Access DB2 on Cloud using Python

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 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.

In [1]:
```
# These libraries are pre-installed in SN Labs. If running in another environmen
# !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
```

In [9]:
```
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 [3]:
```
#Replace the placeholder values with the actuals for your Db2 Service Credential
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"            # e.g. "BLUDB"
dsn_hostname = "1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databa
dsn_port = "32286"                    # e.g. "50000"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
dsn_uid = "dxc70089"                # e.g. "abc12345"
dsn_pwd = "A9vq58iRSQlvpgZk"                # 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

In [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, ds

try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on ho

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

```
Connected to database:  bludb as user:  dxc70089 on host:  1bbf73c5-d84a-4bb0-85b
9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.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:

Table definition

INSTRUCTOR

COLUMN NAME	DATA TYPE	NULLABLE
ID	INTEGER	N
FNAME	VARCHAR	Y
LNAME	VARCHAR	Y
CITY	VARCHAR	Y
CCODE	CHARACTER	Y

In [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)
/tmp/ipykernel_1084/595635871.py in <module>
      3
      4 #Now execute the drop statment
----> 5 dropStmt = ibm_db.exec_immediate(conn, dropQuery)

Exception: [IBM][CLI Driver][DB2/LINUXX8664] SQL0204N  "DXC70089.INSTRUCTOR" is a
n 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

In [6]:
```
#Construct the Create Table DDL statement - replace the ... with rest of the sta
createQuery = "create table INSTRUCTOR(id INTEGER PRIMARY KEY NOT NULL, fname va

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

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_1084/640938283.py in <module>
      3
      4 #Now fill in the name of the method and execute the statement
----> 5 createStmt = ibm_db.replace_with_name_of_execution_method(conn, createQue
ry)

AttributeError: module 'ibm_db' has no attribute 'replace_with_name_of_execution_
method'
```

▶ 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 INTEGER	FNAME VARCHAR(20)	LNAME VARCHAR(20)	CITY VARCHAR(20)	CCODE 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 [7]:
```
#Construct the query - replace ... with the insert statement
insertQuery = "insert into instructor values (1,'rav', 'ahuja', 'torontoo', 'ca'

#execute the insert statement
insertStmt = ibm_db.exec_immediate(conn, insertQuery)
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
/tmp/ipykernel_1084/2784377471.py in <module>
      3
      4 #execute the insert statement
----> 5 insertStmt = ibm_db.exec_immediate(conn, insertQuery)

Exception: [IBM][CLI Driver][DB2/LINUXX8664] SQL0204N  "DXC70089.INSTRUCTOR" is a
n undefined name.  SQLSTATE=42704 SQLCODE=-204
```

▶ Click here for the solution

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

In [8]:
```
#replace ... with the insert statement that inerts the remaining two rows of dat
insertQuery2 = "insert into instructor values (2, 'raul', 'chong', 'markham', 'c

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

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
/tmp/ipykernel_1084/2884797019.py in <module>
      3
      4 #execute the statement
----> 5 insertStmt2 = ibm_db.exec_immediate(conn, insertQuery2)

Exception: [IBM][CLI Driver][DB2/LINUXX8664] SQL0204N  "DXC70089.INSTRUCTOR" is a
n undefined name.  SQLSTATE=42704 SQLCODE=-204
```

▶ 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 [ ]:
```
#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
...
```

▶ Click here for the solution

In [ ]:
```
#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(selec
```

▶ Click here for the solution

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

In [ ]:
```
#Enter your code below
```

▶ 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 [ ]:
```
import pandas
import ibm_db_dbi
```

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

In [ ]:
```
#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]
```

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

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

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

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

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
2020-08-28	2.0	Lavanya	Moved lab to course repo in GitLab