



Access DB2 on Cloud using Python

Objectives

- 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

Task 1: Import the `ibm_db` Python library

We import the `ibm_db` library into our Python Application

In [214]:

```
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"

In [215]:

```
#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 = "dashdb-txn-sbox-yp-dal09-12.services.dal.ibm.com" # e.g. "dashdb-txn-sbox-yp-dal09-04.services.dal.ibm.com"
dsn_port = "50000" # e.g. "50000"
dsn_protocol = "TCPIP" # i.e. "TCPIP"
dsn_uid = "tnm91075" # e.g. "abc12345"
dsn_pwd = "krrsj3j7zjn@nphp" # 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

In [216]:

```
#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)

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

Connected to database: BLUDB as user: tnm91075 on host: dashdb-txn-sbox-yp-dal09-12.services.dal.ibmcloud.net

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

In [217]:

```
#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/LINUX8664] SQL0204N "ABC12345.INSTRUCTOR" is an undefined name. SQLSTATE=42704 SQLCODE=-204

In [218]:

```
#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:

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

In [219]:

```
#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

In [220]:

```
#replace ... with the insert statement that inserts 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](#)

Task 6: Query data in the table

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

In [229]:

```
#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
#both Returns a dictionary, which is indexed by both column name and position,
##representing a row in a result set.
ibm_db.fetch_both(selectStmt)
```

Out[229]:

```
{'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](#)

In [230]:

```
#Fetch the rest of the rows and print the ID and FNAME for those rows
#Optional: If you called the ibm_db.fetch_row function, for each iteration through the
#result set,
#retrieve a value from a specified column by calling the ibm_db.result function. You can
#specify
#the column by passing either an integer that represents the position of the column in
#the row (starting with 0)
#or a string that represents the name of the column.
#Continue fetching rows until the fetch method returns False, which indicates that you
#have reached
#the end of the result set.
# column position 0 is for ID and column position 1 is for FNAME
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

In [231]:

```
#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

In [124]:

```
import pandas
import ibm_db_dbi
```

In [130]:

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

In [234]:

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

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

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

Out[234]:

'Ahuja'

In [103]:

```
#print the entire data frame
df
```

Out[103]:

	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

In [101]:

```
df.shape
```

Out[101]:

(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.

In [104]:

```
ibm_db.close(conn)
```

Out[104]:

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.

Author

[Rav Ahuja \(https://www.linkedin.com/in/ravahuja/\)](https://www.linkedin.com/in/ravahuja/)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-08-28	2.0	Lavanya	Moved lab to course repo in GitLab

© IBM Corporation 2020. All rights reserved.

In []: