Data Wrangling Operations in `Python`

Steven Wun

2023-08-01

Introduction

In order to run `Python` in `R`, we need to load the `reticulate` package first. Further, unless marked as `R code`, all the codes below are Python codes.

```
# R code
# load the required library to run Python in R
library(reticulate)
```

Using Databases With `Python`

We import the module `sqlite3` and use the function `connect()` to create an object, `conn`, to connect to the `SQLite` driver to manipulate the database `University.db`. If the database `University.db` exists in your working directory, the following code chunk will remove it.

```
# this makes sure you can run this notebook multiple times without errors
import os
try:
  os.remove('university.db')
except OSError:
  pass
```

```
# connect to the SQLite driver
import sqlite3
conn = sqlite3.connect('university.db')
```

Creating Tables Using `Python`

Now we are going to create some tables to the database `University.db`. Like before, we will create the tables using the data saved in the CSV files. We first load the CSV files into `DataFrame` in Python:

```
# creating tables using python
import pandas as pd
student = pd.read_csv("Course Files/Block 3/student.csv")
course = pd.read_csv("Course Files/Block 3/course.csv")
grade = pd.read_csv("Course Files/Block 3/grade.csv")
```

We then write record stored in DataFrames student, grade and course as tables to the database University.db using the DataFrame method to_sql().

```
# write tables to the database
# 'index = False` to ensure the DataFrame row index is not written into the SQL tables
student.to_sql('Student', con = conn, index = False)
```

```
course.to_sql('Course', con = conn, index = False)
```

```
grade.to_sql('Grade', con = conn, index = False)
```

Manipulate Databases Using Python

We can manipulate databases in Python by the execute() and fetchall() methods from the sqlite3 module which performs SQL commands. This allows us to leverage the SQL commands we have learned to manipulate the databases in Python. We first need to create a cursor object c:

```
# create a cursor object 'c'
c = conn.cursor()
```

After that we can execute the SQL commands we learned before using the function execute() and fetchall(). For example, if we want to get all the tables in the database, we can run:

```
# get all the tables in a database
c.execute('''
SELECT name
FROM sqlite_master
WHERE type='table'
''')
```

```
## <sqlite3.Cursor object at 0x0000000050A82260>
```

The result is not returned until we run fetchall():

```
# return the result
c.fetchall()
```

```
## [('Student',), ('Course',), ('Grade',)]
```

We can see there are three tables in the database. If we want to browse the table Student we can run (here we display the results as pandas DataFrame):

```
# browse the contents of a table
q = c.execute("SELECT * FROM Student").fetchall()
pd.DataFrame(q)
```

```
##            0              1  2
## 0  201921323      Ava Smith  2
## 1  201832220    Ben Johnson  3
## 2  202003219  Charlie Jones  1
## 3  202045234      Dan Norris  1
## 4  201985603      Emily Wood  1
## 5  201933222  Freddie Harris  2
## 6  201875940    Grace Clarke  2
```

Note here we combine the use of `execute()` and `fetchall()` in one line.

Add a New Table

We can add a new table by running the SQL command through `execute()`:

```
# add a new table
c.execute('''
CREATE TABLE Teacher (staff_id TEXT PRIMARY KEY, name TEXT)
''')
```

```
## <sqlite3.Cursor object at 0x0000000050A82260>
```

```
conn.commit()     # save (commit) the changes
```

When we list the tables, we can see four tables.

```
# list the tables
c.execute('''
SELECT name
FROM sqlite_master
WHERE type='table'
''').fetchall()
```

```
## [('Student',), ('Course',), ('Grade',), ('Teacher',)]
```

Delete a Table

We can delete a table by running the `SQL` command through `execute()`:

```
# delete a table
c.execute("DROP TABLE Teacher")
```

```
## <sqlite3.Cursor object at 0x0000000050A82260>
```

```
conn.commit()
```

When we list the tables, we can see three tables.

```
# list the tables
c.execute('''
SELECT name
FROM sqlite_master
WHERE type='table'
''').fetchall()
```

```
## [('Student',), ('Course',), ('Grade',)]
```

Insert Tuples/Rows

Insert the year 1 student Harper Taylor with student ID 202029744 to Student:

```
# insert rows
c.execute("INSERT INTO Student VALUES(202029744, 'Harper Taylor', 1)")
```

```
## <sqlite3.Cursor object at 0x0000000050A82260>
```

```
conn.commit()
```

When we browse the table, we can see the new row is added.

```
# browse the table
q = c.execute("SELECT * FROM Student").fetchall()
pd.DataFrame(q)
```

```
##            0               1  2
## 0  201921323       Ava Smith  2
## 1  201832220     Ben Johnson  3
## 2  202003219   Charlie Jones  1
## 3  202045234      Dan Norris  1
## 4  201985603      Emily Wood  1
## 5  201933222  Freddie Harris  2
## 6  201875940    Grace Clarke  2
## 7  202029744   Harper Taylor  1
```