```
"""
Thierry Bertin-Mahieux (2010) Columbia University
tb2332@columbia.edu

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

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

Copyright 2010, 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 posgtresql, no N to specify unicode)
    EXAMPLE:
      That's my boy! -> 'That''s my boy!'
    """
    return "'"+s.replace("'","''")+"'"

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

# connect to the SQLite database
conn = sqlite3.connect(dbfile)

```

```
85 # from that connection, get a cursor to do queries
86 c = conn.cursor()
87
88 # SCHEMA OVERVIEW
89 # we got 3 tables
90 # table1: name=artists      #cols=1   (artist_id text)
91 #    One row per artists, no duplicates, usually alphabetical order
92 # table2: name=terms        #cols=1   (term text)
93 #    One row per term, no duplicates, usually alphabetical order
94 # table3: name=artist_term  #cols=2   (artist_id text, term text)
95 #    One row per pair artist_id/term, no duplicate pairs
96 #    Entries in table3 are constrained by table1 and table2,
97 # e.g. an artist_id must exist in table1 before it is used in table3.
98 # NOT ALL ARTISTS HAVE TERMS. They will still all be in table1, but
99 # some artists are not in table3 at all.
100
101 print '*************** GENERAL SQLITE DEMO ***************************'
```

```
*************** GENERAL SQLITE DEMO ***************************
```

```
78
79 # list all tables in that dataset
80 # note that sqlite does the actual job when we call fetchall() or fetchone()
81 q = "SELECT name FROM sqlite_master WHERE type='table' ORDER BY name"
82 res = c.execute(q)
83 print "* tables contained in that SQLite file/database (there should be 3):"
```

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

```
84 print res.fetchall()
```

```
[(u'artist_mbtag',), (u'artist_term',), (u'artists',), (u'mbtags',), (u'
    terms',)]
```

```
85
86 # list all indices
87 q = "SELECT name FROM sqlite_master WHERE type='index' ORDER BY name"
88 res = c.execute(q)
89 print '* indices in the database to make reads faster:'
```

```
* indices in the database to make reads faster:
```

```
90 print res.fetchall()
```

```
[(u'idx_artist_id_mbtag',), (u'idx_artist_id_term',), (u'
    idx_mbtag_artist_id',), (u'idx_term_artist_id',), (u'
    sqlite_autoindex_artists_1',), (u'sqlite_autoindex_mbtags_1',), (u'
    sqlite_autoindex_terms_1',)]
```

```
92
93 print '*************** ARTISTS TABLE DEMO ***************************'
```

```
*************** ARTISTS TABLE DEMO ***************************
```

```
93
94 # list all artists
95 q = "SELECT * FROM artists"
96 res = c.execute(q)
97 print '* list all known artists in the database (display first 3):'
```

```
* list all known artists in the database (display first 3):
```

```
98 print res.fetchall()[:3]
```

```
[(u'AR002UA1187B9A637D',), (u'AR003FB1187B994355',), (u'AR006821187FB5192B
    ',)]
```

```

# list all artists that id starts with ARB
q = "SELECT artist_id FROM artists WHERE SUBSTR(artist_id,1,3)='ARB' LIMIT 2
    "
res = c.execute(q)
print '* list artists whose ID starts with ARB (we ask for 2 of them):'
```

```
* list artists whose ID starts with ARB (we ask for 2 of them):
```

```
print res.fetchall()
```

```
[(u'ARB00VV1187B9AF612',), (u'ARB01F41187B9894C9',)]
```

```

# count all artists
q = "SELECT COUNT(artist_id) FROM artists"
res = c.execute(q)
print '* count the number of artists (with or without tags):'
```

```
* count the number of artists (with or without tags):
```

```
print res.fetchone()
```

```
(44745,)
```

```

print '*************** TERMS TABLE DEMO ******************************'
```

```
*************** TERMS TABLE DEMO ******************************
```

```

# list all terms (=tags)
q = "SELECT * FROM terms"
res = c.execute(q)
print '* list all known terms in the database (display first 3):'
```

```
* list all known terms in the database (display first 3):
```

```
print res.fetchall()[:3]
```

```
[(u'00s',), (u'00s alternative',), (u'00s country',)]
```

```

# list all terms that start with 'indie'
q = "SELECT term FROM terms WHERE SUBSTR(term,1,5)='indie' LIMIT 3"
res = c.execute(q)
print "* list terms that start with 'indie' (we ask for 3 of them):"
```

```
* list terms that start with 'indie' (we ask for 3 of them):
```

```
print res.fetchall()
```

```
[(u'indie',), (u'indie acoustic',), (u'indie alternative',)]
```

```

# check if a tag is inthe dataset
q1 = "SELECT term FROM terms WHERE term='rock' LIMIT 1"
q2 = "SELECT term FROM terms WHERE term='abc123xyz'"
res = c.execute(q1)
res1_str = str(res.fetchone())
res = c.execute(q2)
res2_str = str(res.fetchone())
print '* we check if two tags are in the database, (the first one is):'
```

```
* we check if two tags are in the database, (the first one is):
```

```
print 'rock:',res1_str,', abc123xyz:',res2_str
```

```
rock: (u'rock',) , abc123xyz: None
```

```
135
136 # similar for mtags, list all mbtags
137 q = "SELECT * FROM mbtags"
138 res = c.execute(q)
139 print '* btags work the same as terms, e.g. list all known mbtags (display
    first 3):'
```

```
* btags work the same as terms, e.g. list all known mbtags (display first
    3):
```

```
140 print res.fetchall()[:3]
```

```
[(u'00s',), (u'00s 10s',), (u'1 13 165900 150 7672 22647 34612 48720 59280
    74602 87545 95495 107182 131087 141522 153710',)]
```

```
141
142 # get one badly encoded, fix it...
143 # is it a problem only when we write to file???
144 # we want to show the usage of t.encode('utf-8')  with t a term
145
146 print '*************** ARTIST / TERM TABLE DEMO **********************'
```

```
*************** ARTIST / TERM TABLE DEMO **********************
```

```
147
148 # note that the Beatles artist ID is: AR6XZ861187FB4CECD
149
150 # get all tags from the Beatles
151 q = "SELECT term FROM artist_term WHERE artist_id='AR6XZ861187FB4CECD'"
152 res = c.execute(q)
153 print "* we get all tags applied to the Beatles (we know their artist ID),
    we show 4:"
```

```
* we get all tags applied to the Beatles (we know their artist ID), we
    show 4:
```

```
154 print res.fetchall()[:4]
```

```
[(u'60s',), (u'acoustic',), (u'am pop',), (u'ambient',)]
```

```
155
156 # count number of tags applied to The Beatles
157 q = "SELECT COUNT(term) FROM artist_term WHERE artist_id='AR6XZ861187FB4CECD
    '"
158 res = c.execute(q)
159 print "* we count the number of unique tags applied to The Beatles:"
```

```
* we count the number of unique tags applied to The Beatles:
```

```
160 print res.fetchone()
```

```
(30,)
```

```
161
162 # get artist IDs that ahve been tagged with 'jazz'
163 # note the encode_string function, that mostly doubles the ' sign
164 q = "SELECT artist_id FROM artist_term WHERE term="+encode_string('jazz')
165 q += " ORDER BY RANDOM() LIMIT 2"
166 res = c.execute(q)
167 print "* we get artists tagged with 'jazz', we display 2 at random:"
```

```
* we get artists tagged with 'jazz', we display 2 at random:
```

```
168 print res.fetchall()
```

```
[(u'ARLJWVM11F50C4DD6E',), (u'ARHBVWA1187B99172D',)]
```

```
169
170 # count number of artists tagged with 'rock'
171 q = "SELECT COUNT(artist_id) FROM artist_term WHERE term="+encode_string('
    rock')
172 res = c.execute(q)
173 print "* we count the number of unique artists that got term 'rock':"
```

```
* we count the number of unique artists that got term 'rock':
```

```
174 print res.fetchone()
```

```
(27274,)
```

```
175
176 # count number of artists mb tagged with 'rock'
177 q = "SELECT COUNT(artist_id) FROM artist_mbtag WHERE mbtag="+encode_string('
    rock')
178 res = c.execute(q)
179 print "* samething with musicbrainz tag 'rock':"
```

```
* samething with musicbrainz tag 'rock':
```

```
180 print res.fetchone()
```

```
(843,)
```

```
181
182 # get artists that have term 'rock' but not mbtag 'rock'
183 q = "SELECT artist_id FROM artist_term WHERE term="+encode_string('rock')
184 q += " EXCEPT SELECT artist_id FROM artist_mbtag WHERE mbtag="+encode_string
    ('rock')
185 q += " LIMIT 1"
186 res = c.execute(q)
187 print "* one artist that has term 'rock' but not mbtag 'rock':"
```

```
* one artist that has term 'rock' but not mbtag 'rock':
```

```
188 print res.fetchone()
```

```
(u'AR003FB1187B994355',)
```

```
189
190 # get artists that have no terms
191 # simple with the EXCEPT keyword
192 # other cool keywords: UNION, UNION ALL, INTERSECT
193 q = "SELECT artist_id FROM artists EXCEPT SELECT artist_id FROM artist_term
    LIMIT 1"
194 res = c.execute(q)
195 artist_notag = res.fetchone()
196 print '* we show an artist with no terms:'
```

```
* we show an artist with no terms:
```

```
197 if artist_notag is None:
198     # debug, make sure all artists have at least one tag, can be slow
199     q = "SELECT * FROM artists"
200     res = c.execute(q)
201     allartists = map(lambda x: x[0], res.fetchall())
202     for art in allartists:
203         q = "SELECT COUNT(term) FROM artist_term WHERE artist_id='"+art+"'"
204         res = c.execute(q)
205         assert res.fetchone()[0] > 0
206     print '(found no artist with no terms, we double-checked)'
207 else:
208     print artist_notag
```

```
(u'AR1A5C01187B99DD70',)
```

```
209
210 # DONE
211 # close the cursor and the connection
212 # (if for some reason you added stuff to the db or alter
213 #  a table, you need to also do a conn.commit())
214 c.close()
215 conn.close()
```