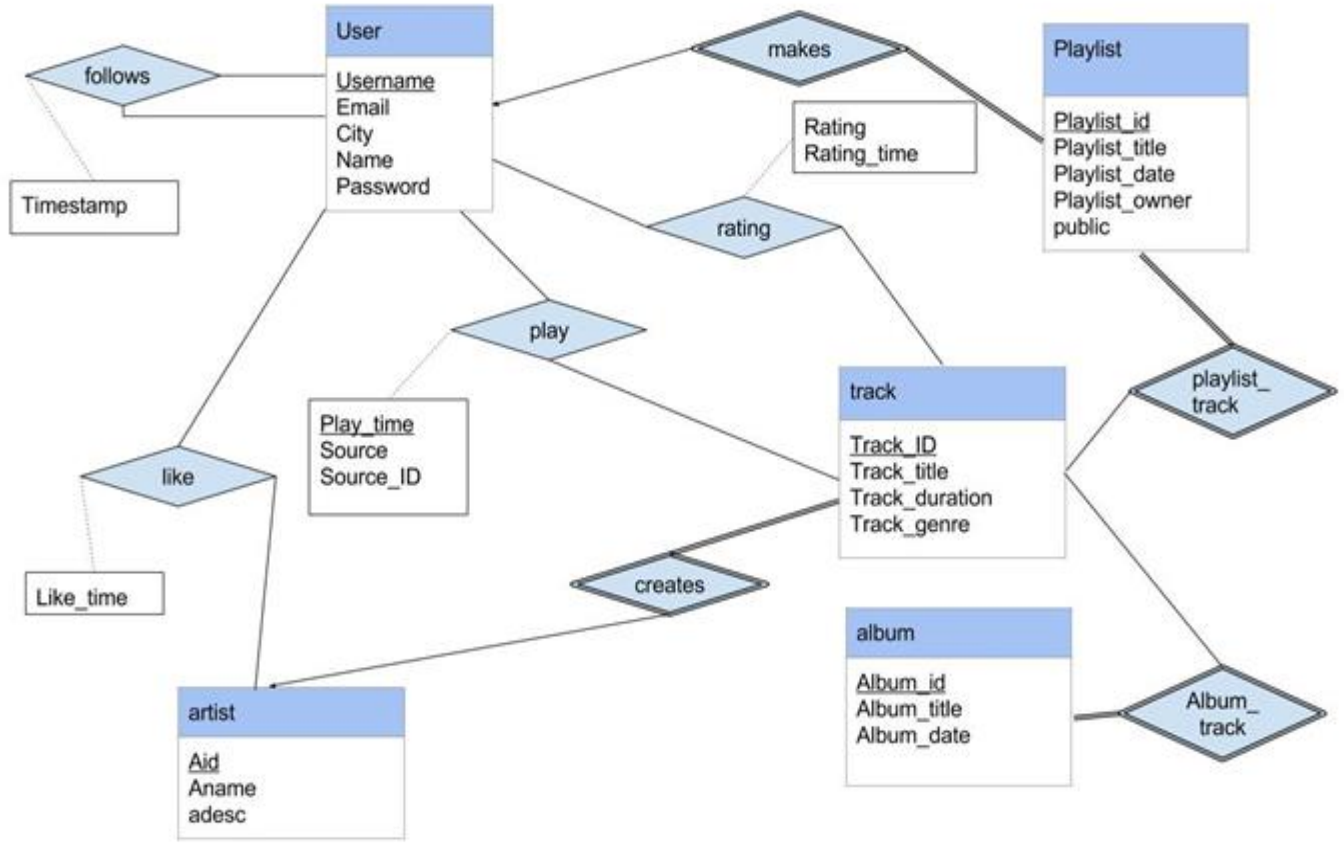
Project Members:

Seo Pallichirayil (sgp322)

Rahul Purushottam Gaonkar(rpg283)

Project Part 1

(a) Design, justify, and create an appropriate relational schema for the above situation. Make sure your schema is space efficient and suitably normalized. Show an ER diagram of your design, and a translation into relational format. Identify keys and foreign key constraints. Note that you may have to revisit your design if it turns out later that the design is not suitable for some of the queries or functionality. Provide a detailed explanation of your design decisions!
Ans:



Relational Schema:

Album (<u>Album_id</u>, Album_title, Album_date)
Album_track (<u>Album_id</u>, <u>Track_id</u>)
Artist (<u>Aid</u>, Aname, Adesc)
follow (<u>Username</u>, <u>Following_id</u>, Timestamp)
like (<u>Username</u>, <u>Aid</u>, Like_time)
play (<u>Username</u>, <u>Track_id</u>, <u>Play_time</u>, Source, Source_ID)
playlist (<u>Playlist_id</u>, Playlist_title, Playlist_date, Playlist_owner, public)
playlist_track (<u>Playlist_id</u>, <u>Track_id</u>)
rating (<u>Username</u>, <u>Track_id</u>, Rating, Rating_time)
track (<u>Track_id</u>, Track_title, Track_duration, Track_genre, Track_aid)
User (<u>Username</u>, Email, City, Name, Password)

Foreign Keys:

Album_track.Album_id references Album.Album_id
Album_track.Track_id references track.Track_id
follow.Username references User.Username
follow.Following_id references User.Username
like.Username references User.Username
like.Aid references Artist.aid
play.Username references User.Username
play.Track_id references track.Track_id
playlist.Playlist_owner references User.Username
playlist_track.Playlist_id references playlist.Playlist_id
playlist_track.Track_id references track.Track_id
rating.Username references User.Username
rating.Track_id references track.Track_id
track.Track_aid references Artist.Aid

Assumptions:

1. Each playlist and Album should contain atleast one track.
2. A user can't follow himself.

Design Decisions :

1. The 'public' attribute of 'playlist' relation will be a Boolean value which will be set depending on if the playlist is public or private.
2. The 'Source' attribute of the 'Play' relation will have three values namely 'Album', 'Playlist' and 'Null' depending on whether the track being played is a part of any Album, Playlist or is an individual track respectively.
3. The 'Source_ID' attribute of 'Play' relation will have values from the 'Album_id' or 'Playlist_id' attribute of Album or Playlist relation respectively if track being played is a part of any Album or Playlist and the value of Source_ID would be null if it is an individual track.
4. The Album relation is normalized and divided into two relations Album and Album_Track to reduce redundancy as a single Album can have multiple tracks and the Album_id would be repeated for each record and it would also violate the primary key(Album_id) constraint of the Album relation if not normalized.
5. The playlist relation is normalized and divided into two relations playlist and playlist_track to reduce redundancy as a single playlist can have multiple tracks and the Playlist_id would be repeated for each record and it would also violate the primary key(Playlist_id) constraint of the playlist relation if not normalized.
6. A User can play same track multiple times, so we define (Username, Track_id, Play_time) as the primary key of the Play table.
7. We only store one record for the rating given by a user to a particular product in the rating relation, if the user changes his rating for that product in future then we update the old record.The rating can be any value between 0 and 5.
8. The record is inserted in the follow and like relation if a user follows another user or likes an artist respectively and a record is deleted in the follow and like relation if a user unfollows another user or unlike an artist respectively.

9. The hash value of the password will be stored in the 'password' attribute of the User relation for every User record. (This will be implemented in the second part of the project)

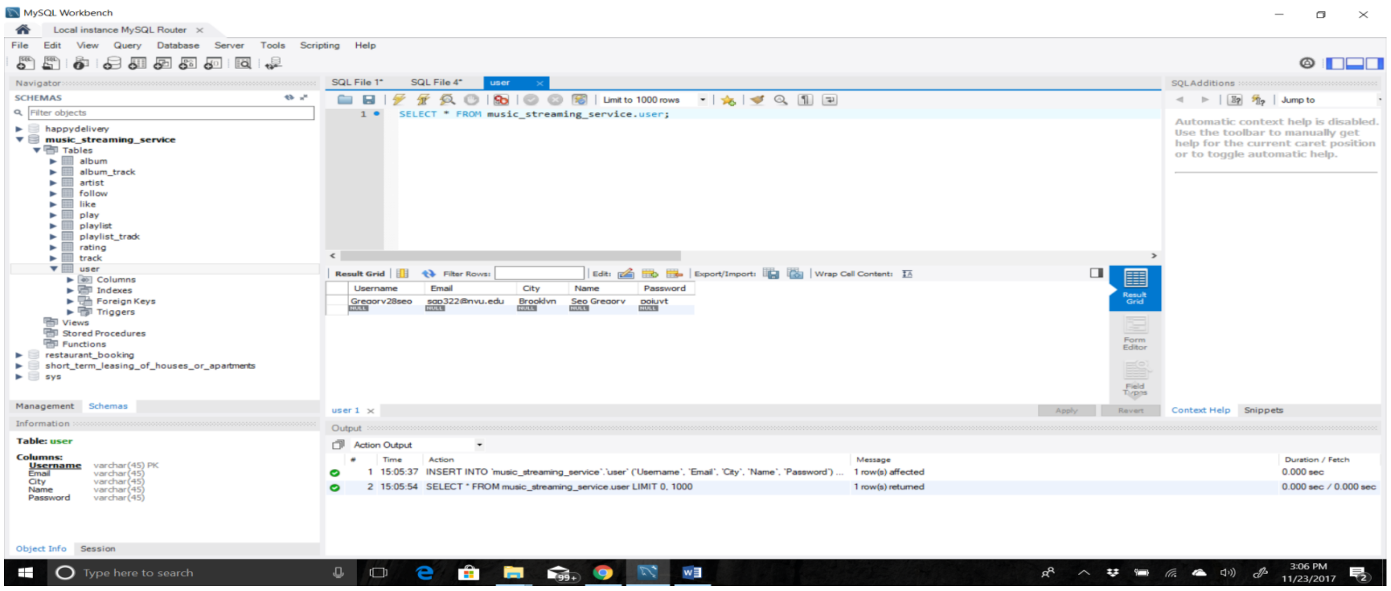
(b) Create the database schema, together with key, foreign key, and other constraints.

(c) Write SQL queries (or sequences of SQL queries) for the following tasks:

- Create a record for a new user account, with a name, a login name, and a password.

Ans:

INSERT INTO `music_streaming_service`.`user` (`Username`, `Email`, `City`, `Name`, `Password`) VALUES ('Gregory28seo', 'sgp322@nyu.edu', 'Brooklyn', 'Seo Gregory', 'poiuyt');



```
SELECT * FROM music_streaming_service.user;
```

Username	Email	City	Name	Password
Gregory28seo	sgp322@nyu.edu	Brooklyn	Seo Gregory	poiuyt
NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                                                                   Message              Duration / Fetch
1  15:05:37  INSERT INTO `music_streaming_service`.`user` (`Username`, `Email`, `City`, `Name`, `Password`) ...  1 row(s) affected    0.000 sec
2  15:05:54  SELECT * FROM music_streaming_service.user LIMIT 0, 1000                                 1 row(s) returned    0.000 sec / 0.000 sec
```

Username	Email	City	Name	Password
Gregory28seo	sgp322@nyu.edu	Brooklyn	Seo Gregory	poiuyt

- For each artist, list their ID, name, and how many times their tracks have been played by users.

Ans:

Explanation: The Number_of_times_track_played would be 0 in two cases:

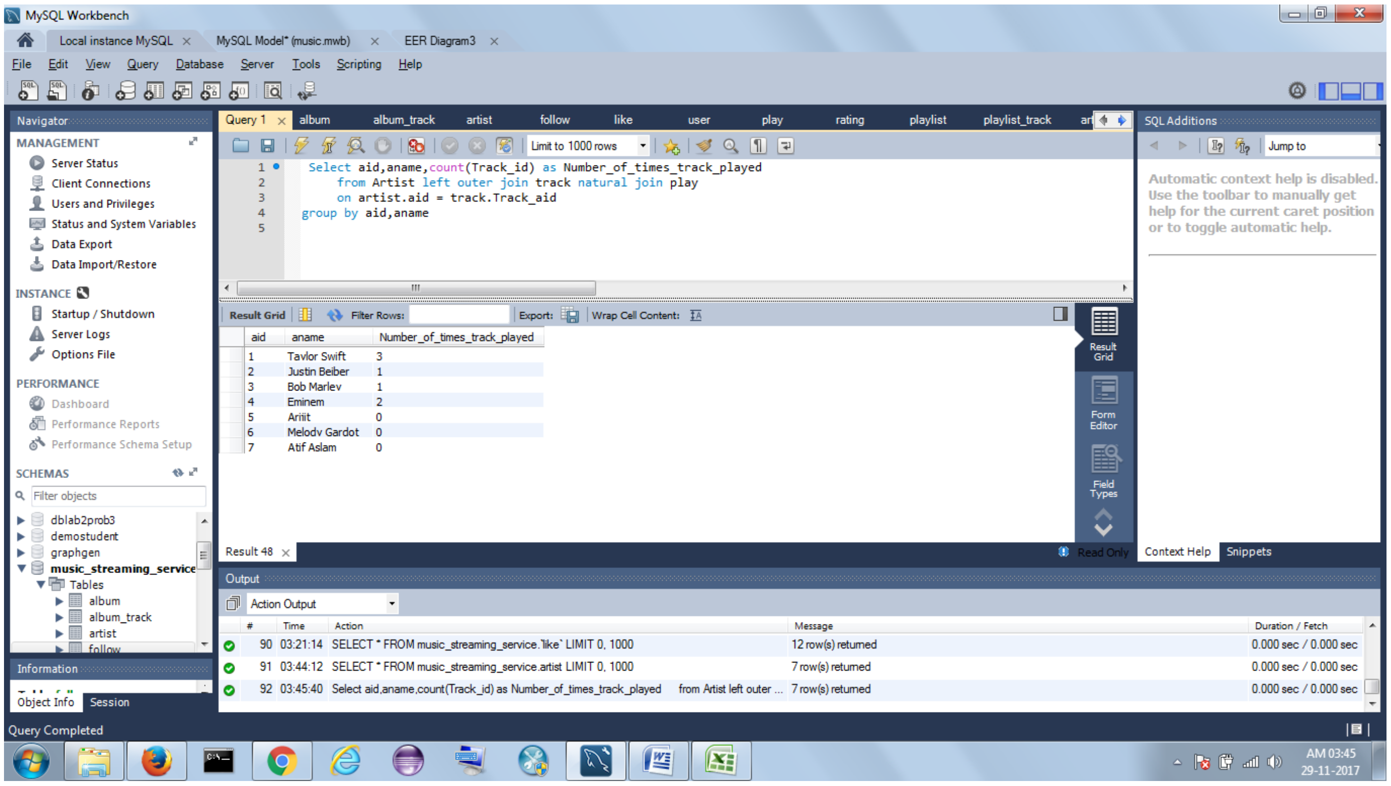
1. **If for a particular artist,all the tracks created by him are never played.For example : Artist 'Arijit' and 'Melody Gardot' in the below result set.**
2. **If a particular artist has not created any track.For example : Artist 'Atif Aslam' in the below result set.**

Select aid,aname,count(Track_id) as Number_of_times_track_played

from Artist left outer join track natural join play

on artist.aid = track.Track_aid

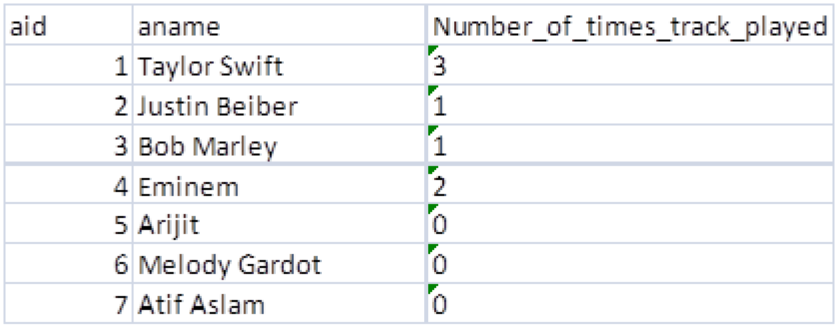
group by aid,aname



```
Select aid,aname,count(Track_id) as Number_of_times_track_played
    from Artist left outer join track natural join play
    on artist.aid = track.Track_aid
group by aid,aname
```

```
aid  aname          Number_of_times_track_played
1    Tavlor Swift   3
2    Justin Beiber  1
3    Bob Marlev     1
4    Eminem         2
5    Ariiit         0
6    Melodv Gardot  0
7    Atif Aslam     0
```

```
#   Time      Action                                                                              Message            Duration / Fetch
90  03:21:14  SELECT * FROM music_streaming_service.`like` LIMIT 0, 1000                          12 row(s) returned 0.000 sec / 0.000 sec
91  03:44:12  SELECT * FROM music_streaming_service.artist LIMIT 0, 1000                           7 row(s) returned  0.000 sec / 0.000 sec
92  03:45:40  Select aid,aname,count(Track_id) as Number_of_times_track_played    from Artist left outer ...  7 row(s) returned  0.000 sec / 0.000 sec
```

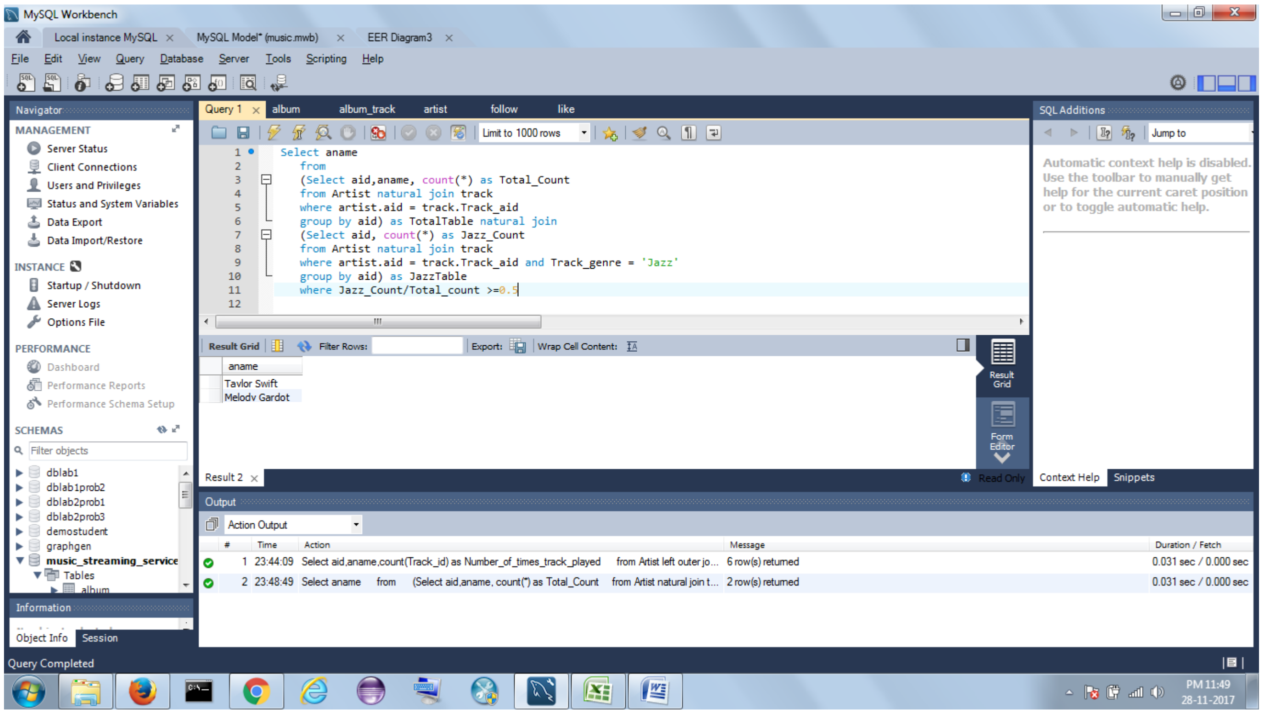


aid	aname	Number_of_times_track_played
1	Taylor Swift	3
2	Justin Beiber	1
3	Bob Marley	1
4	Eminem	2
5	Arijit	0
6	Melody Gardot	0
7	Atif Aslam	0

- List all artists that are mainly playing Jazz, meaning that at least half of their tracks are of genre Jazz.

Ans:

```
Select aname
from
(Select aid,aname, count(*) as Total_Count
from Artist natural join track
where artist.aid = track.Track_aid
group by aid) as TotalTable natural join
(Select aid, count(*) as Jazz_Count
from Artist natural join track
where artist.aid = track.Track_aid and Track_genre = 'Jazz'
group by aid) as JazzTable
where Jazz_Count/Total_count >=0.5
```



```
Select aname
   from
   (Select aid,aname, count(*) as Total_Count
   from Artist natural join track
   where artist.aid = track.Track_aid
   group by aid) as TotalTable natural join
   (Select aid, count(*) as Jazz_Count
   from Artist natural join track
   where artist.aid = track.Track_aid and Track_genre = 'Jazz'
   group by aid) as JazzTable
   where Jazz_Count/Total_count >=0.5
```

```
aname
Tavlor Swift
Melodv Gardot
```

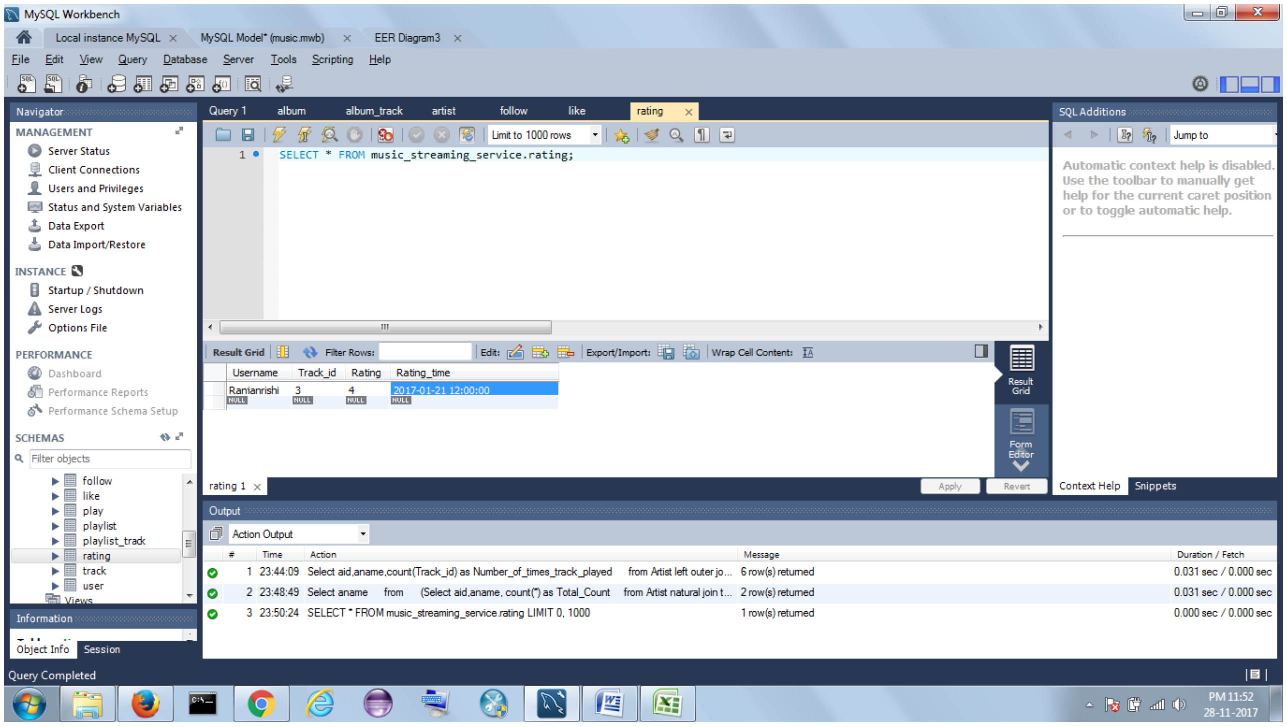
```
#  Time      Action                                                                         Message            Duration / Fetch
1  23:44:09  Select aid,aname,count(Track_id) as Number_of_times_track_played   from Artist left outer jo...  6 row(s) returned  0.031 sec / 0.000 sec
2  23:48:49  Select aname   from   (Select aid,aname, count(*) as Total_Count   from Artist natural join t...  2 row(s) returned  0.031 sec / 0.000 sec
```

aname
Taylor Swift
Melody Gardot

- Insert a new rating given by a user for a track.

Ans:

INSERT INTO `music_streaming_service`.`rating` (`Username`, `Track_id`, `Rating`, `Rating_time`) VALUES ('Ranjanrishi', '3', '4', '2017-1-21 12:00:00');



```
SELECT * FROM music_streaming_service.rating;
```

```
Username     Track_id  Rating  Rating_time
Ranjanrishi  3         4       2017-01-21 12:00:00
NULL         NULL      NULL    NULL
```

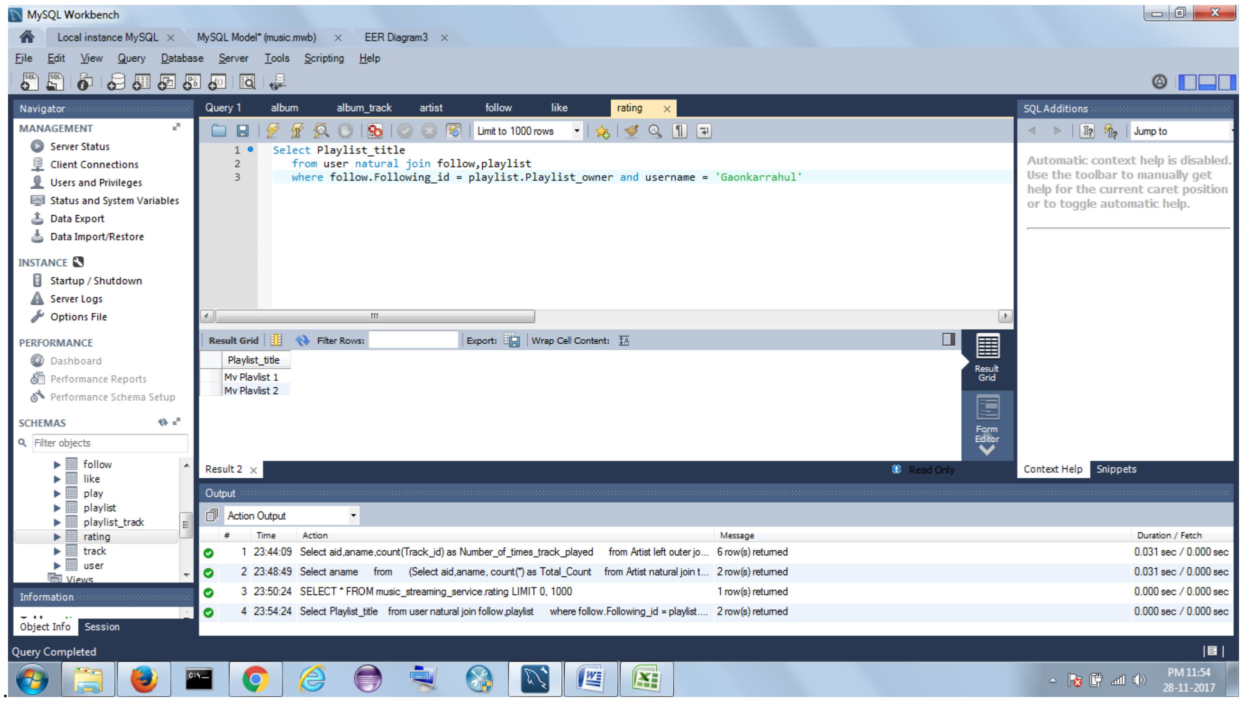
```
#  Time      Action                                                                         Message            Duration / Fetch
1  23:44:09  Select aid,aname,count(Track_id) as Number_of_times_track_played   from Artist left outer jo...  6 row(s) returned  0.031 sec / 0.000 sec
2  23:48:49  Select aname   from   (Select aid,aname, count(*) as Total_Count   from Artist natural join t...  2 row(s) returned  0.031 sec / 0.000 sec
3  23:50:24  SELECT * FROM music_streaming_service.rating LIMIT 0, 1000                      1 row(s) returned  0.000 sec / 0.000 sec
```

Username	Track_id	Rating	Rating_time
Ranjanrishi	3	4	2017-01-21 12:00:00

- For a particular user, say “NancyInQueens”, list all playlists that were made by users that she follows.

Ans:
Test Case 1: A User follows multiple users and if any of those multiple users have created playlists.
Explanation:
User Gaonkarrahul follows Gregory28Seo and Nadikanoop .My Playlist 1 is created by Gregory28Seo and My Playlist 2 is created by Nadikanoop. So we get, My Playlist 1 and My Playlist 2 in the result set.

Select Playlist_title
from user natural join follow,playlist
where follow.Following_id = playlist.Playlist_owner and username = 'Gaonkarrahul'



```
Select Playlist_title
   from user natural join follow,playlist
   where follow.Following_id = playlist.Playlist_owner and username = 'Gaonkarrahul'
```

Playlist_title
My Playlist 1
My Playlist 2

```
#  Time      Action                                                                                        Message            Duration / Fetch
1  23:44:09  Select aid,aname,count(Track_id) as Number_of_times_track_played    from Artist left outer jo...  6 row(s) returned  0.031 sec / 0.000 sec
2  23:48:49  Select aname    from    (Select aid,aname, count(*) as Total_Count    from Artist natural join t...  2 row(s) returned  0.031 sec / 0.000 sec
3  23:50:24  SELECT * FROM music_streaming_service.rating LIMIT 0, 1000                                          1 row(s) returned  0.000 sec / 0.000 sec
4  23:54:24  Select Playlist_title    from user natural join follow,playlist    where follow.Following_id = playlist....  2 row(s) returned  0.000 sec / 0.000 sec
```

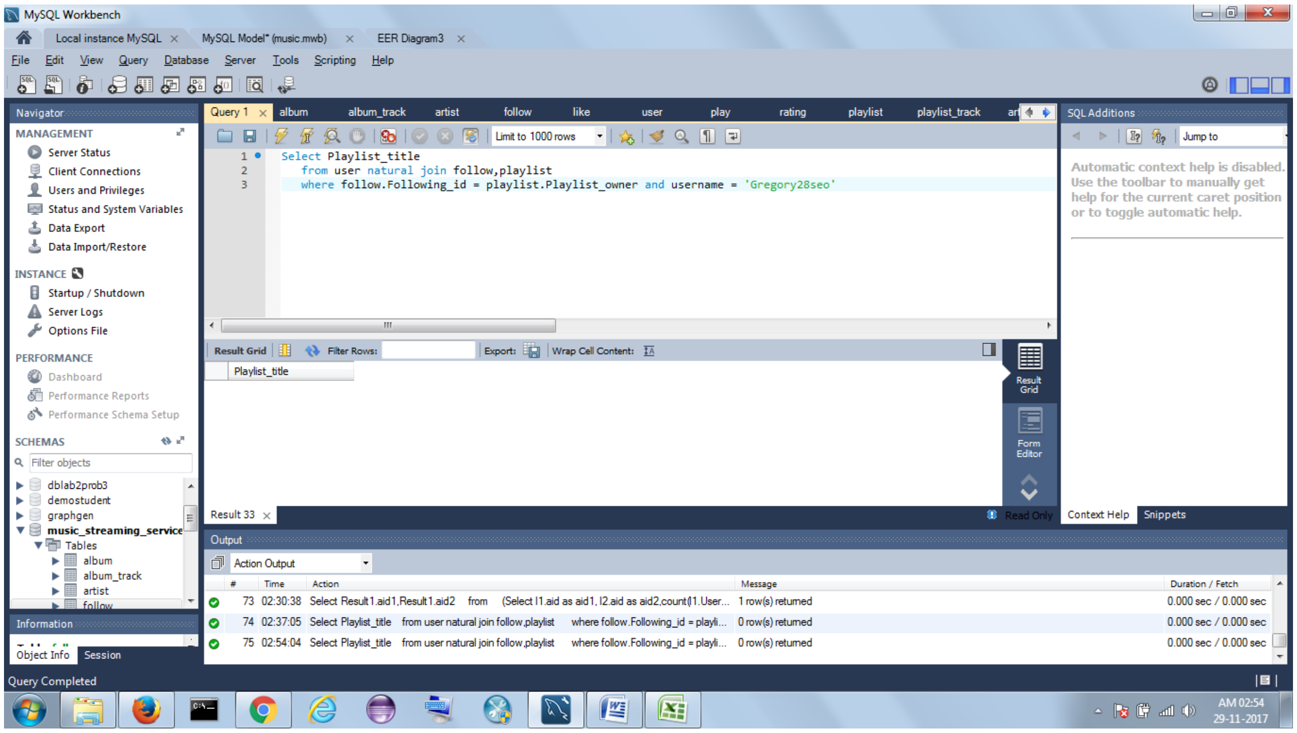
Playlist_title
My Playlist 1
My Playlist 2

Test Case 2: A User follows multiple users and if none of those multiple users have created playlists.
Explanation:
User Gregory28seo follows Newalkarbhushan but Newalkarbhushan has not created any playlist. So we get empty result set.

Select Playlist_title

from user natural join follow,playlist

where follow.Following_id = playlist.Playlist_owner and username = 'Gregory28seo'



```
Select Playlist_title
   from user natural join follow,playlist
   where follow.Following_id = playlist.Playlist_owner and username = 'Gregory28seo'
```

Playlist_title

```
#   Time      Action                                                                                 Message             Duration / Fetch
73  02:30:38  Select Result1.aid1,Result1.aid2   from   (Select l1.aid as aid1, l2.aid as aid2,count(l1.User...  1 row(s) returned   0.000 sec / 0.000 sec
74  02:37:05  Select Playlist_title   from user natural join follow,playlist    where follow.Following_id = playli...  0 row(s) returned   0.000 sec / 0.000 sec
75  02:54:04  Select Playlist_title   from user natural join follow,playlist    where follow.Following_id = playli...  0 row(s) returned   0.000 sec / 0.000 sec
```

<u>Test Case 3 :</u> A User doesn’t follow any other user.

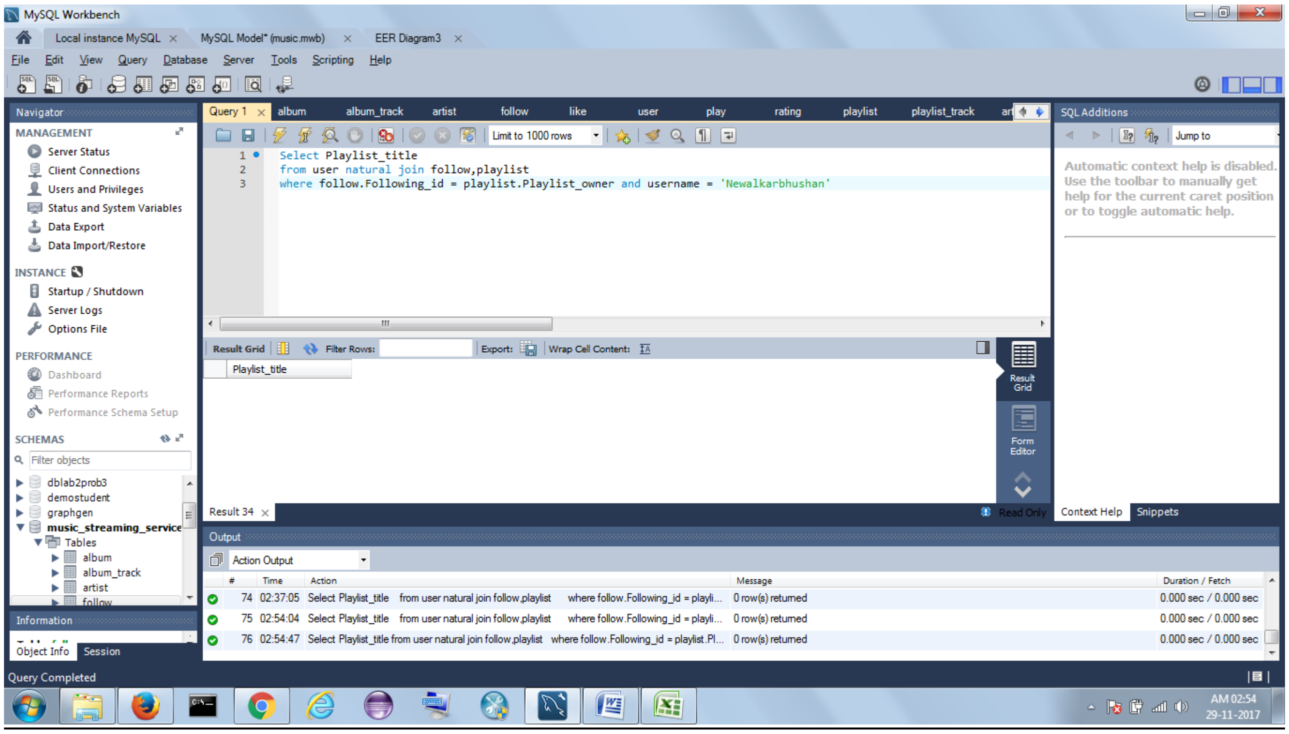
Explanation:

User Newalkarbhushan doesn’t follow any other User.So we get empty result set.

Select Playlist_title

from user natural join follow,playlist

where follow.Following_id = playlist.Playlist_owner and username = 'Newalkarbhushan'



```
Select Playlist_title
from user natural join follow,playlist
where follow.Following_id = playlist.Playlist_owner and username = 'Newalkarbhushan'
```

Playlist_title

```
#   Time      Action                                                                                 Message             Duration / Fetch
74  02:37:05  Select Playlist_title   from user natural join follow,playlist    where follow.Following_id = playli...  0 row(s) returned   0.000 sec / 0.000 sec
75  02:54:04  Select Playlist_title   from user natural join follow,playlist    where follow.Following_id = playli...  0 row(s) returned   0.000 sec / 0.000 sec
76  02:54:47  Select Playlist_title from user natural join follow,playlist  where follow.Following_id = playlist.Pl...  0 row(s) returned   0.000 sec / 0.000 sec
```

- List all songs where the track title or artist title matches some set of keywords (if possible, use ``contains", or otherwise ``like", for this query).

Ans:

The search functionality defined below will match each keyword from the input set of keywords provided and checks if it matches with either the name of the artist (aname) or the title of the track (Track_title) and returns the records of the songs where a match is found.

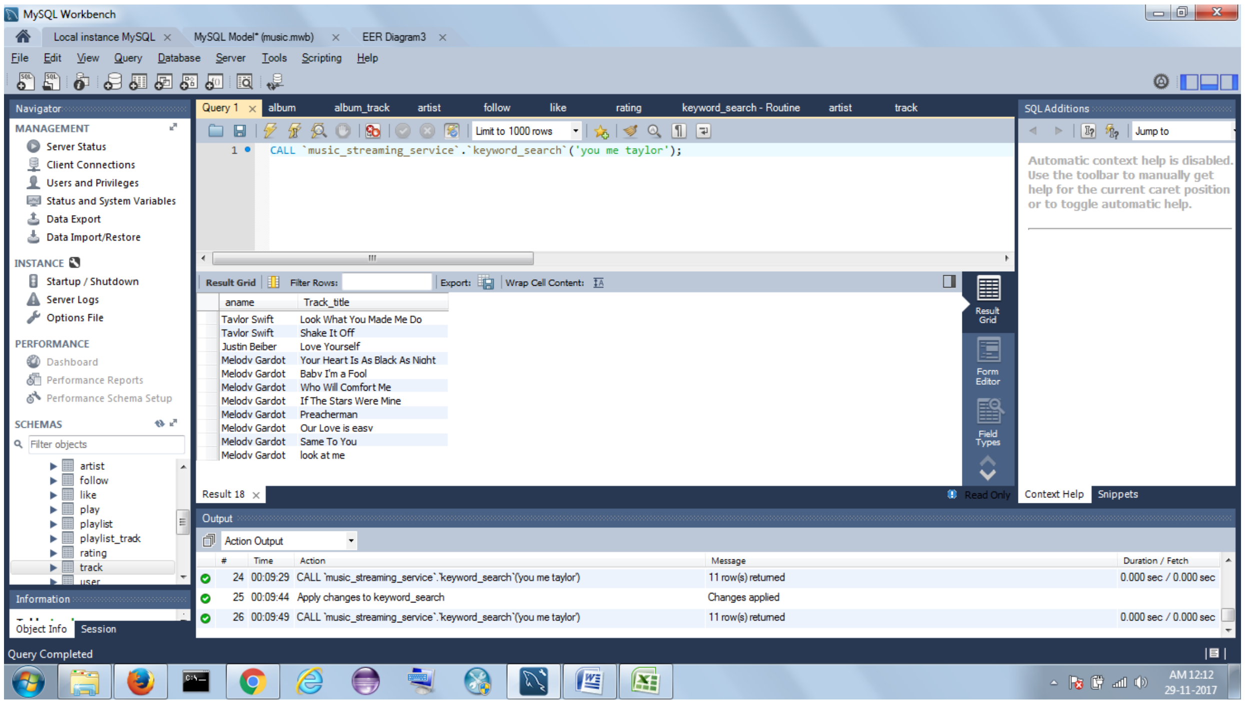
Procedure:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `keyword_search`(IN keyword varchar(100))
BEGIN
set @query = CONCAT('Select aname,Track_title from artist,track where aid = Track_aid and
(Track_title like
',REPLACE(CONCAT('\'%',keyword,'%\''),' ','%\' or Track_title like \'%'),' or aname like
',REPLACE(CONCAT('\'%',keyword,'%\''),' ','%\' or aname like \'%'),')');
PREPARE stmt FROM @query;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
END
```

Procedure Call:

Test Case 1

CALL `music_streaming_service`.`keyword_search`('you me taylor');



```
CALL `music_streaming_service`.`keyword_search`('you me taylor');
```

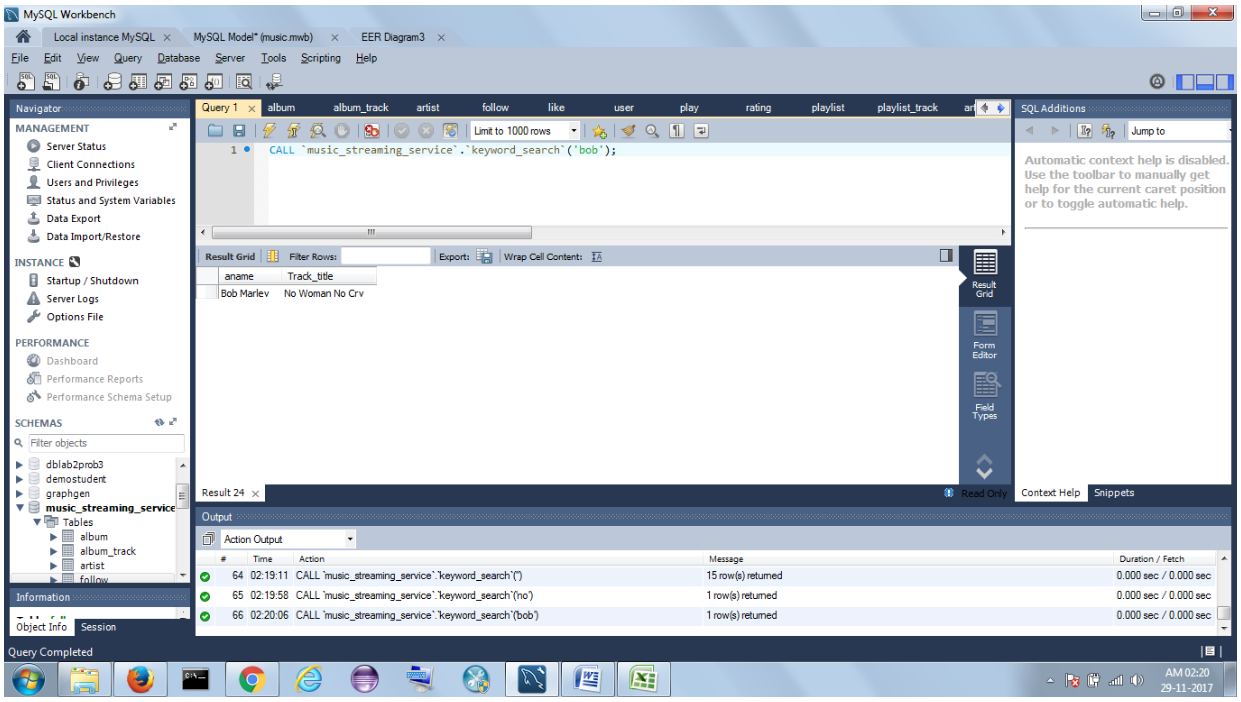
aname	Track_title
Tavlor Swift	Look What You Made Me Do
Tavlor Swift	Shake It Off
Justin Beiber	Love Yourself
Melodv Gardot	Your Heart Is As Black As Niaht
Melodv Gardot	Babv I'm a Fool
Melodv Gardot	Who Will Comfort Me
Melodv Gardot	If The Stars Were Mine
Melodv Gardot	Preacherman
Melodv Gardot	Our Love is easv
Melodv Gardot	Same To You
Melodv Gardot	look at me

#	Time	Action	Message	Duration / Fetch
24	00:09:29	CALL `music_streaming_service`.`keyword_search`('you me taylor')	11 row(s) returned	0.000 sec / 0.000 sec
25	00:09:44	Apply changes to keyword_search	Changes applied	
26	00:09:49	CALL `music_streaming_service`.`keyword_search`('you me taylor')	11 row(s) returned	0.000 sec / 0.000 sec

aname	Track_title
Taylor Swift	Look What You Made Me Do
Taylor Swift	Shake It Off
Justin Beiber	Love Yourself
Melody Gardot	Your Heart Is As Black As Night
Melody Gardot	Baby I'm a Fool
Melody Gardot	Who Will Comfort Me
Melody Gardot	If The Stars Were Mine
Melody Gardot	Preacherman
Melody Gardot	Our Love is easy
Melody Gardot	Same To You
Melody Gardot	look at me

Test Case 2

CALL `music_streaming_service`.`keyword_search`('bob');



```
CALL `music_streaming_service`.`keyword_search`('bob');
```

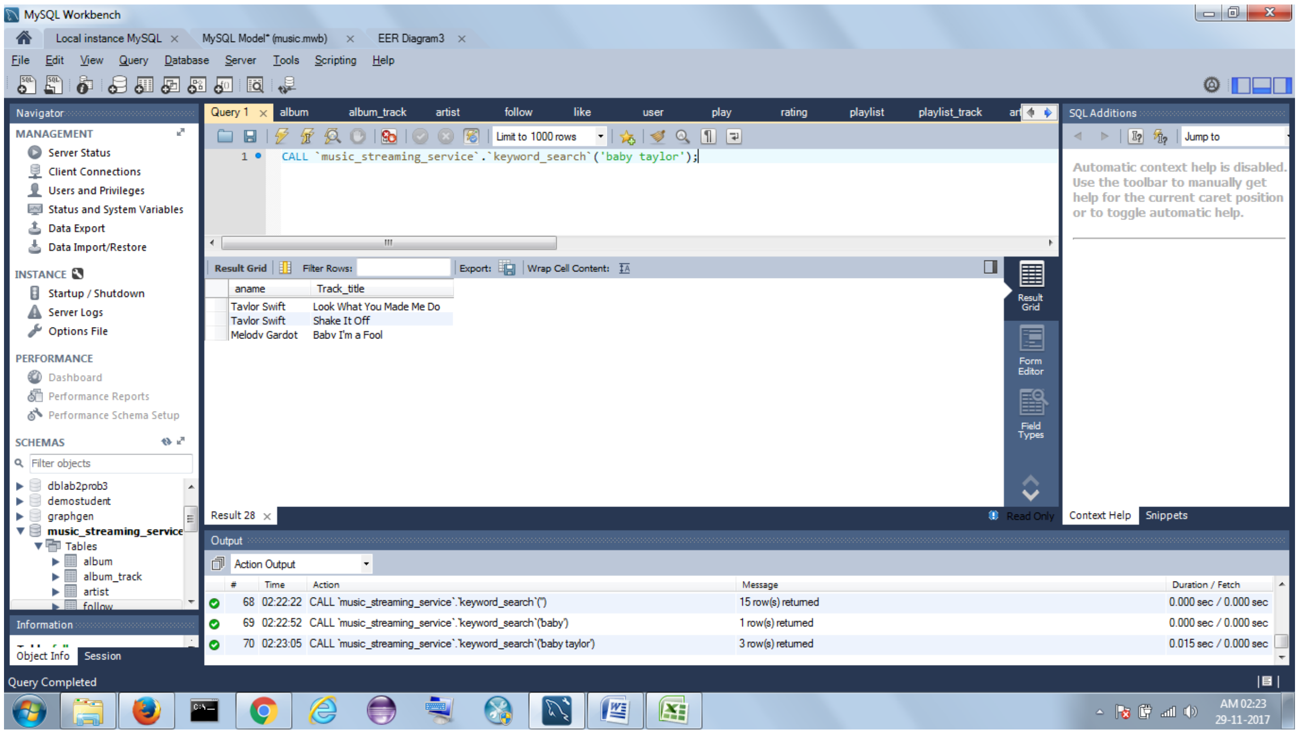
```
aname        Track_title
Bob Marley   No Woman No Cry
```

```
#   Time      Action                                                   Message             Duration / Fetch
64  02:19:11  CALL `music_streaming_service`.`keyword_search`('')      15 row(s) returned  0.000 sec / 0.000 sec
65  02:19:58  CALL `music_streaming_service`.`keyword_search`('no')    1 row(s) returned   0.000 sec / 0.000 sec
66  02:20:06  CALL `music_streaming_service`.`keyword_search`('bob')   1 row(s) returned   0.000 sec / 0.000 sec
```

aname	Track_title
Bob Marley	No Woman No Cry

Test case 3:

CALL `music_streaming_service`.`keyword_search`('baby taylor');



```
CALL `music_streaming_service`.`keyword_search`('baby taylor');
```

```
aname          Track_title
Tavlor Swift   Look What You Made Me Do
Tavlor Swift   Shake It Off
Melodv Gardot  Babv I'm a Fool
```

```
#   Time      Action                                                        Message              Duration / Fetch
68  02:22:22  CALL `music_streaming_service`.`keyword_search`('')           15 row(s) returned   0.000 sec / 0.000 sec
69  02:22:52  CALL `music_streaming_service`.`keyword_search`('baby')       1 row(s) returned    0.000 sec / 0.000 sec
70  02:23:05  CALL `music_streaming_service`.`keyword_search`('baby taylor') 3 row(s) returned   0.015 sec / 0.000 sec
```

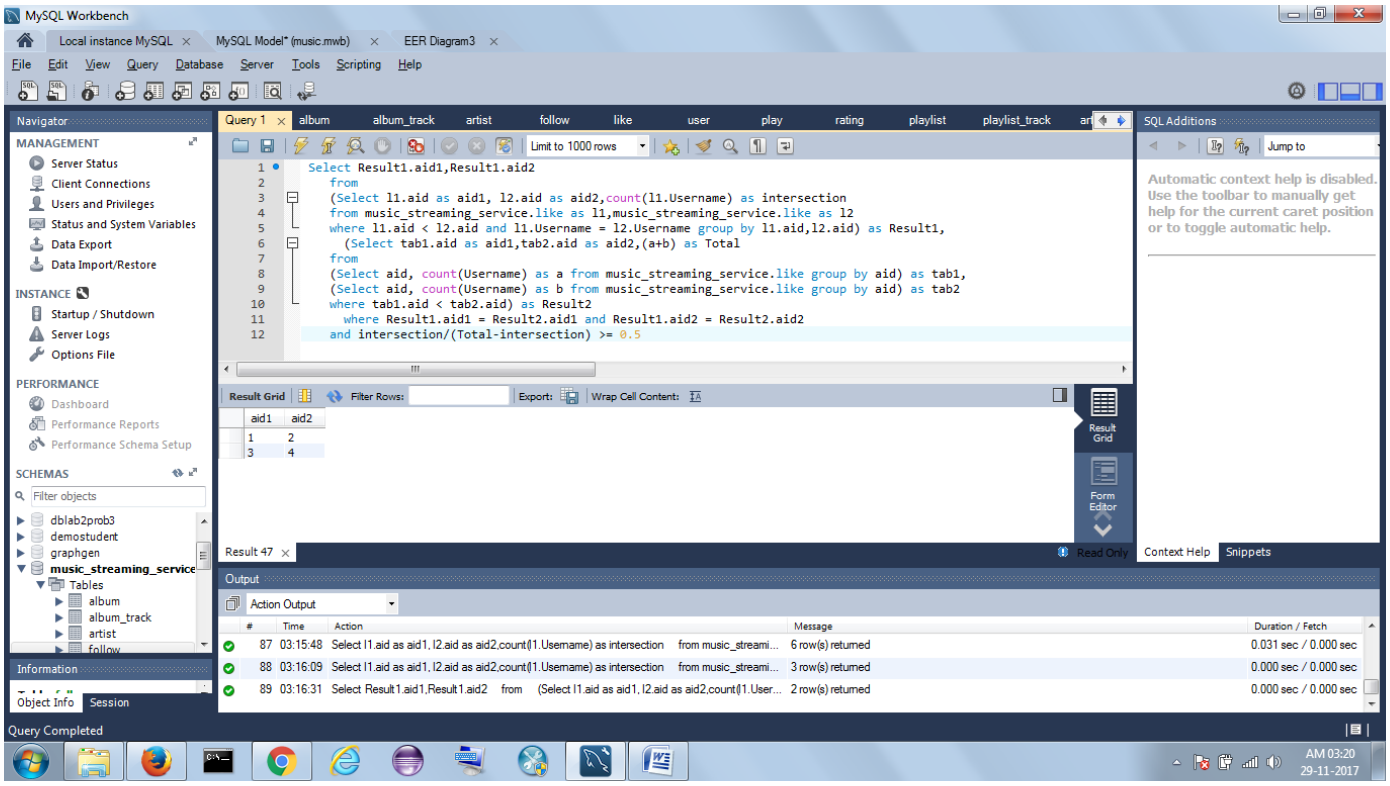
aname	Track_title
Taylor Swift	Look What You Made Me Do
Taylor Swift	Shake It Off
Melody Gardot	Baby I'm a Fool

Find pairs of related artists, where two artists are related if they have many fans in common. (Define this appropriately.)

Ans:

We use Jaccard similarity with a value of 0.5 or greater, which is the size of the intersection of fans(common fans) of artists over the size of the union of the fans of the artists to find pairs of related artists.

```
Select Result1.aid1,Result1.aid2
from
(Select l1.aid as aid1, l2.aid as aid2,count(l1.Username) as intersection
from music_streaming_service.like as l1,music_streaming_service.like as l2
where l1.aid < l2.aid and l1.Username = l2.Username group by l1.aid,l2.aid) as Result1,
(Select tab1.aid as aid1,tab2.aid as aid2,(a+b) as Total
from
(Select aid, count(Username) as a from music_streaming_service.like group by aid) as tab1,
(Select aid, count(Username) as b from music_streaming_service.like group by aid) as tab2
where tab1.aid < tab2.aid) as Result2
where Result1.aid1 = Result2.aid1 and Result1.aid2 = Result2.aid2
and intersection/(Total-intersection) >= 0.5
```



```
Select Result1.aid1,Result1.aid2
  from
  (Select l1.aid as aid1, l2.aid as aid2,count(l1.Username) as intersection
  from music_streaming_service.like as l1,music_streaming_service.like as l2
  where l1.aid < l2.aid and l1.Username = l2.Username group by l1.aid,l2.aid) as Result1,
    (Select tab1.aid as aid1,tab2.aid as aid2,(a+b) as Total
  from
  (Select aid, count(Username) as a from music_streaming_service.like group by aid) as tab1,
  (Select aid, count(Username) as b from music_streaming_service.like group by aid) as tab2
  where tab1.aid < tab2.aid) as Result2
    where Result1.aid1 = Result2.aid1 and Result1.aid2 = Result2.aid2
  and intersection/(Total-intersection) >= 0.5
```

aid1	aid2
1	2
3	4

```
#   Time      Action                                                                                   Message           Duration / Fetch
87  03:15:48  Select l1.aid as aid1, l2.aid as aid2,count(l1.Username) as intersection   from music_streami...   6 row(s) returned   0.031 sec / 0.000 sec
88  03:16:09  Select l1.aid as aid1, l2.aid as aid2,count(l1.Username) as intersection   from music_streami...   3 row(s) returned   0.000 sec / 0.000 sec
89  03:16:31  Select Result1.aid1,Result1.aid2   from   (Select l1.aid as aid1, l2.aid as aid2,count(l1.User...   2 row(s) returned   0.000 sec / 0.000 sec
```

aid1	aid2
1	2
3	4

(d) Populate your database with some sample data, and test the queries you have written in part (c). Make sure to input interesting and meaningful data and to test a number of cases. Limit yourself to a few entries each, but make sure there is enough data to generate interesting test cases. It is suggested that you design your test data very carefully. Show your test data as tables, not as long lists of insert statements, and discuss the structure of the data. Print out and submit your testing.

Dataset:

Album

Album_id	Album_title	Album_date
1	Forever Alone	2017-01-01 00:00:00
2	Unchanied	2017-01-02 00:00:00
3	Live and Learn	2017-02-02 00:00:00
4	Glass House	2017-02-03 00:00:00
5	Blank Canvas	2017-05-05 00:00:00

Album_Track

Album_id	Track_id
1	3
1	4
3	5
5	5
1	6
4	6
3	7
5	7
2	8
2	9

Artist

aid	aname	adesc
1	Taylor Swift	American singer-songwriter
2	Justin Beiber	Young and Upcoming
3	Bob Marley	High on life
4	Eminem	Slim Shady
5	Arijit	Sufi Singer
6	Melody Gardot	Jazz Singer
7	Atif Aslam	Romantic Songs

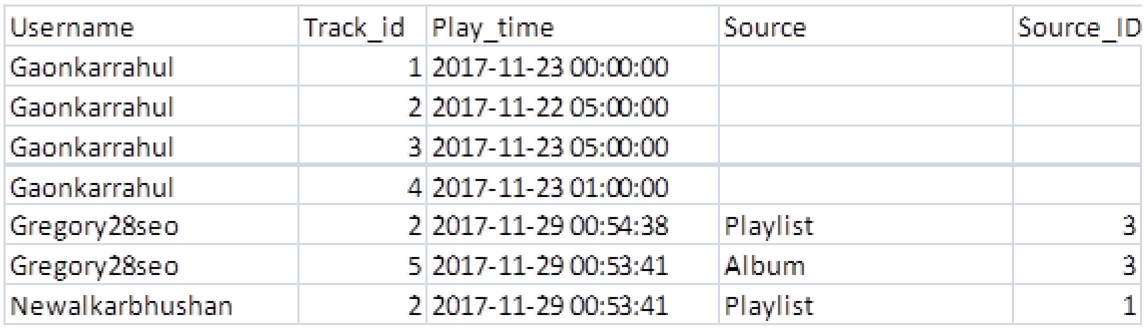
Follow

Username	Following_id	Timestamp
Gaonkarrahul	Gregory28seo	2017-01-21 12:00:00
Gaonkarrahul	Nadikanoop	2017-01-21 13:00:00
Gaonkarrahul	Newalkarbhushan	2017-01-21 14:00:00
Gregory28seo	Newalkarbhushan	2017-01-21 15:00:00
Ranjanrishi	Gaonkarrahul	2016-01-21 15:00:00
Ranjanrishi	Gregory28seo	2017-01-22 15:00:00
Ranjanrishi	Newalkarbhushan	2017-01-21 18:00:00

Like

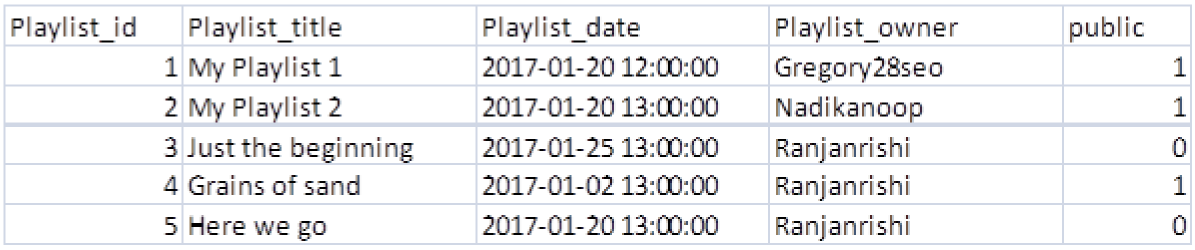
Username	aid	Like_time
Gaonkarrahul	3	2017-11-23 00:00:00
Gaonkarrahul	4	2017-11-23 00:00:00
Gregory28seo	3	2017-11-23 00:00:00
Gregory28seo	4	2017-11-23 00:00:00
Nadikanoop	1	2017-11-23 00:00:00
Nadikanoop	2	2017-11-23 00:00:00
Nadikanoop	3	2017-11-23 00:00:00
Nadikanoop	4	2017-11-23 00:00:00
Newalkarbhushan	1	2017-11-23 00:00:00
Newalkarbhushan	2	2017-11-23 00:00:00
Newalkarbhushan	3	2017-11-23 00:00:00
Ranjanrishi	3	2017-11-23 00:00:00

Play



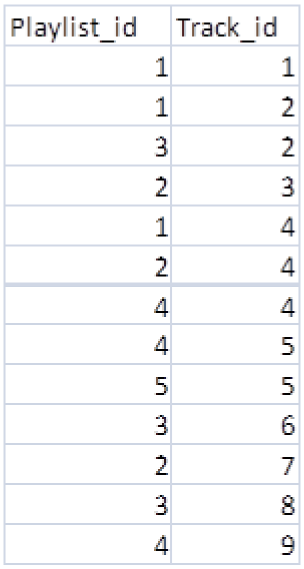
Username	Track_id	Play_time	Source	Source_ID
Gaonkarrahul	1	2017-11-23 00:00:00		
Gaonkarrahul	2	2017-11-22 05:00:00		
Gaonkarrahul	3	2017-11-23 05:00:00		
Gaonkarrahul	4	2017-11-23 01:00:00		
Gregory28seo	2	2017-11-29 00:54:38	Playlist	3
Gregory28seo	5	2017-11-29 00:53:41	Album	3
Newalkarbhushan	2	2017-11-29 00:53:41	Playlist	1

Playlist



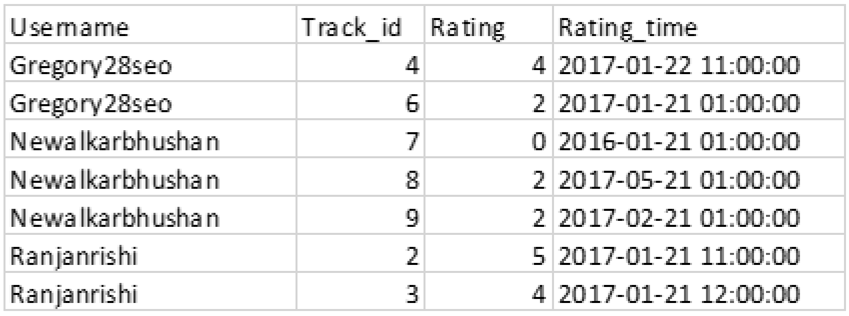
Playlist_id	Playlist_title	Playlist_date	Playlist_owner	public
1	My Playlist 1	2017-01-20 12:00:00	Gregory28seo	1
2	My Playlist 2	2017-01-20 13:00:00	Nadikanoop	1
3	Just the beginning	2017-01-25 13:00:00	Ranjanrishi	0
4	Grains of sand	2017-01-02 13:00:00	Ranjanrishi	1
5	Here we go	2017-01-20 13:00:00	Ranjanrishi	0

Playlist_Track



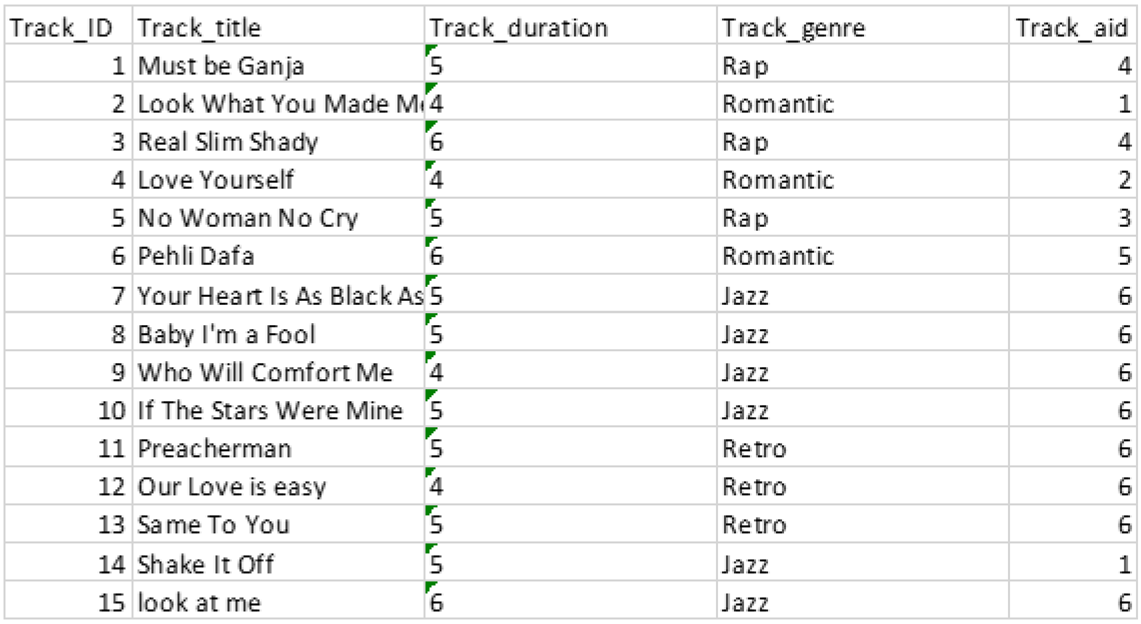
Playlist_id	Track_id
1	1
1	2
3	2
2	3
1	4
2	4
4	4
4	5
5	5
3	6
2	7
3	8
4	9

Rating



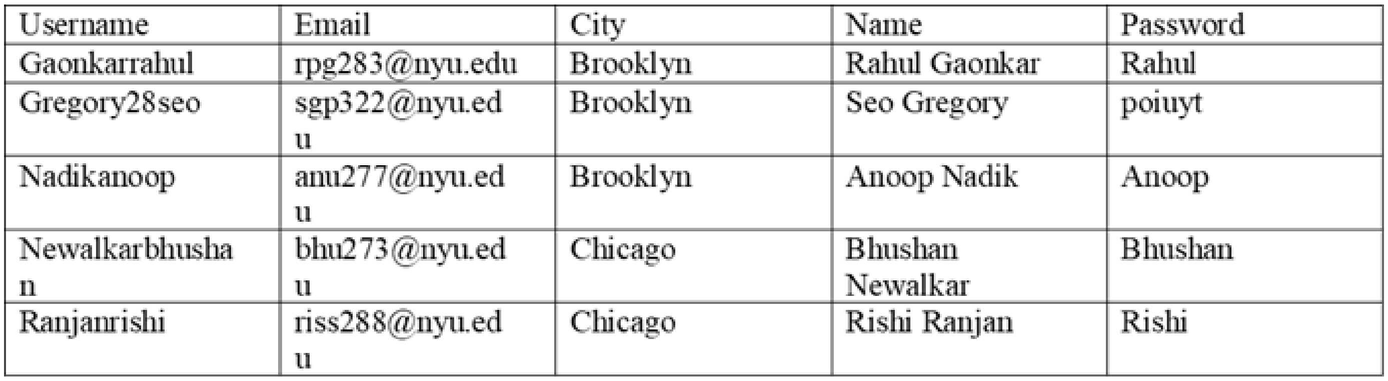
Username	Track_id	Rating	Rating_time
Gregory28seo	4	4	2017-01-22 11:00:00
Gregory28seo	6	2	2017-01-21 01:00:00
Newalkarbhushan	7	0	2016-01-21 01:00:00
Newalkarbhushan	8	2	2017-05-21 01:00:00
Newalkarbhushan	9	2	2017-02-21 01:00:00
Ranjanrishi	2	5	2017-01-21 11:00:00
Ranjanrishi	3	4	2017-01-21 12:00:00

Track



Track_ID	Track_title	Track_duration	Track_genre	Track_aid
1	Must be Ganja	5	Rap	4
2	Look What You Made M	4	Romantic	1
3	Real Slim Shady	6	Rap	4
4	Love Yourself	4	Romantic	2
5	No Woman No Cry	5	Rap	3
6	Pehli Dafa	6	Romantic	5
7	Your Heart Is As Black As	5	Jazz	6
8	Baby I'm a Fool	5	Jazz	6
9	Who Will Comfort Me	4	Jazz	6
10	If The Stars Were Mine	5	Jazz	6
11	Preacherman	5	Retro	6
12	Our Love is easy	4	Retro	6
13	Same To You	5	Retro	6
14	Shake It Off	5	Jazz	1
15	look at me	6	Jazz	6

User



Username	Email	City	Name	Password
Gaonkarrahul	rpg283@nyu.edu	Brooklyn	Rahul Gaonkar	Rahul
Gregory28seo	sgp322@nyu.edu	Brooklyn	Seo Gregory	poiuyt
Nadikanoop	anu277@nyu.edu	Brooklyn	Anoop Nadik	Anoop
Newalkarbhushan	bhu273@nyu.edu	Chicago	Bhushan Newalkar	Bhushan
Ranjanrishi	riss288@nyu.edu	Chicago	Rishi Ranjan	Rishi