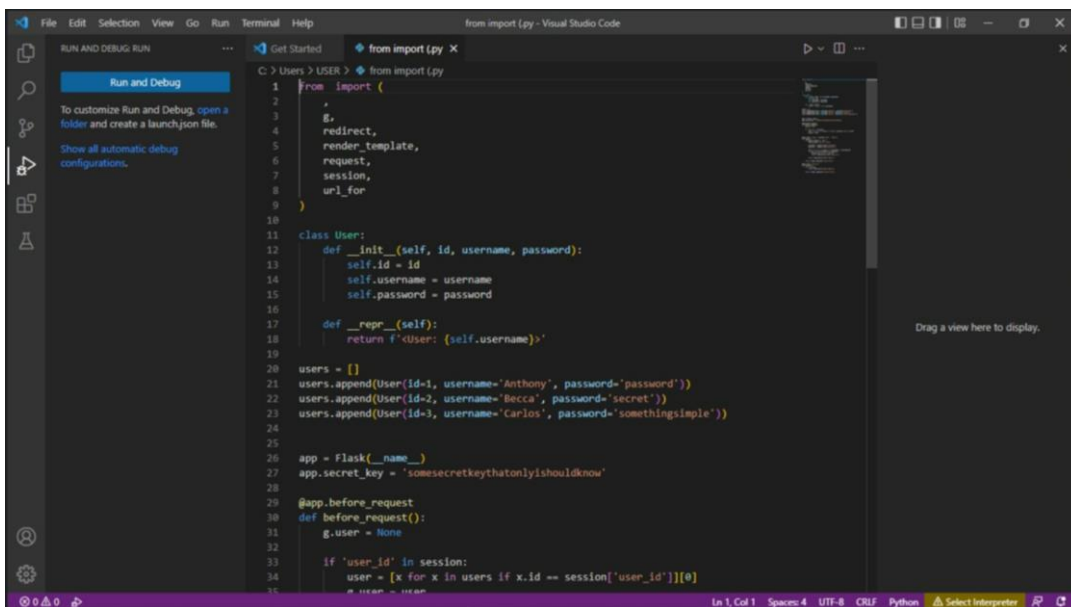
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>Login</title>
7 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/css/bootstrap.min.css">
8 <script defer src="https://use.fontawesome.com/releases/v5.3.1/js/all.js"></script>
9 </head>
10 <body>
11 <section class="hero is-info is-fullheight">
12 <div class="hero-body">
13 <div class="container">
14 <div class="columns is-centered">
15 <div class="column is-half">
16 <form method="POST">
17 <div class="field">
18 <label class="label">Username</label>
19 <div class="control has-icons-left has-icons-right">
20 <input class="input" type="text" name="username" placeholder="Username">
21 <span class="icon is-small is-left"><i class="fas fa-user"></i></span>
22 </div>
23 </div>
24 <div class="field">
25 <label class="label">Password</label>
26 <div class="control has-icons-left">
27 <input class="input" type="password" name="password" placeholder="Password">
28 <span class="icon is-small is-left"><i class="fas fa-lock"></i></span>
29 </div>
30 </div>
31 <div class="field">
32 <div class="control">
33 <button type="submit" class="button is-success">
34 Login
35 </button>
36 </div>
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
```

Source code for welcome page



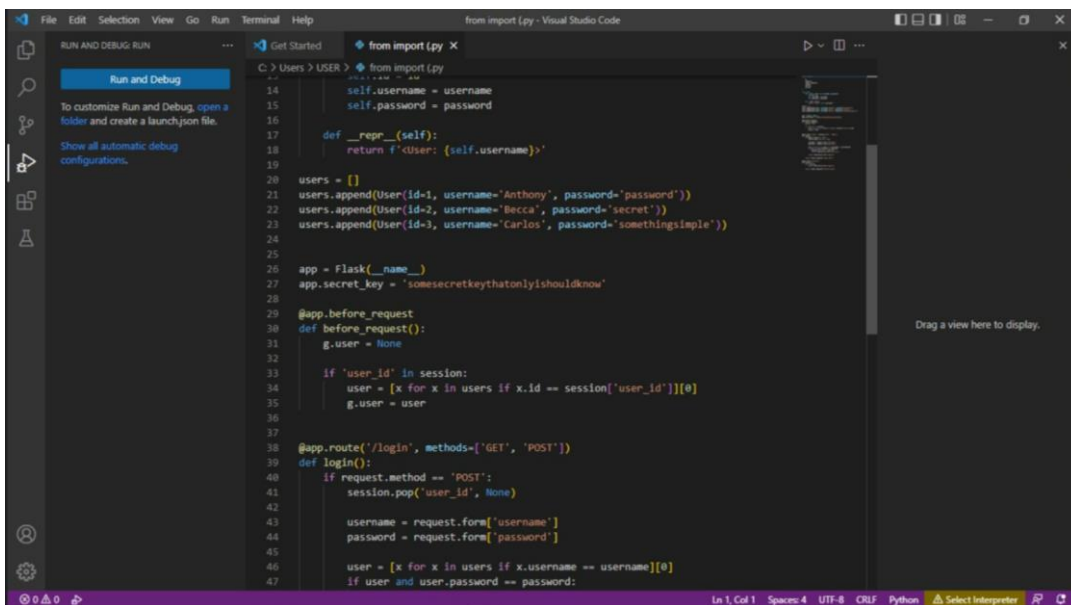
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>Profile Page</title>
7 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.8.0/css/bulma.min.css">
8 <script defer src="https://use.fontawesome.com/releases/v5.3.1/js/all.js"></script>
9 </head>
10 <body>
11 <section class="hero is-success is-fullheight">
12 <div class="hero-body">
13 <div class="container">
14 <h1 class="title">
15 {{(g.user.username)}}'s Profile
16 </h1>
17 <h2 class="subtitle">
18 You are user number #{{(g.user.id)}}
19 </h2>
20 </div>
21 </div>
22 </section>
23 </body>
24 </html>
```

Flask code



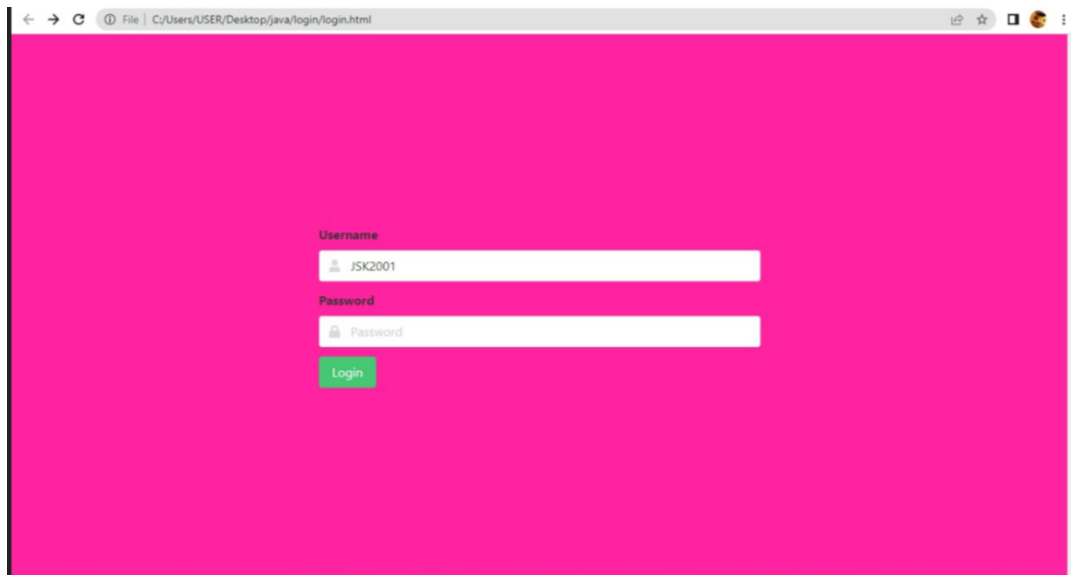
```
1 from import (
2
3     redirect,
4     render_template,
5     request,
6     session,
7     url_for
8 )
9
10
11 class User:
12     def __init__(self, id, username, password):
13         self.id = id
14         self.username = username
15         self.password = password
16
17     def __repr__(self):
18         return f'User: {self.username}'
19
20 users = []
21 users.append(User(id=1, username='Anthony', password='password'))
22 users.append(User(id=2, username='Becca', password='secret'))
23 users.append(User(id=3, username='Carlos', password='somethingsimple'))
24
25
26 app = Flask(__name__)
27 app.secret_key = 'somesecretkeythatonlyishouldknow'
28
29 @app.before_request
30 def before_request():
31     g.user = None
32
33     if 'user_id' in session:
34         user = [x for x in users if x.id == session['user_id']][0]
35         g.user = user
```

Flask code



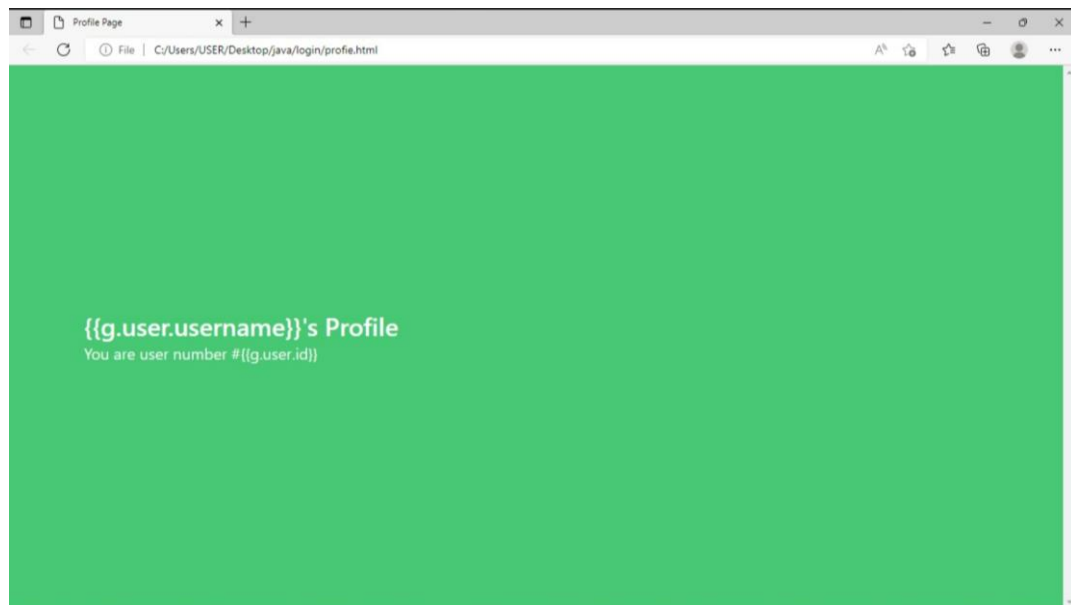
```
14 self.username = username
15 self.password = password
16
17 def __repr__(self):
18     return f'User: {self.username}'
19
20 users = []
21 users.append(User(id=1, username='Anthony', password='password'))
22 users.append(User(id=2, username='Becca', password='secret'))
23 users.append(User(id=3, username='Carlos', password='somethingsimple'))
24
25
26 app = Flask(__name__)
27 app.secret_key = 'somesecretkeythatonlyishouldknow'
28
29 @app.before_request
30 def before_request():
31     g.user = None
32
33     if 'user_id' in session:
34         user = [x for x in users if x.id == session['user_id']][0]
35         g.user = user
36
37
38 @app.route('/login', methods=['GET', 'POST'])
39 def login():
40     if request.method == 'POST':
41         session.pop('user_id', None)
42
43         username = request.form['username']
44         password = request.form['password']
45
46         user = [x for x in users if x.username == username][0]
47         if user and user.password == password:
```

Login Page Output



A screenshot of a web browser displaying a login page. The page has a solid pink background. In the center, there is a white login form. The form contains two input fields: the first is labeled 'Username' and contains the text 'JSK2001'; the second is labeled 'Password' and contains the text 'Password'. Below these fields is a green button with the text 'Login' in white. The browser's address bar shows the file path 'C:/Users/USER/Desktop/java/login/login.html'.

Welcome Page Output



A screenshot of a web browser displaying a welcome page. The page has a solid green background. On the left side, there is white text that reads '{{g.user.username}}'s Profile' followed by 'You are user number #[[g.user.id]]' on the next line. The browser's address bar shows the file path 'C:/Users/USER/Desktop/java/login/profile.html'.