

Assignment - 3

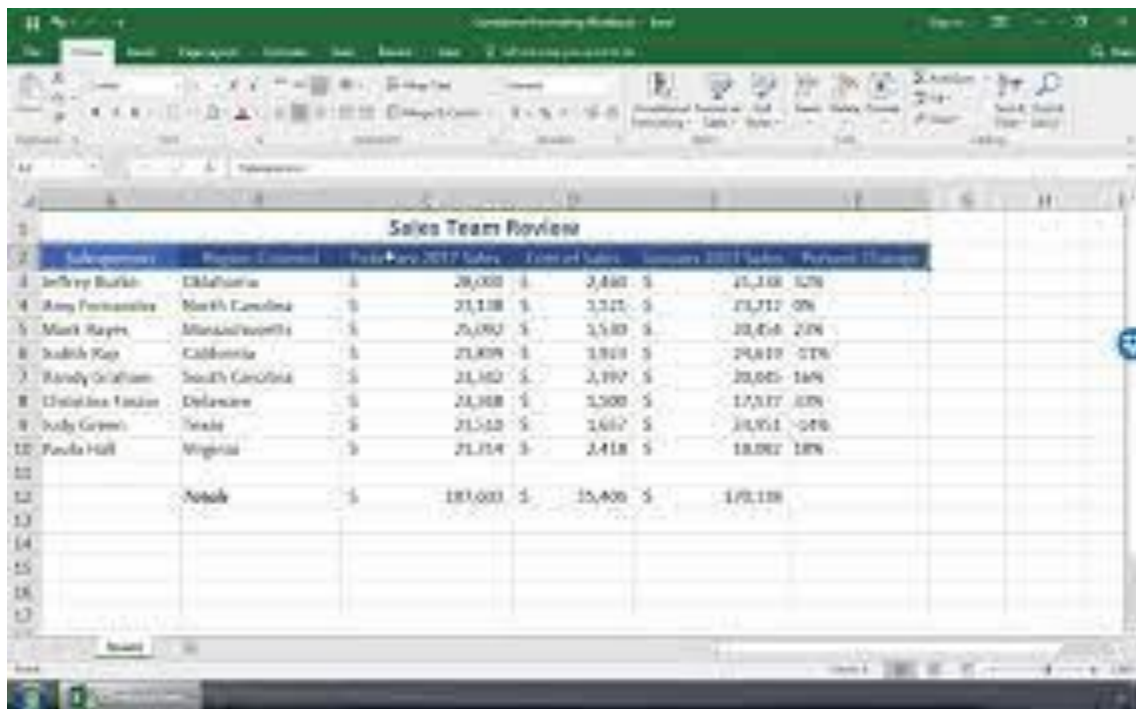
Assignment Date	29 September 2022
Student Name	Sameer Mohammed.S
Student Roll No	812419104056
Maximum Marks	2 Marks

Question-1:

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

Solution:

Excel comma file



Salesperson	Region	Current Sales	Target Sales	Percent Change
Jeffrey Burke	Illinois	28,000	2,400	12%
Amy Perkinson	North Carolina	21,100	2,210	0%
Mark Hayes	Mississippi	26,000	2,500	20%
Scotty Kay	California	21,800	2,800	25%
Randy Graham	South Carolina	21,500	2,900	14%
Christina Taylor	Delaware	21,100	2,500	10%
Judy Green	Texas	21,500	2,600	10%
Paula Hall	Virginia	21,100	2,400	10%
Total		181,000	25,400	14%

User table in Db2

Service Details - IBM Cloud x IBM Db2 on Cloud x IBM x +

bs2ipcul0apon0juf80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F95df8426c3654866805d628e020bbb10%3Ab2b03ed2-bde4-4e...

IBM Db2 on Cloud

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

Find schemas or tables Refresh

Tables New table +

Name	Schema	Properties
DEMO	KNJ63348	...
USER	KNJ63348	...

Total: 2, selected: 0

Table definition

USER

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

View data

94°F Mostly sunny 14:06 24-09-2022

Data in User table

Service Details - IBM Cloud x IBM Db2 on Cloud x IBM x +

bs2ipcul0apon0juf80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F95df8426c3654866805d628e020bbb10%3Ab2b03ed2-bde4-4e...

IBM Db2 on Cloud

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

KNJ63348.USER Back

Export to CSV

	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@gmail.it	GGofNmDYwM	ATF1005	
4	lee	anthonylee69@timetmail.com	LiBkqvFuwp	ATF1010	
5	messier81	tinkergirl@asifboot.com	kIhJWhldsc	ATF1003	
6	novastick	bgoldman191@hacktoy.com	nvKpjU2h0R	ATF1002	
7	raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	
8	rollsyoda	kokorinsanja@gmailvn.net	V'inWt^Tql	ATF1007	
9	tracksjuno	katyskat@gmailvn.net	ThTSMujcNW	ATF1006	
10	xylophone	hedob40703@dnitem.com	88pnH3tRmN	ATF1001	
11					

94°F Mostly sunny 14:05 24-09-2022

Question-2:

Perform INSERT, SELECT, DELETE Queries in user table

Solution:

Insert Query

The screenshot shows the IBM Db2 on Cloud console. On the left, the 'My script' tab is active, displaying a list of templates. The main editor area shows the following SQL query:

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

Below the editor, the 'History' tab shows the execution results:

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

The screenshot shows the IBM Db2 on Cloud console with the 'Tables' tab selected. The table 'KNJ63348.USER' is displayed with the following data:

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	LiBkivFuwp	ATF1010	
messier81	tinkergirl@asifboot.com	kIHJWnldsc	ATF1003	
novastick	bgoldman191@hacktoy.com	nvKpjU2h0R	ATF1002	
raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	

Update Query

The screenshot shows the IBM Db2 on Cloud console interface. On the left, a sidebar contains navigation icons and a 'My script' section with a list of templates. The main area displays an SQL editor with the following query:

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

Below the editor, a 'History' tab shows a table of executed scripts:

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

The bottom of the screen shows a Windows taskbar with various application icons and system status information (95°F, Mostly sunny, 15:14, 24-09-2022).

Update Query Output

The screenshot shows the IBM Db2 on Cloud console displaying the output of an update query. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The main content area shows the table 'KNJ63348.USER' with a 'Back' button. Below the table name, there is an 'Export to CSV' button. The table data is as follows:

USERNAME	EMAIL	PASSWORD	ROLL_NUMBER	COLUMN_4
anto	anthonylee69@timetmail.com	LiBkvFuwp	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	kHjWnldsc	ATF1003	
novastick	bgoldman191@hacktoy.com	nvKpjU2h0R	ATF1002	
raspberry	gary441@tigo.tk	W4UIEQFSau	ATF1004	

The bottom of the screen shows a Windows taskbar with various application icons and system status information (95°F, Partly sunny, 15:38, 24-09-2022).

Delete Query

IBM Db2 on Cloud

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

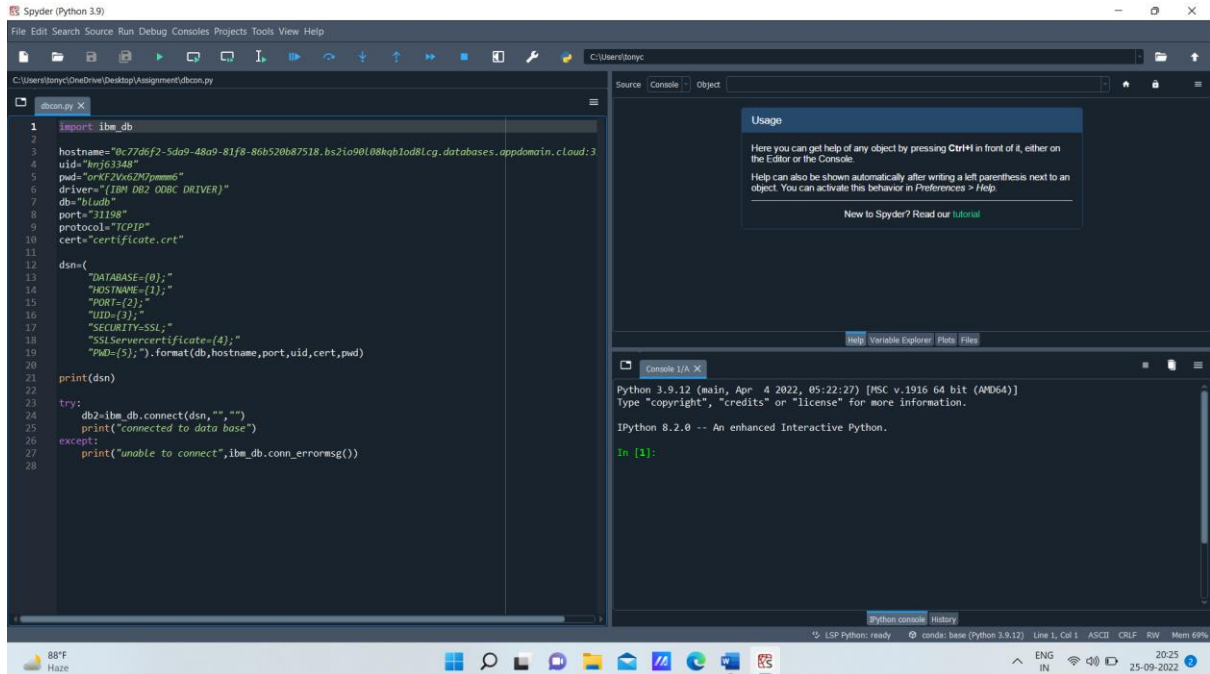
USERNAME	EMAIL	PASSWORD	ROLL_NUMBER	COLUMN_4
anto	anthonylee69@timetmail.com	LiBkqvFuwp	ATF1010	
aquarius	svetmak08@nealheardtrainers.com	moxVWuikTM	ATF1009	
batdonut	texakazi@hacktoy.com	tMXpLcNnPY	ATF1008	
cookiesfly	r4m4h@getmail.lt	GGofNmDYwM	ATF1005	
messier81	tinkergirl@asifboot.com	kIHJWnldsc	ATF1003	
novastick	bgoldman191@hacktoy.com	nvKpjU2h0R	ATF1002	
raspberry	gary441@tigo.tk	W4UIEQFSAu	ATF1004	
rollsodya	kokorinsanja@gmailvn.net	V'inWt^Tql	ATF1007	

Question-3:

Connect Python to DB2

Solution:

Source Code for Python to DB2 Connection



The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script named `dbcon.py` located at `C:\Users\tonyc\OneDrive\Desktop\Assignment\dbcon.py`. The script imports the `ibm_db` module and defines connection parameters for a DB2 database. It constructs a DSN (Data Source Name) string and attempts to connect to the database. The console window on the right shows the output of the script, indicating a successful connection.

```
1 import ibm_db
2
3 hostname="9c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08qb1od8l.c.cloud:3
4 uid="my63348"
5 pwd="orKF2Vx6ZNDpmme6"
6 driver="{IBM DB2 ODBC DRIVER}"
7 db="bladb"
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};"
20 }.format(db,hostname,port,uid,cert,pwd)
21
22 print(dsn)
23
24 try:
25     db2=ibm_db.connect(dsn,"","")
26     print("connected to data base")
27 except:
28     print("unable to connect",ibm_db.conn_errormsg())
```

Console Output:

```
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 8.2.0 -- An enhanced Interactive Python.
In [1]:
```

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

Source code for login page

Source code for welcome page

Flask code

The screenshot shows the Visual Studio Code editor with a Python file named 'from import (py)'. The code defines a User class, a list of users, a Flask app, and a before_request hook. The User class has attributes id, username, and password. The users list contains three users: Anthony, Becca, and Carlos. The Flask app is configured with a secret key and a before_request hook that sets the user attribute based on the session. The code is as follows:

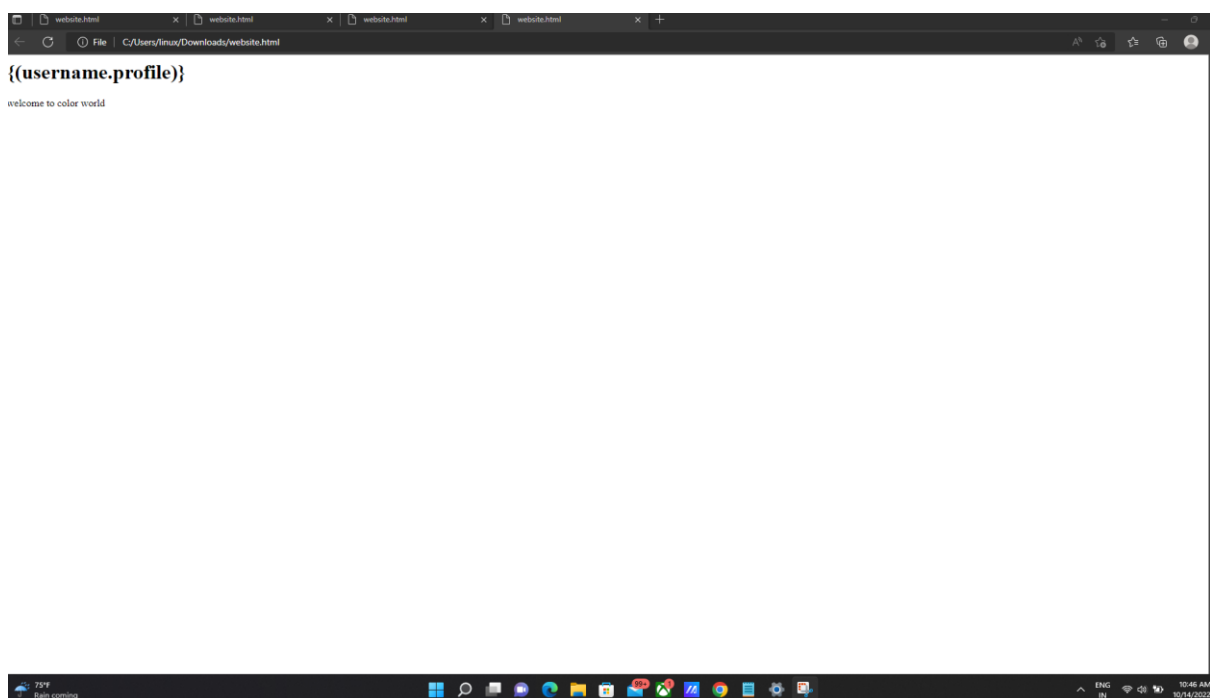
```
1 from import (
2     <
3     <
4     redirect,
5     render_template,
6     request,
7     session,
8     url_for
9 )
10
11 class User:
12     def __init__(self, id, username, password):
13         self.id = id
14         self.username = username
15         self.password = password
16
17     def __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='somethingimple'))
24
25 app = Flask(__name__)
26 app.secret_key = 'somesecretkeythatonlyishouldknow'
27
28 @app.before_request
29 def before_request():
30     g.user = None
31
32     if 'user_id' in session:
33         user = [x for x in users if x.id == session['user_id']][0]
34         g.user = user
```

Flask code

The screenshot shows the Visual Studio Code editor with a Python file named 'from import (py)'. The code defines a User class, a list of users, a Flask app, and a login route. The User class has attributes id, username, and password. The users list contains three users: Anthony, Becca, and Carlos. The Flask app is configured with a secret key and a before_request hook. The login route is defined as follows:

```
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='somethingimple'))
24
25 app = Flask(__name__)
26 app.secret_key = 'somesecretkeythatonlyishouldknow'
27
28 @app.before_request
29 def before_request():
30     g.user = None
31
32     if 'user_id' in session:
33         user = [x for x in users if x.id == session['user_id']][0]
34         g.user = user
35
36
37 @app.route('/login', methods=['GET', 'POST'])
38 def login():
39     if request.method == 'POST':
40         session.pop('user_id', None)
41
42         username = request.form['username']
43         password = request.form['password']
44
45         user = [x for x in users if x.username == username][0]
46         if user and user.password == password:
```

Login Page Output



Welcome Page Output