A PYTHON PROJECT

ON

PANDAS AND SQLITE

COMPLETED BY TEAM TORNADO

GUIDE: MR. SHOBHIT NIGGAM

Submitted By:

Dolly Bagaria
Devvrat Vaidya
Ishant Tiwari
Pramodh Narayan L

TECHNOLOGIES USED

The project uses **Python 3** as the coding platform and uses two libraries i.e.,

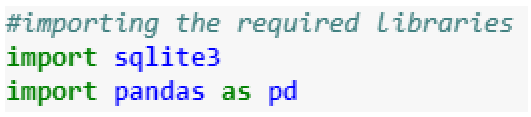
1. **Pandas -** an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming
2. **SQLite 3 -** SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine

SOURCE CODE

IMPLEMENTATION USING PANDAS AND SQLITE3

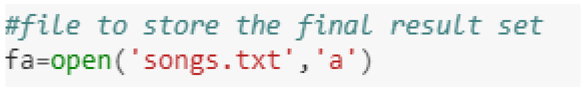
a) Import

Importing essential libraries pandas and sqlite3



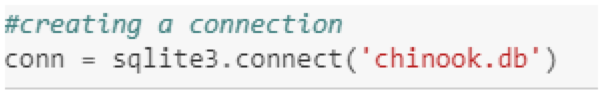
```
#importing the required libraries
import sqlite3
import pandas as pd
```

b) Opening a file to store the final result set



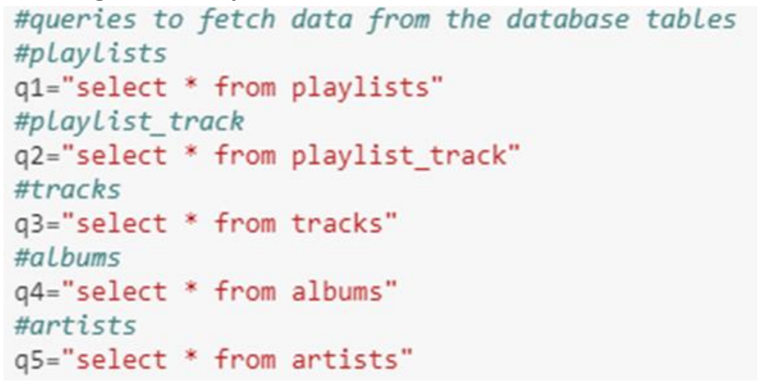
```
#file to store the final result set
fa=open('songs.txt','a')
```

c) Creating a connection between python and the database



```
#creating a connection
conn = sqlite3.connect('chinook.db')
```

d) Fetching all the required data from the tables in database



```
#queries to fetch data from the database tables
#playlists
q1="select * from playlists"
#playlist_track
q2="select * from playlist_track"
#tracks
q3="select * from tracks"
#albums
q4="select * from albums"
#artists
q5="select * from artists"
```

e) **Storing the fetched data in data frames**

```
#storing the fetched data in dataframes
playlist = pd.read_sql_query(q1, conn)
playlist_track= pd.read_sql_query(q2, conn)
track= pd.read_sql_query(q3, conn)
album= pd.read_sql_query(q4, conn)
artist= pd.read_sql_query(q5, conn)
```

f) **Declaring the required datastructures**

```
listfinal=[]
d={}
```

g) **Function to populate a dictionary with keys as column names and values as a list of table names to which the columns belong to**

```
#function to create a dictionary with keys as a specific column names and values as list of table names ha
ving that column
def insert_dict(lista,tablename):
    for i in lista:
        global d
        if i not in d:
            d[i]=[]
        d[i].append(tablename)
```

h) **Creating a list of column names for each table**

```
#creating a list of column names in each table
listplaylist=list(playlist.columns)
listplaylisttrack=list(playlist_track.columns)
listtrack=list(track.columns)
listalbum=list(album.columns)
listartist=list(artist.columns)
```

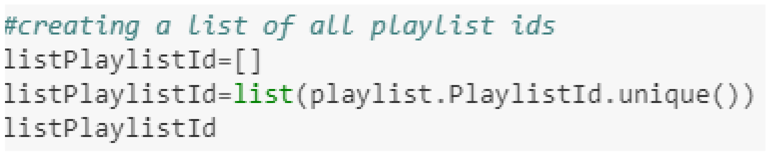
i) **Calling the helper function for each table**

```
#creating a dictionary- tables to which a column belongs to
insert_dict(listplaylist,'playlists')
insert_dict(listplaylisttrack,'playlists_track')
insert_dict(listtrack,'tracks')
insert_dict(listalbum,'albums')
insert_dict(listartist,'artists')
```

j) **Finding the required table to fetch PlaylistId**

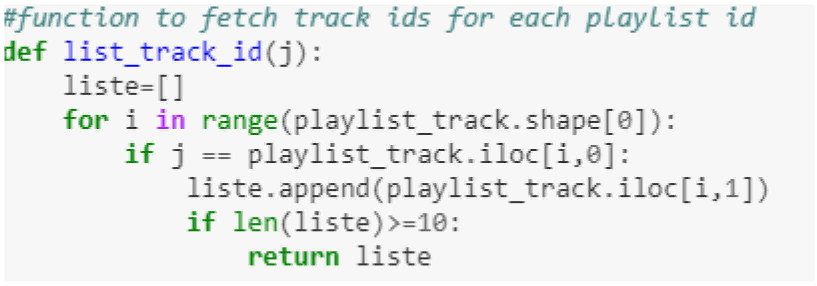
```
seta=set(d['PlaylistId'])
setb=set(d['TrackId'])
```

k) **Creating a list of all playlists ids**



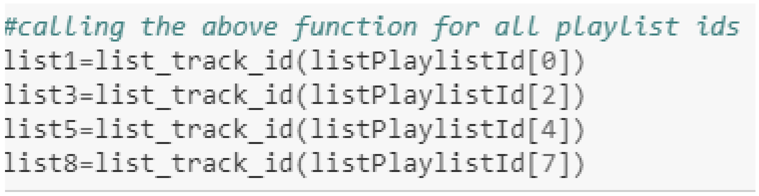
```
#creating a list of all playlist ids
listPlaylistId=[]
listPlaylistId=list(playlist.PlaylistId.unique())
listPlaylistId
```

l) **Function to fetch track ids for corresponding playlist ids**



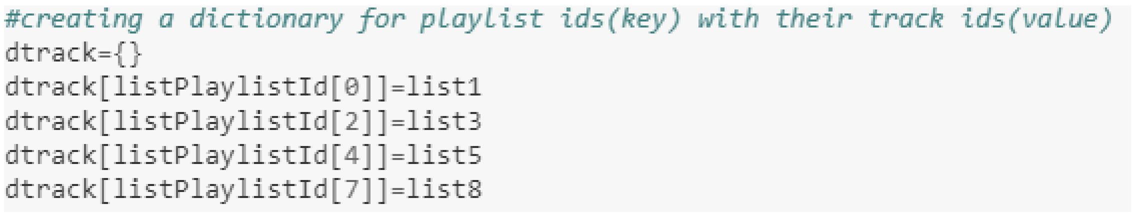
```
#function to fetch track ids for each playlist id
def list_track_id(j):
    liste=[]
    for i in range(playlist_track.shape[0]):
        if j == playlist_track.iloc[i,0]:
            liste.append(playlist_track.iloc[i,1])
            if len(liste)>=10:
                return liste
```

m) **Calling the helper function for four playlist ids**



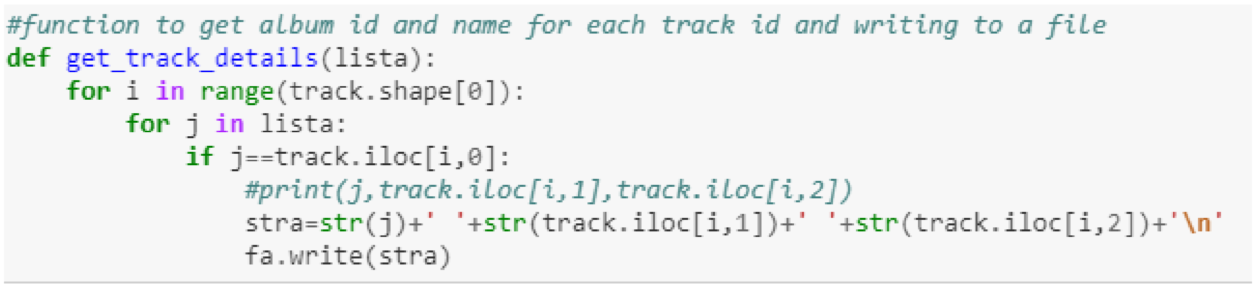
```
#calling the above function for all playlist ids
list1=list_track_id(listPlaylistId[0])
list3=list_track_id(listPlaylistId[2])
list5=list_track_id(listPlaylistId[4])
list8=list_track_id(listPlaylistId[7])
```

m) **Creating a dictionary with keys as playlist ids and values as trackids**



```
#creating a dictionary for playlist ids(key) with their track ids(value)
dtrack={}
dtrack[listPlaylistId[0]]=list1
dtrack[listPlaylistId[2]]=list3
dtrack[listPlaylistId[4]]=list5
dtrack[listPlaylistId[7]]=list8
```

n) **Function to write to a file the required data**



```
#function to get album id and name for each track id and writing to a file
def get_track_details(lista):
    for i in range(track.shape[0]):
        for j in lista:
            if j==track.iloc[i,0]:
                #print(j,track.iloc[i,1],track.iloc[i,2])
                stra=str(j)+' '+str(track.iloc[i,1])+' '+str(track.iloc[i,2])+'\n'
                fa.write(stra)
```

o) Calling the helper function for trackids

```
#caloling the function for 4 track ids
get_track_details(dtrack[1])
get_track_details(dtrack[3])
get_track_details(dtrack[5])
get_track_details(dtrack[8])
```

p) Closing the file

```
#closing the file
fa.close()
```

SAMPLE OUTPUT

```
3389 Revelations 271
3390 One and the Same 271
3391 Sound of a Gun 271
3392 Until We Fall 271
3393 Original Fire 271
3394 Broken City 271
3395 Somedays 271
3396 Shape of Things to Come 271
3397 Jewel of the Summertime 271
3402 Band Members Discuss Tracks from "Revelations" 271
2819 Battlestar Galactica: The Story So Far 226
2820 Occupation / Precipice 227
2821 Exodus, Pt. 1 227
2822 Exodus, Pt. 2 227
2823 Collaborators 227
2824 Torn 227
2825 A Measure of Salvation 227
2826 Hero 227
2827 Unfinished Business 227
3250 Pilot 254
51 We Die Young 7
52 Man In The Box 7
```

.