DB0201EN-Week3-1-2-Querying-Solved

July 1, 2022

1 Access DB2 on Cloud using Python

Estimated time needed: **15** minutes

1.1 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.

1.2 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

The following required modules are pre-installed in the Skills Network Labs environment. However if you run this notebook commands in a different Jupyter environment (e.g. Watson Studio or Ananconda) you may need to install these libraries by removing the `#` sign before `!pip` in the code cell below.

[1]:
```
# These libraries are pre-installed in SN Labs. If running in another␣
 ↪environment please uncomment lines below to install them:
# !pip install --force-reinstall ibm_db==3.1.0 ibm_db_sa==0.3.3
# Ensure we don't load_ext with sqlalchemy>=1.4 (incompadible)
# !pip uninstall sqlalchemy==1.4 -y && pip install sqlalchemy==1.3.24
# !pip install ipython-sql
```

[2]:
```
import ibm_db
```

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

1.3 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"

[3]:
```
#Replace the placeholder values with your actual Db2 hostname, username, and␣
 ↪password:
dsn_hostname = "ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.
 ↪databases.appdomain.cloud"
dsn_uid = "lqh43420"
dsn_pwd = "lb76AfXMQnkkX6lw"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "31505"
dsn_protocol = "TCPIP"
dsn_security = "SSL"
```

1.4 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

[4]:
```
#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:  lqh43420 on host:  ea286ace-86c7-4d5b-
8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud
```

1.5 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)
```

1.6 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]:
```
createQuery = "create table INSTRUCTOR(ID INTEGER PRIMARY KEY NOT NULL, FNAME␣
 ↪VARCHAR(20), LNAME VARCHAR(20), CITY VARCHAR(20), CCODE CHAR(2))"
createStmt = ibm_db.exec_immediate(conn,createQuery)
```

Click here for the solution

```
createQuery = "create table INSTRUCTOR(ID INTEGER PRIMARY KEY NOT NULL, FNAME VARCHAR(20), LNAI

createStmt = ibm_db.exec_immediate(conn,createQuery)
```

1.7 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]: insertQuery = "insert into INSTRUCTOR values (1, 'Rav', 'Ahuja', 'TORONTO',␣
       ↪'CA')"
     insertStmt = ibm_db.exec_immediate(conn, insertQuery)
```

Click here for the solution

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

insertStmt = ibm_db.exec_immediate(conn, insertQuery)
```

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

```
[8]: insertQuery2 = "insert into INSTRUCTOR values (2, 'Raul', 'Chong', 'Markham',␣
       ↪'CA'), (3, 'Hima', 'Vasudevan', 'Chicago', 'US')"
     insertStmt2 = ibm_db.exec_immediate(conn, insertQuery2)
```

Click here for the solution

```
insertQuery2 = "insert into INSTRUCTOR values (2, 'Raul', 'Chong', 'Markham', 'CA'), (3, 'Hima

insertStmt2 = ibm_db.exec_immediate(conn, insertQuery2)
```

1.8 Task 6: Query data in the table

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

```
[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)
     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

```
#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)
ibm_db.fetch_both(selectStmt)
```

[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

```
#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'
```

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

[11]:
```
updateQuery = "update INSTRUCTOR set CITY='MOOSETOWN' where FNAME='Rav'"
updateStmt = ibm_db.exec_immediate(conn, updateQuery)
```

Click here for the solution

```
updateQuery = "update INSTRUCTOR set CITY='MOOSETOWN' where FNAME='Rav'"
updateStmt = ibm_db.exec_immediate(conn, updateQuery)
```

1.9 Task 7: Retrieve data into Pandas

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

[12]:
```
import pandas
import ibm_db_dbi
```

[13]:
```
#connection for pandas
pconn = ibm_db_dbi.Connection(conn)
```

[14]:
```
#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]
```

```
C:\Users\ONKAR\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy
connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2
connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

[14]:
```
'Ahuja'
```

[15]:
```
#print the entire data frame
pdf
```

[15]:
```
   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

[16]:
```
pdf.shape
```

[16]:
```
(3, 5)
```

1.10 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.

[17]:
```
ibm_db.close(conn)
```

[17]:
```
True
```

1.11 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.

1.12 Author

Rav Ahuja

1.13 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-11-17	2.2	Lakshmi	Updated library
2021-07-09	2.1	Malika	Updated connection string

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

##