DB0201EN-Week3-1-2-Querying-3-py

September 29, 2018

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 IBMő 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

```
In [11]: 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"

```
In [12]: dsn_driver = "{IBM DB2 ODBC DRIVER}"
         dsn_database = "BLUDB"            # e.g. "BLUDB"
```

```
dsn_hostname = "dashdb-entry-yp-dal10-01.services.dal.bluemix.net" # e.g.: "awh-yp-smal
dsn_port = 50000                # e.g. "50000"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
dsn_uid = "dash6345"        # e.g. "dash104434"
dsn_pwd = "_Oc3Qpk8ZhJ_"        # e.g. "7dBZ3wWt9xN6$o0JiX!m"
```

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

```
In [13]: #Create database connection
         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,

         try:
             conn = ibm_db.connect(dsn, "", "")
             print ("Connected!")

         except:
             print ("Unable to connect to database")
```

```
Connected!
```

1.4 Task 4: Create a table in the database

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

```
In [20]: #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-20-83413676a2ca> in <module>()
  3
```

```
      4 #Now execute the drop statment
----> 5 dropStmt = ibm_db.exec_immediate(conn, dropQuery)


Exception: [IBM][CLI Driver][DB2/LINUXX8664] SQL0204N  "DASH6345.INSTRUCTOR" is an undef
```

1.5 Dont worry

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 "DASH1234.INSTRUCTOR" is an undefined name. SQLSTATE=42704 SQLCODE=-204

```
In [21]: #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(1

         #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

```
In [24]: #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)
```

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

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

```
In [26]: #replace ... with the insert statement that inerts the remaining two rows of data
         insertQuery2 = "insert into INSTRUCTOR values (2, 'Raul', 'Chong', 'Markham', 'CA'), (3

         #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.

```
In [28]: #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)
```

```
Out[28]: {0: 1,
          1: 'Rav',
          2: 'Ahuja',
          3: 'TORONTO',
          4: 'CA  ',
          'CCODE': 'CA  ',
          'CITY': 'TORONTO',
          'FNAME': 'Rav',
          'ID': 1,
          'LNAME': 'Ahuja'}
```

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

```
In [35]: while ibm_db.fetch_row(selectStmt) != False:
             print (" ID:",  ibm_db.result(selectStmt, 0), " FNAME:",  ibm_db.result(selectStmt,
```

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

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

```
In [37]: selectQuery = "UPDATE INSTRUCTOR SET CITY = 'MOOSETOWN' WHERE ID = 1;"
         selectStmt = ibm_db.exec_immediate(conn, selectQuery)
```

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

1.8 Task 7: Retrieve data into Pandas

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

```
In [39]: import pandas
         import ibm_db_dbi
```

```
In [40]: #connection for pandas
         pconn = ibm_db_dbi.Connection(conn)
```

```
In [41]: #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]

Out[41]: 'Chong'

In [42]: #print the entire data frame
         pdf

Out[42]:    ID FNAME       LNAME       CITY  CCODE
         0   2  Raul       Chong    Markham  CA
         1   3  Hima  Vashudevan    CHICAGO  US
         2   1   Rav       Ahuja  MOOSETOWN  CA
```

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 [43]: pdf.shape

Out[43]: (3, 5)
```

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.

```
In [44]: ibm_db.close(conn)

Out[44]: True
```

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.