

DB0201EN-Week3-1-2-Querying-v4-py

October 26, 2019

Lab: Access DB2 on Cloud using Python

1 Introduction

This notebook illustrates how to access your database instance using Python by following the steps below: 1. Import the `ibm_db` Python library 1. Identify and enter the database connection credentials 1. Create the database connection 1. Create a table 1. Insert data into the table 1. Query data from the table 1. Retrieve the result set into a pandas dataframe 1. 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.

1.1 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

```
[1]: import ibm_db
```

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

1.2 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 your actual Db2 hostname, username, and
    ↪password:
```

```

dsn_hostname = "dashdb-txn-sbox-yp-lon02-02.services.eu-gb.bluemix.net" # e.g.: ↵
↳ "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net"
dsn_uid = "mx117625" # e.g. "abc12345"
dsn_pwd = "ngp3l2w+gkg206d9" # e.g. "7dBZ3wWt9XN6$o0J"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB" # e.g. "BLUDB"
dsn_port = "50000" # e.g. "50000"
dsn_protocol = "TCPIP" # i.e. "TCPIP"

```

1.3 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};").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: mx117625 on host: dashdb-txn-sbox-yp-lon02-02.services.eu-gb.bluemix.net

1.4 Task 4: Create a table in the database

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

```

[15]: #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)
```

1.5 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

```
[16]: #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(30), lname varchar(30), city varchar(30), ccode char(2))"

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

Double-click [here](#) for the solution.

1.6 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

```
[17]: #Construct the query - replace ... with the insert statement
insertQuery = "insert into instructor (id ,fname ,lname ,city, ccode) values_
      ↪(1, 'rav','ahuja','toronto','ca')"
```

```
#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

```
[22]: #replace ... with the insert statement that inserts the remaining two rows of
      ↪data
insertQuery2 = "insert into instructor (id ,fname ,lname ,city, ccode) values_
      ↪(2, 'raul','chong','markham','ca'),(3,'hima','vasudevan','chicago','us')"
```

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

Double-click [here](#) for the solution.

1.7 Task 6: Query data in the table

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

```
[25]: #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)
```

```
[25]: {'ID': 1,
      0: 1,
      'FNAME': 'rav',
      1: 'rav',
      'LNAME': 'ahuja',
      2: 'ahuja',
      'CITY': 'toronto',
      3: 'toronto',
      'CCODE': 'ca',
      4: 'ca'}
```

Double-click [here](#) for the solution.

```
[26]: #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
```

Double-click [here](#) for the solution.

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

```
[27]: #Enter your code below
update = "update instructor set city = 'moosetown' where fname ='rav'"

selectupdate = ibm_db.exec_immediate(conn,update)
```

Double-click [here](#) for the solution.

1.8 Task 7: Retrieve data into Pandas

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

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

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

```
[30]: #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]
```

```
[30]: 'ahuja'
```

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

```
[31]:
```

	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

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

1.9 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)
```

1.10 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.

Copyright © 2017-2018 cognitiveclass.ai. This notebook and its source code are released under the terms of the [MIT License](https://creativecommons.org/licenses/by/4.0/).

```
[ ]:
```