How to use Python to work with as database – ch 17

* How to connect to a SQLite database
* How to execute SELECT statements
* How to get the rows in as a result set
* How to execute INSERT, UPDATE, and DELETE statements
* How to test the database code
* How to handle with encapsulation

***The syntax for a SELECT statement that gets all columns***

SELECT \* FROM table

[WHERE selection-criteria]

[ORDER BY column-1 [ASC|DESC] [, column-2 [ASC|DESC] ...]]

***A SELECT statement that gets all columns***

SELECT \* FROM Movie

WHERE categoryID =2

***The syntax for a SELECT statement that gets selected columns***

SELECT column-1[, column-2] ...

FROM table

[WHERE selection-criteria]

[ORDER BY column-1 [ASC|DESC][, column-2 [ASC|DESC] ...]]

***A SELECT statement that gets selected columns and rows***

SELECT name, minutes

FROM Movie

WHERE minutes < 90

ORDER BY minutes ASC

***The syntax for a SELECT statement that joins two tables***

SELECT column-1 [AS alias-1] [[, column-2] [AS alias-2]]...

FROM table-1

[INNER ]JOIN table-2 ON table-1.column-1 = table-2.column-2

***A statement that gets data from two related tables***

SELECT Movie.name, Category.name AS categoryName, minutes

FROM Movie

JOIN Category ON Category.categoryID = Movie.categoryID

WHERE minutes < 90

ORDER BY minutes ASC

***The syntax for the INSERT statement***

INSERT INTO table-name [(column-list)]

VALUES (value-list)

***A statement that uses a column list to add one row***

INSERT INTO Movie (name, year, minutes, categoryID)

VALUES ('Juno', 2007, 96, 2)

***A statement that doesn’t use a column list to add one row***

INSERT INTO Movie

VALUES (14, 2, 'Juno', 2007, 96)

***The syntax for the UPDATE statement***

UPDATE table-name SET expression-1 [, expression-2] ...

WHERE selection-criteria

***A statement that updates a column in one row***

UPDATE Movie

SET minutes = 84

WHERE movieID = 4

***The syntax for the DELETE statement***

DELETE FROM table-name

WHERE selection-criteria

***A statement that deletes one row from a table***

DELETE FROM Movie

WHERE movieID = 14

***A statement that deletes multiple rows from a table***

DELETE FROM Movie

WHERE year = 1979

***How to open a database***

1. Start DB Browser for SQLite.

2. Click the Open Database button or choose the FileOpen Database command. Then, use the dialog box to select the SQLite database you want to open.

3. This displays the tree structure of the database in the Database Structure tab.

***How to view a table***

1. Click the Browse Data tab.

2. Use the drop-down Tables list to select the table that you want to view.

3. To sort the table by the values in a column, click on the column name.

***How to edit a table***

• To edit the data in the rows, change the data in the table.

• To add or delete rows, use the New Record or Delete Record button.

• To save the changes, click the Write Changes button.

• To cancel the changes, click the Revert Changes button.

***How to import the module that supports SQLite databases***

import sqlite3

***The syntax for connecting to a database and returning a connection object***

conn = sqlite3.connect(path\_to\_database\_file)

***How to connect to a database that’s in the working directory***

conn = sqlite3.connect("movies.sqlite")

***How to connect to a database that’s not in the working directory***

import sys

import os

if sys.platform == "win32": else: # Windows

DB\_FILE = "/murach/python/\_db/movies.sqlite"

else: # Mac OS X and Linux

HOME = os.environ["HOME"]

DB\_FILE = HOME + "/Documents/murach/python/\_db/movies.sqlite"

conn = sqlite3.connect(DB\_FILE)

***How to use the close() method to close a connection object***

if conn:

conn.close()

***How to get a cursor object from the connection object***

c = conn.cursor()

***How to execute a SELECT statement that doesn’t have parameters***

query = '''SELECT \* FROM Movie'''

c.execute(query)

***How to execute a SELECT statement that has a parameter***

query = '''SELECT \* FROM Movie

WHERE minutes < ?'''

c.execute(query, (90,))

***How to automatically close the cursor object***

***The code for importing the closing() function***

from contextlib import closing

***The syntax for automatically closing the cursor object***

with closing(resource) as name:

statements

***How to automatically close the cursor object***

with closing(conn.cursor()) as c:

query = '''SELECT \* FROM Movie'''

c.execute(query)

***How to use the fetchone() method to get a row from of a table***

with closing(conn.cursor()) as c:

query = '''SELECT \* FROM Movie

WHERE movieID = ?'''

c.execute(query, (5,)

movie = c.fetchone()

***How to access columns by index***

print("Name: “ + movie[2])

print("Year: " + str(movie[3]))

print("Minutes: " + str(movie[4]))

***How to access columns by name***

***How to use the row\_factory attribute to enable name access***

conn.row\_factory = sqlite3.Row

***How to access columns by name***

print("Name: “+ movie["name"]) "

print(“Year: “ + str(movie["year"]))

print("Minutes: " + str(movie["minutes"]))

***How to use the fetchall() method to retrieve all rows in the cursor***

with closing(conn.cursor()) as c:

query = '''SELECT \* FROM Movie

WHERE minutes < ?'''

c.execute(query, (90,))

movies = c.fetchall()

***How to loop through all rows***

for movie in movies:

print(movie["name"], "|", movie["year"], "|", movie["minutes"])

***How to execute an INSERT statement***

name = "Juno"

year = 2007

minutes = 96 categoryID = 2

with closing(conn.cursor()) as c:

sql = '''INSERT INTO Movie (name, year, minutes, categoryID)

VALUES (?, ?, ?, ?)'''

c.execute(sql, (name, year, minutes, categoryID))

conn.commit()

***How to execute an UPDATE statement***

id = 4

minutes = 84

with closing(conn.cursor()) as c:

sql = '''UPDATE Movie

SET minutes = ?

WHERE movieID = ?'''

c.execute(sql, (minutes, id))

conn.commit()

***How to execute a DELETE statement***

id = 12

with closing(conn.cursor()) as c:

sql = '''DELETE FROM Movie

WHERE movieID = ?'''

c.execute(query, (id,))

conn.commit()