**Access DB2 on Cloud using Python[¶](https://jupyterlab-0-labs-prod-jupyterlab-us-east-3.labs.cognitiveclass.ai/user/akulasamson/lab/tree/labs/DB0201EN/DB0201EN-Week3-1-2-Querying-v4-py.ipynb" \l "Access-DB2-on-Cloud-using-Python" \t "_self)**

Estimated time needed: **15** minutes

**Objectives**

After completing this lab you will be able to:

* Create a table
* Insert data into the table
* Query data from the table
* Retrieve the result set into a pandas dataframe
* 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](https://pypi.python.org/pypi/ibm_db/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDB0201ENSkillsNetwork20127838-2021-01-01)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

[1]:



**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"

[2]:



*#Replace the placeholder values with the actuals for your Db2 Service Credentials*

dsn\_driver **=** "{IBM DB2 ODBC DRIVER}"

dsn\_database **=** "BLUDB" *# e.g. "BLUDB"*

dsn\_hostname **=** "54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud" *# e.g.: "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net"*

dsn\_port **=** "32733" *# e.g. "50000"*

dsn\_protocol **=** "TCPIP" *# i.e. "TCPIP"*

dsn\_uid **=** "kvn77701" *# e.g. "abc12345"*

dsn\_pwd **=** "INISp92TCJkh15dQ" *# e.g. "7dBZ3wWt9XN6$o0J"*

dsn\_security **=** "SSL" *#i.e. "SSL"*

**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

[3]:



*#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};"

"SECURITY={7};").format(dsn\_driver, dsn\_database, dsn\_hostname, dsn\_port, dsn\_protocol, dsn\_uid, dsn\_pwd,dsn\_security)

​

**try**:

conn **=** ibm\_db.connect(dsn, "", "")

print ("Connected to database: ", dsn\_database, "as user: ", dsn\_uid, "on host: ", dsn\_hostname)

​

**except**:

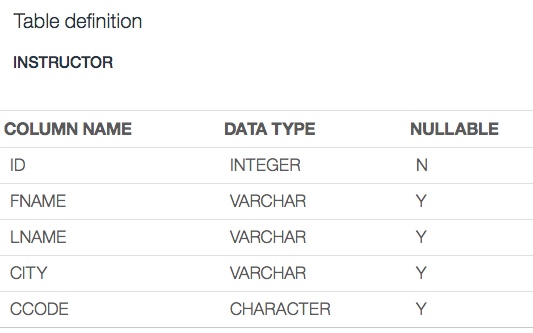
print ("Unable to connect: ", ibm\_db.conn\_errormsg() )

​

Connected to database: BLUDB as user: kvn77701 on host: 54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud

**Task 4: Create a table in the database**

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



[5]:



*#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)

---------------------------------------------------------------------------

Exception Traceback (most recent call last)

<ipython-input-5-83413676a2ca> in <module>

**3**

**4** #Now execute the drop statment

----> 5 dropStmt = ibm\_db.exec\_immediate(conn, dropQuery)

Exception: [IBM][CLI Driver][DB2/LINUXX8664] SQL0204N "KVN77701.INSTRUCTOR" is an undefined name. SQLSTATE=42704 SQLCODE=-204

**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

[6]:



*#Construct the Create Table DDL statement - replace the ... with rest of the statement*

createQuery **=** "create table INSTRUCTOR(ID INTEGER PRIMARY KEY NOT NULL, FNAME VARCHAR(20), LNAME VARCHAR(20), CITY VARCHAR(20), CCODE CHAR(2))"

​

*#Now fill in the name of the method and execute the statement*

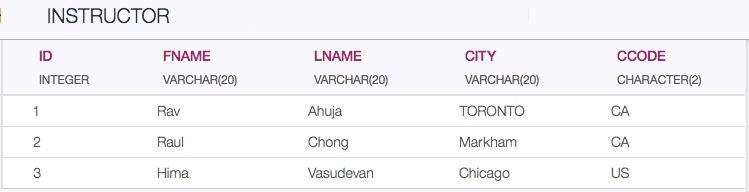
createStmt **=** ibm\_db.exec\_immediate(conn,createQuery)

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:



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

[7]:



*#Construct the query - replace ... with the insert statement*

insertQuery **=** "insert into INSTRUCTOR values (1, 'Rav', 'Ahuja', 'TORONTO', 'CA')"

​

*#execute the insert statement*

insertStmt **=** ibm\_db.exec\_immediate(conn, insertQuery)

Click here for the solution

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

[8]:



*#replace ... with the insert statement that inerts the remaining two rows of data*

insertQuery2 **=** "insert into INSTRUCTOR values (2, 'Raul', 'Chong', 'Markham', 'CA'), (3, 'Hima', 'Vasudevan', 'Chicago', 'US')"

​

*#execute the statement*

insertStmt2 **=** ibm\_db.exec\_immediate(conn, insertQuery2)

Click here for the solution

insertQuery2 **=** "insert into INSTRUCTOR values (2, 'Raul', 'Chong', 'Markham', 'CA'), (3, 'Hima', 'Vasudevan', 'Chicago', 'US')"

insertStmt2 **=** ibm\_db.exec\_immediate(conn, insertQuery2)

**Task 6: Query data in the table**

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

**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.**](https://cocl.us/ibm_watson_studio_infobox)

[9]:



*#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*

ibm\_db.fetch\_both(selectStmt)

[9]:

{'ID': 1,

0: 1,

'FNAME': 'Rav',

1: 'Rav',

'LNAME': 'Ahuja',

2: 'Ahuja',

'CITY': 'TORONTO',

3: 'TORONTO',

'CCODE': 'CA',

4: 'CA'}

Click here for the solution

[10]:



*#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"))

ID: 2 FNAME: Raul

ID: 3 FNAME: Hima

Click here for the solution

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

[12]:



*#Enter your code below*

updateQuery **=** "update INSTRUCTOR set CITY='MOOSETOWN' where FNAME='Rav'"

updateStmt **=** ibm\_db.exec\_immediate(conn, updateQuery)

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

[13]:



**import** pandas

**import** ibm\_db\_dbi

[14]:



*#connection for pandas*

pconn **=** ibm\_db\_dbi.Connection(conn)

[15]:



*#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]

[15]:

'Ahuja'

[16]:



*#print the entire data frame*

pdf

[16]:

|  | **ID** | **FNAME** | **LNAME** | **CITY** | **CCODE** |
| --- | --- | --- | --- | --- | --- |
| **0** | 1 | Rav | Ahuja | MOOSETOWN | CA |
| **1** | 2 | Raul | Chong | Markham | CA |
| **2** | 3 | Hima | Vasudevan | Chicago | US |

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

[17]:



pdf.shape

[17]:

(3, 5)

**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.

[18]:



ibm\_db.close(conn)

[18]:

True

**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.