DJANGO UNCHAINED

By

Siddarth Naik

Sai Balaguru Ganess

V. Jahnavi Reddy

Swetha Vemparala

1. INTRODUCTION:

The aim of this project is to traverse through a list of tables which are interlinked through a primary key and then print the columns required from the last table. Using the attributes of first table, which are not primary keys, we search for a table which has the non-key attributes of first table as the primary key of second table. Here, finding this new table is a task since we are also running a piece of code to find its location.

1.1 SQLITE3:

SQLITE is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. However, SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion is not possible.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

2. SOURCE CODE:

```
import pandas as pd
import sqlite3
```

Here we are importing pandas and SQLITE3 packages.

```
con = sqlite3.connect("chinook.db")
cur = con.cursor()
query = """ Select name from sqlite_master where type = "table" """
```

Here we are establishing the connection with the database and selecting the table names present in the database and also creating a cursor to get the data.

```
res = cur.execute(query)
list_tables = res.fetchall()
```

Here we are executing the query and fetching the data that is in the cursor and putting all the table names in “chinook.db” into “list_tables”

```
table_name_columns = {}
for t_name in list_tables:
    q1 = 'SELECT * FROM {}'.format(str(t_name[0]))
    cursor = con.execute(q1)
    temp1 = []
    for i in range(len(cursor.description)):
        temp1.append(cursor.description[i][0])
    table_name_columns[str(t_name[0])] = temp1
```

Here we are creating a dictionary “table_name_columns” and inserting the column names of each table into the dictionary with “Table Name” as Keys.

```
name_col = table_name_columns['playlists'][0]
```

```
for key in table_name_columns.keys():
    if name_col in table_name_columns[key]:
    new_table = key
```

We search for the table which has a column named “playlist_id” and it’s stored in the variable new_table. The table is “playlist_tracks”

```
q2 = "select * from {}".format(new_table)
cur2 = con.execute(q2)
playlist_track = cur2.fetchall()
```

Here we are fetching the track_id from playlist_track table.

```
id = [1,3,5,8]
track_id = {}
for j in id:
             counter = 0
             temp = []
             for i in range(len(playlist_track)):
                if playlist_track[i][0] == j:
                    if(counter!=10):
                        counter+=1
                        temp.append(playlist_track[i][1])
                    else:
                        break
             track_id[j] = temp
```

We have created list named “ID” and hard-coded the values as [1,3,5,8]. We fetch the respective “track_id” for each Playlist_id and store it in a dictionary – “track_id”

```
name_col1 = table_name_columns['playlist_track'][1]
for key in table_name_columns.keys():
  if name_col1 in table_name_columns[key] and key != 'playlist_track':
             new_table1 = key
```

We search for another table which contains the column – “Track_id” and store it in new_table1. The Table is: “Tracks”

```
q3 = "select * from {}".format(new_table1)

cur3 = con.execute(q3)

tracks = cur3.fetchall()
```

Here we are fetching all the values from table “Tracks”.

```
track_name = {}

for id in track_id.keys():

                        for i in track_id[id]:

                            for j in range(len(tracks)):

                                   if tracks[j][0] == i:

                                      track_name[i]=[tracks[j][1],tracks[j][2]]
```

Creating a dictionary that has track_id as the key and the list of track_name, album_id as the values.

.

```
name_col2 = table_name_columns['tracks'][2]
for key in table_name_columns.keys():
        if name_col2 in table_name_columns[key] and key != 'tracks':
                new_table2 = key
```

Here we search for another table that contains the column “album_id” and it’s stored in new_table2. The table is: “albums”

```
q4 = "select * from {}".format(new_table2)

     cur4 = con.execute(q4)

     albums = cur4.fetchall()
```

To fetch the contents of table albums

```
for id in track_name.keys():
    for j in range(len(albums)):
        if albums[j][0] == track_name[id][1]:
                     track_name[id].append(albums[j][1])
                      track_name[id].append(albums[j][2])
```

With album_id, we are fetching album_name and artist_id and it’s appended to the values of respective “track_id” in the dictionary “track_name”

```
name_col3 = table_name_columns['albums'][2]
for key in table_name_columns.keys():
    if name_col3 in table_name_columns[key] and key != 'albums':
        new_table3 = key
```

We traverse to the “artist” table using “Artist_id”. The table name is stored in new_table3.

```
q5 = "select * from {}".format(new_table3)
cur5 = con.execute(q5)
artists = cur5.fetchall()
```

We store the data in the list “artists”

```
for id in track_name.keys():
   for j in range(len(artists)):
    if artists[j][0] == track_name[id][3]:
     track_name[id].append(artists[j][1])
```

Appending the artist_name from artists table to the values of track_name.

```
df = pd.DataFrame()
```

Creating a new dataframe

```
df['track_id'] = track_name.keys()
song_name = []
album_id = []
album_name = []
```

```
artist_id = []
artist_name = []
```

Creating lists for all columns

```
for key in track_name.keys():
    song_name.append(track_name[key][0])
    album_id.append(track_name[key][1])
     album_name.append(track_name[key][2])
    artist_id.append(track_name[key][3])
    artist_name.append(track_name[key][4])
df['Song title'] = song_name
df['Album_id'] = album_id
df['Album Name'] = album_name
df['Artist ID'] = artist_id
df['Artist Name'] = artist_name
```

Appending the list for each column and concatenating it to the Data frame.

Inserting the lists into dataframes

```
play_id = []
```

Creating new list for the playlist_id.

```
x = list(df['track_id'])
for i in x:
        if i in track_id[1]:
                 play_id.append(1)
        elif i in track_id[3]:
                play_id.append(3)
        elif i in track_id[5]:
                play_id.append(5)
        elif i in track_id[8]:
                play_id.append(8)
```

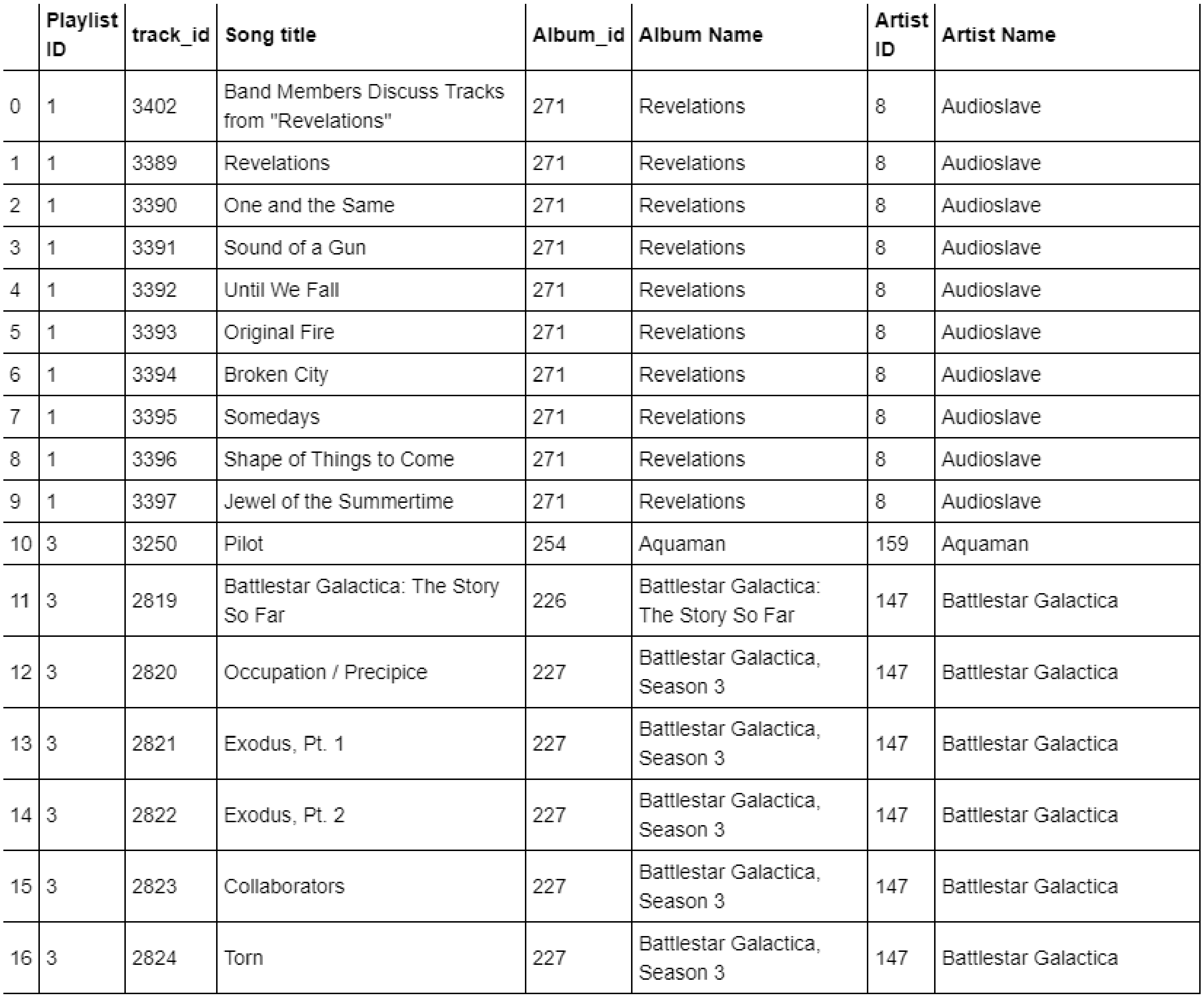
Inserting values into the list play_id in accordance with the Track_id.

```
df.insert(0,'Playlist ID',play_id)
```

Inserting the column “Playlist_id” to the Data Frame “df”

```
df
```

Viewing contents of data frame.



	Playlist ID	track_id	Song title	Album_id	Album Name	Artist ID	Artist Name
0	1	3402	Band Members Discuss Tracks from "Revelations"	271	Revelations	8	Audioslave
1	1	3389	Revelations	271	Revelations	8	Audioslave
2	1	3390	One and the Same	271	Revelations	8	Audioslave
3	1	3391	Sound of a Gun	271	Revelations	8	Audioslave
4	1	3392	Until We Fall	271	Revelations	8	Audioslave
5	1	3393	Original Fire	271	Revelations	8	Audioslave
6	1	3394	Broken City	271	Revelations	8	Audioslave
7	1	3395	Somedays	271	Revelations	8	Audioslave
8	1	3396	Shape of Things to Come	271	Revelations	8	Audioslave
9	1	3397	Jewel of the Summertime	271	Revelations	8	Audioslave
10	3	3250	Pilot	254	Aquaman	159	Aquaman
11	3	2819	Battlestar Galactica: The Story So Far	226	Battlestar Galactica: The Story So Far	147	Battlestar Galactica
12	3	2820	Occupation / Precipice	227	Battlestar Galactica, Season 3	147	Battlestar Galactica
13	3	2821	Exodus, Pt. 1	227	Battlestar Galactica, Season 3	147	Battlestar Galactica
14	3	2822	Exodus, Pt. 2	227	Battlestar Galactica, Season 3	147	Battlestar Galactica
15	3	2823	Collaborators	227	Battlestar Galactica, Season 3	147	Battlestar Galactica
16	3	2824	Torn	227	Battlestar Galactica, Season 3	147	Battlestar Galactica

3. Conclusion:

The given task has been completed and executed successfully.