working_sqlite_db

June 3, 2019

Database 1. Relational Ex: sqlite3, MySQL, MS SQL, Oracle 2. Non-Relational Ex: Mongo DB, Couch DB

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

Connecting to database; and create the db if not present

```
In [2]: conn = sqlite3.connect('population.db')
```

```
In [4]: print(dir(conn))
```

```
['DataError', 'DatabaseError', 'Error', 'IntegrityError', 'InterfaceError', 'InternalError', 'I
```

```
In [5]: cur = conn.cursor()
```

```
In [6]: print(dir(cur))
```

```
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getatt
```

```
In [7]: cur.execute('CREATE TABLE PopByRegion(Region TEXT, Population INTEGER)' )

        cur.execute('''CREATE TABLE PopByRegion
                                        (Region TEXT NOT NULL,
                                         Population INTEGER NOT NULL,
                                     PRIMARY KEY (Region))''');
```

```
Out[7]: <sqlite3.Cursor at 0x5c693b0>
```

DataType Equivalents

DB Type	Python Equivalent	Use
NULL	None	nothing
INTEGER	int (or) long	integers
REAL	float	8-byte floating point numbers
TEXT	str (or) unicode	Strings of characters
BLOB	buffer	Binary data

```
In [8]: cur.execute('INSERT INTO PopByRegion VALUES("Central Africa", 330993)' )
```

```
Out[8]: <sqlite3.Cursor at 0x5c693b0>
```

```
In [9]: cur.execute('INSERT INTO PopByRegion VALUES("Southeastern Africa", 743112)' )
```

```
Out[9]: <sqlite3.Cursor at 0x5c693b0>
```

```
In [10]: cur.execute('INSERT INTO PopByRegion VALUES("Japan", 100562)' )
```

```
Out[10]: <sqlite3.Cursor at 0x5c693b0>
```

```
In [11]: cur.execute('INSERT INTO PopByRegion VALUES("Northern Africa", 1037463)' )
         cur.execute('INSERT INTO PopByRegion VALUES("Southern Asia", 2051941)' )
         cur.execute('INSERT INTO PopByRegion VALUES("Asia Pacific", 785468)' )
         cur.execute('INSERT INTO PopByRegion VALUES("Middle East", 687630)' )
         cur.execute('INSERT INTO PopByRegion VALUES("Eastern Asia", 1362955)' )
         cur.execute('INSERT INTO PopByRegion VALUES("South America", 593121)' )
         cur.execute('INSERT INTO PopByRegion VALUES("Eastern Europe", 223427)' )
         cur.execute('INSERT INTO PopByRegion VALUES("North America", 661157)' )
         cur.execute('INSERT INTO PopByRegion VALUES("Western Europe", 387933)' )

         # ref  http://www.worldmapper.org
```

```
Out[11]: <sqlite3.Cursor at 0x5c693b0>
```

Saving changes

```
In [12]: conn.commit()
```

Retrieving Data

```
In [13]: cur.execute('SELECT Region, Population FROM PopByRegion' )

         print(cur.fetchone())
```

```
('Central Africa', 330993)
```

fetchone method returns each record as a tuple, in order specified in query. If no more records found, It will return **None**.

```
In [14]: print(cur.fetchone())
```

```
('Southeastern Africa', 743112)
```

```
In [15]: print(cur.fetchall())
```

```
[('Japan', 100562), ('Northern Africa', 1037463), ('Southern Asia', 2051941), ('Asia Pacific',
In [16]: # Re-querying
         cur.execute('SELECT Region, Population FROM PopByRegion ORDER BY Region' )
         cur.fetchall()

Out[16]: [('Asia Pacific', 785468),
          ('Central Africa', 330993),
          ('Eastern Asia', 1362955),
          ('Eastern Europe', 223427),
          ('Japan', 100562),
          ('Middle East', 687630),
          ('North America', 661157),
          ('Northern Africa', 1037463),
          ('South America', 593121),
          ('Southeastern Africa', 743112),
          ('Southern Asia', 2051941),
          ('Western Europe', 387933)]

In [17]: cur.execute('SELECT Region, Population FROM PopByRegion ORDER BY Population DESC')
         cur.fetchall()

Out[17]: [('Southern Asia', 2051941),
          ('Eastern Asia', 1362955),
          ('Northern Africa', 1037463),
          ('Asia Pacific', 785468),
          ('Southeastern Africa', 743112),
          ('Middle East', 687630),
          ('North America', 661157),
          ('South America', 593121),
          ('Western Europe', 387933),
          ('Central Africa', 330993),
          ('Eastern Europe', 223427),
          ('Japan', 100562)]

In [18]: cur.execute('SELECT Region FROM PopByRegion' )
         cur.fetchall()

Out[18]: [('Central Africa',),
          ('Southeastern Africa',),
          ('Japan',),
          ('Northern Africa',),
          ('Southern Asia',),
          ('Asia Pacific',),
          ('Middle East',),
          ('Eastern Asia',),
          ('South America',),
          ('Eastern Europe',),
          ('North America',),
          ('Western Europe',)]
```

```
In [19]: cur.execute('SELECT * FROM PopByRegion')
         cur.fetchall()

Out[19]: [('Central Africa', 330993),
          ('Southeastern Africa', 743112),
          ('Japan', 100562),
          ('Northern Africa', 1037463),
          ('Southern Asia', 2051941),
          ('Asia Pacific', 785468),
          ('Middle East', 687630),
          ('Eastern Asia', 1362955),
          ('South America', 593121),
          ('Eastern Europe', 223427),
          ('North America', 661157),
          ('Western Europe', 387933)]
```

Query Conditions
SQL Relational operations

```
------------------------------------------------
Operator         Description
------------------------------------------------
=                Equal to
!=               Not equal to
>                Greater than
<                Lesser than
>=               Greater than or equal to
<=               Lesser than or equal to
```

```
In [20]: cur.execute('SELECT Region FROM PopByRegion WHERE Population > 1000000' )
         cur.fetchall()

Out[20]: [('Northern Africa',), ('Southern Asia',), ('Eastern Asia',)]

In [21]: cur.execute('SELECT Region FROM PopByRegion WHERE Population > 1000000 AND Region < "
         cur.fetchall()

Out[21]: [('Eastern Asia',)]
```

Updating Records

```
In [22]: cur.execute('SELECT * FROM PopByRegion WHERE Region = "Japan"' )
         cur.fetchone()

Out[22]: ('Japan', 100562)

In [23]: cur.execute('UPDATE PopByRegion SET Population = 100600 WHERE Region = "Japan"')

         cur.execute('SELECT * FROM PopByRegion WHERE Region = "Japan"' )
         cur.fetchone()
```

```
Out[23]: ('Japan', 100600)
```

Deleting Records

```
In [24]: cur.execute('SELECT * FROM PopByRegion WHERE Region < "L"' )
         cur.fetchall()

Out[24]: [('Central Africa', 330993),
          ('Japan', 100600),
          ('Asia Pacific', 785468),
          ('Eastern Asia', 1362955),
          ('Eastern Europe', 223427)]

In [25]: cur.execute('DELETE FROM PopByRegion WHERE Region < "L"' )

Out[25]: <sqlite3.Cursor at 0x5c693b0>

In [26]: cur.execute('SELECT * FROM PopByRegion WHERE Region < "L"' )
         cur.fetchall()

Out[26]: []

In [27]: cur.execute('SELECT * FROM PopByRegion');
         cur.fetchall()

Out[27]: [('Southeastern Africa', 743112),
          ('Northern Africa', 1037463),
          ('Southern Asia', 2051941),
          ('Middle East', 687630),
          ('South America', 593121),
          ('North America', 661157),
          ('Western Europe', 387933)]

In [29]: # placing record, back into the table
         cur.execute('INSERT INTO PopByRegion VALUES ("Japan", 100562)' )

         cur.execute('SELECT * FROM PopByRegion');
         cur.fetchall()

Out[29]: [('Southeastern Africa', 743112),
          ('Northern Africa', 1037463),
          ('Southern Asia', 2051941),
          ('Middle East', 687630),
          ('South America', 593121),
          ('North America', 661157),
          ('Western Europe', 387933),
          ('Japan', 100562),
          ('Japan', 100562)]
```

Deleting the table

```
In [30]: cur.execute('DROP TABLE PopByRegion' );

In [31]: cur.execute('SELECT * FROM PopByRegion');


        ---------------------------------------------------------------------------

        OperationalError                          Traceback (most recent call last)

        <ipython-input-31-0833369e7fd5> in <module>
    ----> 1 cur.execute('SELECT * FROM PopByRegion');


        OperationalError: no such table: PopByRegion


In [32]: conn.close()

In [33]: cur.execute('SELECT * FROM PopByRegion');


        ---------------------------------------------------------------------------

        ProgrammingError                          Traceback (most recent call last)

        <ipython-input-33-0833369e7fd5> in <module>
    ----> 1 cur.execute('SELECT * FROM PopByRegion');


        ProgrammingError: Cannot operate on a closed database.
```

Re-opening the Db connection

```
In [34]: conn = sqlite3.connect('population.db')
         cur = conn.cursor()

In [35]: cur.execute('CREATE TABLE PopByCountry(Region TEXT, Country TEXT,Population INTEGER)'

Out[35]: <sqlite3.Cursor at 0x5f5e3b0>

In [41]: cur.execute('INSERT INTO PopByCountry VALUES("Eastern Asia", "China",1285238)')

Out[41]: <sqlite3.Cursor at 0x5f5e3b0>

In [42]: countries = [
             ("Eastern Asia", "DPR Korea", 24056),
             ("Eastern Asia","Hong Kong (China)", 8764),
             ("Eastern Asia", "Mongolia", 3407),
             ("Eastern Asia", "Republic of Korea", 41491),
             ("Eastern Asia", "Taiwan", 1433),
             ("North America", "Bahamas", 368),
             ("North America", "Canada", 40876),
             ("North America", "Greenland", 43),
             ("North America", "Mexico", 126875),
             ("North America", "United States", 493038)
         ]

         for c in countries:
             cur.execute('INSERT INTO PopByCountry VALUES (?, ?, ?)' , (c[0], c[1], c[2]))

         conn.commit()
```

```
In [43]: cur.execute('SELECT * from PopByCountry')
         cur.fetchall()
```

```
Out[43]: [('Eastern Asia', 'China', 1285238),
          ('Eastern Asia', 'DPR Korea', 24056),
          ('Eastern Asia', 'Hong Kong (China)', 8764),
          ('Eastern Asia', 'Mongolia', 3407),
          ('Eastern Asia', 'Republic of Korea', 41491),
          ('Eastern Asia', 'Taiwan', 1433),
          ('North America', 'Bahamas', 368),
          ('North America', 'Canada', 40876),
          ('North America', 'Greenland', 43),
          ('North America', 'Mexico', 126875),
          ('North America', 'United States', 493038)]
```

```
In [44]: cur.execute('''
             SELECT PopByRegion.Region, PopByCountry.Country FROM PopByRegion
                 INNER JOIN PopByCountry WHERE (PopByRegion.Region = PopByCountry.Region) AND
             ''')

         cur.fetchall()
```

```
Out[44]: [('Eastern Asia', 'China'),
          ('Eastern Asia', 'DPR Korea'),
          ('Eastern Asia', 'Hong Kong (China)'),
          ('Eastern Asia', 'Mongolia'),
          ('Eastern Asia', 'Republic of Korea'),
          ('Eastern Asia', 'Taiwan')]
```

```
In [45]: cur.execute('''
             SELECT PopByRegion.Region FROM PopByRegion
                 INNER JOIN PopByCountry WHERE (PopByRegion.Region = PopByCountry.Region)
                                             AND ((PopByCountry.Population * 1.0) / PopByRegio
         cur.fetchall()
```

```
Out[45]: [('Eastern Asia',), ('North America',), ('North America',)]
```

To get unique results, use **DISTINCT**

```
In [47]: cur.execute('''
             SELECT DISTINCT PopByRegion.Region FROM PopByRegion
                 INNER JOIN PopByCountry WHERE (PopByRegion.Region = PopByCountry.Region)
                                             AND ((PopByCountry.Population * 1.0) / PopByRegio
         cur.fetchall()
```

```
Out[47]: [('Eastern Asia',), ('North America',)]
```

Table with Primary Key

```
In [48]: cur.execute('''CREATE TABLE
                             PopByRegion (Region TEXT NOT NULL,
                             Population INTEGER NOT NULL,
                             PRIMARY KEY (Region))''');
```

```
        ---------------------------------------------------------------------------

        OperationalError                          Traceback (most recent call last)

        <ipython-input-48-bd2bf232954b> in <module>
          2                     PopByRegion (Region TEXT NOT NULL,
          3                     Population INTEGER NOT NULL,
    ----> 4                     PRIMARY KEY (Region))''');


        OperationalError: table PopByRegion already exists
```

```
In [49]: cur.execute('''
         CREATE TABLE PopByCountry(
         Region TEXT NOT NULL,
         Country TEXT NOT NULL,
         Population INTEGER NOT NULL,
         CONSTRAINT Country_Key PRIMARY KEY (Region, Country))
         ''')
```

```
        ---------------------------------------------------------------------------

        OperationalError                          Traceback (most recent call last)

        <ipython-input-49-d882c476edb0> in <module>
          5 Population INTEGER NOT NULL,
          6 CONSTRAINT Country_Key PRIMARY KEY (Region, Country))
```

```
----> 7 ''')


    OperationalError: table PopByCountry already exists
```

DB Aggregate Functions

```
------------------------------------------------
Aggregate function      Description
------------------------------------------------
    AVG                 Average of the values
    MIN                 Minimum value
    MAX                 Maximum value
    COUNT               Number of non-null values
    SUM                 Sum of the values
```

```
In [51]: cur.execute('SELECT SUM (Population) FROM PopByRegion' )
         cur.fetchone()

Out[51]: (8965762,)

In [53]: cur.execute('SELECT Region, SUM (Population) FROM PopByCountry GROUP BY Region' )
         cur.fetchall()

Out[53]: [('Eastern Asia', 1364389), ('North America', 661200)]
```

Joins

```
In [54]: cur.execute('''
             SELECT A.Country, B.Country FROM PopByCountry A
                                          INNER JOIN PopByCountry B
             WHERE (ABS(A.Population - B.Population) <= 1000) AND (A.Country != B.Country)''')
         cur.fetchall()

Out[54]: [('Republic of Korea', 'Canada'),
          ('Bahamas', 'Greenland'),
          ('Canada', 'Republic of Korea'),
          ('Greenland', 'Bahamas')]
```

Nested Query

```
In [55]: cur.execute('''
             SELECT DISTINCT Region
             FROM PopByCountry
             WHERE (PopByCountry.Population != 8764)
         ''')
         cur.fetchall()
```

```
Out[55]: [('Eastern Asia',), ('North America',)]

In [56]: cur.execute('''
             SELECT DISTINCT Region
             FROM PopByCountry
             WHERE (PopByCountry.Population = 8764)
         ''')
         cur.fetchall()

Out[56]: [('Eastern Asia',)]

In [57]: cur.execute('''
             SELECT DISTINCT Region
             FROM PopByCountry
             WHERE Region NOT IN
                     (SELECT DISTINCT Region
                     FROM PopByCountry
                     WHERE (PopByCountry.Population = 8764))
         ''')
         cur.fetchall()

Out[57]: [('North America',)]

In [58]: conn.close()
```