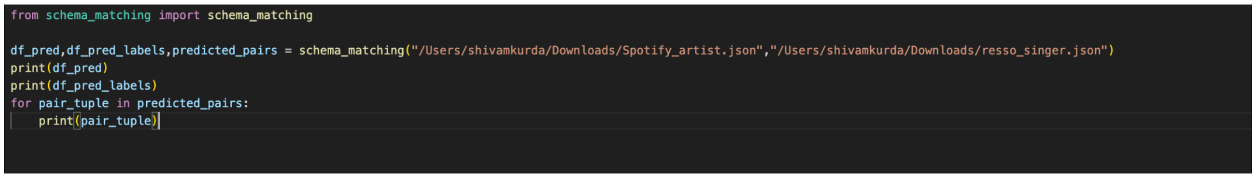
SCHEMA MATCHING/MAPPING

We tried to implement schema matching/mapping using ‘schema-matching’ library of python. This library uses machine learning algorithm to find matching between columns of two tables or schemas. Library takes two tables as input and returns a correlation matrix representing the amount of correlation between different columns of two tables. If a value in the matrix is high then , it is very likely that those two attributes will match. The library employs linguistic matching and structural matching algorithms to find out matching.
The sample code is given below:



```
from schema_matching import schema_matching

df_pred,df_pred_labels,predicted_pairs = schema_matching("/Users/shivamkurda/Downloads/Spotify_artist.json","/Users/shivamkurda/Downloads/resso_singer.json")
print(df_pred)
print(df_pred_labels)
for pair_tuple in predicted_pairs:
    print(pair_tuple)
```

The matching that we used is as follows:

Global_Music	Spotify
Artist(Artist_name)	Artist(Artist_name)
Artist(Age)	Artist(Age)
Artist(Followers(Spotify))	Artist(Followers)
Genre(Gen_Name)	Genre(Name)
Songs(Song_Name)	Songs(Name)
Songs(Song_Genre)	Songs(Genre)
Songs(Song_Artist)	Songs(Artist)
Songs(LaunchYear)	Songs(ReleaseYear)
Songs(Song_Platform)	“Spotify”
SONG_RECORD(Play_Date)	Plays(Play_Date)
SONG_RECORD(User_Age)	Customer(Age)
SONG_RECORD(User_Gender)	Customer(Gender)

SONG_RECORD(User_Location)	Customer(Address)
SONG_RECORD(Song_Artist)	Artist(Artist_Name)
SONG_RECORD(Song_Genre)	Genre(Name)

SCHEMA MAPPING BETWEEN GLOBAL MUSIC AND APPLE Music DATABASE

Global_Music	Apple Music
Artist(Artist_name)	Artist(Name)
Artist(Age)	Artist(Age)
Artist(Followers(Apple Music))	Artist(Followers)
Genre(Gen_Name)	Genre(Genre_Name)
Songs(Song_Name)	Songs(Song_Name)
Songs(Song_Genre)	Songs(Song_Genre)
Songs(Song_Artist)	Songs(Song_Artist)
Songs(LaunchYear)	Songs(Song_Release_Year)
Songs(Song_Platform)	“Apple Music”
SONG_RECORD(Play_Date)	Plays(Play_Date)
SONG_RECORD(User_Age)	Customer(Age)
SONG_RECORD(User_Gender)	Customer(Gender)
SONG_RECORD(User_Location)	Customer(Location)
SONG_RECORD(Song_Artist)	Artist(Name)
SONG_RECORD(Song_Genre)	Genre(Genre_Name)

SCHEMA MAPPING BETWEEN GLOBAL MUSIC AND Resso DATABASE

Global_Music	RESSO
Artist(Artist_name)	Singer(name)
Artist(Age)	Singer(age)
Artist(Followers(Resso))	Singer(followers)
Genre(Gen_Name)	Genre(name)
Songs(Song_Name)	Songs(name)
Songs(Song_Genre)	Songs(genre)
Songs(Song_Artist)	Songs(singer)
Songs(LaunchYear)	Songs(releaseyear)
Songs(Song_Platform)	“Resso”
SONG_RECORD(Play_Date)	Plays(playdate)
SONG_RECORD(User_Age)	User(Age)
SONG_RECORD(User_Gender)	User(Gender)
SONG_RECORD(User_Location)	User(Address)
SONG_RECORD(Song_Artist)	Singer(Artist_Name)
SONG_RECORD(Song_Genre)	Genre(Name)

SCHEMA MAPPING BETWEEN GLOBAL MUSIC AND YOUTUBE MUSIC DATABASE

Global_Music	YOUTUBE MUSIC
Artist(Artist_name)	Artist(name)
Artist(Age)	Artist(age)
Artist(Followers(YOUTUBE MUSIC))	Artist(followers)
Genre(Gen_Name)	Genre(name)
Songs(Song_Name)	Songs(name)
Songs(Song_Genre)	Songs(genre#)
Songs(Song_Artist)	Songs(artist#)
Songs(LaunchYear)	Songs(releaseyear)
Songs(Song_Platform)	“Youtube Music”

ETL/DATA EXCHANGE/Data Propagation

For ETL/Data exchange/Data Propagation, we do not require any external ETL tools as data stored in our local databases are homogenous and are stored as MYSQL relational databases.ETL tools are useful when local data sources are heterogeneous like when one is csv and another is json and one other is any relational database. But this is not the case for our project. Since all the databases are mysql we can simply use sql queries to extract and load data from local sources to global database. Mapping between local and global database is used for transforming data in local databases to fit in global database. Example for this is query given below:

```
INSERT INTO Artist
SELECT name as Artist_Name,
date_of_birth as DOB,
followers as followers_resso ,
NULL AS followers_spotify,
NULL AS followers_youtube,
NULL AS  followers_apple
FROM resso.singer;
SELECT * FROM ARTIST;
```

Here ARTIST table is in global databases.

The above query transforms the data of singer schema in local database “resso” to fit into the ARTIST Table of global database(“Global Music”).

Adding/Removing of Data

For handling of adding and removing of tuple from local sources, we will use exact change algorithm to handle changes in the global database. Algorithm maintains the global database without accessing it so it is computationally efficient.

If there is deletion of tuple, and if the tuple is produced by algorithm , then the tuple will not be present in the updated global database. Same is the case for insertion. Triggers are also used for efficient handling.

If the data source is deleted then we are handling it using Events of MySql, we have created an event which basically checks if there is a database deletion after every 5 seconds, if it is the case then we are updating the global database with deleting the records of that database from the global database. If a new data source is added, then MYSQL queries will be used to insert new data into global database. Following is the example of event_creation query:

```
DELIMITER //

CREATE EVENT Maintain
ON SCHEDULE EVERY 5 SECOND
DO
BEGIN

  IF NOT EXISTS (SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
WHERE SCHEMA_NAME = 'spotify') THEN
```

```
    UPDATE ARTIST SET followers_spotify=NULL;
  END IF;

  IF NOT EXISTS (SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA
WHERE SCHEMA_NAME = 'apple_music') THEN
    UPDATE ARTIST SET followers_apple=NULL;
  END IF;

END //
DELIMITER ;
```