



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh  
**TRUNG TÂM TIN HỌC**

# **DATABASE AND SQL FOR DATA SCIENCE**

*Phân tích dữ liệu với Python*

**Phòng LT & Mạng**

<http://csc.edu.vn/kiem-thu-phan-mem>

**2019**



**Data  
Science**

1. **IBM\_db API**
2. Kết nối database sử dụng **IBM\_db API**
3. Tạo bảng, tải và truy vấn dữ liệu sử dụng **IBM\_db API**
4. Sử dụng Magic SQL (%) với **IBM\_db API**
5. Phân tích dữ liệu với **Python**
6. Làm việc với **SQLite** và **PostgreSQL**

- ❑ Thư viện `ibm_db` API cung cấp nhiều hàm hữu ích để truy cập vào thao tác dữ liệu trong máy chủ dữ liệu của IBM.
- ❑ `ibm_db` API sử dụng IBM Data Server Driver for ODBC và CLI APIs để kết nối tới IBM, DB2 và Informix

❑ Để kết nối tới DB2 Warehouse yêu cầu các thông tin:

- driver name
- database name
- host DNS name or IP address
- host port
- connection protocol
- user ID
- user password

1. `ibm_db` API
2. Kết nối database sử dụng `ibm_db` API
3. Tạo bảng, tải và truy vấn dữ liệu sử dụng `ibm_db` API
4. Sử dụng Magic SQL (%) với `ibm_db` API
5. Phân tích dữ liệu với Python
6. Làm việc với SQLite và PostgreSQL

# Kết nối database sử dụng ibm\_db API

---

- ❑ Cài đặt thư viện

**`pip install ibm_db`**



- ❑ Truy cập địa chỉ

**`https://console.bluemix.net/dashboard/apps`**

- ❑ Đăng ký tài khoản và đăng nhập vào hệ thống

- ❑ Trong màn hình dashboard, chọn dịch vụ DB2-xx

# Kết nối database sử dụng ibm\_db API

 IBM Cloud Catalog Docs Support Manage 

Dashboard


RESOURCE GROUP  
All Resources ▾

CLOUD FOUNDRY ORG  
All Organizations ▾

CLOUD FOUNDRY SPACE  
All Spaces ▾

LOCATION  
All Locations ▾

CATEGORY  
All Categories ▾

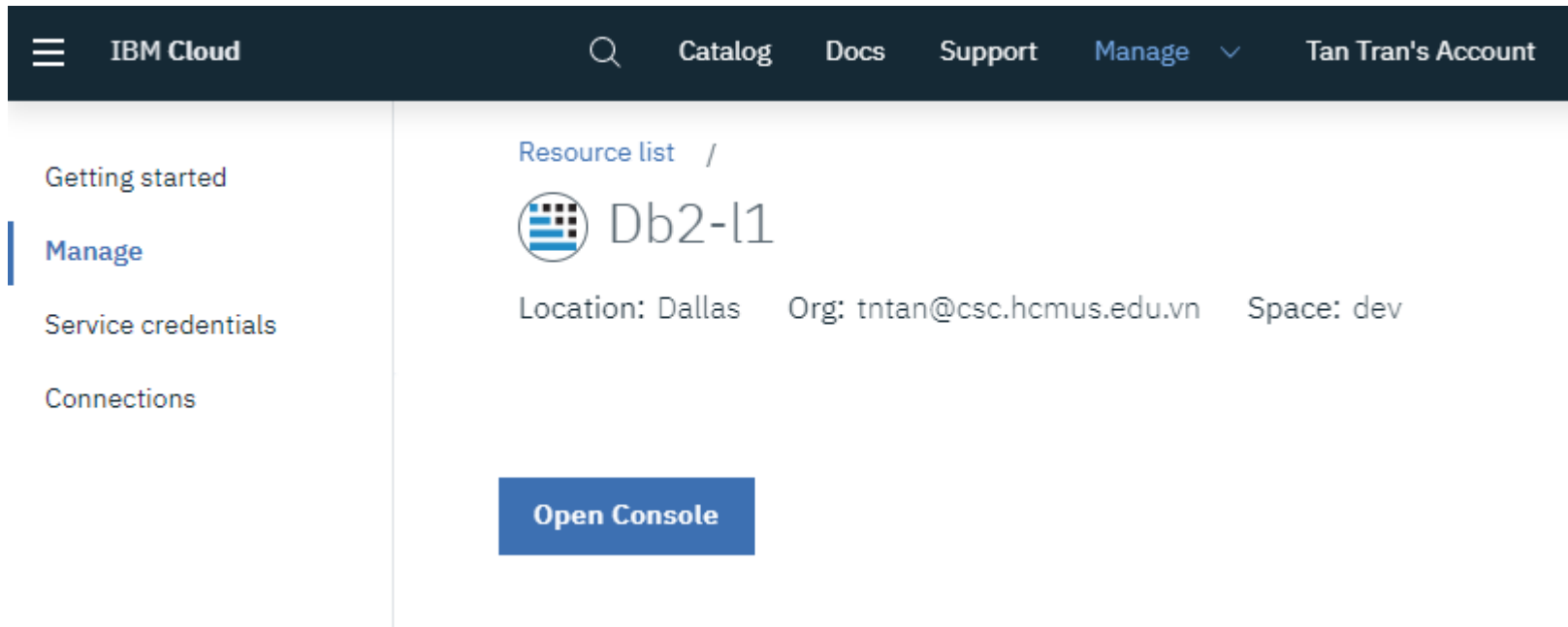


Cloud Foundry Services

Name ▴	Region	CF Org	CF Space	Plan
Db2-s6	Dallas	xagurah@kht...	dev	Lite

# Kết nối database sử dụng ibm\_db API

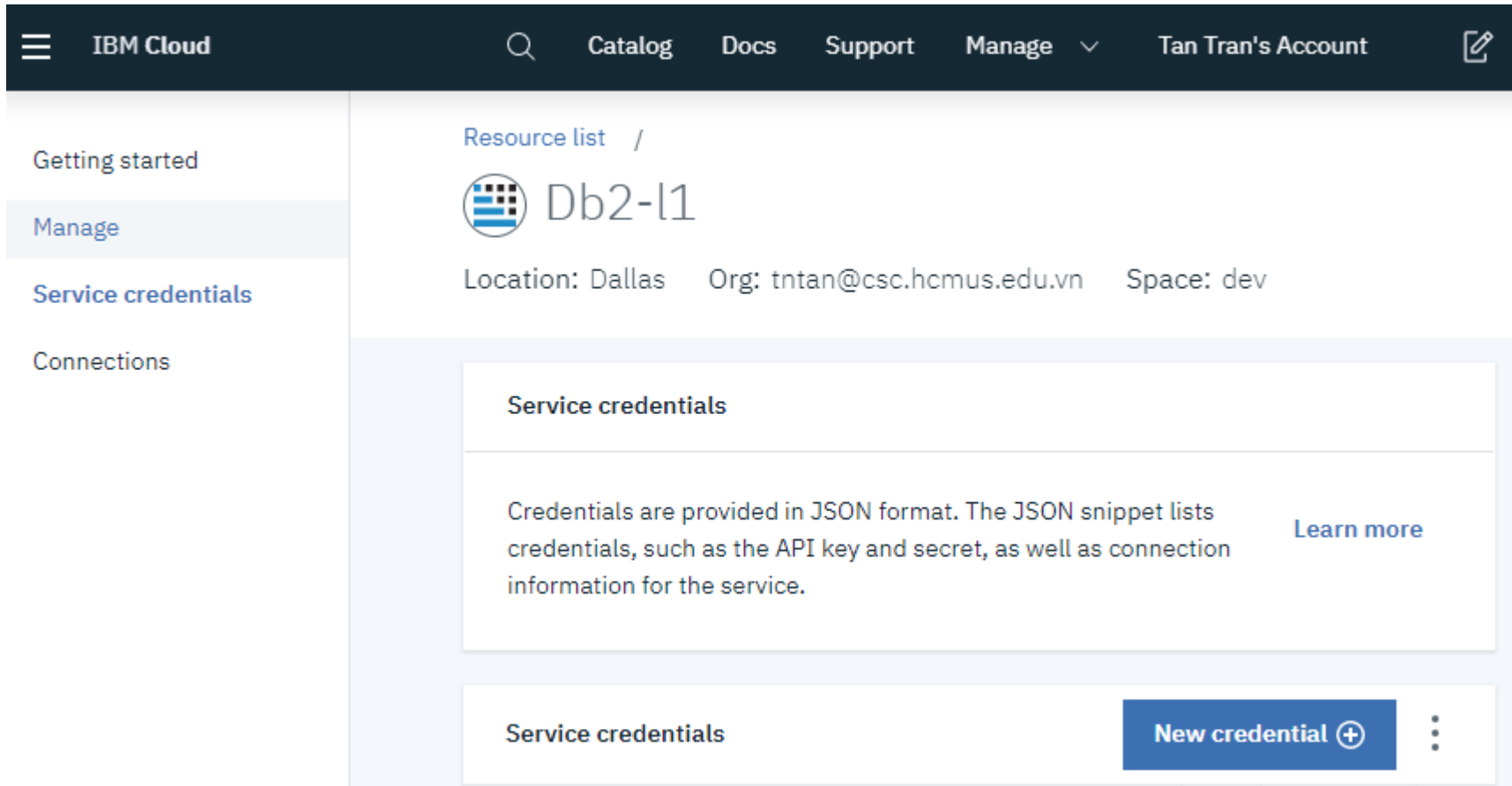
- ❑ Trong thực đơn bên trái nhấn chọn Service credentials






# Kết nối database sử dụng ibm\_db API

- ❑ Nhấn nút New credential để tạo credential, điền tên và nhấn nút Add để tạo credential



IBM Cloud

Resource list /

 Db2-l1

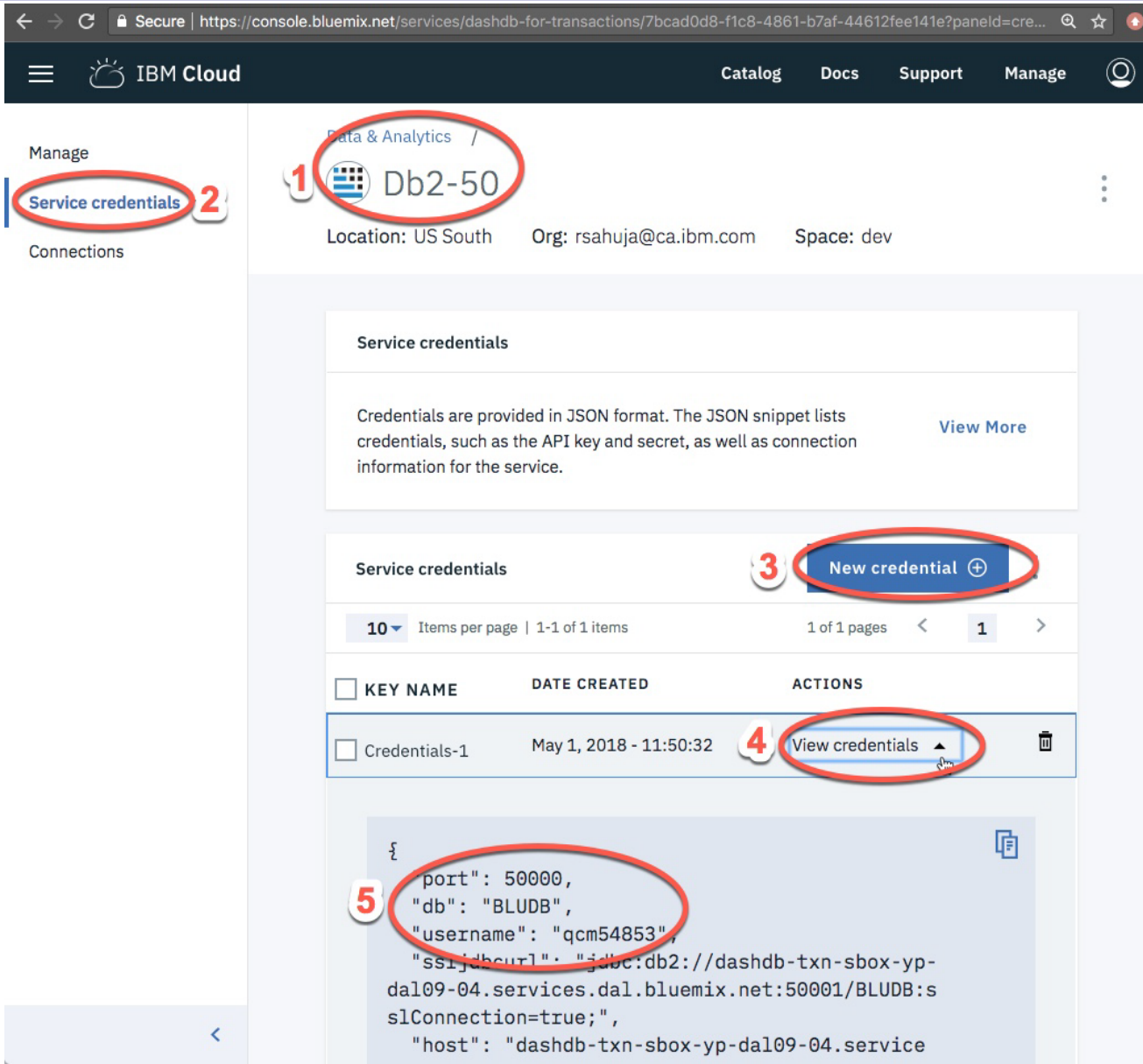
Location: Dallas Org: tntan@csc.hcmus.edu.vn Space: dev

**Service credentials**

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. [Learn more](#)

**Service credentials** [New credential +](#)

# Kết nối database sử dụng ibm\_db API



Manage  
**Service credentials**  
Connections

Data & Analytics / **Db2-50**  
Location: US South Org: rsahuja@ca.ibm.com Space: dev

**Service credentials**

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. [View More](#)

**Service credentials** **New credential**

10 Items per page | 1-1 of 1 items 1 of 1 pages 1

<input type="checkbox"/>	KEY NAME	DATE CREATED	ACTIONS
<input type="checkbox"/>	Credentials-1	May 1, 2018 - 11:50:32	<b>View credentials</b>

```
{
  "port": 50000,
  "db": "BLUDB",
  "username": "qcm54853",
  "ssljdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-
dal09-04.services.dal.ibmcloud.net:50001/BLUDB:s
slConnection=true;",
  "host": "dashdb-txn-sbox-yp-dal09-04.service
```

# Kết nối database sử dụng ibm\_db API

---

## ❑ Ghi nhận các thông tin

- Port: cổng giao tiếp của database
- Db: tên của database
- Host: tên máy chủ của database instance
- Username: tên đăng nhập
- Password: mật khẩu
- URI: địa chỉ kết nối khi sử dụng SQL Magic trong Jupyter notebooks

# Kết nối database sử dụng ibm\_db API

## □ Các bước kết nối đến DB2

- Bước 1: tham chiếu thư viện ibm\_db

```
import ibm_db
```

- Bước 2 khai báo các thông tin kết nối

```
dsn_database = "BLUDB"  
dsn_hostname = "dashdb-txn-sbox-yp-dal09-03.services.dal.ibmcloud.net"  
dsn_port = "50000"  
dsn_protocol = "TCPIP"  
dsn_uid = "bmr71783"  
dsn_pwd = "t^j600d24tbzd2xk"
```

# Kết nối database sử dụng ibm\_db API

## □ Các bước kết nối đến DB2

### ● Bước 3: tạo kết nối

```
#Create database connection
dsn = (
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "PROTOCOL={3};"
    "UID={4};"
    "PWD={5};").format(dsn_database, dsn_hostname, dsn_port, dsn_protocol, dsn_uid, dsn_pwd)

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

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

Connected!

# Kết nối database sử dụng ibm\_db API

---

## ❑ Các bước kết nối đến DB2

- Bước 4: thực hiện các câu lệnh truy vấn
- Bước 5: đóng kết nối

`ibm_db.close(conn)`

1. `ibm_db` API
2. Kết nối database sử dụng `ibm_db` API
3. Tạo bảng, tải và truy vấn dữ liệu sử dụng `ibm_db` API
4. Sử dụng Magic SQL (%) với `ibm_db` API
5. Phân tích dữ liệu với Python
6. Làm việc với SQLite và PostgreSQL

# Thao tác dữ liệu sử dụng ibm\_db API

---

## ❑ Thực thi câu lệnh truy vấn đơn:

- Sử dụng hàm `exec_immediate`
- Cú pháp:

`ibm_db.exec_immediate(< IBM_DBConnection>, <query statement>)`



# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Tạo bảng sử dụng ibm\_db API:

```
#Construct the Create Table DDL 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)
```

## ❑ Xóa bảng sử dụng ibm\_db API

```
dropQuery = "Drop table INSTRUCTOR"

dropStmt = ibm_db.exec_immediate(conn, dropQuery)
```

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Thêm dữ liệu sử dụng ibm\_db API:

### ● Thêm 1 dòng dữ liệu:

```
#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)
```

### ● Thêm nhiều dòng dữ liệu

```
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)
```

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Truy vấn dữ liệu sử dụng ibm\_db API:

### ● Câu lệnh truy vấn đơn:

```
#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)
```

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

# Thao tác dữ liệu sử dụng ibm\_db API

---

## ❑ Truy vấn dữ liệu sử dụng ibm\_db API:

- Câu lệnh truy vấn có tham số:

- Sử dụng hàm prepare để chuyển đổi các tham số được liệt kê

`stmt = ibm_db.prepare(connection, statement)`

- Sử dụng hàm bind\_param để truyền giá trị cho tham số (optional)

`ibm_db.bind_param(stmt, parameter_number, value)`

- Sử dụng hàm execute để thi hành câu truy vấn

`ibm_db.execute(stmt)`

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Truy vấn dữ liệu sử dụng ibm\_db API:

- Câu lệnh truy vấn có tham số:

```
selectQuery = "select * from INSTRUCTOR WHERE CITY = ? and CCODE = ?"
stmt = ibm_db.prepare(conn, selectQuery)
city = 'Chicago'
ccode = 'US'
# Explicitly bind parameters
ibm_db.bind_param(stmt, 1, city)
ibm_db.bind_param(stmt, 2, ccode)
ibm_db.execute(stmt)
```

# Thao tác dữ liệu sử dụng `ibm_db` API

---

- ❑ Duyệt qua từng dòng trong dữ liệu kết quả
  - **`fetch_tuple`**: Trả về một tuple, được lập chỉ mục theo vị trí cột, biểu thị một hàng trong tập kết quả.
  - **`fetch_assoc`**: Trả về một dictionary, được lập chỉ mục theo tên cột, biểu thị một hàng trong tập kết quả.

# Thao tác dữ liệu sử dụng `ibm_db` API

---

- ❑ Duyệt qua từng dòng trong dữ liệu kết quả
  - **`fetch_both`**: Trả về một dictionary, được lập chỉ mục bởi cả tên và vị trí cột, biểu thị một hàng trong tập kết quả.
  - **`fetch_row`**: Đặt con trỏ tập kết quả sang hàng tiếp theo hoặc hàng được yêu cầu. Sử dụng chức năng này để lặp qua một tập kết quả.

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Duyệt qua từng dòng trong dữ liệu kết quả

- fetch\_both:

```
#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)
```

```
dictionary = ibm_db.fetch_both(selectStmt)
```

```
while dictionary != False:
```

```
    print(dictionary)
```

```
    dictionary = ibm_db.fetch_both(selectStmt)
```

```
{ 'ID': 2, 0: 2, 'FNAME': 'Raul', 1: 'Raul', 'LNAME': 'Chong', 2: 'Chong',  
  'CITY': 'Markham', 3: 'Markham', 'CCODE': 'CA', 4: 'CA' }
```

```
{ 'ID': 3, 0: 3, 'FNAME': 'Hima', 1: 'Hima', 'LNAME': 'Vasudevan', 2: 'Vasu  
devan', 'CITY': 'Chicago', 3: 'Chicago', 'CCODE': 'US', 4: 'US' }
```



# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Duyệt qua từng dòng trong dữ liệu kết quả

- fetch\_tuple:

```
selectQuery = "select * from INSTRUCTOR WHERE CITY = ? and CCODE = ?"
stmt = ibm_db.prepare(conn, selectQuery)
city = 'Chicago'
ccode = 'US'
# Explicitly bind parameters
ibm_db.bind_param(stmt, 1, city)
ibm_db.bind_param(stmt, 2, ccode)
ibm_db.execute(stmt)

#Fetch the rest of the rows and print the ID and FNAME for those rows
tuple = ibm_db.fetch_tuple(stmt)
while tuple != False:
    print(tuple)
    tuple = ibm_db.fetch_tuple(stmt)
```

(3, 'Hima', 'Vasudevan', 'Chicago', 'US')

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Duyệt qua từng dòng trong dữ liệu kết quả

### ● fetch\_assoc:

```
selectQuery = "select * from INSTRUCTOR WHERE CITY = ? and CCODE = ?"
stmt = ibm_db.prepare(conn, selectQuery)
city = 'Chicago'
ccode = 'US'
# Explicitly bind parameters
ibm_db.bind_param(stmt, 1, city)
ibm_db.bind_param(stmt, 2, ccode)
ibm_db.execute(stmt)

dictionary = ibm_db.fetch_assoc(stmt)
while dictionary != False:
    print(dictionary)
    dictionary = ibm_db.fetch_assoc(stmt)
```

```
{'ID': 3, 'FNAME': 'Hima', 'LNAME': 'Vasudevan', 'CITY': 'Chicago', 'CCODE': 'US'}
```

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Duyệt qua từng dòng trong dữ liệu kết quả

- `fetch_row`:

```
#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)
while ibm_db.fetch_row(selectStmt) != False:
    print ("The Employee number is : ", ibm_db.result(selectStmt, 0))
    print ("The last name is : ", ibm_db.result(selectStmt, "LNAME"))
```

```
The Employee number is : 1
The last name is : Ahuja
The Employee number is : 2
The last name is : Chong
The Employee number is : 3
The last name is : Vasudevan
```

# Thao tác dữ liệu sử dụng ibm\_db API

## ❑ Sử dụng Pandas để đọc dữ liệu

```
# Using Pandas
import pandas
import ibm_db_dbi

pconn = ibm_db_dbi.Connection(conn)
df = pandas.read_sql('SELECT * FROM INSTRUCTOR', pconn)
```

df

	ID	FNAME	LNAME	CITY	CCODE
0	1	Rav	Ahuja	TORONTO	CA
1	2	Raul	Chong	Markham	CA
2	3	Hima	Vasudevan	Chicago	US

1. `ibm_db` API
2. Kết nối database sử dụng `ibm_db` API
3. Tạo bảng, tải và truy vấn dữ liệu sử dụng `ibm_db` API
4. Sử dụng Magic SQL (%) với `ibm_db` API
5. Phân tích dữ liệu với Python
6. Làm việc với SQLite và PostgreSQL

# Sử dụng Magic SQL (%) với ibm\_db API

## ❑ Load thư viện ipython-sql

```
%load_ext sql
```

## ❑ Tạo kết nối

- Sử dụng thuộc tính URI trong Credentials, thay thế db2:// bằng ibm\_db\_sa://

```
%sql ibm_db_sa://bmr71783:t%5Ej600d24tbzd2xk@dashdb-txn-sbox-yp-dal09-03
```

```
'Connected: bmr71783@BLUDB'
```

# Sử dụng Magic SQL (%) với ibm\_db API

## □ Tạo cấu trúc bảng

```
%%sql
```

```
CREATE TABLE INTERNATIONAL_STUDENT_TEST_SCORES (  
country VARCHAR(50),  
first_name VARCHAR(50),  
last_name VARCHAR(50),  
test_score INT  
);
```

```
* ibm_db_sa://bmr71783:***@dashdb-txn-sbox-yp-dal09-03.servic  
es.dal.ibmcloud.com:50000/BLUDB  
Done.
```

# Sử dụng Magic SQL (%) với ibm\_db API

## ❑ Thêm dữ liệu vào bảng

```
%%sql
```

```
INSERT INTO INTERNATIONAL_STUDENT_TEST_SCORES (country, first_name, last_name, test_score)
VALUES
('United States', 'Marshall', 'Bernadot', 54),
('Ghana', 'Celinda', 'Malkin', 51),
('Ukraine', 'Guillermo', 'Furze', 53),
('Greece', 'Aharon', 'Tunnow', 48),
('Russia', 'Bail', 'Goodwin', 46),
('Poland', 'Cole', 'Winteringham', 49),
('Sweden', 'Emlyn', 'Erricker', 55),
('Russia', 'Cathee', 'Sivewright', 49),
('China', 'Barney', 'Ingerson', 57);
```

```
* ibm_db_sa://bmr71783:***@dashdb-txn-sbox-yp-dal09-03.services.dal.ibmcloud.com:50000/BLUDB
9 rows affected.
```



# Sử dụng Magic SQL (%) với ibm\_db API

## ❑ Truy vấn dữ liệu sử dụng biến

```
country = "Russia"
%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = :country

* ibm_db_sa://bmr71783:***@dashdb-txn-sbox-yp-dal09-03.services.dal.ibmcloud.com:50000/BLUDB
Done.
```

country	first_name	last_name	test_score
Russia	Bail	Goodwin	46
Russia	Cathee	Sivewright	49

# Sử dụng Magic SQL (%) với ibm\_db API

## ❑ Gán kết quả truy vấn cho biến

```
%sql test_score_distribution <<
SELECT test_score as "Test Score", count(*) as "Frequency"
from INTERNATIONAL_STUDENT_TEST_SCORES
GROUP BY test_score;
```

```
* ibm_db_sa://bmr71783:***@dashdb-txn-sbox-yp-dal09-03.services.dal.ibmcloud.com:50000/BLUDB
Done.
Returning data to local variable test_score_distribution
```

### • Hoặc

```
test_score_distribution = %sql SELECT test_score as "Test Score", count(*) as "F
```

```
<
* ibm_db_sa://bmr71783:***@dashdb-txn-sbox-yp-dal09-03.services.dal.ibmcloud.com:50000/BLUDB
Done.
Returning data to local variable test_score_distribution
```

# Sử dụng Magic SQL (%) với ibm\_db API

## ❑ Xem kết quả

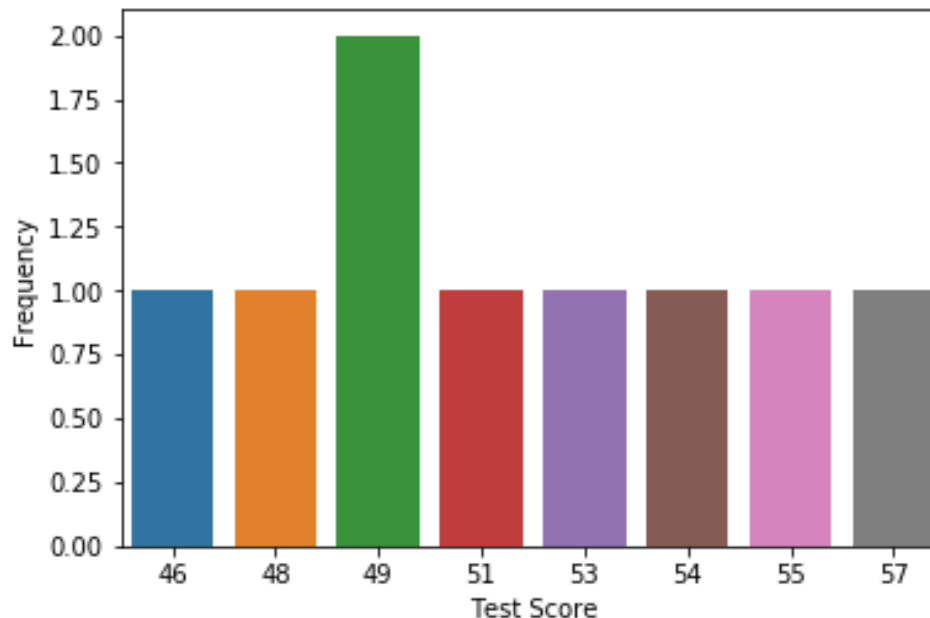
```
test_score_distribution
```

Test Score	Frequency
46	1
48	1
49	2
51	1
53	1
54	1
55	1
57	1

# Sử dụng Magic SQL (%) với ibm\_db API

## ☐ Trực quan hóa

```
dataframe = test_score_distribution.DataFrame()  
  
%matplotlib inline  
import seaborn  
  
plot = seaborn.barplot(x='Test Score',y='Frequency', data=dataframe)
```

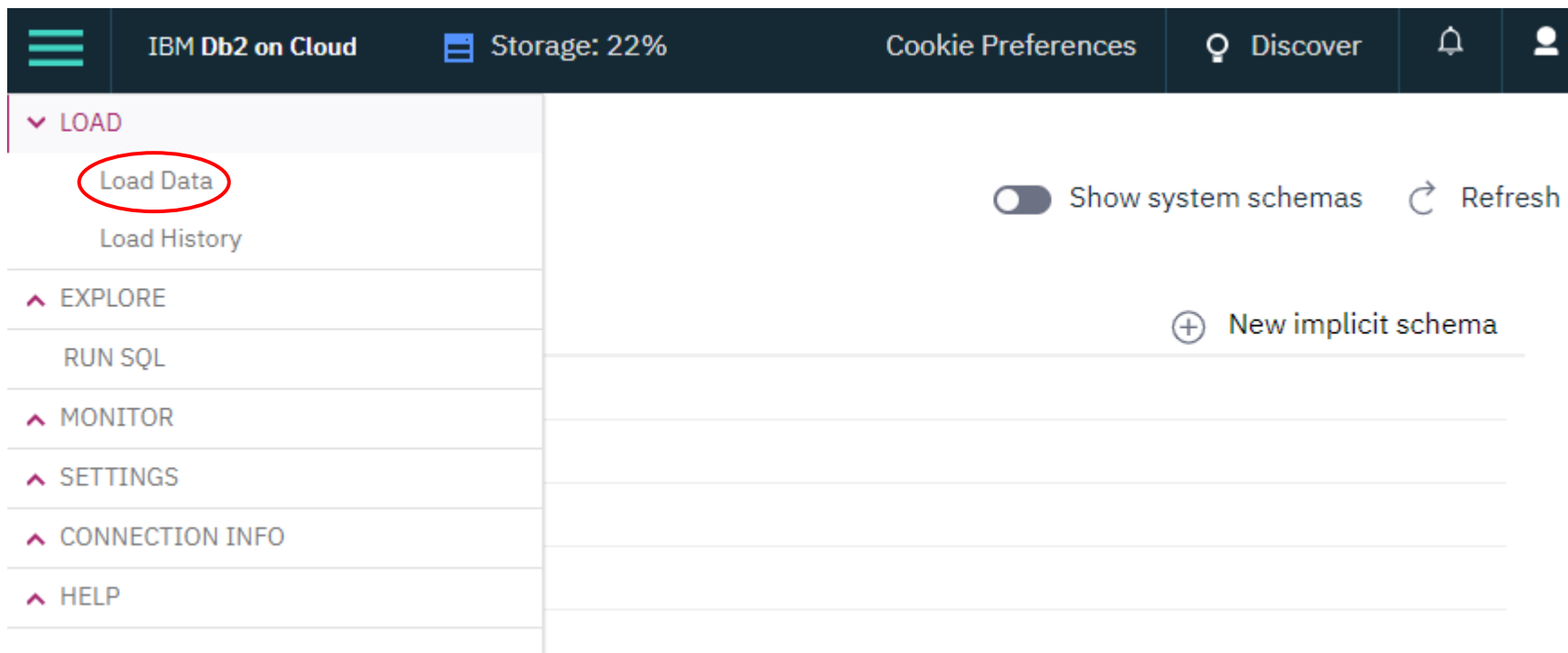


1. `ibm_db` API
2. Kết nối database sử dụng `ibm_db` API
3. Tạo bảng, tải và truy vấn dữ liệu sử dụng `ibm_db` API
4. Sử dụng Magic SQL (%) với `ibm_db` API
5. Phân tích dữ liệu với Python
6. Làm việc với SQLite và PostgreSQL

# Phân tích dữ liệu với Python

## ❑ Upload CSV file lên DB2 warehouse

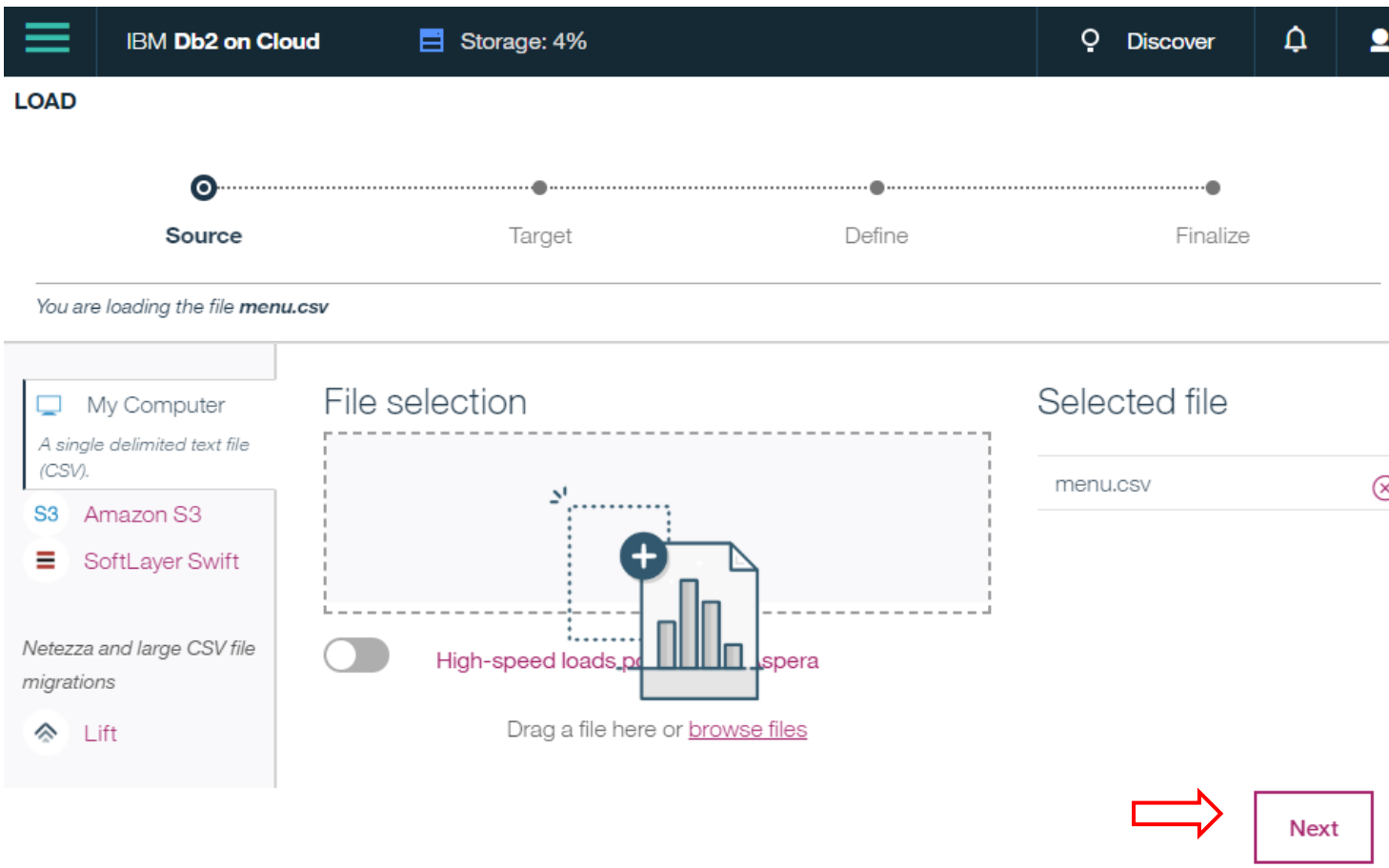
- Trong cửa sổ DB2 warehouse, chọn Load → Load Data ở menu bên phải



# Phân tích dữ liệu với Python

## ❑ Upload CSV file lên DB2 warehouse

- Chọn tập tin csv để upload và nhấn Next

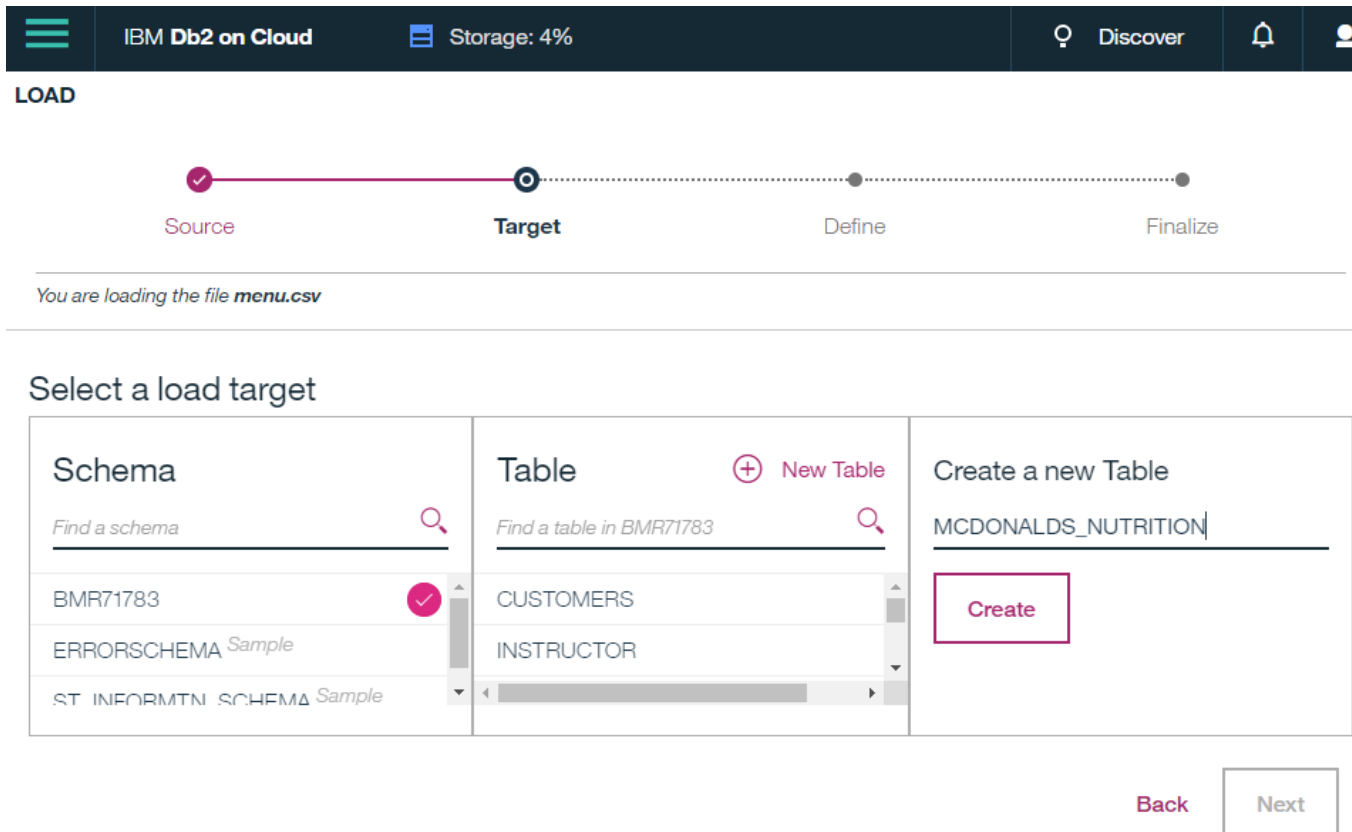


The screenshot shows the IBM Db2 on Cloud console interface. At the top, there's a navigation bar with a hamburger menu, the text "IBM Db2 on Cloud", a storage indicator "Storage: 4%", and buttons for "Discover", a bell icon, and a user profile icon. Below the navigation bar, the word "LOAD" is displayed. A progress bar shows four steps: "Source" (selected), "Target", "Define", and "Finalize". Below the progress bar, a message states "You are loading the file **menu.csv**". The main area is divided into three sections: "File selection" on the left, "File selection" in the center, and "Selected file" on the right. The "File selection" section on the left lists various storage options: "My Computer" (described as "A single delimited text file (CSV)."), "S3 Amazon S3", "SoftLayer Swift", "Netezza and large CSV file migrations", and "Lift". The central "File selection" area features a large dashed box with a plus icon and a document icon, a toggle switch for "High-speed loads, powered by Aspera", and the text "Drag a file here or [browse files](#)". The "Selected file" section on the right shows "menu.csv" with a close button (X). At the bottom right, there is a red arrow pointing to a "Next" button.

# Phân tích dữ liệu với Python

## ❑ Upload CSV file lên DB2 warehouse

- Chọn target Schema và Table để chứa dữ liệu (có thể chọn Table đã có hoặc tạo Table mới)



LOAD

Source Target Define Finalize

You are loading the file **menu.csv**

Select a load target

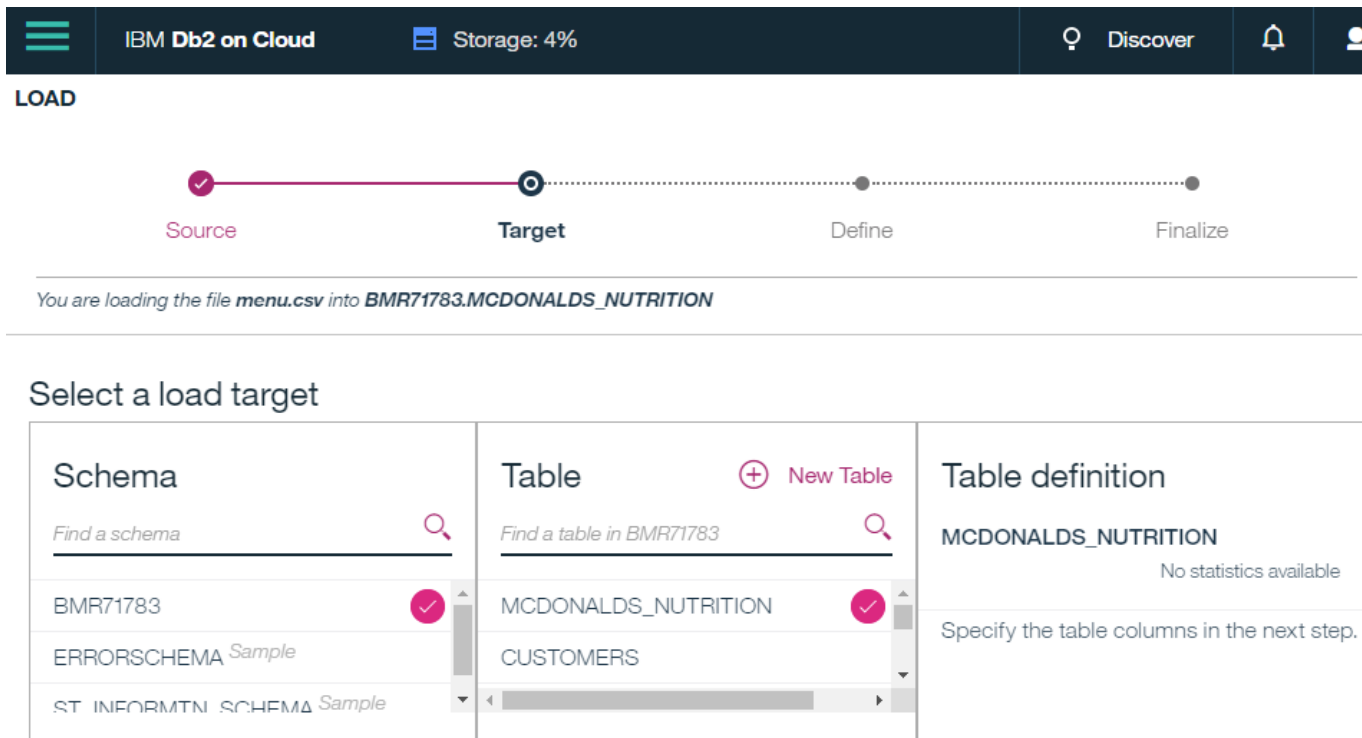
Schema	Table	Create a new Table
<input type="text" value="Find a schema"/>	<input type="text" value="Find a table in BMR71783"/>	<input type="text" value="MCDONALDS_NUTRITION"/>
BMR71783 ✓	CUSTOMERS	<div>Create</div>
ERRORSCHEMA <i>Sample</i>	INSTRUCTOR	
ST_INFORMTN_SCHEMA <i>Sample</i>		

Back Next



## ❑ Upload CSV file lên DB2 warehouse

- Chọn target Schema và Table để chứa dữ liệu (có thể chọn Table đã có hoặc tạo Table mới)



LOAD

Source Target Define Finalize

You are loading the file **menu.csv** into **BMR71783.MCDONALDS\_NUTRITION**

Select a load target

Schema	Table	Table definition
<i>Find a schema</i>	<i>Find a table in BMR71783</i>	<b>MCDONALDS_NUTRITION</b> No statistics available
BMR71783 ✓	MCDONALDS_NUTRITION ✓	Specify the table columns in the next step.
ERRORSCHEMA <i>Sample</i>	CUSTOMERS	
ST INFORMTN SCHEMA <i>Sample</i>		

Back






Next




# Phân tích dữ liệu với Python

## ❑ Upload CSV file lên DB2 warehouse


- Định nghĩa các cột dữ liệu

 IBM Db2 on Cloud  Storage: 4%  Discover  

LOAD




You are loading the file **menu.csv** into **BMR71783.MCDONALDS\_NUTRITION**






Code page (character encoding): **1208 (UTF-8)** 



Separator: **,**

Header in first row: ☒

Time & date format: 



Detect data types: ☒




	<b>Category</b>  VARCHAR(18)	<b>Item</b>  VARCHAR(61)	<b>Serving_Size</b>  VARCHAR(17)	<b>Calories</b>  SMALLINT	<b>Calories_from_Fat</b>  SMALLINT
1	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120
2	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70
3	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200
4	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250
5	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210
6	Breakfast	Steak & Egg McMuffin	6.5 oz (185 g)	430	210
7	Breakfast	Bacon, Egg & Cheese Biscuit (Regular Biscuit)	5.3 oz (150 g)	460	230
8	Breakfast	Bacon, Egg & Cheese Biscuit (Large Biscuit)	5.8 oz (164 g)	520	270
9	Breakfast	Bacon, Egg & Cheese Biscuit with Egg Whites (Regular Biscuit)	5.4 oz (153 g)	410	180
10	Breakfast	Bacon, Egg & Cheese Biscuit with Egg Whites (Large Biscuit)	5.8 oz (164 g)	520	270

 Back  Next


## ❑ Upload CSV file lên DB2 warehouse


- Kiểm tra lại các thiết lập và nhấn Begin Load


 IBM Db2 on Cloud  Storage: 4%


 Discover  

LOAD

 Source

 Target

 Define

 Finalize

You are loading the file **menu.csv** into **BMR71783.MCDONALDS\_NUTRITION**

Review settings

Summary

Code page:

1208 (Default)

Separator:

, (Default)

Header in first row:

Yes (Default)

Time format:

HH:MM:SS (Default)

Date format:

YYYY-MM-DD (Default)

Timestamp format:

YYYY-MM-DD  
HH:MM:SS (Default)

String delimiter:

"(Default)

Option

Maximum number of warnings

1000

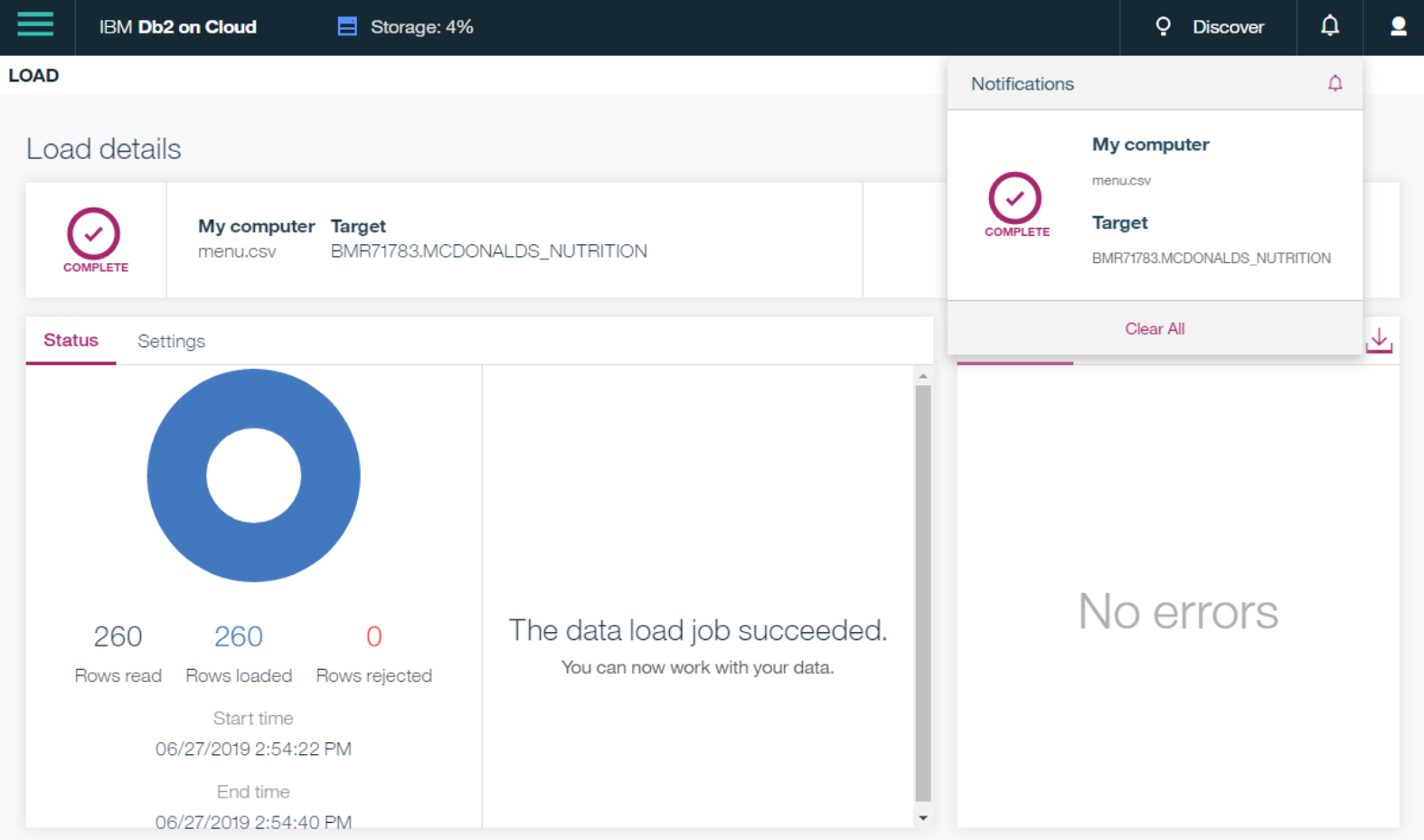
Back

Begin Load

# Phân tích dữ liệu với Python

## ❑ Upload CSV file lên DB2 warehouse

### ● Kết quả sau khi Load thành công



The screenshot displays the IBM Db2 on Cloud interface. At the top, a dark navigation bar shows 'IBM Db2 on Cloud' and 'Storage: 4%'. Below this, the 'LOAD' section is active. The 'Load details' card shows a 'My computer' source (menu.csv) loading into a 'Target' (BMR71783.MCDONALDS\_NUTRITION). The status is 'COMPLETE'. A 'Notifications' panel on the right also shows the completion of the load. The 'Status' tab is selected, showing a large blue donut chart with 260 rows read, 260 rows loaded, and 0 rows rejected. The start time is 06/27/2019 2:54:22 PM and the end time is 06/27/2019 2:54:40 PM. A message states: 'The data load job succeeded. You can now work with your data.' The 'No errors' section is empty.

Source	Target	Status
My computer menu.csv	BMR71783.MCDONALDS_NUTRITION	COMPLETE

Rows read	Rows loaded	Rows rejected
260	260	0

Start time: 06/27/2019 2:54:22 PM  
End time: 06/27/2019 2:54:40 PM

The data load job succeeded.  
You can now work with your data.

No errors

# Phân tích dữ liệu với Python

## ❑ Sử dụng SQL kiểm chứng lại dữ liệu đã Load

```
selectQuery = "Select count(*) from BMR71783.MCDONALDS_NUTRITION"  
stmt = ibm_db.exec_immediate(conn, selectQuery)  
  
ibm_db.fetch_both(stmt)  
  
{ '1': '260', 0: '260' }
```

## ❑ Sử dụng Pandas để nhận dữ liệu từ truy vấn

```
In [6]: import pandas as pd
import ibm_db_dbi
pconn = ibm_db_dbi.Connection(conn)
df = pd.read_sql("select * from BMR71783.MCDONALDS_NUTRITION", pconn)
df
```

Out[6]:

	Category	Item	Serving_Size	Calories	Calories_from_Fat	Total_Fat	Total
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	
4	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210	23.0	
5	Breakfast	Steak & Egg McMuffin	6.5 oz (185 g)	430	210	23.0	

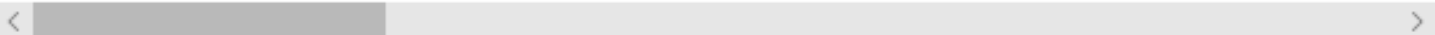
## ❑ Sử dụng Pandas để nhận dữ liệu từ truy vấn

In [7]: `df.head(5)`

Out[7]:

	Category	Item	Serving_Size	Calories	Calories_from_Fat	Total_Fat	Total_Fat____Daily_\\
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	
4	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210	23.0	

5 rows x 24 columns



# Phân tích dữ liệu với Python

```
In [8]: df.tail(5)
```

Out[8]:

	Category	Item	Serving_Size	Calories	Calories_from_Fat	Total_Fat	Total_Fat____Daily_Value_	Saturated_Fat	Satura
255	Smoothies & Shakes	McFlurry with Oreo Cookies (Small)	10.1 oz (285 g)	510	150	17.0	26	9.0	
256	Smoothies & Shakes	McFlurry with Oreo Cookies (Medium)	13.4 oz (381 g)	690	200	23.0	35	12.0	
257	Smoothies & Shakes	McFlurry with Oreo Cookies (Snack)	6.7 oz (190 g)	340	100	11.0	17	6.0	
258	Smoothies & Shakes	McFlurry with Reese's Peanut Butter Cups (Medium)	14.2 oz (403 g)	810	290	32.0	50	15.0	
259	Smoothies & Shakes	McFlurry with Reese's Peanut Butter Cups (Snack)	7.1 oz (202 g)	410	150	16.0	25	8.0	

5 rows x 24 columns



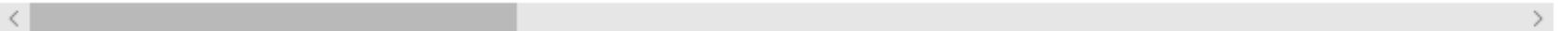
# Phân tích dữ liệu với Python

In [9]: `df.describe(include='all')`

Out[9]:

	Category	Item	Serving_Size	Calories	Calories_from_Fat	Total_Fat	Total_Fat____Daily_Value_	Saturated_Fat	!
count	260	260	260	260.000000	260.000000	260.000000	260.000000	260.000000	
unique	9	260	107	NaN	NaN	NaN	NaN	NaN	
top	Coffee & Tea	Sweet Tea (Child)	16 fl oz cup	NaN	NaN	NaN	NaN	NaN	
freq	95	1	45	NaN	NaN	NaN	NaN	NaN	
mean	NaN	NaN	NaN	368.269231	127.096154	14.165385	21.815385	6.007692	
std	NaN	NaN	NaN	240.269886	127.875914	14.205998	21.885199	5.321873	
min	NaN	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	NaN	NaN	NaN	210.000000	20.000000	2.375000	3.750000	1.000000	
50%	NaN	NaN	NaN	340.000000	100.000000	11.000000	17.000000	5.000000	
75%	NaN	NaN	NaN	500.000000	200.000000	22.250000	35.000000	10.000000	
max	NaN	NaN	NaN	1880.000000	1060.000000	118.000000	182.000000	20.000000	

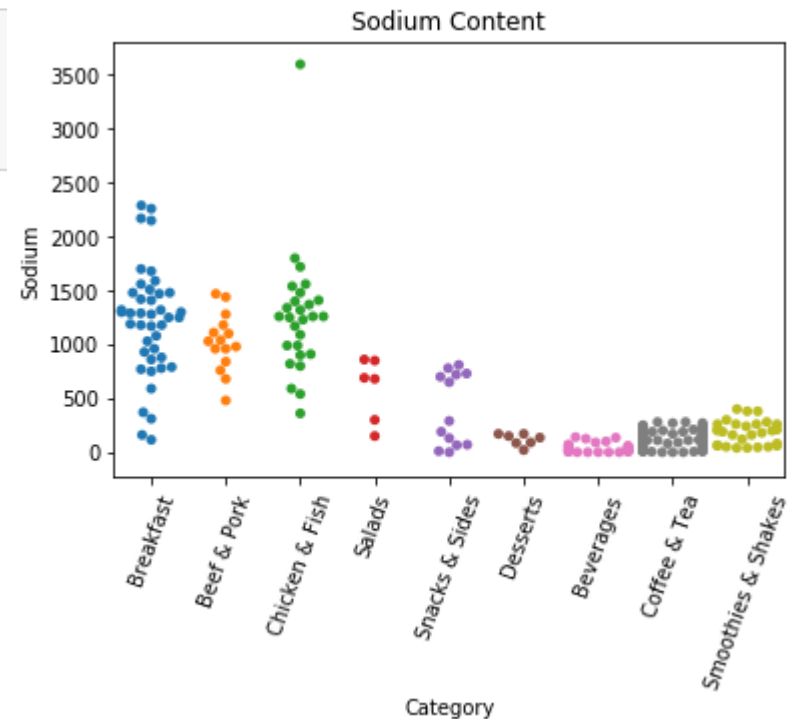
11 rows × 24 columns



## □ Thống kê hàm lượng Natri (Sodium) trong mỗi nhóm thực phẩm

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sb
```

```
plot = sb.swarmplot(x="Category", y="Sodium", data=df)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title("Sodium Content")
plt.show()
```



❑ Cho biết hàm lượng Natri cao nhất trong loại thực phẩm nào?

```
In [12]: df["Sodium"].describe()
```

```
Out[12]: count      260.000000  
mean        495.750000  
std         577.026323  
min           0.000000  
25%        107.500000  
50%        190.000000  
75%        865.000000  
max       3600.000000  
Name: Sodium, dtype: float64
```

```
In [13]: df["Sodium"].idxmax()
```

```
Out[13]: 82
```

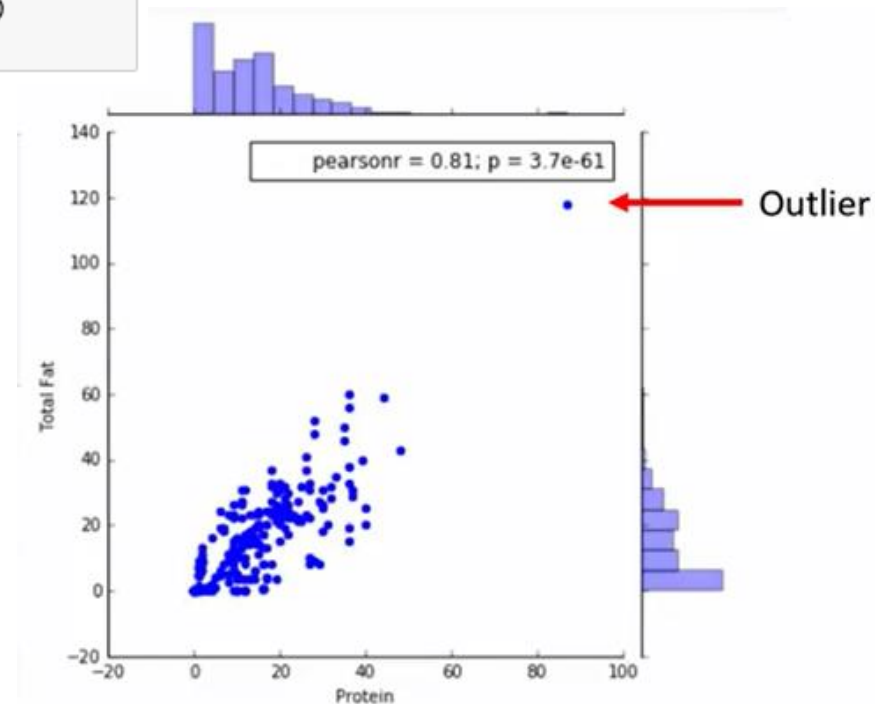
```
In [14]: df.at[82, 'Item']
```

```
Out[14]: 'Chicken McNuggets (40 piece)'
```

## □ Thể hiện mối liên quan giữa hàm lượng Protein và Total Fat

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sb

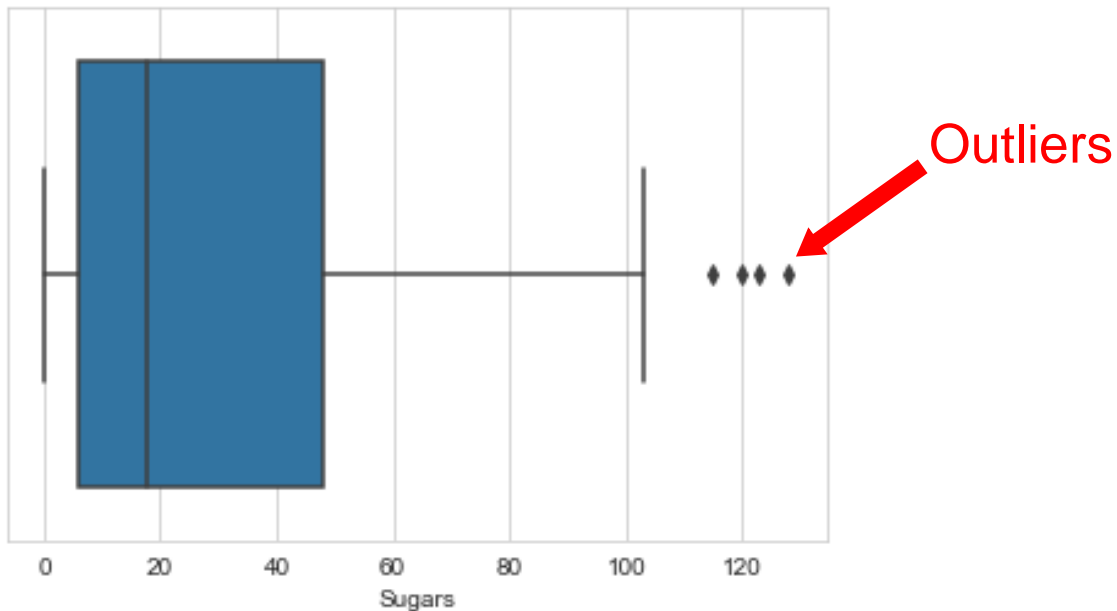
plot = sb.jointplot(x="Protein", y="Total_Fat", data=df)
plot.show()
```



# Phân tích dữ liệu với Python

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sb

plot = sb.set_style("whitegrid")
ax = sb.boxplot(x=df["Sugars"])
plt.show()
```



1. `ibm_db` API
2. Kết nối database sử dụng `ibm_db` API
3. Tạo bảng, tải và truy vấn dữ liệu sử dụng `ibm_db` API
4. Sử dụng Magic SQL (%) với `ibm_db` API
5. Phân tích dữ liệu với Python
6. Làm việc với SQLite và PostgreSQL

# Làm việc với SQLite và PostgreSQL

---

## ❑ Làm việc với SQLite

```
import sqlite3
```

- Kết nối database: (nếu database chưa có sẽ tạo database mới)

```
conn = sqlite3.connect('example.db')
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với SQLite

- Thực thi câu lệnh truy vấn

```
c = conn.cursor()

# Create table
c.execute('''CREATE TABLE stocks
            (date text, trans text, symbol text, qty real, price real)''')
```

```
<sqlite3.Cursor at 0x17b2322a650>
```

```
# Insert a row of data
c.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

# Save (commit) the changes
conn.commit()

conn.close()
```



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với SQLite

- Thực thi câu lệnh truy vấn với tham số

```
# Do this instead
t = ('RHAT',)
c.execute('SELECT * FROM stocks WHERE symbol=?', t)
print(c.fetchone())

('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với SQLite

- Thực thi câu lệnh truy vấn với tham số

```
# Larger example that inserts many records at a time
purchases = [('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
              ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
              ('2006-04-06', 'SELL', 'IBM', 500, 53.00),
              ]
c.executemany('INSERT INTO stocks VALUES (?, ?, ?, ?, ?)', purchases)

<sqlite3.Cursor at 0x278f1226880>
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với SQLite

- Thực thi câu lệnh truy vấn với tham số, đặt tên cho tham số, sử dụng **:<tên\_tham\_số>** thay cho ?

```
purchases = [{ 'ngay': '2006-03-28', 'hang': 'IBM', 'loai': 'BUY', 'soluong': 1000, 'gia': 45.00 },  
               { 'ngay': '2006-04-05', 'hang': 'MSFT', 'loai': 'BUY', 'soluong': 1000, 'gia': 72.00 },  
               ( 'ngay': '2006-04-06', 'hang': 'IBM', 'loai': 'SELL', 'soluong': 500, 'gia': 53.00 )  
             ]  
sql = 'INSERT INTO stocks VALUES (:ngay, :loai, :hang, :soluong, :gia)'  
cursor.executemany(sql, purchases)
```

```
<sqlite3.Cursor at 0x2a77c1558f0>
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với SQLite

- Duyệt dữ liệu kết quả

```
c.execute('SELECT * FROM stocks ORDER BY price')  
print(c.fetchone())  
print()  
print(c.fetchall())
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
```

```
[('2006-03-28', 'BUY', 'IBM', 1000.0, 45.0), ('2006-04-06', 'SELL', 'IBM', 500.0, 53.0), ('2006-04-05', 'BUY', 'MSFT', 1000.0, 72.0)]
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với SQLite

- Duyệt từng dòng dữ liệu kết quả

```
c.execute('SELECT * FROM stocks ORDER BY price')  
for row in c:  
    print(row)
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)  
( '2006-03-28', 'BUY', 'IBM', 1000.0, 45.0)  
( '2006-04-06', 'SELL', 'IBM', 500.0, 53.0)  
( '2006-04-05', 'BUY', 'MSFT', 1000.0, 72.0)
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Cài đặt PostgreSQL

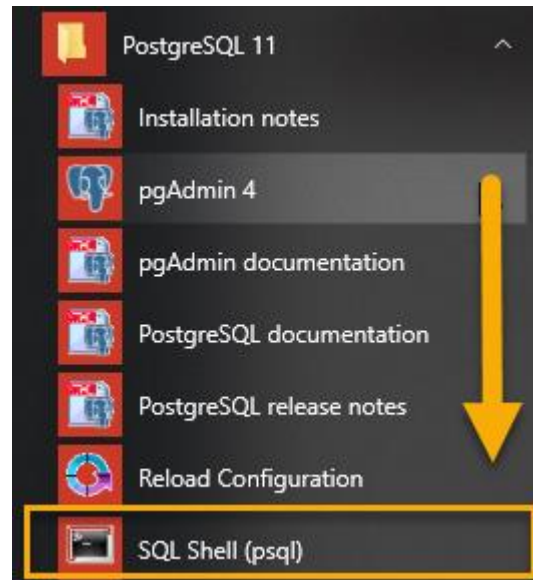
<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
12.3	N/A	N/A	<a href="#">Download</a>	<a href="#">Download</a>	N/A
11.8	N/A	N/A	<a href="#">Download</a>	<a href="#">Download</a>	N/A
10.13	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
9.6.18	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
9.5.22	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
9.4.26 (Not Supported)	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>
9.3.25 (Not Supported)	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">Download</a>

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

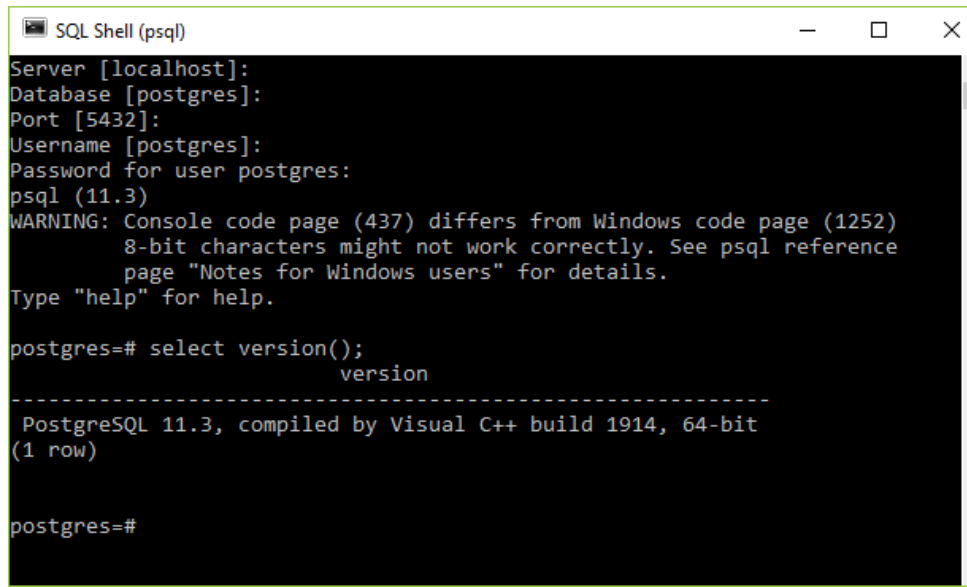
- Cài đặt PostgreSQL theo các thiết lập mặc định
- Kiểm tra kết quả cài đặt: khởi động SQL Shell (psql)



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Kiểm tra kết quả cài đặt:
  - khởi động SQL Shell (psql)
  - Nhấn Enter để nhập các giá trị mặc định và nhập mật khẩu để đăng nhập PostgreSQL
  - Gõ lệnh **select Version();** để xem kết quả



```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (11.3)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# select version();
              version
-----
PostgreSQL 11.3, compiled by Visual C++ build 1914, 64-bit
(1 row)

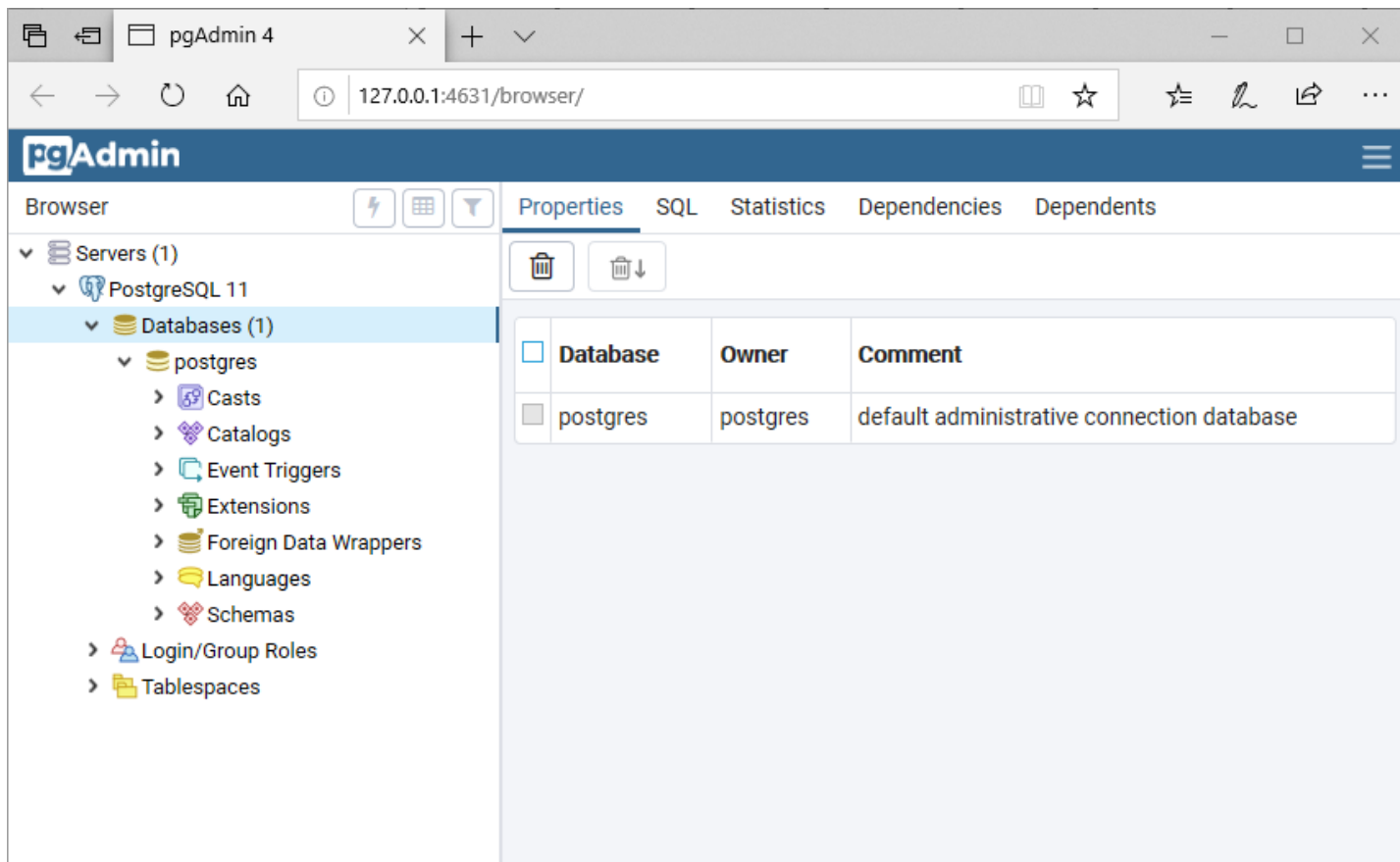
postgres=#
```



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

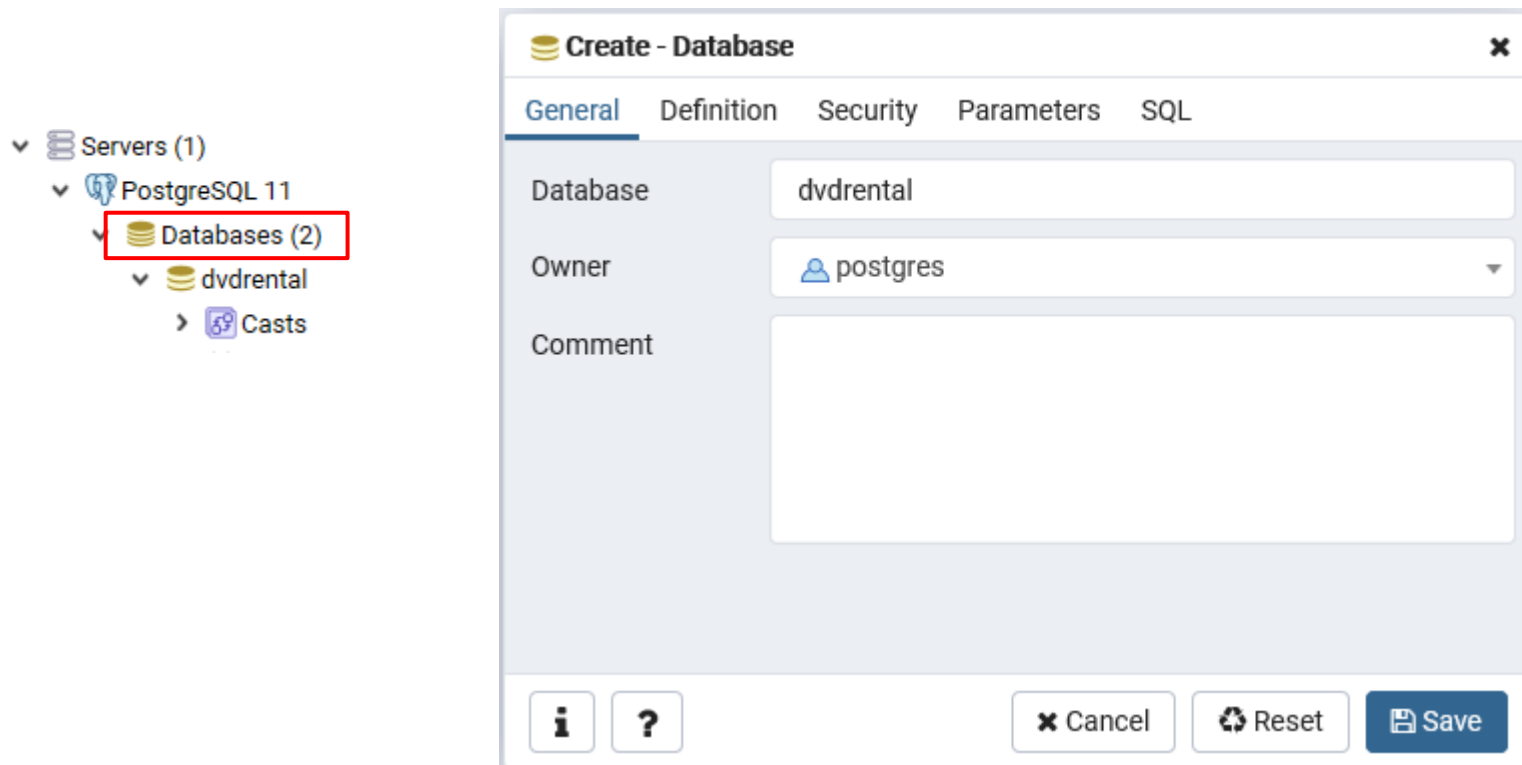
- Quản trị PostgreSQL với pgAdmin



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

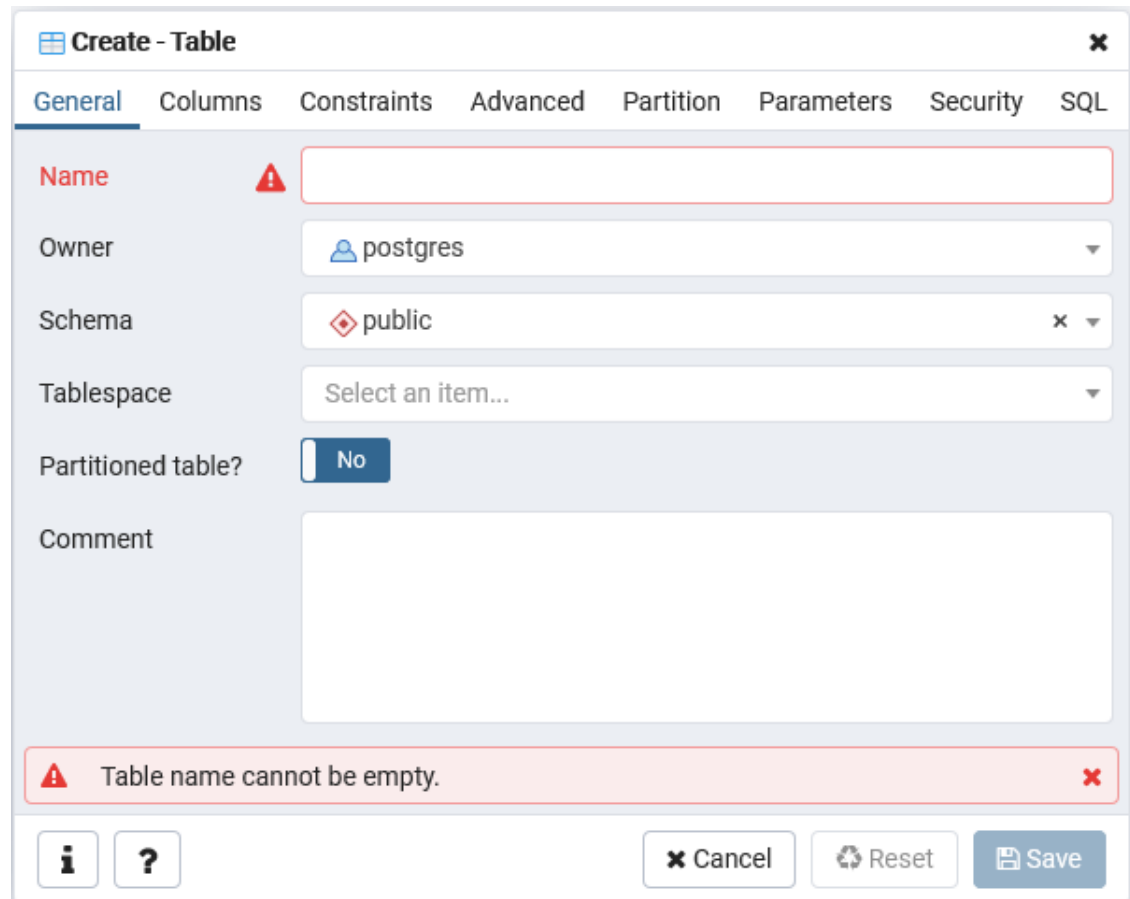
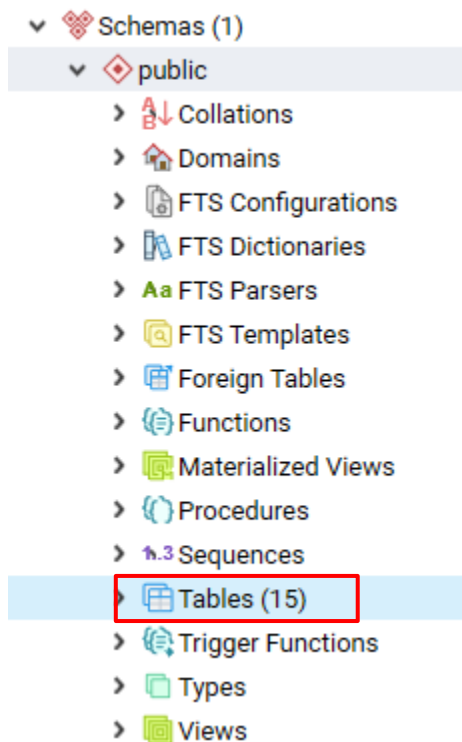
- Tạo Database: nhấn chuột phải lên Databases và chọn Create -> Database...



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

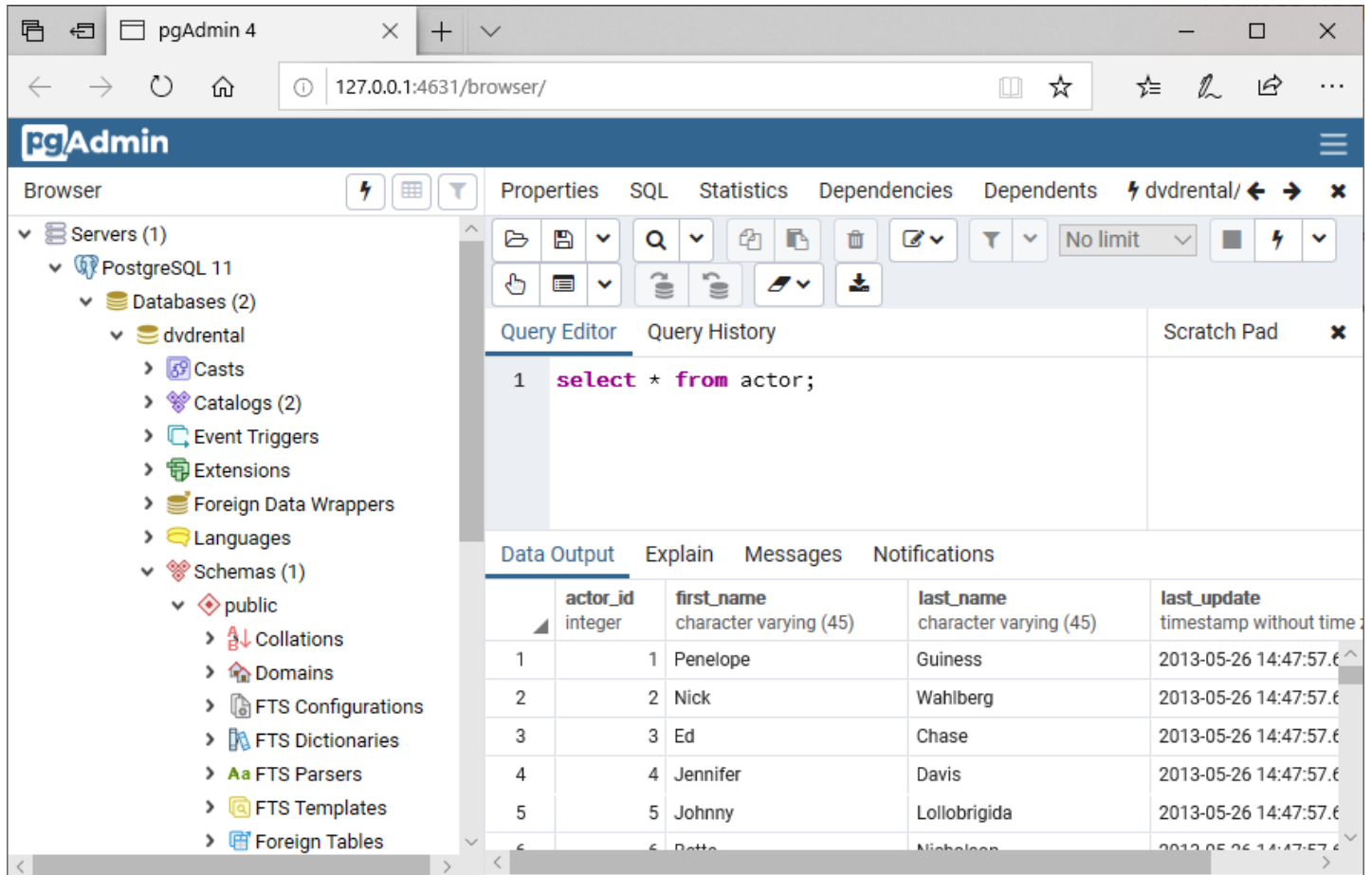
- Tạo bảng: nhấn chuột phải lên Tables và chọn Create -> Table...



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Làm việc với câu lệnh SQL: chọn menu Tools -> Query Tool



The screenshot shows the pgAdmin 4 web interface. The left sidebar displays the database structure: Servers (1) > PostgreSQL 11 > Databases (2) > dvdrental. The main panel is divided into several tabs: Properties, SQL, Statistics, Dependencies, Dependents, and a tab for the 'dvdrental' database. The 'Query Editor' tab is active, showing a SQL query: `1 select * from actor;`. Below the query editor, the 'Data Output' tab is selected, displaying a table with the results of the query. The table has four columns: `actor_id` (integer), `first_name` (character varying (45)), `last_name` (character varying (45)), and `last_update` (timestamp without time zone). The table contains six rows of data.

	actor_id integer	first_name character varying (45)	last_name character varying (45)	last_update timestamp without time zone
1	1	Penelope	Guinness	2013-05-26 14:47:57.6
2	2	Nick	Wahlberg	2013-05-26 14:47:57.6
3	3	Ed	Chase	2013-05-26 14:47:57.6
4	4	Jennifer	Davis	2013-05-26 14:47:57.6
5	5	Johnny	Lollobrigida	2013-05-26 14:47:57.6
6	6	Bette	Nicholson	2013-05-26 14:47:57.6

# Làm việc với SQLite và PostgreSQL

---

## ❑ Làm việc với PostgreSQL

- Cài đặt module psycopg2

`pip install psycopg2`

- Kết nối Python với PostgreSQL

- Username: tên đăng nhập vào Database PostgreSQL
- Password: mật khẩu đăng nhập vào PostgreSQL
- Hostname: tên máy hoặc địa chỉ IP cài PostgreSQL
- DatabaseName: tên database
- Port: cổng kết nối đến PostgreSQL

## ❑ Làm việc với PostgreSQL

### ● Kết nối Python với PostgreSQL

```
import psycopg2
try:
    connection = psycopg2.connect(user = "postgres",
                                   password = "123456",
                                   host = "127.0.0.1",
                                   port = "5432",
                                   database = "dvdrental")

    cursor = connection.cursor()
    # Print PostgreSQL Connection properties
    print ( connection.get_dsn_parameters(),"\n")
    # Print PostgreSQL version
    cursor.execute("SELECT version();")
    record = cursor.fetchone()
    print("You are connected to - ", record,"\n")
    #closing database connection.
    if(connection):
        cursor.close()
        connection.close()
        print("PostgreSQL connection is closed")
except (Exception, psycopg2.Error) as error :
    print ("Error while connecting to PostgreSQL", error)
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Thực thi câu lệnh truy vấn

```
import psycopg2

connection = psycopg2.connect(user = "postgres",
                              password = "123456",
                              host = "127.0.0.1",
                              port = "5432",
                              database = "dvdrental")

cursor = connection.cursor()
# Create table
cursor.execute('''CREATE TABLE stocks
                 (date text, trans text, symbol text, qty real, price real)''')

# Insert a row of data
cursor.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

# Save (commit) the changes
connection.commit()

# connection.close()
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Thực thi câu lệnh truy vấn với tham số

```
# Do this instead  
t = ('RHAT',)  
cursor.execute('SELECT * FROM stocks WHERE symbol=%s', t)  
print(cursor.fetchone())
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
```



# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Thực thi câu lệnh truy vấn với tham số

```
# Large example that insert many records at a time
purchases = [( '2006-03-28', 'BUY', 'IBM', 1000, 45.00),
              ( '2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
              ( '2006-04-06', 'SELL', 'IBM', 500, 53.00)
            ]
sql = 'INSERT INTO stocks VALUES (%s, %s, %s, %s, %s)'
cursor.executemany(sql, purchases)
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Duyệt dữ liệu kết quả

```
cursor.execute('SELECT * FROM stocks ORDER BY price')
print(cursor.fetchone())
print()
print(cursor.fetchall())
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
```

```
[('2006-03-28', 'BUY', 'IBM', 1000.0, 45.0), ('2006-04-06', 'SELL', 'IBM', 500.0, 53.0), ('2006-04-05', 'BUY', 'MSFT', 1000.0, 72.0)]
```

# Làm việc với SQLite và PostgreSQL

## ❑ Làm việc với PostgreSQL

- Duyệt từng dòng dữ liệu kết quả

```
cursor.execute('SELECT * FROM stocks ORDER BY price')
for row in cursor:
    print(row)
```

```
('2006-01-05', 'BUY', 'RHAT', 100.0, 35.14)
('2006-03-28', 'BUY', 'IBM', 1000.0, 45.0)
('2006-04-06', 'SELL', 'IBM', 500.0, 53.0)
('2006-04-05', 'BUY', 'MSFT', 1000.0, 72.0)
```

