rec09

April 8, 2021

1 CS 1656 – Introduction to Data Science

1.1 Instructor: Xiaowei Jia

1.2 Teaching Assistant: Evangelos Karageorgos

1.3 Additional Credits: Xiaoting Li, Tahereh Arabghalizi, Agha Zuha, Anatoli Shein

1.4 Advanced SQL

In this recitation we will learn how to create SQLite Databases, create tables, populate tables, and execute SQL queries.

Start off by importing slite3, which comes installed with Anaconda's package list.

```
In [1]: import  sqlite3 as lite
```

1.4.1 Introduction to SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files.

1.4.2 Creating and Connecting to SQLite Database

To connect to a database, use the connect() method which returns a connection object. If a database with that name does not exist, connect() method creates a database.

```
In [2]: con = lite.connect('cs1656wed.sqlite')
```

1.4.3 Create/Drop Tables & Insert Data

From the connection, we get the cursor object. The cursor is used to traverse the records from the result set. By using the with keyword, the Python interpreter automatically releases the resources by closing the connection, provides error handling and **commits** the changes. Otherwise, each update to the database has to be committed manually. You can think of commit as saving the changes.

We call the execute() method of the cursor to execute the SQL statements.Let's start by creating a Rankings table in the database.

```
In [3]: with con:
            cur = con.cursor()
            cur.execute('DROP TABLE IF EXISTS Courses')
            cur.execute("CREATE TABLE Courses(cid INT, number INT, professor TEXT, major TEXT, y

            cur.execute('DROP TABLE IF EXISTS Majors')
            cur.execute("CREATE TABLE Majors(sid INT, major TEXT)")

            cur.execute('DROP TABLE IF EXISTS Grades')
            cur.execute("CREATE TABLE Grades(sid INT, cid INT, credits INT, grade INT)")

            cur.execute('DROP TABLE IF EXISTS Students')
            cur.execute("CREATE TABLE Students(sid INT, firstName TEXT, lastName TEXT, yearStart
```

Now data can be inserted in the table using two ways. You could either insert each row one by one as shown below,

```
In [4]: import pandas
        from sqlalchemy import create_engine

        engine = create_engine("sqlite:///cs1656wed.sqlite")
        df1 = pandas.read_csv('students.csv')
        df1.to_sql('students', engine, if_exists='append', index=False)

        df2 = pandas.read_csv('grades.csv')
        df2.to_sql('grades', engine, if_exists='append', index=False)

        df3 = pandas.read_csv('courses.csv')
        df3.to_sql('courses', engine, if_exists='append', index=False)

        df4 = pandas.read_csv('majors.csv')
        df4.to_sql('majors', engine, if_exists='append', index=False)
```

Or a easier way to insert all rows together is by using executemany() method. But before we try the second method of inserting data, let's first drop the exising table and create it again.

1.4.4 Select, Where, Orderby

To select all data from the table,

```
In [5]: cur.execute("SELECT * FROM students")

Out[5]: <sqlite3.Cursor at 0x137c6868420>
```

To retrieve data after executing a SELECT statement, you can either treat the cursor as an iterator and call the cursor's fetchone() method to retrieve a single matching row, or call fetchall() to get a list of the matching rows.

```
In [6]: for row in cur.execute("select * from students"):
            print(row)
        cur.execute("select * from students")
        df5 = pandas.DataFrame(cur.fetchall(), columns=[column[0] for column in cur.description]
        df5
```

```
(555, 'Solange', 'Knowles', 2012)
(666, 'Peter', 'Weiner', 2012)
(1111, 'Ya', 'Boi', 2020)
(1234, 'Michael', 'Scott', 2009)
(1337, 'Beyonce', 'Knowles', 1985)
(1345, 'Julius', 'Caesar', -60)
(1865, 'Abraham', 'Lincoln', 2012)
(3321, 'Mark', 'Brandanowitz', 1992)
(4224, 'Michelle', 'Young', 1984)
(4444, 'Grace', 'Hopper', 1944)
(5376, 'Poverty', 'Jones', 1969)
(5432, 'Mark', 'Wahlberg', 2000)
(6969, 'Thug', 'Nugget', 1862)
(7928, 'John', 'Cash', 1950)
(9191, 'Margaret', 'Mead', 1919)
(9878, 'First', 'Last', 2014)
(9999, 'Elon', 'Musk', 1932)
(14325, 'John', 'Doe', 1999)
(69420, 'Ray', 'Zimmerman', 2017)
(90210, 'Kappa', 'Pride', 2018)
(314158, 'Mr.', 'Pie', 1000)
(999831, 'John', 'Cena', 2003)
(89990, 'BoJack', 'Horseman', 2012)
```

```
Out[6]:        sid firstName      lastName  yearStarted
        0      555   Solange       Knowles         2012
        1      666     Peter        Weiner         2012
        2     1111        Ya           Boi         2020
        3     1234   Michael         Scott         2009
        4     1337   Beyonce       Knowles         1985
        5     1345    Julius        Caesar          -60
        6     1865   Abraham       Lincoln         2012
        7     3321      Mark  Brandanowitz         1992
        8     4224  Michelle         Young         1984
        9     4444     Grace        Hopper         1944
        10    5376   Poverty         Jones         1969
        11    5432      Mark      Wahlberg         2000
        12    6969      Thug        Nugget         1862
        13    7928      John          Cash         1950
        14    9191  Margaret          Mead         1919
        15    9878     First          Last         2014
```

```
16    9999     Elon        Musk        1932
17   14325     John         Doe        1999
18   69420      Ray   Zimmerman        2017
19   90210    Kappa       Pride        2018
20  314158      Mr.         Pie        1000
21  999831     John        Cena        2003
22   89990   BoJack    Horseman        2012
```

Now, let's find out how many courses were passed per semester (plus year)

```
In [7]: q3a = """
        SELECT year, semester, count(*)
        FROM courses natural join grades
        WHERE grade > 0
        GROUP BY year, semester
        """
        cur.execute(q3a)
        cur.fetchall()
```

```
Out[7]: [(-59, 'Fall', 2),
         (-58, 'Fall', 1),
         (1776, 'Summer', 4),
         (1920, 'Fall', 2),
         (1951, 'Spring', 1),
         (1966, 'Summer', 1),
         (1969, 'Spring', 1),
         (1986, 'Summer', 1),
         (1993, 'Spring', 2),
         (1994, 'Fall', 1),
         (1999, 'Spring', 2),
         (2002, 'Fall', 2),
         (2009, 'Spring', 1),
         (2013, 'Fall', 1),
         (2016, 'Fall', 3),
         (2016, 'Spring', 1),
         (2017, 'Fall', 1),
         (2017, 'Spring', 3)]
```

Let's create a view called 'alldata' that compiles student grades, and show the view using a dataframe.

```
In [8]: cur.execute("DROP VIEW IF EXISTS allgrades")
        q4c = """
        create view allgrades as
        SELECT s.firstName, s.lastName, m.major as ms,
               c.number, c.major as mc, g.grade
        FROM students as s, majors as m, grades as g, courses as c
        WHERE s.sid = m.sid AND g.sid = s.sid AND g.cid = c.cid
        """
```

```
cur.execute(q4c)
pandas.DataFrame(cur.execute("select * from allgrades").fetchall(), columns=[column[0] f
```

```
Out[8]:    firstName      lastName                         ms  number  \
        0      Peter        Weiner            Women's Studies       8
        1      Peter        Weiner            Women's Studies      13
        2      Peter        Weiner            Women's Studies    1567
        3      Peter        Weiner            Women's Studies    1111
        4         Ya           Boi  Underwater Basket Weaving     420
        5         Ya           Boi  Underwater Basket Weaving    1113
        6         Ya           Boi  Underwater Basket Weaving    2011
        7    Michael         Scott             Paper Supplies       1
        8     Julius        Caesar                   Classics    1568
        9     Julius        Caesar                   Classics    1567
        10    Julius        Caesar                       MILT    1568
        11    Julius        Caesar                       MILT    1567
        12   Abraham       Lincoln                    Theatre       4
        13      Mark  Brandanowitz             Urban Planning       2
        14      Mark  Brandanowitz             Urban Planning       2
        15  Michelle         Young                 Film Study    2000
        16     Grace        Hopper      American Sign Language       2
        17     Grace        Hopper      American Sign Language     420
        18     Grace        Hopper      American Sign Language    1656
        19     Grace        Hopper      American Sign Language    1313
        20     Grace        Hopper      American Sign Language   73652
        21     Grace        Hopper               Anthropology       2
        22     Grace        Hopper               Anthropology     420
        23     Grace        Hopper               Anthropology    1656
        24     Grace        Hopper               Anthropology    1313
        25     Grace        Hopper               Anthropology   73652
        26     Grace        Hopper                         CS       2
        27     Grace        Hopper                         CS     420
        28     Grace        Hopper                         CS    1656
        29     Grace        Hopper                         CS    1313
        30     Grace        Hopper                         CS   73652
        31      Mark      Wahlberg             Communications       8
        32      Mark      Wahlberg             Communications     101
        33      Thug        Nugget                 Basketball    1069
        34      John          Cash                      Music     101
        35      John          Cash                      Music    1101
        36  Margaret          Mead               Anthropology    1313
        37  Margaret          Mead               Anthropology   73652
        38     First          Last                      Stuff     245
        39     Kappa         Pride               KappaPriding      13
        40       Mr.           Pie             Home Economics    1999
        41       Mr.           Pie                  Seinfeld     1999
        42      John          Cena             U Can't See Me      15
        43    BoJack      Horseman                    Justice       8
```

```
44    BoJack      Horseman                    Justice      80

                           mc  grade
0   Administration of Justice      3
1                KappaPriding      3
2                        MILT      4
3              Women's Studies      0
4                          CS      3
5   Underwater Basket Weaving      4
6   Underwater Basket Weaving      4
7                       Paper      1
8                        MILT      4
9                        MILT      3
10                       MILT      4
11                       MILT      3
12                    Theatre      3
13             Urban Planning      4
14                 Film Study      4
15                 Film Study      1
16                         CS      4
17                         CS      4
18                         CS      4
19               Anthropology      3
20               Anthropology      4
21                         CS      4
22                         CS      4
23                         CS      4
24               Anthropology      3
25               Anthropology      4
26                         CS      4
27                         CS      4
28                         CS      4
29               Anthropology      3
30               Anthropology      4
31  Administration of Justice      3
32             Communications      2
33                    Seventy      4
34                      Music      4
35  Administration of Justice      0
36               Anthropology      3
37               Anthropology      4
38                     Murder      4
39               KappaPriding      4
40                   Seinfeld      4
41                   Seinfeld      4
42             U Can't See Me      4
43  Administration of Justice      4
44                    Justice      0
```

1.4.5 Tasks

ATTENTION: Use this notebook only to test and debug your queries, NOT as the submission.

T1) Show how many courses were passed (grade>0) per student per semester (plus year). Show student id, year, semester and the count. Sort the results by student id, year and semester.

```
In [9]: cur.execute("""

        """)
        cur.fetchall()

Out[9]: []
```

T2) Same as T1, but show student first and last name instead of student id. Also only show results for students passing at least two courses for every semester. Sort the results by first name, last name, year and semester.

```
In [10]: cur.execute("""

         """)
         cur.fetchall()

Out[10]: []
```

T3) Show the students that have failed at a course in their majors (firstName, lastName, major, courseNumber), utilizing the 'allgrades' view. Sort the results by first name, last name, major and courseNumber.

```
In [11]: cur.execute("""

         """)
         cur.fetchall()

Out[11]: []
```

T4) Same as T3, but without utilizing the view.

```
In [12]: cur.execute("""

         """)
         cur.fetchall()

Out[12]: []
```

T5) Show the professors in decreasing order of 'success' (professor, success). Success will be defined as the number of students passing any of the courses with grade >= 2. Sort by success in descending order and professor in ascending order.

```
In [13]: cur.execute("""

         """)
         cur.fetchall()
```

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

T6) Show a report of the courses (course_number, student_names, avg_grade). Column 'student_names' will contain the first and last names (seperated by a space) of all students taking the course, each name being seperated by ', ' (eg. 'John Doe, Mary Jane'). Only students that passed a specific course (grade>=2) will be considered. Also, the report should only contain courses with avg_grade > 3. Sort the results by avg_grade (descending order), student_names and course_number.

```
In [14]: cur.execute("""

         """)
         cur.fetchall()
```

```
Out[14]: []
```

```
In [15]: cur.close()
         con.close()
```