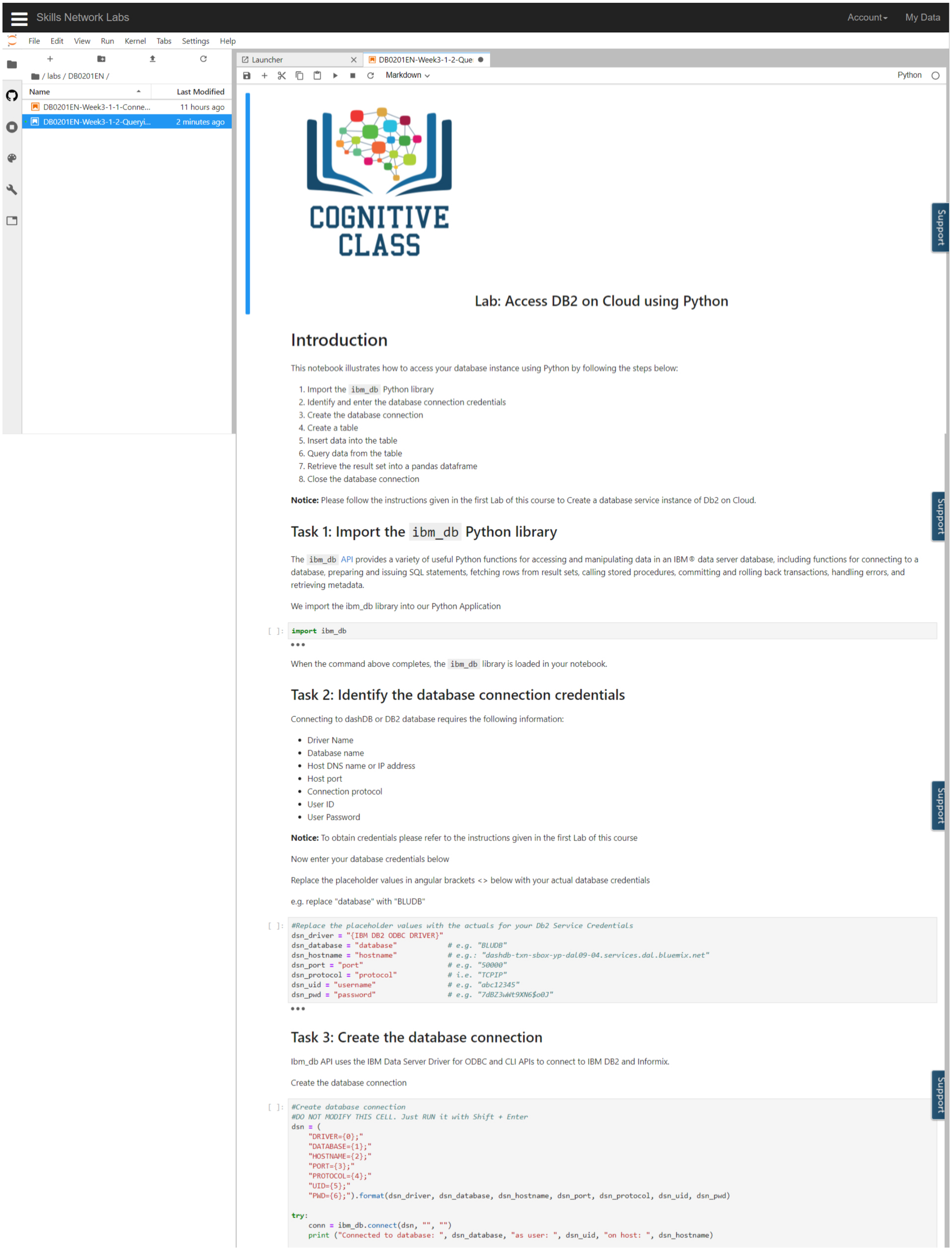

Skills Network Labs
Account
My Data
File Edit View Run Kernel Tabs Settings Help
/ labs / DB0201EN /
Name
Last Modified
DB0201EN-Week3-1-1-Conne... 11 hours ago
DB0201EN-Week3-1-2-Queryi... 2 minutes ago
Launcher
DB0201EN-Week3-1-2-Que
Markdown
Python
Support

COGNITIVE CLASS

Lab: Access DB2 on Cloud using Python

Introduction

This notebook illustrates how to access your database instance using Python by following the steps below:

1. Import the `ibm_db` Python library
2. Identify and enter the database connection credentials
3. Create the database connection
4. Create a table
5. Insert data into the table
6. Query data from the table
7. Retrieve the result set into a pandas dataframe
8. Close the database connection

Notice: Please follow the instructions given in the first Lab of this course to Create a database service instance of Db2 on Cloud.

Task 1: Import the `ibm_db` Python library

The `ibm_db` API provides a variety of useful Python functions for accessing and manipulating data in an IBM® data server database, including functions for connecting to a database, preparing and issuing SQL statements, fetching rows from result sets, calling stored procedures, committing and rolling back transactions, handling errors, and retrieving metadata.

We import the ibm_db library into our Python Application

```
[ ]: import ibm_db
```

When the command above completes, the `ibm_db` library is loaded in your notebook.

Task 2: Identify the database connection credentials

Connecting to dashDB or DB2 database requires the following information:

- Driver Name
- Database name
- Host DNS name or IP address
- Host port
- Connection protocol
- User ID
- User Password

Notice: To obtain credentials please refer to the instructions given in the first Lab of this course

Now enter your database credentials below

Replace the placeholder values in angular brackets <> below with your actual database credentials

e.g. replace "database" with "BLUDB"

```
[ ]: #Replace the placeholder values with the actuals for your Db2 Service Credentials
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "database"            # e.g. "BLUDB"
dsn_hostname = "hostname"            # e.g.: "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net"
dsn_port = "port"                    # e.g. "50000"
dsn_protocol = "protocol"            # i.e. "TCPIP"
dsn_uid = "username"                 # e.g. "abc12345"
dsn_pwd = "password"                 # e.g. "7dBZ3wWt9XN6$o0J"
```

Task 3: Create the database connection

Ibm_db API uses the IBM Data Server Driver for ODBC and CLI APIs to connect to IBM DB2 and Informix.

Create the database connection

```
[ ]: #Create database connection
#DO NOT MODIFY THIS CELL. Just RUN it with Shift + Enter
dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd)

try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ", dsn_hostname)
```

```python
except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )
```

Task 4: Create a table in the database

In this step we will create a table in the database with following details:

Table definition

INSTRUCTOR

COLUMN NAME	DATA TYPE	NULLABLE
ID	INTEGER	N
FNAME	VARCHAR	Y
LNAME	VARCHAR	Y
CITY	VARCHAR	Y
CCODE	CHARACTER	Y

```python
#Lets first drop the table INSTRUCTOR in case it exists from a previous attempt
dropQuery = "drop table INSTRUCTOR"

#Now execute the drop statment
dropStmt = ibm_db.exec_immediate(conn, dropQuery)
```

Dont worry if you get this error:

If you see an exception/error similar to the following, indicating that INSTRUCTOR is an undefined name, that's okay. It just implies that the INSTRUCTOR table does not exist in the table - which would be the case if you had not created it previously.

Exception: [IBM][CLI Driver][DB2/LINUXX8664] SQL0204N "ABC12345.INSTRUCTOR" is an undefined name. SQLSTATE=42704 SQLCODE=-204

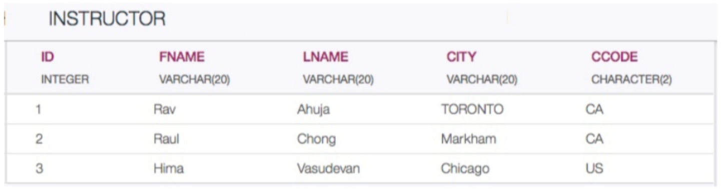
```python
#Construct the Create Table DDL statement - replace the ... with rest of the statement
createQuery = "create table INSTRUCTOR(id INTEGER PRIMARY KEY NOT NULL, fname ...)"

#Now fill in the name of the method and execute the statement
createStmt = ibm_db.replace_with_name_of_execution_method(conn, createQuery)
```

Double-click **here** for the solution.

Task 5: Insert data into the table

In this step we will insert some rows of data into the table.

The INSTRUCTOR table we created in the previous step contains 3 rows of data:



INSTRUCTOR

ID	FNAME	LNAME	CITY	CCODE
INTEGER	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	CHARACTER(2)
1	Rav	Ahuja	TORONTO	CA
2	Raul	Chong	Markham	CA
3	Hima	Vasudevan	Chicago	US

We will start by inserting just the first row of data, i.e. for instructor Rav Ahuja

```python
#Construct the query - replace ... with the insert statement
insertQuery = "..."

#execute the insert statement
insertStmt = ibm_db.exec_immediate(conn, insertQuery)
```

Double-click **here** for the solution.

Now use a single query to insert the remaining two rows of data

```python
#replace ... with the insert statement that inerts the remaining two rows of data
insertQuery2 = "..."

#execute the statement
insertStmt2 = ibm_db.exec_immediate(conn, insertQuery2)
```

Double-click **here** for the solution.

Task 6: Query data in the table

In this step we will retrieve data we inserted into the INSTRUCTOR table.

```python
#Construct the query that retrieves all rows from the INSTRUCTOR table
selectQuery = "select * from INSTRUCTOR"

#Execute the statement
selectStmt = ibm_db.exec_immediate(conn, selectQuery)

#Fetch the Dictionary (for the first row only) - replace ... with your code
...
```

Double-click **here** for the solution.

```python
#Fetch the rest of the rows and print the ID and FNAME for those rows
while ibm_db.fetch_row(selectStmt) != False:
   print (" ID:",  ibm_db.result(selectStmt, 0), " FNAME:",  ibm_db.result(selectStmt, "FNAME"))
```

Double-click **here** for the solution.

Bonus: now write and execute an update statement that changes the Rav's CITY to MOOSETOWN

```python
#Enter your code below
```

Double-click **here** for the solution.

Task 7: Retrieve data into Pandas

In this step we will retrieve the contents of the INSTRUCTOR table into a Pandas dataframe

Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in. Learn more here.

```
[ ]: import pandas
     import ibm_db_dbi
```

```
[ ]: #connection for pandas
     pconn = ibm_db_dbi.Connection(conn)
```

```
[ ]: #query statement to retrieve all rows in INSTRUCTOR table
     selectQuery = "select * from INSTRUCTOR"

     #retrieve the query results into a pandas dataframe
     pdf = pandas.read_sql(selectQuery, pconn)

     #print just the LNAME for first row in the pandas data frame
     pdf.LNAME[0]
```

```
[ ]: #print the entire data frame
     pdf
```

Once the data is in a Pandas dataframe, you can do the typical pandas operations on it.

For example you can use the shape method to see how many rows and columns are in the dataframe

```
[ ]: pdf.shape
```

Task 8: Close the Connection

We free all resources by closing the connection. Remember that it is always important to close connections so that we can avoid unused connections taking up resources.

```
[ ]: ibm_db.close(conn)
```

Summary

In this tutorial you established a connection to a database instance of DB2 Warehouse on Cloud from a Python notebook using ibm_db API. Then created a table and insert a few rows of data into it. Then queried the data. You also retrieved the data into a pandas dataframe.

```
[ ]:
```