This code demo the use of the artist_similarity.db
This is almost the same code as demo_artist_similarity.py
in the github repository.

This is part of the Million Song Dataset project from
LabRossa (Columbia University) and The Echo Nest.

Copyright 2011, Thierry Bertin-Mahieux

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

---

```
import os
import sys
import glob
import time
import datetime
import numpy as np

try:
    import sqlite3
except ImportError:
    print('you need sqlite3 installed to use this program')
    sys.exit(0)

def encode_string(s):
    """
    Simple utility function to make sure a string is proper
    to be used in a SQLite query
    (different than postgresql, no N to specify unicode)
    EXAMPLE:
    That’s my boy! -> 'That''s my boy!'
    """
    return '"' + s.replace('"', '\"') + '"'

# PATH TO artist_similarity.db
# CHANGE THIS TO YOUR LOCAL CONFIGURATION
# IT SHOULD BE IN THE ADDITIONAL FILES
# (you can use 'subset_artist_similarity.db')
dbfile = '/home/thierry/Columbia/MSongsDB/Tasks_Demos/SQLite/
    artist_similarity.db'

# connect to the SQLite database
conn = sqlite3.connect(dbfile)
# from that connection, get a cursor to do queries
# NOTE: we could query directly from the connection object
c = conn.cursor()
print '****************** GENERAL SQLITE DEMO *******************************'

# list all tables in that dataset
# note that sqlite does the actual job when we call fetchall() or fetchone()
q = "SELECT name FROM sqlite_master WHERE type='table' ORDER BY name"
res = c.execute(q)
print 

* tables contained in that SQLite file/database (there should be 3):

res.fetchall()

[(u'artists'), (u'similarity')]

# list all indices
q = "SELECT name FROM sqlite_master WHERE type='index' ORDER BY name"
res = c.execute(q)
print 

* indices in the database to make reads faster:

res.fetchall()

[(u'idx_sim_target'), (u'idx_target_sim'), (u'sqlite_autoindex_artists_1')]

# list all artist ID
q = "SELECT artist_id FROM artists"
res = c.execute(q)
print 

* number of artist Echo Nest ID in 'artists' table:

print len(res.fetchall())

44745

# get a random similarity relationship
q = "SELECT target,similar FROM similarity LIMIT 1"
res = c.execute(q)
a,s = res.fetchone()
print 

* one random similarity relationship (A→B means B similar to A):

print a,'->',s

AR002UA118789A637D -> ARQDÖR81187FB3B06C
# count number of similar artist to a in previous call
q = "SELECT Count(similar) FROM similarity WHERE target="+encode_string(a)
res = c.execute(q)
print 'artist ',a,' has that many similar artists in the dataset:'

print res.fetchone()[0]

# count number of artist s (c queries up) is similar to
q = "SELECT Count(target) FROM similarity WHERE similar="+encode_string(s)
res = c.execute(q)
print 'artist ',s,' is similar to that many artists in the dataset:'

print res.fetchone()[0]

# DONE
# close cursor and connection
# (if for some reason you added stuff to the db or alter
# a table, you need to also do a conn.commit())
c.close()
conn.close()