Hands-on Lab: Accessing Databases using Python Script

Estimated Effort: 30 mins

Using databases is an important and useful method of sharing information. To preserve repeated storage of the same files containing the required data, it is a good practice to save the said data on a database on a server and access the required subset of information using database management systems.

In this lab, you'll learn how to create a database, load data from a CSV file as a table, and then run queries on the data using Python.

Objectives

In this lab you'll learn how to:

1. Create a database using Python
2. Load the data from a CSV file as a table to the database
3. Run basic "queries" on the database to access the information

Scenario

Consider a dataset of employee records that is available with an HR team in a CSV file. As a Data Engineer, you are required to create the database called `STAFF` and load the contents of the CSV file as a table called `INSTRUCTORS`. The headers of the available data are :

Header	Description
ID	Employee ID
FNAME	First Name
LNAME	Last Name
CITY	City of residence
CCODE	Country code (2 letters)

Setting Up

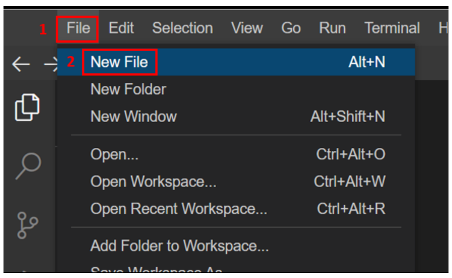
Usually, the database for storing data would be created on a server to which the other team members would have access. For the purpose of this lab, we are going to create the database on a dummy server using SQLite3 library.

Note: SQLite3 is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. SQLite3 comes bundled with Python and does not require installation.

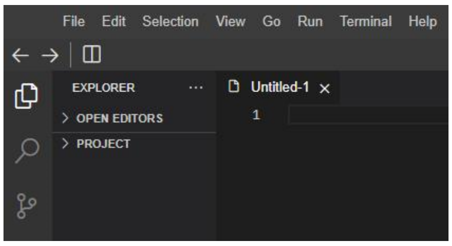
Initial steps

For this lab, you will need a Python file in the `project` folder. You can name it `db_code.py`. The process to create the file is shown in the images below.

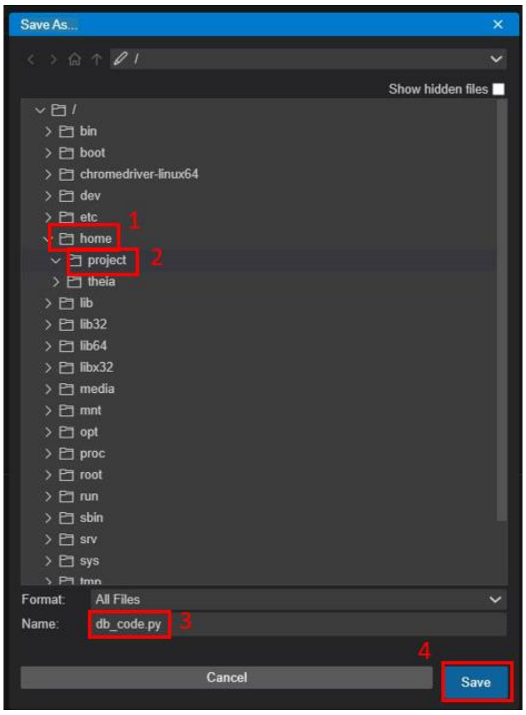
1. In the `File` menu, click the option `New File`.



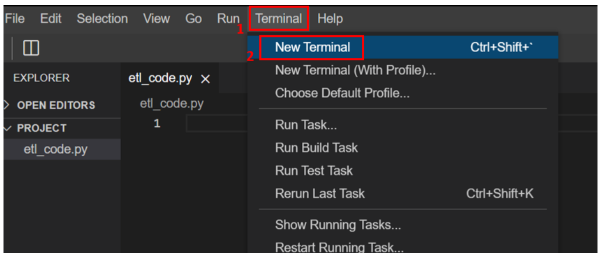
This should open an `Untitled` file in the editor tab.



2. Use `Ctlr+S` to save the file. The `Save As` interface would pop up. Navigate to the path `/home/project/` as shown in the image below and name the file `db_code.py`. Click `Save`.



You also need the CSV data to be available in the same location `/home/project/`. For this, open a new terminal from the `Terminal` tab in the menu as shown below.


File Edit Selection View Go Run Terminal Help
1
2
New Terminal Ctrl+Shift+`
New Terminal (With Profile)...
Choose Default Profile...
Run Task...
Run Build Task
Run Test Task
Rerun Last Task Ctrl+Shift+K
Show Running Tasks...
Restart Running Task...
EXPLORER
OPEN EDITORS
PROJECT
etl_code.py

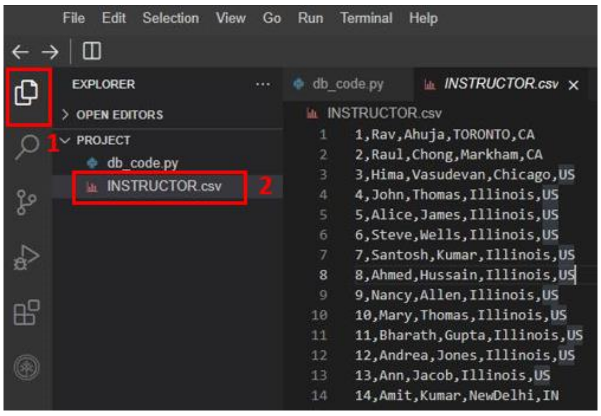
Run the following command in the terminal. Make sure the current directory in the terminal window is `/home/project/`.

```
1. 1

1. wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/INSTRUCTOR.csv
```

Copied! Executed!

The file `INSTRUCTOR.csv` should now be available in the location `/home/project/`. You can check its contents by clicking it from the `Explorer` menu.



```
1,Rav,Ahuja,TORONTO,CA
2,Raul,Chong,Markham,CA
3,Hima,Vasudevan,Chicago,US
4,John,Thomas,Illinois,US
5,Alice,James,Illinois,US
6,Steve,Wells,Illinois,US
7,Santosh,Kumar,Illinois,US
8,Ahmed,Hussain,Illinois,US
9,Nancy,Allen,Illinois,US
10,Mary,Thomas,Illinois,US
11,Bharath,Gupta,Illinois,US
12,Andrea,Jones,Illinois,US
13,Ann,Jacob,Illinois,US
14,Amit,Kumar,NewDelhi,IN
```

Further, to read the CSV and interact with the database, you'll need the `pandas` library. This library will first have to be installed in the Cloud IDE framework. For this, run the below mentioned statement in a terminal window.

```
python3.11 -m pip install pandas
```

Copied! Executed!

Python Scripting: Database initiation

Let us first create a database using Python.

Open `db_code.py` and import the `sqlite3` library using the below mentioned command.

```
import sqlite3
```

Copied!

Import the `pandas` library in `db_code.py` using the following code.

```
import pandas as pd
```

Copied!

Now, you can use SQLite3 to create and connect your process to a new database `STAFF` using the following statements.

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

Copied!

Remember to save the file using `Ctrl+S`.

Python Scripting: Create and Load the table

To create a table in the database, you first need to have the attributes of the required table. Attributes are columns of the table. Along with their names, the knowledge of their data types are also required. The attributes for the required tables in this lab were shared in the Lab Scenario.

Add the following statements to `db_code.py` to feed the required table name and attribute details for the table.

```
table_name = 'INSTRUCTOR'
attribute_list = ['ID', 'FNAME', 'LNAME', 'CITY', 'CCODE']
```

Copied!

Note: This information can be updated for the case of any other kind of table.

Save the file using `Ctrl+S`.

Reading the CSV file

Now, to read the CSV using Pandas, you use the `read_csv()` function. Since this CSV does not contain headers, you can use the keys of the `attribute_dict` dictionary as a list to assign headers to the data. For this, add the commands below to `db_code.py`.

```
file_path = '/home/project/INSTRUCTOR.csv'
df = pd.read_csv(file_path, names = attribute_list)
```

Copied!

Loading the data to a table

The `pandas` library provides easy loading of its dataframes directly to the database. For this, you may use the `to_sql()` method of the `dataframe` object.

However, while you load the data for creating the table, you need to be careful if a table with the same name already exists in the database. If so, and it isn't required anymore, the tables should be replaced with the one you are loading here. You may also need to append some information to an existing table. For this purpose, `to_sql()` function uses the argument `if_exists`. The possible usage of `if_exists` is tabulated below.

Argument usage	Description
`if_exists = 'fail'`	Default. The command doesn't work if a table with the same name exists in the database.
`if_exists = 'replace'`	The command replaces the existing table in the database with the same name.
`if_exists = 'append'`	The command appends the new data to the existing table with the same name.

As you need to create a fresh table upon execution, add the following commands to the code. The print command is optional, but helps identify the completion of the steps of code until this point.

```
df.to_sql(table_name, conn, if_exists = 'replace', index =False)
print('Table is ready')
```

Copied!

Save the file using `Ctrl+S`.

Python Scripting: Running basic queries on data

Now that the data is uploaded to the table in the database, anyone with access to the database can retrieve this data by executing SQL queries.

Some basic SQL queries to test this data are `SELECT` queries for viewing data, and `COUNT` query to count the number of entries.

SQL queries can be executed on the data using the `read_sql` function in `pandas`.

Now, run the following tasks for data retrieval on the created database.

1. Viewing all the data in the table.

Add the following lines of code to `db_code.py`

```
query_statement = f"SELECT * FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)
```

Copied!

2. Viewing only FNAME column of data.

Add the following lines of code to `db_code.py`

```
query_statement = f"SELECT FNAME FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)
```

Copied!

3. Viewing the total number of entries in the table.

Add the following lines of code to `db_code.py`

```
query_statement = f"SELECT COUNT(*) FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)
```

Copied!

Now try appending some data to the table. Consider the following.
a. Assume the `ID` is `100`.
b. Assume the first name, `FNAME`, is `John`.
c. Assume the last name as `LNAME`, `Doe`.
d. Assume the city of residence, `CITY` is `Paris`.
e. Assume the country code, `CCODE` is `FR`.

Use the following statements to create the dataframe of the new data.

```python
data_dict = {'ID' : [100],
            'FNAME' : ['John'],
            'LNAME' : ['Doe'],
            'CITY' : ['Paris'],
            'CCODE' : ['FR']}
data_append = pd.DataFrame(data_dict)
```

Copied!

Now use the following statement to append the data to the `INSTRUCTOR` table.

```python
data_append.to_sql(table_name, conn, if_exists = 'append', index =False)
print('Data appended successfully')
```

Copied!

Now, repeat the `COUNT` query. You will observe an increase by 1 in the output of the first `COUNT` query and the second one.

Before proceeding with the final execution, you need to add the command to close the connection to the database after all the queries are executed.

Add the following line at the end of `db_code.py` to close the connection to the database.

```python
conn.close()
```

Copied!

Save the file using `Ctrl+S`.

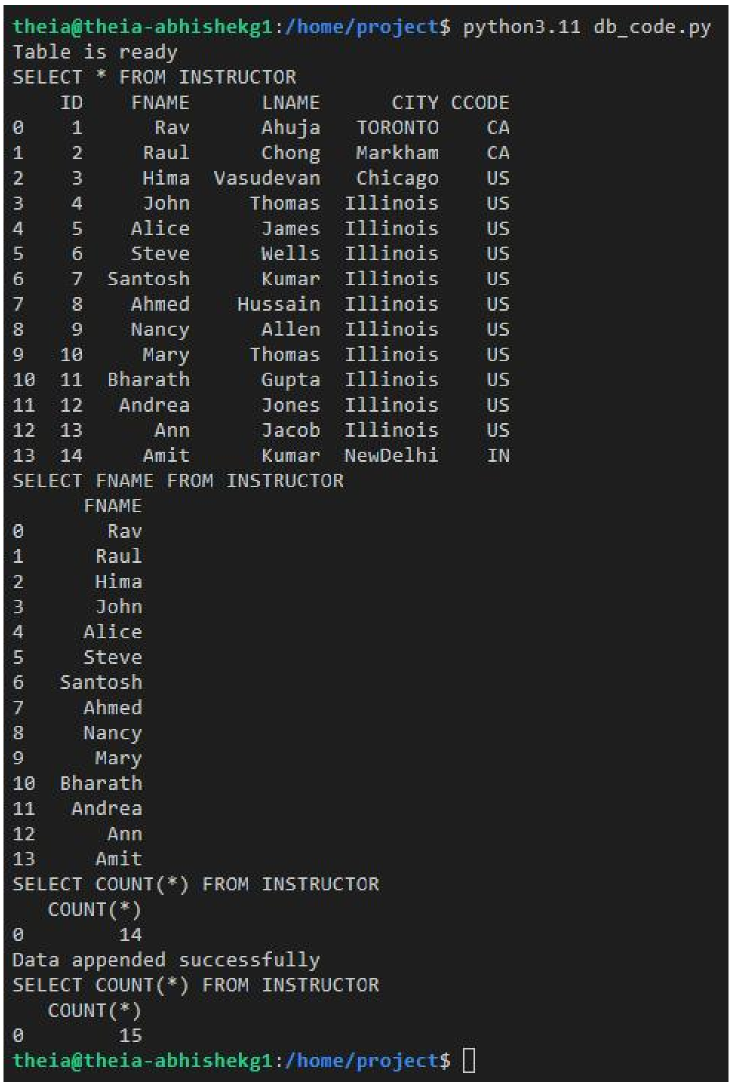
Code Execution

Execute `db_code.py` from the terminal window using the following command.

```
python3.11 db_code.py
```

Copied! Executed!

The output expected is shown in the image below.



```
theia@theia-abhishekg1:/home/project$ python3.11 db_code.py
Table is ready
SELECT * FROM INSTRUCTOR
    ID    FNAME      LNAME      CITY CCODE
0    1      Rav      Ahuja   TORONTO    CA
1    2     Raul      Chong   Markham    CA
2    3     Hima  Vasudevan   Chicago    US
3    4     John     Thomas  Illinois    US
4    5    Alice      James  Illinois    US
5    6    Steve      Wells  Illinois    US
6    7  Santosh      Kumar  Illinois    US
7    8    Ahmed    Hussain  Illinois    US
8    9    Nancy      Allen  Illinois    US
9   10     Mary     Thomas  Illinois    US
10  11  Bharath      Gupta  Illinois    US
11  12   Andrea      Jones  Illinois    US
12  13      Ann      Jacob  Illinois    US
13  14     Amit      Kumar  NewDelhi    IN
SELECT FNAME FROM INSTRUCTOR
      FNAME
0       Rav
1      Raul
2      Hima
3      John
4     Alice
5     Steve
6   Santosh
7     Ahmed
8     Nancy
9      Mary
10  Bharath
11   Andrea
12      Ann
13     Amit
SELECT COUNT(*) FROM INSTRUCTOR
   COUNT(*)
0        14
Data appended successfully
SELECT COUNT(*) FROM INSTRUCTOR
   COUNT(*)
0        15
theia@theia-abhishekg1:/home/project$ 
```

Lab Solution

In case you are not able to get the required output from the code or are facing some errors, the final file for `db_code.py`is shared below. Please note that this is for your help, and we encourage you to first try to resolve the errors on your own.

Also, you may keep a copy of `db_code.py` saved in your local machine since it will be useful in the projects of the course as well.

▼ db_code.py

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
```

```python
import sqlite3
import pandas as pd

# Connect to the SQLite3 service
conn = sqlite3.connect('STAFF.db')

# Define table parameters
table_name = 'INSTRUCTOR'
attribute_list = ['ID', 'FNAME', 'LNAME', 'CITY', 'CCODE']

# Read the CSV data
file_path = '/home/project/INSTRUCTOR.csv'
df = pd.read_csv(file_path, names = attribute_list)

# Load the CSV to the database
df.to_sql(table_name, conn, if_exists = 'replace', index = False)
print('Table is ready')

# Query 1: Display all rows of the table
query_statement = f"SELECT * FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)

# Query 2: Display only the FNAME column for the full table.
query_statement = f"SELECT FNAME FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)

# Query 3: Display the count of the total number of rows.
query_statement = f"SELECT COUNT(*) FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)

# Define data to be appended
data_dict = {'ID' : [100],
            'FNAME' : ['John'],
            'LNAME' : ['Doe'],
            'CITY' : ['Paris'],
            'CCODE' : ['FR']}
data_append = pd.DataFrame(data_dict)

# Append data to the table
data_append.to_sql(table_name, conn, if_exists = 'append', index = False)
print('Data appended successfully')

# Query 4: Display the count of the total number of rows.
query_statement = f"SELECT COUNT(*) FROM {table_name}"
query_output = pd.read_sql(query_statement, conn)
print(query_statement)
print(query_output)

# Close the connection
conn.close()
```

Copied!

Practice Problems

Try the following practice problems to test your understanding of the lab. Please note that the solutions for the following are not shared, and the learners are encouraged to use the discussion forums in case they need help.

1. In the same database `STAFF`, create another table called `Departments`. The attributes of the table are as shown below.

Header	Description
DEPT_ID	Department ID
DEP_NAME	Department Name
MANAGER_ID	Manager ID
LOC_ID	Location ID

2. Populate the `Departments` table with the data available in the CSV file which can be downloaded from the link below using `wget`.

```
1. 1
```

```
1. https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-PY0221EN-Coursera/labs/v2/Departments.csv
```

Copied! Executed!

3. Append the `Departments` table with the following information.

Attribute	Value
DEPT_ID	9
DEP_NAME	Quality Assurance
MANAGER_ID	30010
LOC_ID	L0010

4. Run the following queries on the `Departments` Table:
   a. View all entries
   b. View only the department names
   c. Count the total entries

Conclusion

Congratulations on completing this lab.

In this lab, you have learned how to:

- Create a databse using SQLite3 in Python.
- Create and load a table using data from a CSV file using Pandas.
- Run basic queries on the tables in the database.

Author(s)

Abhishek Gagneja