
# Computer Science and Engineering

# Online Music Community

## Design Specification-Version 2

- Document Number: Design Specification-002
- Team Members:


| Names | E-mails |
|---|---|
| Shantanu Ranjan | Sr3306@nyu.edu |
| Amit Bhandari | Asb634@nyu.edu |

## Revision History

| Date | Version | Description of Change |
|---|---|---|
| 11/23/2014 | 0.1 | Initial Document |
| 12/8/2014 | 0.2 | Final Document |

# Table of Contents

# 1. INTRODUCTION

## 1.1 Purpose

The purpose of this document is to describe the business needs of creating a software product for a startup company.The goal of this company is to provide online community for fans of live music. The system will be called Online Mus ic community that will provide information about bands and concerts.

The main users of this system will be the registered users, administrators of the company,Artists. Registered users could interact with the system to update their profile, could like particular genre of music,follow other users,post concert information ,could become fan of an artist.Admins could post concert information on behalf of artists by seeking their permission and Artists could post their concert information and could update their profile.

# 2. SCOPE

Online music community will be responsible for registration of users and Artists. Once the user is logged in as registered user he could be able to view his profile, update it,vis it other users profile,follow them,could find bands/artists based on particular genre .He could become fan of a particular artist.The user would also be able to post concert information or other relevant information to the community so that other users could see.He could also create a recommendation list about concerts based on different ratings and reviews so that other user could view it.The user is allowed to post about concerts to the community based on particular trust level.This trust level depends upon the no of ratings and reviews given by this user and also the no of recommended list created by him.

The administrators/company memebers have the privilege to post concert information in behalf of artist which is going to be held.In such case they need to seek permission from artists.

The bands/artist could logged into the system to view their profile and update it.They would be able to post information about their upcoming concerts.They could also list their different genres of music.

The system would create a message feed consisting of upcoming concerts recommended by users he follows or the concerts recommended by system based on users past preferences and ratings.

# 3. REQUIREMENTS

## 3.1 Functional Descriptive Detailed Requirements

Background Information: Online Music community for startup company will accomplish these tasks: a)signup registration page for artist/user,create or update profile,follow other user,become fan of band,post review and rating,store concert information,artist information b)Allow users/artist to post additional information about upcoming concert, c) Generate a list of upcoming concert recommended by different users(Recommender system),d)Browse/Search concert based on different genres,city or based on recommendation list(search mechanism). The following requirements specify a system to accomplish the goals.

1. **Registration Interface:** The system will provide an interface to the user/artist to register and log in, display profile to the users/artist, maintain their information in the system's database and perform different tasks.
   **1.1 Registration interface**
   1.1.1 The system has the option to register as user or as artist by providing their basic information like name, date of birth, email, address, city ,state, weblink, genre etc.
   1.1.2 Once registered he can be directed to user interface or artist interface

2. **Profile Interface:** The system directs user/artist/admin to their respective interface.
   **2.1 Create Profile interface**
   2.1.1The system will accept inputs from user and validate the login credentials of the user. This input is verified with the databases and if correct the system will allow the user to use the system.
   2.1.2 Next the user is directed to create profile page where he can provide his basic information and could like genre, sub-genre, follow artist. Once the profile information is updated the user is directed to user profile page.
   **2.2 User interface**
   **2.2.1User Profile**
   2.2.1.1On user profile page the user is able to post artist information, concert information or add new concert.

2.2.1.2On post artist info tab the user is able to search for particular artist and could post information to public posts which would then be visible to other users who are following him.

2.2.1.3On post concert info tab the user could search for particular concert and then post it under public posts so that other users could view it.

2.2.1.4On Add new concert tab the user could add information about concert which would then be inserted into database.

2.2.1.5The user is allowed to post information if his trust score is greater than 5.

**2.2.2 Recommended Artist**

2.2.2.1Under this tab, user is able to view those artist which are recommended by other users to whom he follows.

**2.2.3Recommended Upcoming Concerts**

2.2.3.1Under this tab, the user is able to view those concerts which are recommended by other users and would be coming up in following days.

**2.2.4Public Rate and Review**

2.2.4.1Under this tab the user is able to view ratings and review of concerts which is given by the users to whom he follows.

**2.2.5Concerts Post by Users**

2.2.5.1Under this tab the user is able to view concert information posted by other users.

**2.2.6Concerts Post by Artist**

2.2.6.1Under this tab the user is able to view concert information posted by artists

**2.2.7Artist Post**

2.2.7.1Under this tab the user is able to view artist information posted by other users.

`

**2.2.8Search User**

2.2.8.1The user could search for other user and visit his profile. He could then follow him and view his recommended list of concerts and posts.

**2.2.9Search Artist/Band**

2.2.9.1The user could search for particular band/artist based on genre or by its name and view their profile.He could then become fan.

**2.2.10Search Concert**

2.2.10.1The user could search for particular concert based on location, time, genre.

**2.2.11Concert,Genre,Artist list**

2.2.11.1The user could view those concerts which he has already attended, view his genre liking, artist list, update profile.

### 2.3Artist interface

2.2.1The system will accept inputs from artist and validate the login credentials of the artist. This input is verified with the databases and if correct the system will allow the artist to use the system.

2.2.3The system will redirect him to create profile page where he could give his basic information.

2.2.3The system will display artist profile information and would have the provision to update it.

2.2.4The system will display artist's genre of music.

2.2.5The system will allow artist to post information about their upcoming concert.

2.26The system will allow artist to view his pat concert information.

2.27The system will allow artist to view ratings and reviews given by other users who are fan of him.

## 2.2 Admin interface(Company's Dashboard)

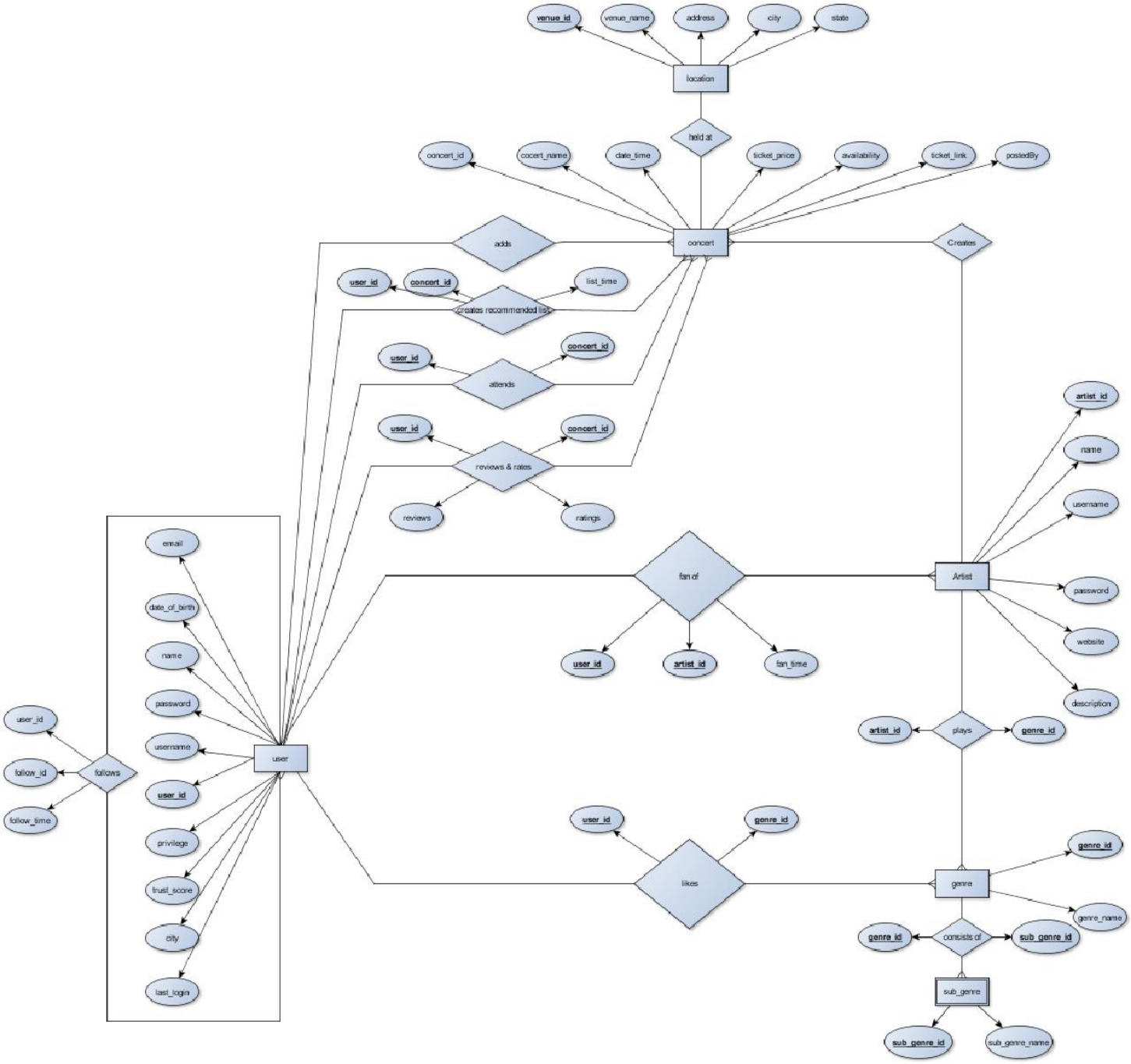
2.3.1 The system will accept inputs from admin and validate the login credentials of the admin. This input is verified with the databases and if correct the system will allow the admin to use the system.

2.3.2The system will allow admin to post concert information of a particular artist by seeking their permission.

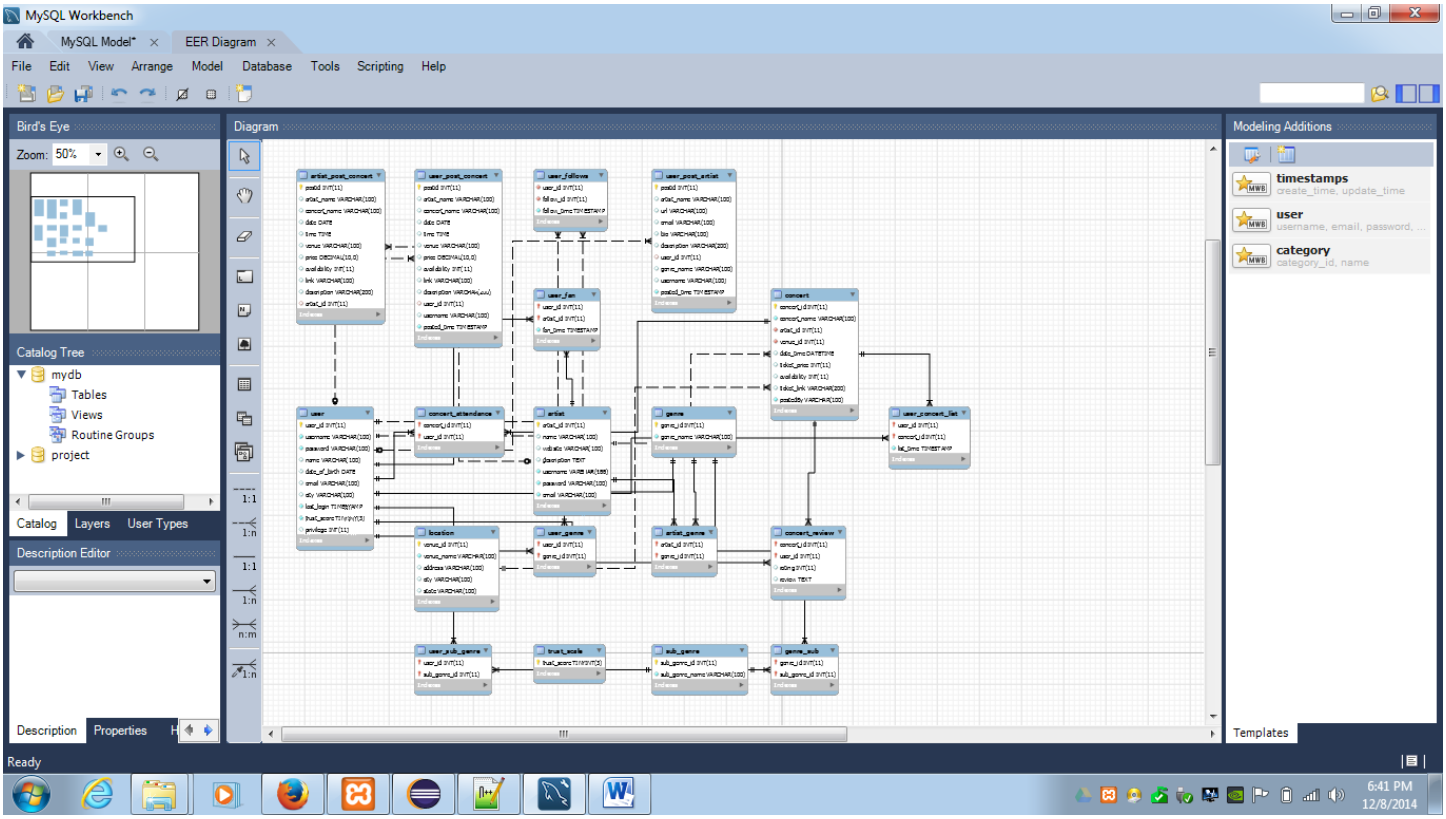
2.3.3The system will allow admin to update its profile.

2.3.4The system will allow admin to search a user or artist and view their profile.

## 3.2 Entity Relationship Diagram



## 3.3 Schema Diagram



# 4. DATABASE ARCHITECTURE
## 4.1 TABLE CREATION LOG:

```
-- phpMyAdmin SQL Dump
-- version 4.2.7.1
-- http://www.phpmyadmin.net
--
-- Host: 127.0.0.1
-- Generation Time: Dec 09, 2014 at 12:43 AM
-- Server version: 5.5.39
-- PHP Version: 5.4.31

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Database: `project`
--

DELIMITER $$
--
-- Procedures
--
CREATE DEFINER=`root`@`localhost` PROCEDURE `add_recommended_concert`(IN userid int,IN
        concertid int)
BEGIN
insert into user_concert_list(user_id,concert_id) values(userid,concertid);
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `artistconcert_rate_review`(IN artistid int(11))
BEGIN
select c.concert_name,r.rating,r.review,u.name from user u
inner join concert_review r on u.user_id=r.user_id
inner join concert c on r.concert_id=c.concert_id
inner join artist a on c.artist_id=a.artist_id
where a.artist_id=8;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `artist_concert_posts`(IN `userid` INT(11))
```

```
select
        ap.concert_name,ap.date,ap.time,ap.venue,ap.price,ap.availability,ap.link,ap.description,ap.artist_id,
        ap.artist_name
from artist_post_concert ap,user_fan uf
where ap.artist_id = uf.artist_id
and uf.user_id = userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `artist_concert_posts_profile`(IN artistid INT(11))
BEGIN
select
        ap.artist_name,ap.concert_name,ap.date,ap.time,ap.venue,ap.price,ap.availability,ap.link,ap.descript
        ion,
a.username from artist_post_concert ap inner join artist a on
ap.artist_id=a.artist_id
where ap.artist_id=artistid;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `artist_past_concert`(IN artistid int(11))
BEGIN
select c.concert_name,v.venue_name,v.city from concert c
inner join location v  on c.venue_id=v.venue_id inner join artist a
on a.artist_id=c.artist_id where c.date_time<curdate()
and a.artist_id=artistid;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `artist_personal_information`(IN `artistid` INT)
   NO SQL
select * from artist where artist_id = artistid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `artist_related_post`(IN userid INT(11))
BEGIN
select
        up.user_id,up.artist_name,up.url,up.email,up.bio,up.description,up.username,up.posted_time,up.ge
        nre_name from user_post_artist up
inner join user_follows uf on up.user_id=uf.follow_id
where uf.user_id=userid order by up.posted_time desc;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `concert_rate_review`(IN `userid` INT(11))
BEGIN
select u.user_id,u.username,r.rating,r.review,r.concert_id,c.concert_name
from concert_review r inner join user_follows uf on r.user_id=uf.follow_id
inner join user u on u.user_id=uf.user_id inner join concert c on
c.concert_id=r.concert_id
where r.user_id=userid;
```

```
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `concert_review`(IN concert_id INT(11),IN user_id
        INT(11),IN rating INT(11),IN review TEXT)
BEGIN
                insert into concert_review values(concert_id,user_id,rating,review);
         END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_artist`(IN `keyword` VARCHAR(100))
   NO SQL
select a.artist_id,a.name,g.genre_name from artist a,artist_genre ag,genre g where a.name like keyword
and ag.artist_id = a.artist_id
and ag.genre_id =g.genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_artist_by_genre`(IN `searchtext`
        VARCHAR(100))
   NO SQL
select a.artist_id,a.name,g.genre_name
from artist a,genre g,artist_genre ag
where g.genre_name = searchtext
and g.genre_id = ag.genre_id
and ag.artist_id = a.artist_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_attended_concert`(IN `userid` INT)
   NO SQL
select c.concert_id,c.concert_name,a.name,date(c.date_time) as date,v.city
from concert c,concert_attendance ca,artist a,location v
where curdate() > c.date_time
and c.concert_id = ca.concert_id
and c.artist_id = a.artist_id
and c.venue_id = v.venue_id
and ca.user_id = userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_concert`(IN `concertid` INT)
   NO SQL
select c.concert_id,c.concert_name,c.artist_id,c.venue_id,date(c.date_time) as date,time(c.date_time) as
        time,c.ticket_price,c.availability,c.ticket_link,a.name,g.genre_name,v.venue_name,v.address,v.city,v.s
        tate
from concert c, artist a,artist_genre ag, genre g,location v
where c.concert_id = concertid
and c.artist_id = a.artist_id
and a.artist_id = ag.artist_id
and ag.genre_id = g.genre_id
and c.venue_id = v.venue_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_concert_by_artist`(IN `searchtext`
        VARCHAR(100))
   NO SQL
```

```
select c.concert_id,c.concert_name,g.genre_name,a.name,date(c.date_time) as date,time(c.date_time) as
        time,v.venue_name
from concert c,artist a,artist_genre ag,genre g,location v
where a.name =searchtext
and a.artist_id =c.artist_id
and a.artist_id = ag.artist_id
and ag.genre_id = g.genre_id
and c.venue_id = v.venue_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_concert_by_date`(IN `searchtext`
        VARCHAR(100))
   NO SQL
select c.concert_id,c.concert_name,g.genre_name,a.name,date(c.date_time) as date,time(c.date_time) as
        time,v.venue_name
from concert c,artist a,artist_genre ag,genre g,location v
where date(c.date_time) = searchtext
and c.artist_id = a.artist_id
and a.artist_id = ag.artist_id
and ag.genre_id = g.genre_id
and c.venue_id = v.venue_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_concert_by_genre`(IN `searchtext`
        VARCHAR(100))
   NO SQL
select c.concert_id,c.concert_name,g.genre_name,a.name,date(c.date_time) as date,time(c.date_time) as
        time,v.venue_name
from concert c,artist a,artist_genre ag,genre g,location v
where g.genre_name = searchtext
and g.genre_id = ag.genre_id
and ag.artist_id = a.artist_id
and a.artist_id = c.artist_id
and c.venue_id = v.venue_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_concert_by_location`(IN `searchtext`
        VARCHAR(100))
   NO SQL
select c.concert_id,c.concert_name,g.genre_name,a.name,date(c.date_time) as date,time(c.date_time) as
        time,v.venue_name
from concert c,artist a,artist_genre ag,genre g,location v
where v.city = searchtext
and v.venue_id = c.venue_id
and c.artist_id = a.artist_id
and a.artist_id = ag.artist_id
and ag.genre_id = g.genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_concert_list`(IN `searchtext` VARCHAR(100))
   NO SQL
```

```
select c.concert_id,c.concert_name,c.artist_id,date(c.date_time) as date,time(c.date_time) as
        time,a.name,g.genre_name,v.venue_name
from concert c, artist a,artist_genre ag, genre g,location v
where c.concert_name = searchtext
and c.artist_id = a.artist_id
and a.artist_id = ag.artist_id
and ag.genre_id = g.genre_id
and c.venue_id = v.venue_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_rate_review`(IN `userid` INT)
   NO SQL
select c.concert_name,a.name as name,date(c.date_time) as date,cr.rating,cr.review
from concert c,artist a,concert_review cr
where cr.user_id = userid
and cr.concert_id = c.concert_id
and c.artist_id = a.artist_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_recommended_concert`(IN `userid` INT)
   NO SQL
select c.concert_id,c.concert_name,a.name,date(c.date_time) as date,v.city
from concert c,user_concert_list ca,artist a,location v
where curdate() > c.date_time
and c.concert_id = ca.concert_id
and c.artist_id = a.artist_id
and c.venue_id = v.venue_id
and ca.user_id = userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_user`(IN `keyword` VARCHAR(100))
   NO SQL
select user_id,name,username from user where name like keyword$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_user_genre`(IN `userid` INT)
   NO SQL
select g.genre_name as genre from genre g,user_genre ug
where ug.user_id = userid
and ug.genre_id = g.genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_user_sub_genre`(IN `userid` INT)
   NO SQL
select g.sub_genre_name as genre
from sub_genre g,user_sub_genre ug
where ug.user_id = userid
and ug.sub_genre_id = g.sub_genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_venue`()
   NO SQL
select venue_id,venue_name from location$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `follow`(IN userid INT(11),IN followid INT(11))
BEGIN
insert into user_follows(user_id,follow_id) values(userid,followid);
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `genre_sub`(IN `genrename` VARCHAR(100))
   NO SQL
select distinct sb.sub_genre_name from sub_genre sb,genre_sub gs,genre g
where g.genre_name = genrename and
g.genre_id = gs.genre_id and gs.sub_genre_id = sb.sub_genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `myArtist`(IN `userid` INT)
   NO SQL
select a.name,a.website,a.description,a.email,g.genre_name from artist a
inner join user_fan uf on uf.artist_id=a.artist_id
inner join user u on uf.user_id=u.user_id
inner join artist_genre ag
on ag.artist_id=a.artist_id
inner join genre g on
g.genre_id=ag.genre_id
where u.user_id=userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `post_concert`(IN concertname VARCHAR(100),IN
        artistid INT(11),IN venueid INT(11),IN date_time datetime,IN price INT(11),IN available
        INT(11),IN tktlink VARCHAR(200))
BEGIN
Declare artistname varchar(100);
                select name into artistname from artist where artist_id=artistid;
                insert into
        concert(concert_name,artist_id,venue_id,date_time,ticket_price,availability,ticket_link,postedBy)
        values(concertname,artistid,venueid,date_time,price,available,tktlink,artistname);
         END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `post_concert_user`(IN userid INT(11),IN
        concertname VARCHAR(100),IN artistid INT(11),IN venueid INT(11),IN concert_datetime
        datetime,IN price INT(11),IN available INT(11),IN tktlink VARCHAR(200))
BEGIN
DECLARE score int(11);
DECLARE newid int(11);
DECLARE tempuser varchar(100);

select trust_score into score from user where user_id=userid;
select username into tempuser from user where user_id=userid;
if score>8 then
insert into
        concert(concert_name,artist_id,venue_id,date_time,ticket_price,availability,ticket_link,postedBy)
        values(concertname,artistid,venueid,concert_datetime,price,available,tktlink,tempuser);
end if;
```

```
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `profile`(IN name VARCHAR(100),IN dob date,IN
        email VARCHAR(100),IN city VARCHAR(100),IN userid int)
BEGIN
                update user set name=name,
                date_of_birth=dob,
                email=email,
                city=city
                where user_id=userid;
        END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `recommend_artist`(IN userid INT(11))
BEGIN
 select artist_id,name,website from artist where artist_id in
(select artist_id from user_fan where user_id in
 (select follow_id from user_follows
where user_id=userid));

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `recommend_concert`(IN userid INT(11))
BEGIN
      DECLARE follow_id INT(11);
      DECLARE concert_id INT(11);
      DECLARE v_finished INTEGER DEFAULT 0;
      DECLARE v_finished_concert INTEGER DEFAULT 0;
      DEClARE followid_cursor CURSOR FOR select follow_id from user_follows where user_id=userid;

   OPEN followid_cursor;
   get_followid: LOOP FETCH followid_cursor INTO follow_id;

 BEGIN
   DEClARE concertid_cursor CURSOR FOR select concert_id from user_concert_list where
         user_id=follow_id;
   OPEN concertid_cursor;
      get_concertid: LOOP FETCH concertid_cursor INTO concert_id;
          select concert_name from concert where concert_id=concert_id and date_time>curdate();
 END LOOP

 get_concertid;
 CLOSE concertid_cursor;
 END;
 END LOOP get_followid;
 CLOSE followid_cursor;
 END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `recommend_concerts`(IN userid INT(11))
```

```
BEGIN
 select c.concert_id,c.concert_name,a.name,c.date_time from concert c
 inner join artist a on c.artist_id=a.artist_id inner join user_concert_list ul
 on ul.concert_id=c.concert_id inner join user_follows uf on uf.follow_id=ul.user_id
 where uf.user_id=userid;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `recommend_genre`(IN userid INT(11),IN genrename
        VARCHAR(100))
BEGIN
         DECLARE follow_id INT(11);
         DECLARE concert_id INT(11);
         DECLARE v_finished INTEGER DEFAULT 0;
         DECLARE v_finished_concert INTEGER DEFAULT 0;

         DEClARE followid_cursor CURSOR FOR select follow_id from user_follows where user_id=userid;
                  DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
                  OPEN followid_cursor;
                           get_followid: LOOP
                                    FETCH followid_cursor INTO follow_id;
                                             IF v_finished = 1 THEN
                                                      LEAVE get_followid;
                                             END IF;
                        BEGIN
                                                      DEClARE concertid_cursor CURSOR FOR select concert_id from
        user_concert_list where user_id=follow_id;
                                                      DECLARE CONTINUE HANDLER FOR NOT FOUND SET
        v_finished_concert = 1;
                                                      OPEN concertid_cursor;
                                                               get_concertid: LOOP
                                                                        FETCH concertid_cursor INTO concert_id;
                                                                                 IF v_finished_concert = 1 THEN
                                                                                          LEAVE get_concertid;
                                                                                 END IF;

                                                                        select c.concert_name,a.name from concert c
        inner join artist_genre a on c.artist_id=a.artist_id
                                                                        inner join user_genre ug on
        ug.genre_id=a.genre_id
                                 where ug.user_id=userid;
                                                                                          END LOOP get_concertid;
                                                      CLOSE concertid_cursor;
                           END;
                                    END LOOP get_followid;
                           CLOSE followid_cursor;
         END$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `search_artist`(IN `keyword` VARCHAR(100))
   NO SQL
select a.artist_id,a.name,a.website,a.email,a.description,g.genre_name from artist a,artist_genre ag,genre g
        where a.name = keyword
and ag.artist_id = a.artist_id
and ag.genre_id =g.genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_artist_profile`(IN artistid int(11))
BEGIN
select a.artist_id,a.name,a.website,a.email,a.description,a.username,g.genre_name from artist a inner join
artist_genre ag on a.artist_id=ag.artist_id inner join genre g on
g.genre_id=ag.genre_id
where a.artist_id=artistid;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_concert`(IN `concertname` VARCHAR(100))
   NO SQL
select c.concert_id,c.concert_name,a.name,date(c.date_time)as date,time(c.date_time) as
        time,c.ticket_price,c.availability,c.ticket_link,v.venue_name
from concert c,artist a,location v
where c.venue_id = v.venue_id and c.artist_id = a.artist_id
and c.concert_name=concertname$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_genre`(IN search_city VARCHAR(100),IN
        genrename VARCHAR(100))
BEGIN
select c.concert_name,a.name,v.venue_name,v.address from concert c inner join artist a on
        c.artist_id=a.artist_id
inner join location v on v.venue_id=c.venue_id inner join artist_genre ag on ag.artist_id=c.artist_id
inner join genre g on ag.genre_id=g.genre_id
where v.city=search_city and g.genre_name=genrename and date(c.date_time)>DATE(current_date() +
        INTERVAL (DAYOFWEEK(current_date())) DAY)
and date(c.date_time)<DATE(current_date() + INTERVAL (8-DAYOFWEEK(current_date())) DAY);
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_newconcert`(IN id INT(11))
BEGIN

         select c.concert_name,a.name,v.address,v.city,c.date_time from concert c inner join artist a on
        c.artist_id=a.artist_id
         inner join location v on c.venue_id=v.venue_id
         where c.date_time>(select last_login  from user where user_id=id);

 END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_recommend`(IN userid INT(11))
BEGIN
```

```
        DECLARE follow_id INT(11);
        DECLARE concert_id INT(11);
        DECLARE v_finished INTEGER DEFAULT 0;
        DEClARE followid_cursor CURSOR FOR select follow_id from user_follows where user_id=userid;

        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

        OPEN followid_cursor;
        get_followid: LOOP
                FETCH followid_cursor INTO follow_id;
                        IF v_finished = 1 THEN
                                LEAVE get_followid;
                        END IF;

                        select concert_id into concert_id from user_concert_list where user_id=follow_id;
                        select c.concert_name,a.name,v.address,v.city,c.date_time from concert c inner join
      artist a on c.artist_id=a.artist_id
                        inner join location v on c.venue_id=v.venue_id
                        where month(c.date_time)=month(current_date())+1;
         END LOOP get_followid;
        CLOSE followid_cursor;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_user`(IN `userid` INT)
   NO SQL
select * from user where user_id = userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_concert_posts`(IN userid INT(11))
BEGIN
select artist_name,concert_name,date,time,venue,price,availability,link,username,description from
        user_post_concert
where user_id in(select follow_id from user_follows where user_id=userid);

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_fan`(IN userid INT(11),IN followid INT(11))
BEGIN
insert into user_fan(user_id,artist_id) values(userid,followid);
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_genre`(IN `userid` INT)
   NO SQL
select distinct g.genre_name from genre g,user_genre ug where ug.genre_id =  g.genre_id and ug.user_id =
        userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_personal_information`(IN `userid` INT)
   NO SQL
select * from user where user_id=userid$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `user_post_artist`(IN userid INT(11))
BEGIN
select user_id,artist_name,url,email,bio,description,username,posted_time,genre_name from
        user_post_artist
where user_id=userid order by posted_time desc;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_post_concert`(IN `userid` INT)
   NO SQL
select * from user_post_concert where user_id =userid order by posted_time desc$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_signup`(IN username VARCHAR(100),IN pass
        varchar(100))
BEGIN
                INSERT INTO user(username,password) VALUES (username,pass);
         END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_sub_genre`(IN `userid` INT)
   NO SQL
select distinct s.sub_genre_name from sub_genre s , user_sub_genre ug where ug.sub_genre_id =
        s.sub_genre_id and ug.user_id=userid$$

DELIMITER ;

-- --------------------------------------------------------

--
-- Table structure for table `artist`
--

CREATE TABLE IF NOT EXISTS `artist` (
`artist_id` int(11) NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `website` varchar(100) DEFAULT NULL,
  `description` text,
  `username` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=42 ;

--
-- Dumping data for table `artist`
--

INSERT INTO `artist` (`artist_id`, `name`, `website`, `description`, `username`, `password`, `email`) VALUES
```

(1, 'the rolling stones', 'www.rollingstones.com', 'Description would be inserted later', 'rollingstone',
'rolllingstone', 'rolllingstone@gmail.com'),
(2, 'anthology', 'www.wirz.de', 'description would be inserted later', 'anthology', 'anthology',
'anthology@gmail.com'),
(3, 'delaney bramllett', 'www.delaneybramlett.com', 'description would be inserted later',
'delaneybramlett', 'delaneybramlett', 'delaneybramlett@yahoo.com'),
(4, 'the black keys', 'www.theblackkeys.com', 'theblackkeys', 'theblackkeys', 'theblackkeys',
'theblackkeys@gmail.com'),
(5, 'ray charles', 'www.raycharles.com', 'description would be inserted later', 'raycharles', 'raycharles',
'raycharles@gmail.com'),
(6, 'steve earles', 'www.steveearles.com', 'description would be inserted later', 'steveearles', 'steveearles',
'steveearles@gmail.com'),
(7, 'tim mcgraw', 'www.timmcgraw.com', 'description would be inserted later', 'timmcgraw', 'timmcgraw',
'timmcgraw@gmail.com'),
(8, 'colt ford', 'www.coltford.com', 'description would be inserted later', 'coltford', 'coltford',
'coltford@gmail.com'),
(9, 'bob dylan', 'www.bobdylan.com', 'description would be inserted later', 'bobdylan', 'bobdylan',
'bobdylan@gmail.com'),
(10, 'waylon jennings', 'www.waylonjennings.com', 'description would be inserted later',
'waylonjennings', 'waylonjennings', 'waylonjennings@gmail.com'),
(11, 'slipknot', 'www.slipknot1.com', 'description would be inserted later', 'slipknot', 'slipknot',
'slipknot@gmail.com'),
(12, 'cannibal corpse', 'www.cannibalcorpse.net', 'description would be inserted later', 'cannibalcorpse',
'cannibalcorpse', 'cannibalcorpse@gmail.com'),
(13, 'Korn', 'www.korn.com', 'description would be inserted later', 'korn', 'korn', 'korn@gmail.com'),
(14, 'metallica', 'www.metallica.com', 'description would be inserted later', 'metallica', 'metallica',
'metallica@gmail.com'),
(15, 'lamb of god', 'www.lamb-of-god.com', 'description would be inserted later', 'lambofgod',
'lambofgod', 'lambofgod@gmail.com'),
(16, 'red hot chilli peppers', 'www.redhotchillipeppers.com', 'description would be inserted later',
'redhotchillipeppers', 'redhotchillipeppers', 'redhotchillipeppers@gmail.com'),
(17, 'guns n roses', 'www.gunsnroses.com', 'description would be inserted later', 'gunsnroses',
'gunsnroses', 'gunsnroses@gmail.com'),
(18, 'pink floyd', 'www.pinkfloyd.com', 'description would be inserted later', 'pinkfloyd', 'pinkfloyd',
'pinkfloyd@gmail.com'),
(19, 'eagles', 'www.eaglesband.com', 'description would be inserted later', 'eagles', 'eagles',
'eagles@gmail.com'),
(20, 'green day', 'www.greenday.com', 'description would be inserted later', 'greenday', 'greenday',
'greenday@gmail.com'),
(21, 'seeeds', 'www.seeedstudio.com', 'description would be inserted later', 'seeeds', 'seeeds',
'seeeds@gmail.com'),
(22, 'sublime', 'www.sublimelbc.com', 'description would be inserted later', 'sublime', 'sublime',
'sublime@gmail.com'),
(23, 'dennis brown', 'www.dennisbrown.com', 'description would be inserted later', 'dennisbrown',
'dennisbrown', 'dennisbrown@gmail.com'),
(24, 'slightly stoopid', 'www.slightlystoopid.com', 'description would be inserted later', 'slightlystoopid',
'slightlystoopid', 'slightlystoopid@gmail.com'),

```
(25, 'lucky', 'www.luckyreggae.com', 'description would be inserted later', 'luckyreggae', 'luckyreggae',
        'luckyreggae@gmail.com'),
(26, 'dicogs', 'www.discogs.com', 'description would be inserted later', 'dicogs', 'dicogs',
        'dicogs@gmail.com'),
(27, 'the prodigy', 'www.theprodigy.com', 'description would be inserted later', 'theprodigy', 'theprodigy',
        'theprodigy@gmail.com'),
(28, 'andy stott', 'www.andystott.com', 'description would be inserted later', 'andystott', 'andystott',
        'andystott@gmail.com'),
(29, 'neal hemphill', 'www.nealhemphill.com', 'description would be inserted later', 'nealhemphill',
        'nealhemphill', 'nealhemphill@gmail.com'),
(30, 'ron flatter', 'www.ronflatter.com', 'description would be inserted later', 'ronflatter', 'ronflatter',
        'ronflatter@gmail.com'),
(31, 'union jack', 'www.unionjack.com', 'description would be inserted later', 'unionjack', 'unionjack',
        'unionjack@gmail.com'),
(32, 'robert young', 'www.robertyoung.com', 'description would be inserted later', 'robertyoung',
        'robertyoung', 'robertyoung@gmail.com'),
(33, 'dryer', 'www.dryer.com', 'description would be inserted later', 'dryer', 'dryer', 'dryer@gmail.com'),
(34, 'neck deep', 'www.neckdeepmusic.com', 'description would be inserted later', 'neckdeep', 'neckdeep',
        'neckdeep@gmail.com'),
(35, 'ranker', 'www.ranker.com', 'description would be inserted later', 'ranker', 'ranker',
        'ranker@gmail.com'),
(36, 'dr dre', 'www.beatsbydre.com', 'description would be inserted later', 'drdre', 'drdre',
        'drdre@gmail.com'),
(37, 'eminem', 'www.eminem.com', 'description would be inserted later', 'eminem', 'eminem',
        'eminem@gmail.com'),
(38, 'haaretz', 'www.haaeretz.com', 'description would be inserted later', 'haaretz', 'haaretz',
        'haaretz@gmail.com'),
(39, 'passalacqua', 'www.passalacqua.com', 'description would be inserted later', 'passalacqua',
        'passalacqua', 'passalacqua@gmail.com'),
(40, 'kanye west', 'www.kanyewest.com', 'description would be inserted later', 'kanyewest', 'kanyewest',
        'kanyewest@gmail.com'),
(41, NULL, NULL, NULL, 'artisttest', 'password', 'artist@artist.com');

-- --------------------------------------------------------

--
-- Table structure for table `artist_genre`
--

CREATE TABLE IF NOT EXISTS `artist_genre` (
  `artist_id` int(11) NOT NULL,
  `genre_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `artist_genre`
--
```

```
INSERT INTO `artist_genre` (`artist_id`, `genre_id`) VALUES
(1, 1),
(2, 1),
(3, 1),
(4, 1),
(5, 1),
(6, 2),
(7, 2),
(8, 2),
(9, 2),
(10, 2),
(11, 3),
(12, 3),
(13, 3),
(14, 3),
(15, 3),
(16, 4),
(17, 4),
(18, 4),
(19, 4),
(20, 4),
(21, 5),
(22, 5),
(23, 5),
(24, 5),
(25, 5),
(26, 6),
(27, 6),
(28, 6),
(29, 6),
(30, 6),
(31, 7),
(32, 7),
(33, 7),
(34, 7),
(35, 7),
(36, 8),
(37, 8),
(38, 8),
(39, 8),
(40, 8);

-- --------------------------------------------------------

--
-- Table structure for table `artist_post_concert`
--
```

```
CREATE TABLE IF NOT EXISTS `artist_post_concert` (
`postid` int(11) NOT NULL,
  `artist_name` varchar(100) DEFAULT NULL,
  `concert_name` varchar(100) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `time` time DEFAULT NULL,
  `venue` varchar(100) DEFAULT NULL,
  `price` decimal(10,0) DEFAULT NULL,
  `availability` int(11) DEFAULT NULL,
  `link` varchar(100) DEFAULT NULL,
  `description` varchar(200) DEFAULT NULL,
  `artist_id` int(11) DEFAULT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;

--
-- Dumping data for table `artist_post_concert`
--

INSERT INTO `artist_post_concert` (`postid`, `artist_name`, `concert_name`, `date`, `time`, `venue`, `price`,
        `availability`, `link`, `description`, `artist_id`) VALUES
(1, 'colt ford', 'Coltford concert', '2014-12-12', '21:00:00', 'New York', '50', 4500, 'www.myticketlink.com',
        'concert information', 8),
(2, 'colt ford', 'Colt post new concert', '2014-12-11', '11:00:00', 'Hard Rock Cafe', '50', 50,
        'www.website.com', 'this is new post', 8),
(3, 'colt ford', 'Test concert', '2014-12-15', '11:00:00', 'Top of The Rock', '50', 40, 'www.website.com',
        'This is test ', 8),
(4, 'colt ford', 'concert 100', '2014-12-12', '21:00:00', 'Ben Hill Griffin Stadium', '34', 340,
        'www.myticket.com', 'posted by artist', 8);

-- --------------------------------------------------------

--
-- Table structure for table `concert`
--

CREATE TABLE IF NOT EXISTS `concert` (
`concert_id` int(11) NOT NULL,
  `concert_name` varchar(100) NOT NULL,
  `artist_id` int(11) NOT NULL,
  `venue_id` int(11) NOT NULL,
  `date_time` datetime DEFAULT NULL,
  `ticket_price` int(11) DEFAULT NULL,
  `availability` int(11) DEFAULT NULL,
  `ticket_link` varchar(200) DEFAULT NULL,
  `postedBy` varchar(100) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=28 ;

--
```

```
-- Dumping data for table `concert`
--

INSERT INTO `concert` (`concert_id`, `concert_name`, `artist_id`, `venue_id`, `date_time`, `ticket_price`,
        `availability`, `ticket_link`, `postedBy`) VALUES
(1, 'Tomorrow Land', 1, 100, '2014-11-29 19:00:00', 40, 125, 'www.myticketbut.com', 'rollingstone'),
(2, 'alipknot', 11, 104, '2014-11-30 20:00:00', 70, 150, 'www.myticketbuy.com', 'slipknot'),
(3, 'firy glaze', 8, 101, '2014-12-05 20:00:00', 50, 200, 'www.myticketbuy.com', 'coltford'),
(4, 'Korn', 13, 102, '2014-12-06 20:00:00', 90, 800, 'www.myticketbuy.com', 'korn'),
(5, 'Sundance', 34, 107, '2014-12-13 20:00:00', 120, 500, 'www.myticketbuy.com', 'neckdeep'),
(6, 'Kanye West', 40, 109, '2014-12-14 20:00:00', 150, 900, 'www.myticketbuy.com', 'kanyewest'),
(7, 'Sunshine', 3, 106, '2014-12-14 20:00:00', 150, 900, 'www.myticketbuy.com', 'delaneybramlett'),
(8, 'Westland', 17, 108, '2014-12-14 20:00:00', 150, 800, 'www.myticketbuy.com', 'gunsnroses'),
(9, 'Wolfsbane', 22, 109, '2014-12-15 20:00:00', 70, 800, 'www.myticketbuy.com', 'sublime'),
(10, 'Moonlight', 27, 107, '2014-12-15 19:00:00', 550, 400, 'www.myticketbuy.com', 'theprodigy'),
(11, 'Mreepy', 11, 107, '2014-12-15 22:00:00', 30, 400, 'www.myticketbuy.com', 'slipknot'),
(12, 'Savana', 34, 108, '2014-12-27 19:00:00', 130, 400, 'www.myticketbuy.com', 'neckdeep'),
(13, 'Horizon', 27, 108, '2014-11-15 19:00:00', 20, 400, 'www.myticketbuy.com', 'the prodigy'),
(15, 'Yankee', 1, 100, '2014-10-06 19:00:00', 30, 300, 'www.myticketbuy.com', 'the rolling stones'),
(16, 'concert', 1, 100, '2014-12-25 00:00:00', 34, 120, 'www.ticket.com', 'admin'),
(18, 'myConcert', 8, 109, '0000-00-00 00:00:00', 50, 450, 'www.myticket.com', 'coltford'),
(22, '', 8, 100, '2014-12-11 21:00:00', 60, 50, 'www.website.com', 'amcol4'),
(23, 'New Conecrt', 8, 100, '2014-12-13 22:00:00', 50, 50, 'www.website.com', 'amcol4'),
(24, 'Colt post new concert', 8, 100, '2014-12-11 11:00:00', 50, 50, 'www.website.com', 'coltford'),
(25, 'Test concert', 8, 103, '2014-12-15 11:00:00', 50, 40, 'www.website.com', 'coltford'),
(26, 'concert for new artist', 37, 108, '2014-12-12 21:00:00', 50, 4500, 'www.myticketlink,.com',
        'amcol4'),
(27, 'concert 100', 8, 105, '2014-12-12 21:00:00', 34, 340, 'www.myticket.com', 'coltford');

-- --------------------------------------------------------

--
-- Table structure for table `concert_attendance`
--

CREATE TABLE IF NOT EXISTS `concert_attendance` (
  `concert_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `concert_attendance`
--

INSERT INTO `concert_attendance` (`concert_id`, `user_id`) VALUES
(1, 2),
(4, 2),
(5, 2),
```

```
(8, 2),
(9, 2),
(16, 2),
(22, 2),
(26, 2);

-- --------------------------------------------------------

--
-- Table structure for table `concert_review`
--

CREATE TABLE IF NOT EXISTS `concert_review` (
  `concert_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `rating` int(11) DEFAULT NULL,
  `review` text
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `concert_review`
--

INSERT INTO `concert_review` (`concert_id`, `user_id`, `rating`, `review`) VALUES
(1, 1, 3, 'thoroughly Enjoyed'),
(1, 2, 4, 'awesome'),
(4, 2, 4, 'dsfsdf');

-- --------------------------------------------------------

--
-- Table structure for table `genre`
--

CREATE TABLE IF NOT EXISTS `genre` (
`genre_id` int(11) NOT NULL,
  `genre_name` varchar(100) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;

--
-- Dumping data for table `genre`
--

INSERT INTO `genre` (`genre_id`, `genre_name`) VALUES
(1, 'blues'),
(2, 'country'),
(3, 'metal'),
(4, 'rock'),
```

```
(5, 'reggae'),
(6, 'techno'),
(7, 'trance'),
(8, 'hip hop');

-- --------------------------------------------------------

--
-- Table structure for table `genre_sub`
--

CREATE TABLE IF NOT EXISTS `genre_sub` (
  `genre_id` int(11) NOT NULL,
  `sub_genre_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `genre_sub`
--

INSERT INTO `genre_sub` (`genre_id`, `sub_genre_id`) VALUES
(1, 1),
(1, 2),
(1, 3),
(1, 4),
(1, 5),
(2, 6),
(2, 7),
(2, 8),
(2, 9),
(2, 10),
(3, 11),
(3, 12),
(3, 13),
(3, 14),
(3, 15),
(4, 16),
(4, 17),
(4, 18),
(4, 19),
(4, 20),
(5, 21),
(5, 22),
(5, 23),
(5, 24),
(5, 25),
(6, 26),
(6, 27),
```

```
(6, 28),
(6, 29),
(6, 30),
(7, 31),
(7, 32),
(7, 33),
(7, 34),
(7, 35),
(8, 36),
(8, 37),
(8, 38),
(8, 39),
(8, 40);

-- --------------------------------------------------------

--
-- Table structure for table `location`
--

CREATE TABLE IF NOT EXISTS `location` (
  `venue_id` int(11) NOT NULL,
  `venue_name` varchar(100) NOT NULL,
  `address` varchar(100) DEFAULT NULL,
  `city` varchar(100) DEFAULT NULL,
  `state` varchar(100) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `location`
--

INSERT INTO `location` (`venue_id`, `venue_name`, `address`, `city`, `state`) VALUES
(100, 'Hard Rock cafe', '645 Americanas', 'New York', 'NY'),
(101, 'Rock World', '345 East 6th 3rd Ave', 'New York', 'NY'),
(102, 'Yankee Stadium', '1 E 161st St, Bronx', 'New York', 'NY'),
(103, 'Top of The Rock', '30 Rockefeller Plaza', 'New York', 'NY'),
(104, 'MetLife Stadium', '1 MetLife Stadium Dr, East Rutherford', 'New Jersey', 'NY'),
(105, 'Ben Hill Griffin Stadium', '157 Gale Lemerand Dr, Gainesville', 'Florida', 'FL'),
(106, 'AT&T Stadium', '1 AT&T Way, Arlington', 'Arlington', 'Texas'),
(107, 'Kezar Stadium', '755 Stanyan Street', 'San Francisco', 'SF'),
(108, 'Koger Center for the Arts', '1051 Greene St', 'Columbia', 'South Carolina'),
(109, 'Bass Concert Hall', '2350 Robert Dedman Dr', 'Austin', 'Texas');

-- --------------------------------------------------------

--
-- Table structure for table `sub_genre`
```

```
--

CREATE TABLE IF NOT EXISTS `sub_genre` (
`sub_genre_id` int(11) NOT NULL,
  `sub_genre_name` varchar(100) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=41 ;

--
-- Dumping data for table `sub_genre`
--

INSERT INTO `sub_genre` (`sub_genre_id`, `sub_genre_name`) VALUES
(1, 'blues rock'),
(2, 'country blues'),
(3, 'Gospel blues'),
(4, 'punk blues'),
(5, 'rhythm and blues'),
(6, 'alternative country'),
(7, 'christian country'),
(8, 'country rap'),
(9, 'country rock'),
(10, 'outlaw country'),
(11, 'alternative metal'),
(12, 'death metal'),
(13, 'nu metal'),
(14, 'thrash metal'),
(15, 'groove metal'),
(16, 'alternative rock'),
(17, 'hard rock'),
(18, 'progressive rock'),
(19, 'soft rock'),
(20, 'punk rock'),
(21, 'dancehall'),
(22, 'reggae fusion'),
(23, 'lovers rock'),
(24, 'reggae rock'),
(25, 'african reggae'),
(26, 'detroit techno'),
(27, 'hardcore techo'),
(28, 'dub techno'),
(29, 'birmingham sound'),
(30, 'minimal techno'),
(31, 'acid trance'),
(32, 'conscious trance'),
(33, 'dream trance'),
(34, 'deep trance'),
(35, 'neo trance'),
(36, 'gangsta rap'),
```

```
(37, 'underground hip hop'),
(38, 'jewish hip hop'),
(39, 'detroit hip hop'),
(40, 'alternative hip hop');

-- --------------------------------------------------------

--
-- Table structure for table `trust_scale`
--

CREATE TABLE IF NOT EXISTS `trust_scale` (
  `trust_score` tinyint(3) unsigned NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `trust_scale`
--

INSERT INTO `trust_scale` (`trust_score`) VALUES
(1),
(2),
(3),
(4),
(5),
(6),
(7),
(8),
(9),
(10);

-- --------------------------------------------------------

--
-- Table structure for table `user`
--

CREATE TABLE IF NOT EXISTS `user` (
`user_id` int(11) NOT NULL,
  `username` varchar(100) NOT NULL,
  `password` varchar(100) NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `date_of_birth` date DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `city` varchar(100) DEFAULT NULL,
  `last_login` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        CURRENT_TIMESTAMP,
  `trust_score` tinyint(3) unsigned NOT NULL DEFAULT '1',
```

```
  `privilege` int(11) DEFAULT '0'
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=11 ;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`user_id`, `username`, `password`, `name`, `date_of_birth`, `email`, `city`, `last_login`,
        `trust_score`, `privilege`) VALUES
(1, 'shantanu.ranjan1', 'shan@61189', 'Shantanu Ranjan', '1989-11-06', 'shantanuranjan3@gmail.com',
        'New York', '2014-11-23 23:15:10', 1, 0),
(2, 'amcol4', 'amb@345', 'Amit Bhandari', '1989-04-06', 'amcool4@gmail.com', 'New York', '2014-12-08
        21:24:01', 3, 0),
(3, 'hdk345', 'pass234', 'Hardik Gohil', '1990-02-06', 'hardik.gohil4@gmail.com', 'Texas', '2014-11-23
        23:15:10', 1, 0),
(4, 'Raj345', 'password', 'Raj Kumar', '1988-04-03', 'rajkumar@gmail.com', 'New Jersey', '2014-11-23
        23:15:10', 1, 0),
(5, 'Jai', 'jafg45', 'Jai Sharma', '1988-03-13', 'jaisharma@gmail.com', 'Connecticut', '2014-11-23 23:15:10',
        1, 0),
(7, 'shantanu', 'ranjan', 'name', '1989-11-06', 'shantanu@gmail.com', 'New York', '2014-11-24 08:04:38',
        1, 0),
(8, 'admin', 'admin', 'admin', '1988-06-04', 'admin@onlinemusiccompany.com', 'new york', '2014-11-24
        08:20:25', 10, 1),
(9, 'amcool4u', 'mypass', NULL, NULL, 'amcool4u@gmail.com', NULL, '2014-11-30 00:42:19', 1, 0),
(10, 'username', 'pass', 'name', '1988-06-04', 'asb@asb.com', 'mumbai', '2014-12-05 23:24:34', 1, 0);

-- --------------------------------------------------------

--
-- Table structure for table `user_concert_list`
--

CREATE TABLE IF NOT EXISTS `user_concert_list` (
  `user_id` int(11) NOT NULL,
  `concert_id` int(11) NOT NULL,
  `list_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `user_concert_list`
--

INSERT INTO `user_concert_list` (`user_id`, `concert_id`, `list_time`) VALUES
(1, 11, '2014-12-06 23:11:56'),
(1, 12, '2014-12-06 23:11:56'),
(2, 2, '2014-11-24 07:40:08'),
(2, 3, '2014-11-24 07:40:08'),
(2, 7, '2014-12-08 22:44:17'),
```

```
(2, 15, '2014-12-08 22:43:40'),
(2, 16, '2014-12-08 03:33:21'),
(2, 22, '2014-12-08 21:39:38'),
(2, 24, '2014-12-08 22:42:52'),
(2, 25, '2014-12-08 22:44:36'),
(2, 26, '2014-12-08 21:42:16'),
(5, 5, '2014-11-24 07:40:08'),
(5, 7, '2014-11-24 07:40:09');

-- --------------------------------------------------------

--
-- Table structure for table `user_fan`
--

CREATE TABLE IF NOT EXISTS `user_fan` (
  `user_id` int(11) NOT NULL,
  `artist_id` int(11) NOT NULL DEFAULT '0',
  `fan_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `user_fan`
--

INSERT INTO `user_fan` (`user_id`, `artist_id`, `fan_time`) VALUES
(1, 1, '2014-11-23 23:31:02'),
(1, 2, '2014-12-08 17:33:45'),
(1, 5, '2014-12-08 17:43:28'),
(1, 6, '2014-12-08 17:34:21'),
(1, 8, '2014-12-08 17:33:30'),
(1, 11, '2014-11-23 23:31:02'),
(1, 13, '2014-12-08 17:34:03'),
(1, 14, '2014-12-08 17:34:13'),
(1, 16, '2014-11-24 08:23:48'),
(2, 2, '2014-12-06 21:25:02'),
(2, 3, '2014-12-06 21:32:03'),
(2, 4, '2014-12-06 21:21:01'),
(2, 5, '2014-12-07 09:06:14'),
(2, 8, '2014-12-08 19:13:39'),
(3, 34, '2014-11-23 23:31:02'),
(3, 40, '2014-11-23 23:31:02'),
(4, 3, '2014-11-23 23:31:02'),
(4, 17, '2014-11-23 23:31:02'),
(5, 22, '2014-11-23 23:31:02'),
(5, 27, '2014-11-23 23:31:02'),
(7, 1, '2014-12-08 22:51:11'),
(7, 2, '2014-12-08 22:51:13'),
```

```
(7, 8, '2014-12-08 22:51:23'),
(7, 9, '2014-12-08 22:51:03'),
(7, 18, '2014-12-08 22:51:04'),
(7, 30, '2014-12-08 22:51:05');

-- --------------------------------------------------------

--
-- Table structure for table `user_follows`
--

CREATE TABLE IF NOT EXISTS `user_follows` (
  `user_id` int(11) NOT NULL,
  `follow_id` int(11) NOT NULL,
  `follow_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `user_follows`
--

INSERT INTO `user_follows` (`user_id`, `follow_id`, `follow_time`) VALUES
(1, 3, '2014-11-23 23:34:35'),
(1, 2, '2014-11-23 23:34:35'),
(1, 5, '2014-11-23 23:34:35'),
(2, 1, '2014-11-23 23:34:35'),
(2, 3, '2014-11-23 23:34:35'),
(3, 2, '2014-11-23 23:34:35'),
(3, 5, '2014-11-23 23:34:35'),
(4, 5, '2014-11-23 23:34:35'),
(4, 1, '2014-11-23 23:34:35'),
(5, 2, '2014-11-23 23:34:35'),
(5, 3, '2014-11-23 23:34:35'),
(1, 7, '2014-11-24 08:13:30'),
(2, 4, '2014-12-08 06:17:57'),
(2, 5, '2014-12-08 06:18:55');

-- --------------------------------------------------------

--
-- Table structure for table `user_genre`
--

CREATE TABLE IF NOT EXISTS `user_genre` (
  `user_id` int(11) NOT NULL,
  `genre_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
-- Dumping data for table `user_genre`
--

INSERT INTO `user_genre` (`user_id`, `genre_id`) VALUES
(1, 1),
(2, 1),
(4, 1),
(5, 1),
(2, 2),
(5, 2),
(2, 3),
(7, 3),
(2, 4),
(4, 4),
(7, 4),
(1, 5),
(2, 5),
(4, 5),
(7, 5),
(1, 6),
(3, 6),
(5, 6),
(3, 7),
(3, 8);

-- --------------------------------------------------------

--
-- Table structure for table `user_post_artist`
--

CREATE TABLE IF NOT EXISTS `user_post_artist` (
`postid` int(11) NOT NULL,
  `artist_name` varchar(100) DEFAULT NULL,
  `url` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `bio` varchar(100) DEFAULT NULL,
  `description` varchar(200) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `genre_name` varchar(100) DEFAULT NULL,
  `username` varchar(100) DEFAULT NULL,
  `posted_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        CURRENT_TIMESTAMP
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;

--
```

```
-- Dumping data for table `user_post_artist`
--

INSERT INTO `user_post_artist` (`postid`, `artist_name`, `url`, `email`, `bio`, `description`, `user_id`,
        `genre_name`, `username`, `posted_time`) VALUES
(1, 'anthology', 'www.wirz.de', 'anthology@gmail.com', 'description would be inserted later', 'maa chuda',
        2, 'blues', 'amcol4', '2014-12-07 07:16:22'),
(2, 'colt ford', 'www.coltford.com', 'coltford@gmail.com', 'description would be inserted later', 'lavde', 2,
        'country', 'amcol4', '2014-12-07 07:17:18'),
(3, 'colt ford', 'www.coltford.com', 'coltford@gmail.com', 'description would be inserted later', 'He is
        awesome', 2, 'country', 'amcol4', '2014-12-07 09:07:12'),
(4, 'slipknot', 'www.slipknot1.com', 'slipknot@gmail.com', 'description would be inserted later', 'post
        information about slipknot', 2, 'metal', 'amcol4', '2014-12-07 17:07:13'),
(5, 'slipknot', 'www.slipknot1.com', 'slipknot@gmail.com', 'description would be inserted later', 'shantanu
        ranjan post', 1, 'metal', 'shantanu.ranjan1', '2014-12-07 17:18:03'),
(6, 'anthology', 'www.wirz.de', 'anthology@gmail.com', 'description would be inserted later', 'This is
        awesome', 2, 'blues', 'amcol4', '2014-12-08 02:55:01'),
(7, 'anthology', 'www.wirz.de', 'anthology@gmail.com', 'description would be inserted later', 'some
        information', 1, 'blues', 'shantanu.ranjan1', '2014-12-08 16:31:14'),
(8, 'eminem', 'www.eminem.com', 'eminem@gmail.com', 'description would be inserted later', 'posting
        eminem information', 2, 'hip hop', 'amcol4', '2014-12-08 21:29:26');

-- --------------------------------------------------------

--
-- Table structure for table `user_post_concert`
--

CREATE TABLE IF NOT EXISTS `user_post_concert` (
`postid` int(11) NOT NULL,
  `artist_name` varchar(100) DEFAULT NULL,
  `concert_name` varchar(100) DEFAULT NULL,
  `date` date DEFAULT NULL,
  `time` time DEFAULT NULL,
  `venue` varchar(100) DEFAULT NULL,
  `price` decimal(10,0) DEFAULT NULL,
  `availability` int(11) DEFAULT NULL,
  `link` varchar(100) DEFAULT NULL,
  `description` varchar(200) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `username` varchar(100) DEFAULT NULL,
  `posted_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        CURRENT_TIMESTAMP
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

--
-- Dumping data for table `user_post_concert`
--
```

```
INSERT INTO `user_post_concert` (`postid`, `artist_name`, `concert_name`, `date`, `time`, `venue`, `price`,
        `availability`, `link`, `description`, `user_id`, `username`, `posted_time`) VALUES
(1, 'the rolling stones', 'Tomorrow Land', '2014-11-29', '19:00:00', 'Hard Rock cafe', '40', 125,
        'www.myticketbut.com', 'This is my first try ', 2, 'amcol4', '2014-12-07 09:02:52'),
(2, 'the rolling stones', 'Tomorrow Land', '2014-11-29', '19:00:00', 'Hard Rock cafe', '40', 125,
        'www.myticketbut.com', 'concert is awesome', 2, 'amcol4', '2014-12-07 09:08:17'),
(3, 'slipknot', 'alipknot', '2014-11-30', '20:00:00', 'MetLife Stadium', '70', 150, 'www.myticketbuy.com',
        'shantanu concert post', 1, 'shantanu.ranjan1', '2014-12-07 17:18:31'),
(4, 'the rolling stones', 'Tomorrow Land', '2014-11-29', '19:00:00', 'Hard Rock cafe', '40', 125,
        'www.myticketbut.com', 'This is awesome', 2, 'amcol4', '2014-12-08 02:55:32'),
(5, 'the rolling stones', 'Tomorrow Land', '2014-11-29', '19:00:00', 'Hard Rock cafe', '40', 125,
        'www.myticketbut.com', 'concert info', 1, 'shantanu.ranjan1', '2014-12-08 16:31:34'),
(6, 'neck deep', 'Sundance', '2014-12-13', '20:00:00', 'Kezar Stadium', '120', 500, 'www.myticketbuy.com',
        '', 2, 'amcol4', '2014-12-08 21:30:16');

-- --------------------------------------------------------

--
-- Table structure for table `user_sub_genre`
--

CREATE TABLE IF NOT EXISTS `user_sub_genre` (
  `user_id` int(11) NOT NULL,
  `sub_genre_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `user_sub_genre`
--

INSERT INTO `user_sub_genre` (`user_id`, `sub_genre_id`) VALUES
(1, 1),
(4, 1),
(5, 1),
(5, 6),
(2, 7),
(2, 8),
(2, 12),
(2, 13),
(2, 14),
(4, 16),
(2, 17),
(2, 18),
(2, 22),
(2, 23),
(1, 24),
(4, 25),
```

```
(1, 26),
(1, 27),
(3, 27),
(5, 30),
(3, 33),
(3, 40);

--
-- Indexes for dumped tables
--

--
-- Indexes for table `artist`
--
ALTER TABLE `artist`
 ADD PRIMARY KEY (`artist_id`);

--
-- Indexes for table `artist_genre`
--
ALTER TABLE `artist_genre`
 ADD PRIMARY KEY (`artist_id`,`genre_id`), ADD KEY `genre_id` (`genre_id`);

--
-- Indexes for table `artist_post_concert`
--
ALTER TABLE `artist_post_concert`
 ADD PRIMARY KEY (`postid`), ADD KEY `artist_id` (`artist_id`);

--
-- Indexes for table `concert`
--
ALTER TABLE `concert`
 ADD PRIMARY KEY (`concert_id`), ADD KEY `artist_id` (`artist_id`), ADD KEY `venue_id` (`venue_id`);

--
-- Indexes for table `concert_attendance`
--
ALTER TABLE `concert_attendance`
 ADD PRIMARY KEY (`concert_id`,`user_id`), ADD KEY `user_id` (`user_id`);

--
-- Indexes for table `concert_review`
--
ALTER TABLE `concert_review`
 ADD PRIMARY KEY (`concert_id`,`user_id`), ADD KEY `user_id` (`user_id`);

--
```

```
-- Indexes for table `genre`
--
ALTER TABLE `genre`
 ADD PRIMARY KEY (`genre_id`);

--
-- Indexes for table `genre_sub`
--
ALTER TABLE `genre_sub`
 ADD PRIMARY KEY (`genre_id`,`sub_genre_id`), ADD KEY `sub_genre_id` (`sub_genre_id`);

--
-- Indexes for table `location`
--
ALTER TABLE `location`
 ADD PRIMARY KEY (`venue_id`);

--
-- Indexes for table `sub_genre`
--
ALTER TABLE `sub_genre`
 ADD PRIMARY KEY (`sub_genre_id`);

--
-- Indexes for table `trust_scale`
--
ALTER TABLE `trust_scale`
 ADD PRIMARY KEY (`trust_score`);

--
-- Indexes for table `user`
--
ALTER TABLE `user`
 ADD PRIMARY KEY (`user_id`);

--
-- Indexes for table `user_concert_list`
--
ALTER TABLE `user_concert_list`
 ADD PRIMARY KEY (`user_id`,`concert_id`), ADD KEY `concert_id` (`concert_id`);

--
-- Indexes for table `user_fan`
--
ALTER TABLE `user_fan`
 ADD PRIMARY KEY (`user_id`,`artist_id`), ADD KEY `artist_id` (`artist_id`);

--
```

```
-- Indexes for table `user_follows`
--
ALTER TABLE `user_follows`
 ADD KEY `user_id` (`user_id`), ADD KEY `follow_id` (`follow_id`);

--
-- Indexes for table `user_genre`
--
ALTER TABLE `user_genre`
 ADD PRIMARY KEY (`user_id`,`genre_id`), ADD KEY `genre_id` (`genre_id`);

--
-- Indexes for table `user_post_artist`
--
ALTER TABLE `user_post_artist`
 ADD PRIMARY KEY (`postid`), ADD KEY `user_id` (`user_id`);

--
-- Indexes for table `user_post_concert`
--
ALTER TABLE `user_post_concert`
 ADD PRIMARY KEY (`postid`), ADD KEY `user_id` (`user_id`);

--
-- Indexes for table `user_sub_genre`
--
ALTER TABLE `user_sub_genre`
 ADD PRIMARY KEY (`user_id`,`sub_genre_id`), ADD KEY `user_subgenre_ibfk_2` (`sub_genre_id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `artist`
--
ALTER TABLE `artist`
MODIFY `artist_id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=42;
--
-- AUTO_INCREMENT for table `artist_post_concert`
--
ALTER TABLE `artist_post_concert`
MODIFY `postid` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=5;
--
-- AUTO_INCREMENT for table `concert`
--
ALTER TABLE `concert`
MODIFY `concert_id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=28;
```

```
--
-- AUTO_INCREMENT for table `genre`
--
ALTER TABLE `genre`
MODIFY `genre_id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=9;
--
-- AUTO_INCREMENT for table `sub_genre`
--
ALTER TABLE `sub_genre`
MODIFY `sub_genre_id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=41;
--
-- AUTO_INCREMENT for table `user`
--
ALTER TABLE `user`
MODIFY `user_id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=11;
--
-- AUTO_INCREMENT for table `user_post_artist`
--
ALTER TABLE `user_post_artist`
MODIFY `postid` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=9;
--
-- AUTO_INCREMENT for table `user_post_concert`
--
ALTER TABLE `user_post_concert`
MODIFY `postid` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=7;
--
-- Constraints for dumped tables
--


--
-- Constraints for table `artist_genre`
--
ALTER TABLE `artist_genre`
ADD CONSTRAINT `artist_genre_ibfk_1` FOREIGN KEY (`artist_id`) REFERENCES `artist` (`artist_id`),
ADD CONSTRAINT `artist_genre_ibfk_2` FOREIGN KEY (`genre_id`) REFERENCES `genre` (`genre_id`);

--
-- Constraints for table `artist_post_concert`
--
ALTER TABLE `artist_post_concert`
ADD CONSTRAINT `artist_post_concert_ibfk_1` FOREIGN KEY (`artist_id`) REFERENCES `artist`
        (`artist_id`);

--
-- Constraints for table `concert`
--
ALTER TABLE `concert`
ADD CONSTRAINT `concert_ibfk_1` FOREIGN KEY (`artist_id`) REFERENCES `artist` (`artist_id`),
```

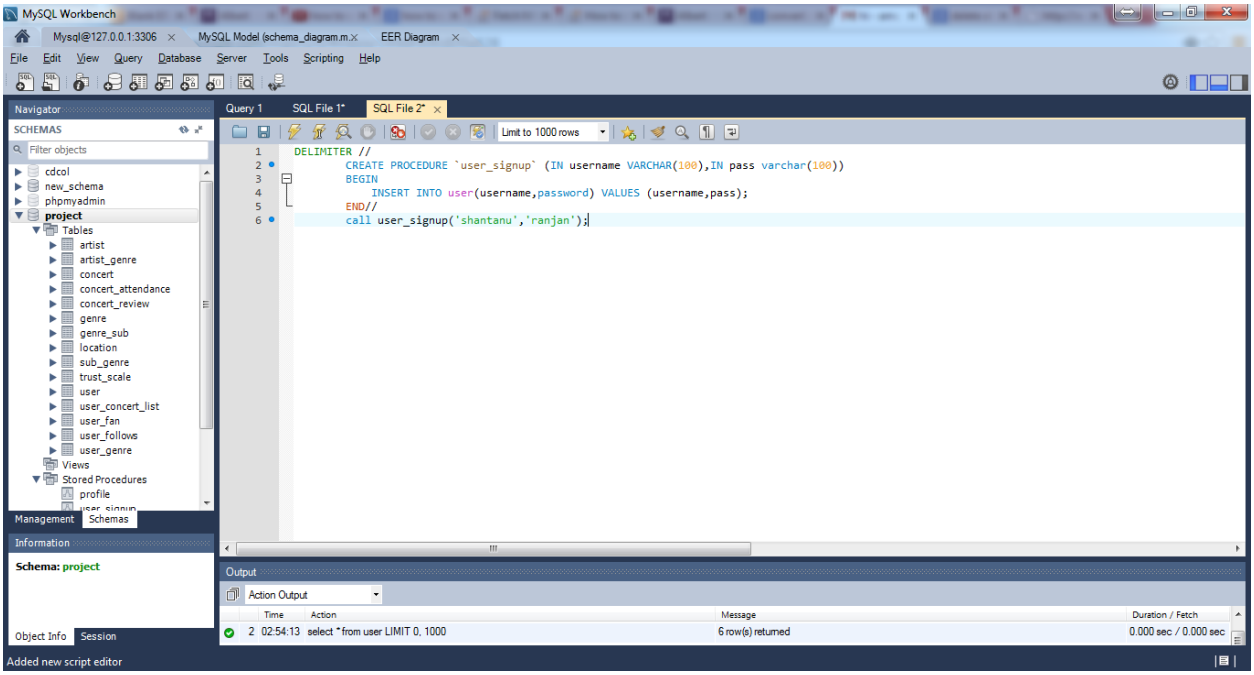
```
ADD CONSTRAINT `concert_ibfk_2` FOREIGN KEY (`venue_id`) REFERENCES `location` (`venue_id`);

--
-- Constraints for table `concert_attendance`
--
ALTER TABLE `concert_attendance`
ADD CONSTRAINT `concert_attendance_ibfk_1` FOREIGN KEY (`concert_id`) REFERENCES `concert`
        (`concert_id`),
ADD CONSTRAINT `concert_attendance_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`);

--
-- Constraints for table `concert_review`
--
ALTER TABLE `concert_review`
ADD CONSTRAINT `concert_review_ibfk_1` FOREIGN KEY (`concert_id`) REFERENCES `concert`
        (`concert_id`),
ADD CONSTRAINT `concert_review_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`);

--
-- Constraints for table `genre_sub`
--
ALTER TABLE `genre_sub`
ADD CONSTRAINT `genre_sub_ibfk_1` FOREIGN KEY (`genre_id`) REFERENCES `genre` (`genre_id`),
ADD CONSTRAINT `genre_sub_ibfk_2` FOREIGN KEY (`sub_genre_id`) REFERENCES `sub_genre`
        (`sub_genre_id`);

--
-- Constraints for table `user_concert_list`
--
ALTER TABLE `user_concert_list`
ADD CONSTRAINT `user_concert_list_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`),
ADD CONSTRAINT `user_concert_list_ibfk_2` FOREIGN KEY (`concert_id`) REFERENCES `concert`
        (`concert_id`);

--
-- Constraints for table `user_fan`
--
ALTER TABLE `user_fan`
ADD CONSTRAINT `user_fan_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`),
ADD CONSTRAINT `user_fan_ibfk_2` FOREIGN KEY (`artist_id`) REFERENCES `artist` (`artist_id`);

--
-- Constraints for table `user_follows`
--
ALTER TABLE `user_follows`
ADD CONSTRAINT `user_follows_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`),
ADD CONSTRAINT `user_follows_ibfk_2` FOREIGN KEY (`follow_id`) REFERENCES `user` (`user_id`);
```

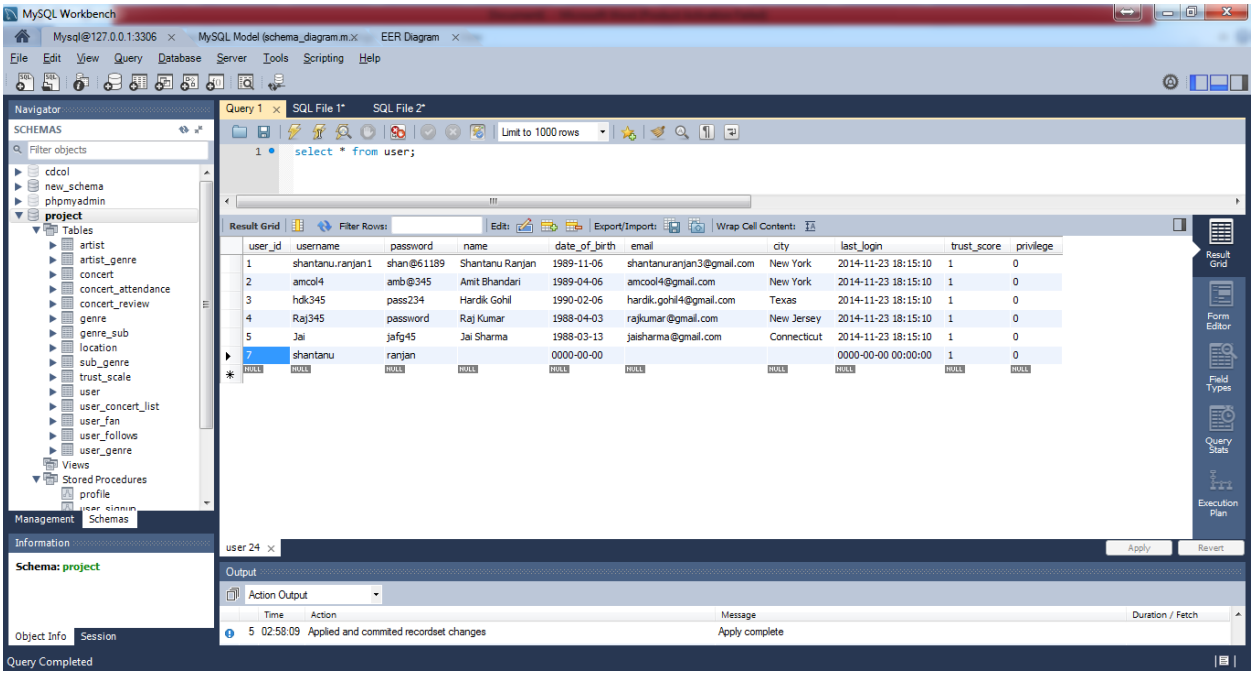
```
--
-- Constraints for table `user_genre`
--
ALTER TABLE `user_genre`
ADD CONSTRAINT `user_genre_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`),
ADD CONSTRAINT `user_genre_ibfk_2` FOREIGN KEY (`genre_id`) REFERENCES `genre` (`genre_id`);

--
-- Constraints for table `user_post_artist`
--
ALTER TABLE `user_post_artist`
ADD CONSTRAINT `user_post_artist_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`);

--
-- Constraints for table `user_post_concert`
--
ALTER TABLE `user_post_concert`
ADD CONSTRAINT `user_post_concert_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`);

--
-- Constraints for table `user_sub_genre`
--
ALTER TABLE `user_sub_genre`
ADD CONSTRAINT `user_subgenre_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`user_id`),
ADD CONSTRAINT `user_subgenre_ibfk_2` FOREIGN KEY (`sub_genre_id`) REFERENCES `sub_genre`
        (`sub_genre_id`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

# 4.2 QUERIES AND TEST CASES

### 4.1.1 USER SIGN UP:

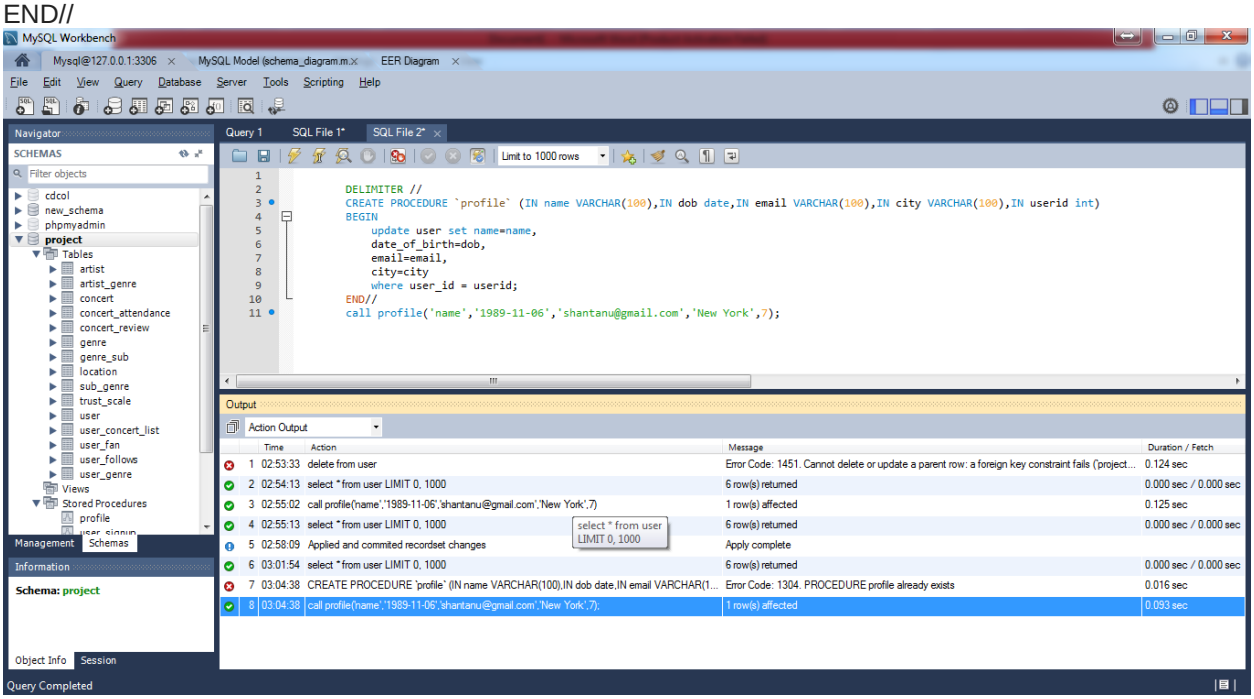
```
DELIMITER //
CREATE PROCEDURE `user_signup` (IN username VARCHAR(100),IN pass varchar(100))
BEGIN
INSERT INTO user(username,password) VALUES (username,pass);
END//
```

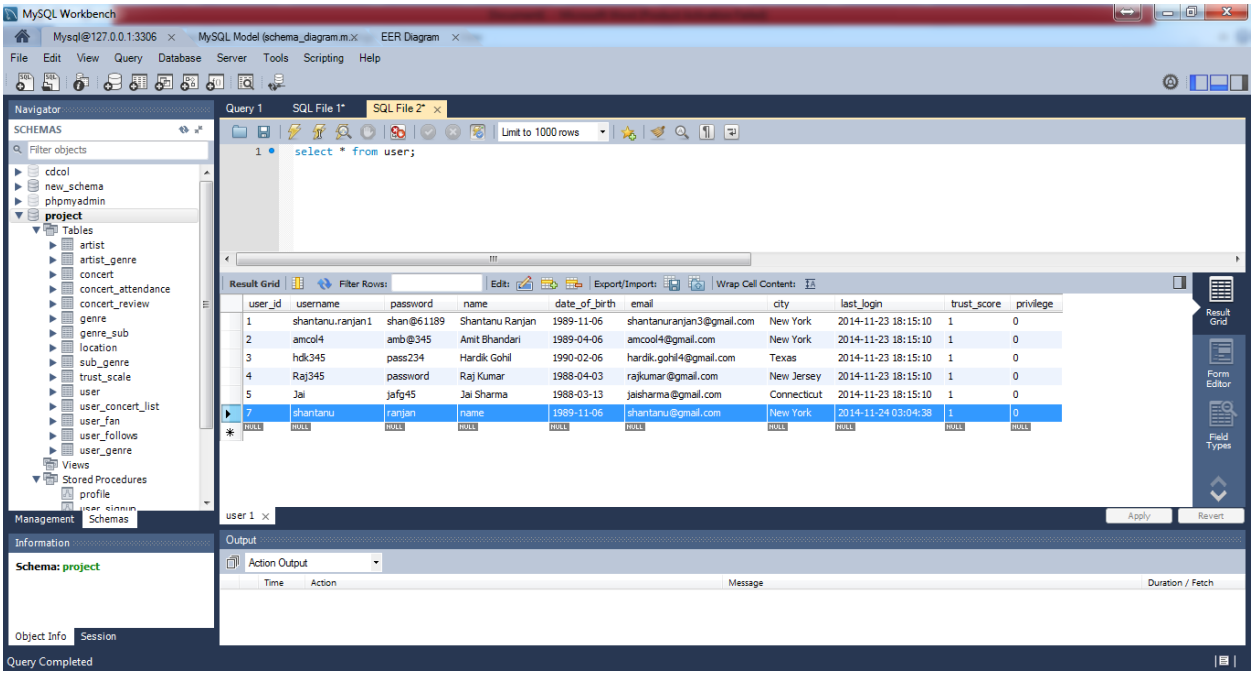



### 4.1.2 CREATE PROFILE:

```
DELIMITER //
CREATE PROCEDURE `profile` (IN name VARCHAR(100),IN dob date,IN email VARCHAR(100),IN city
VARCHAR(100),IN username VARCHAR(100),IN pass VARCHAR(100))
BEGIN
update user set name=name,
date_of_birth=dob,
email=email,
city=city
where username=username and password=pass;
END//
```

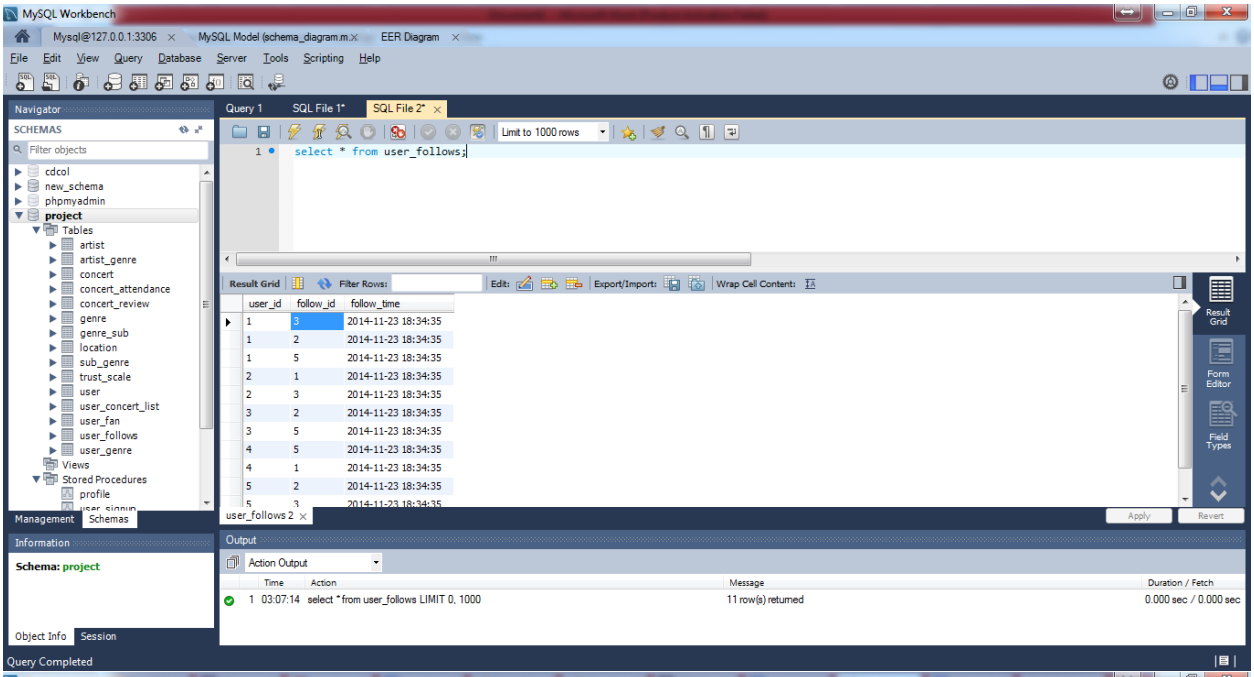


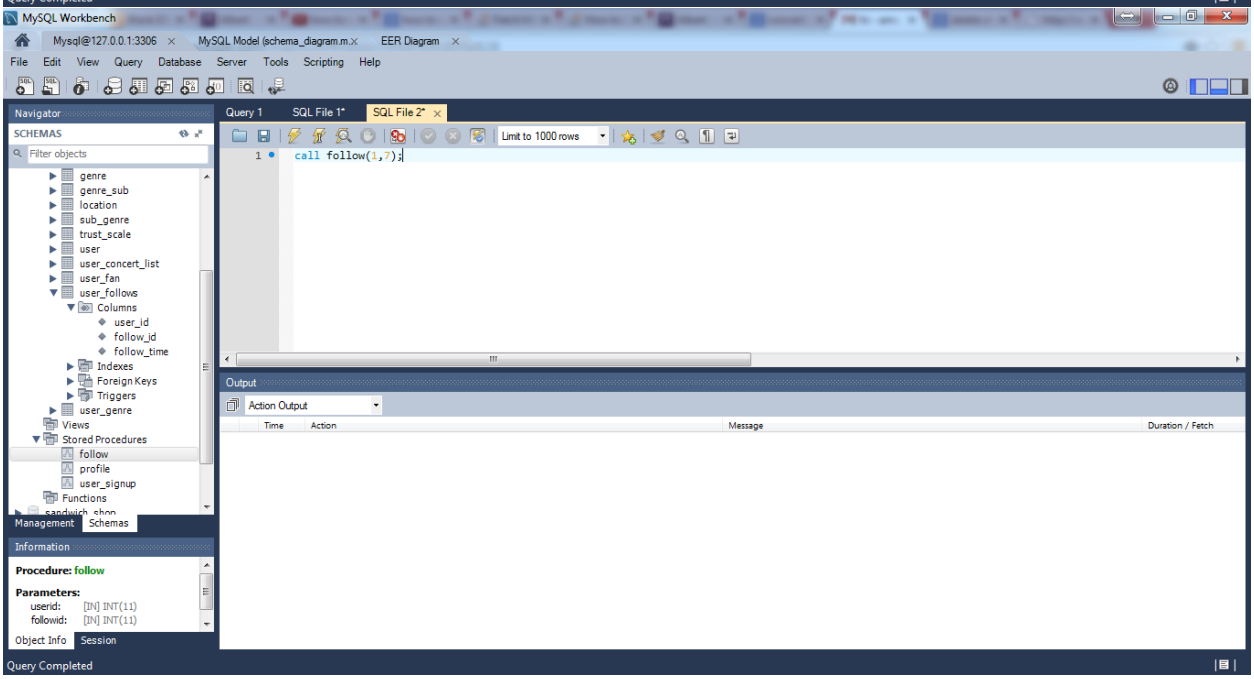



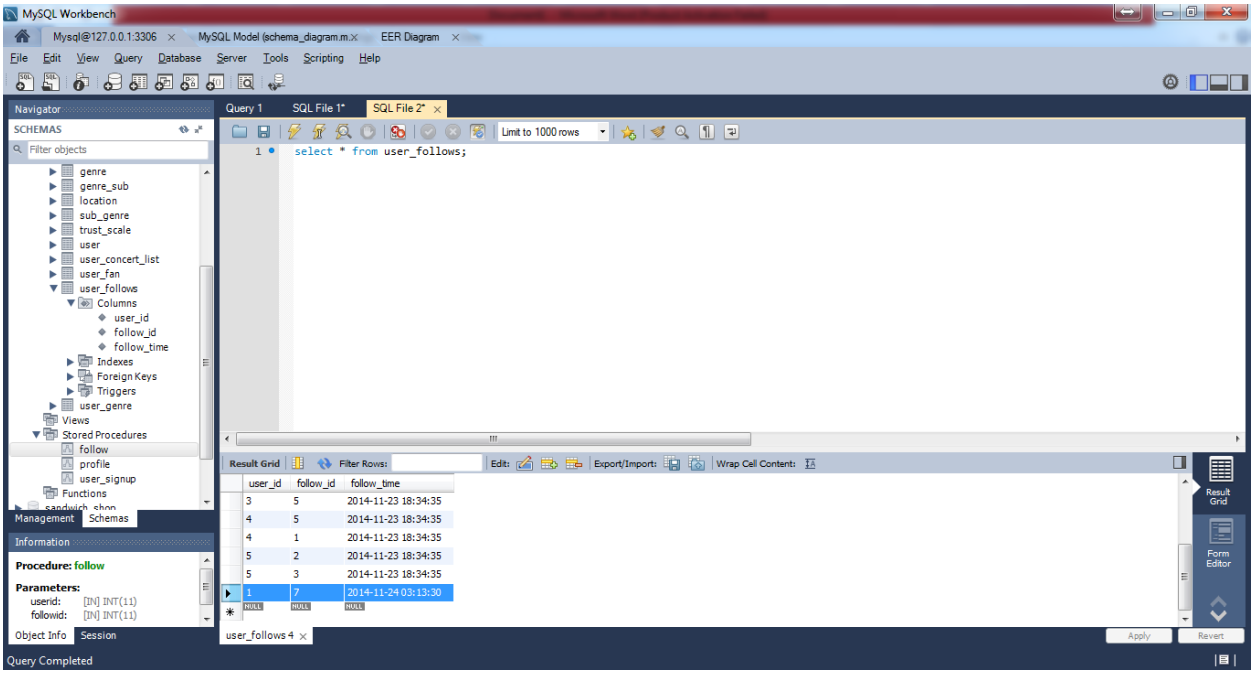

## 4.1.3.FOLLOW ANOTHER USER:

```
DELIMITER //
```

```
CREATE PROCEDURE `follow` (IN userid INT(11),IN followid INT(11))
BEGIN
insert into user_follows(user_id,follow_id) values(userid,followid);
END//
```

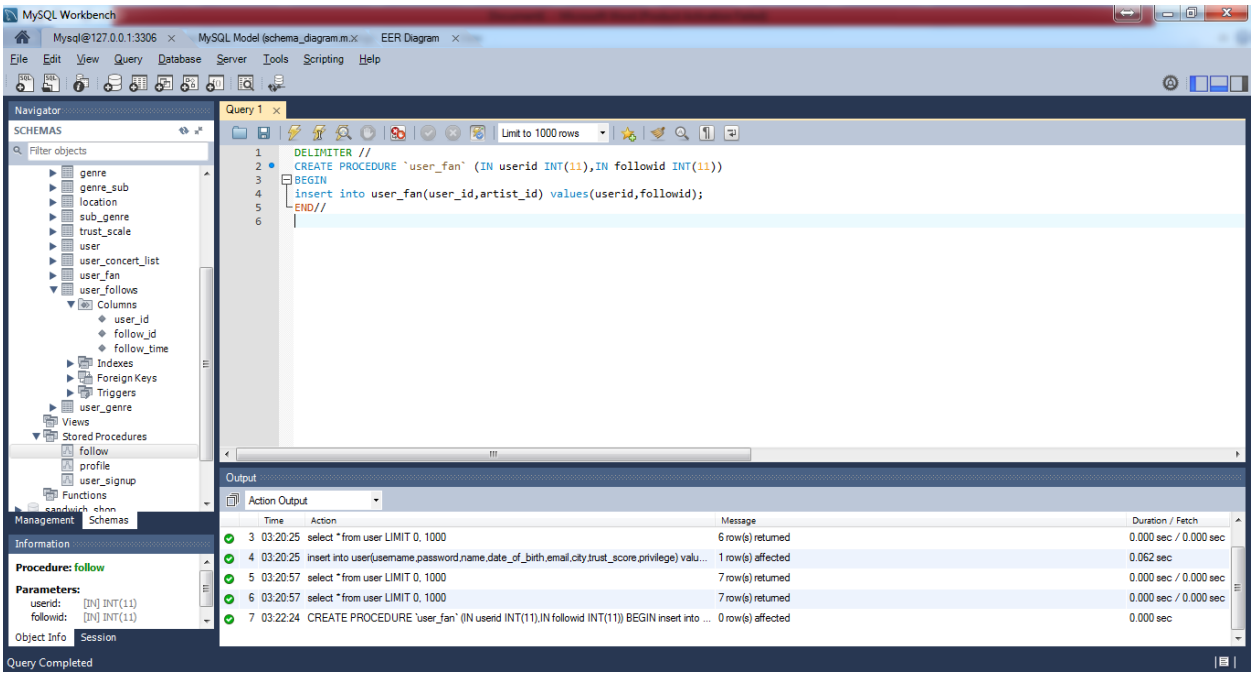


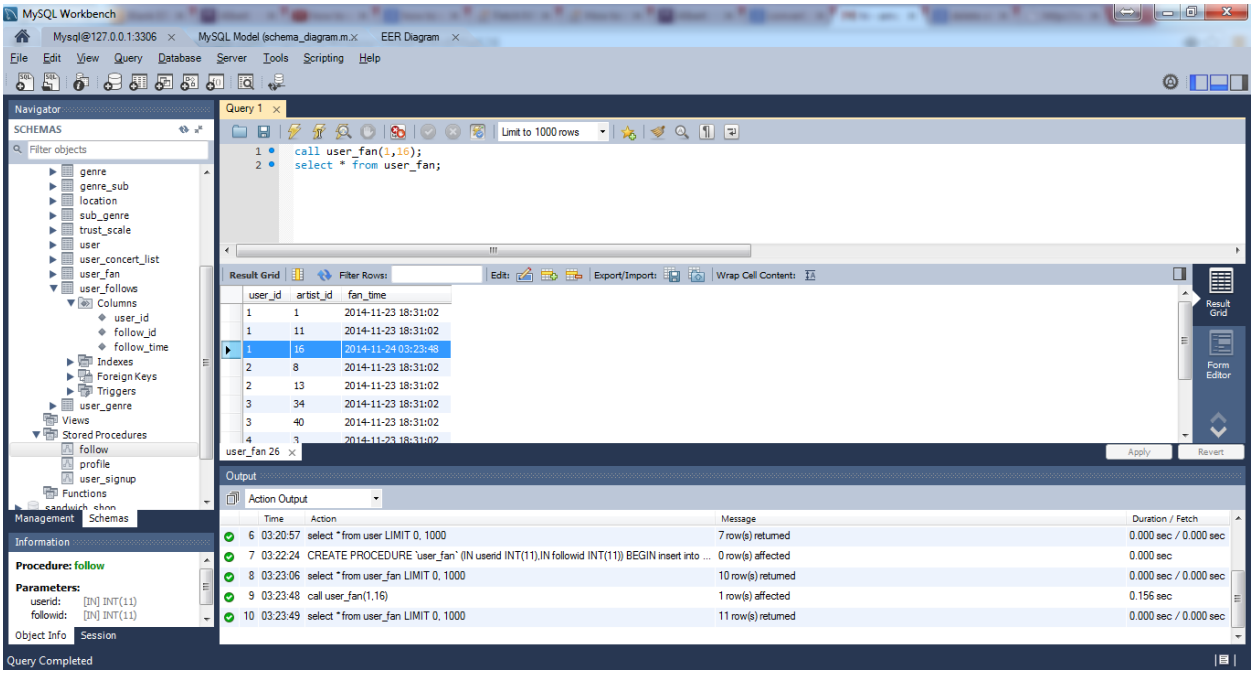






**4.1.4.BECOME FAN OF BAND:**

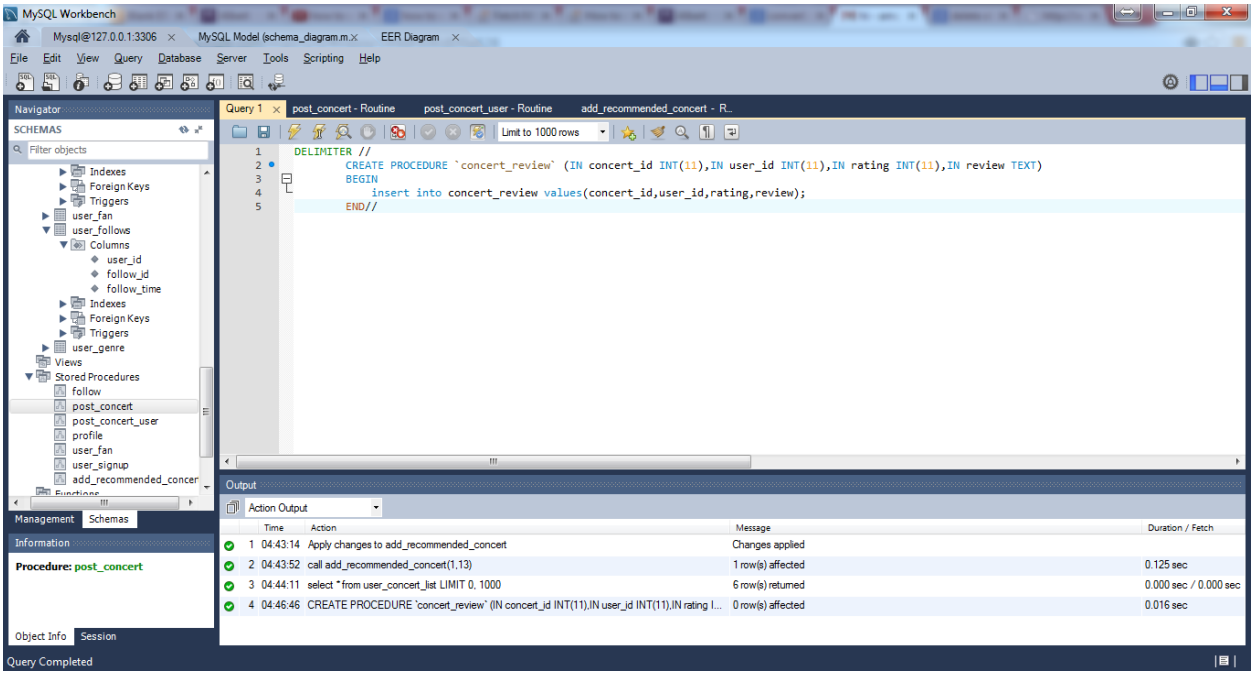
```
DELIMITER //
CREATE PROCEDURE `user_fan` (IN userid INT(11),IN followid INT(11))
BEGIN
insert into user_fan(user_id,artist_id) values(userid,followid);
END//
```

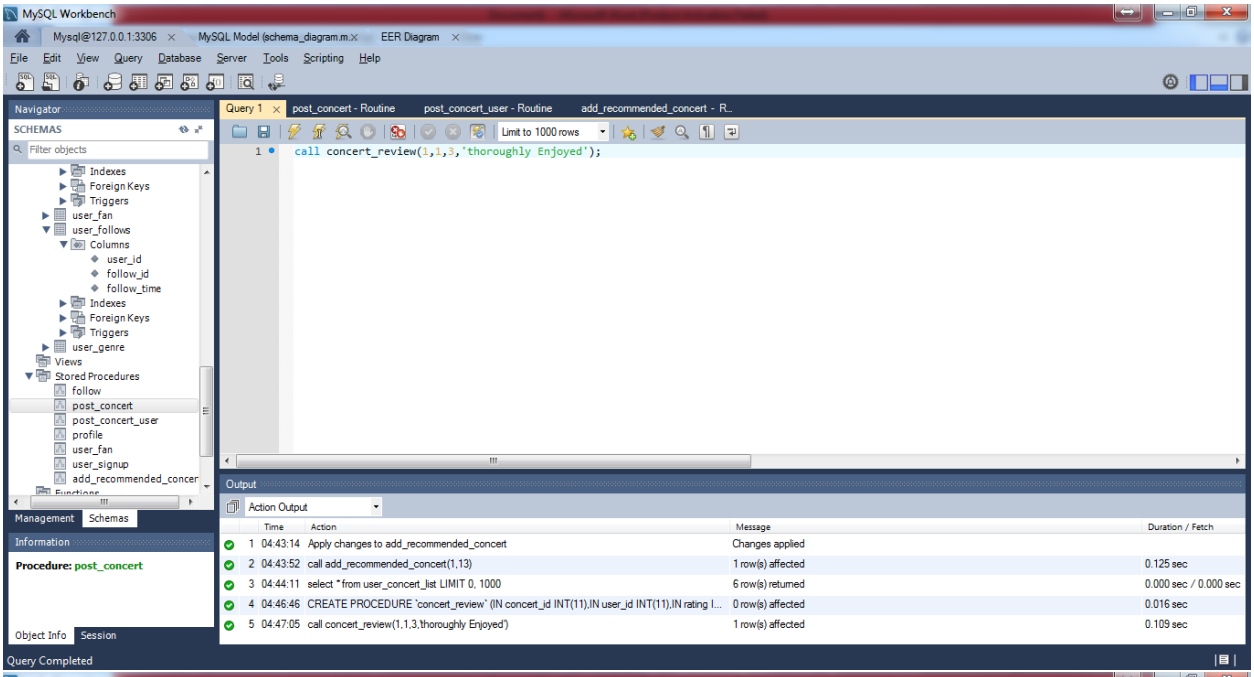


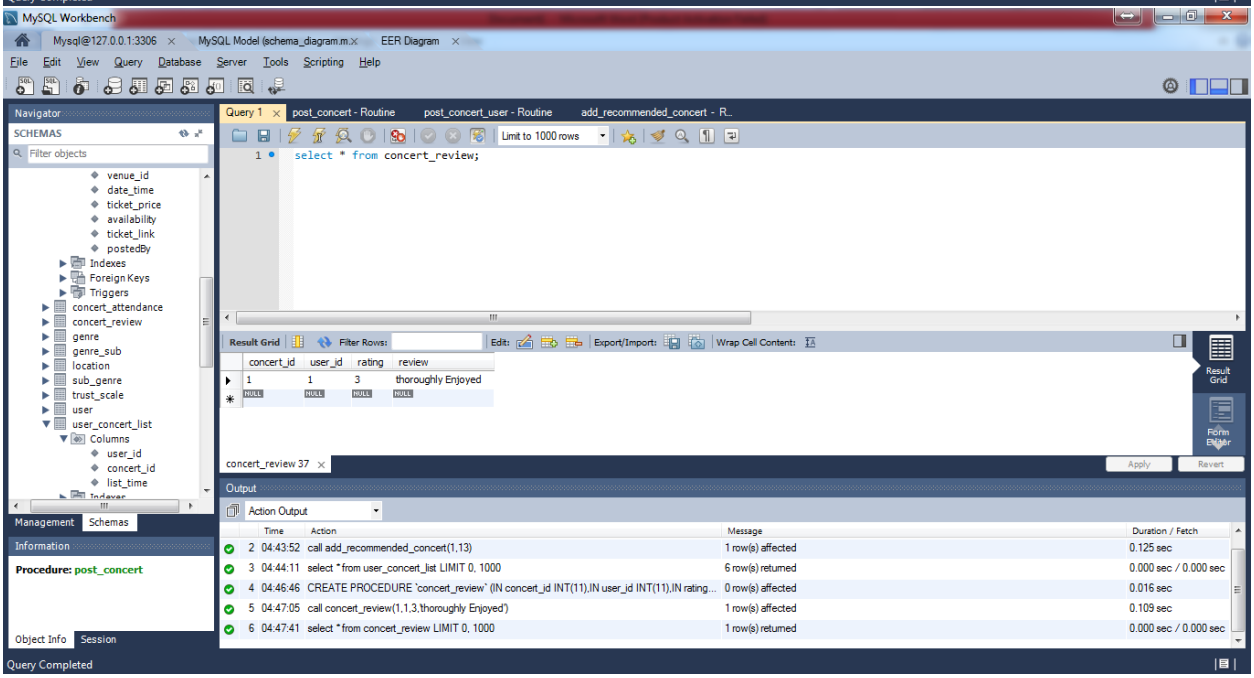



### 4.1.5 POST REVIEW AND RATING :

```
DELIMITER //
CREATE PROCEDURE `concert_review` (IN concert_id INT(11),IN user_id INT(11),IN rating INT(11),IN review TEXT)
BEGIN
insert into concert_review values(concert_id,user_id,rating,review);
END//
```

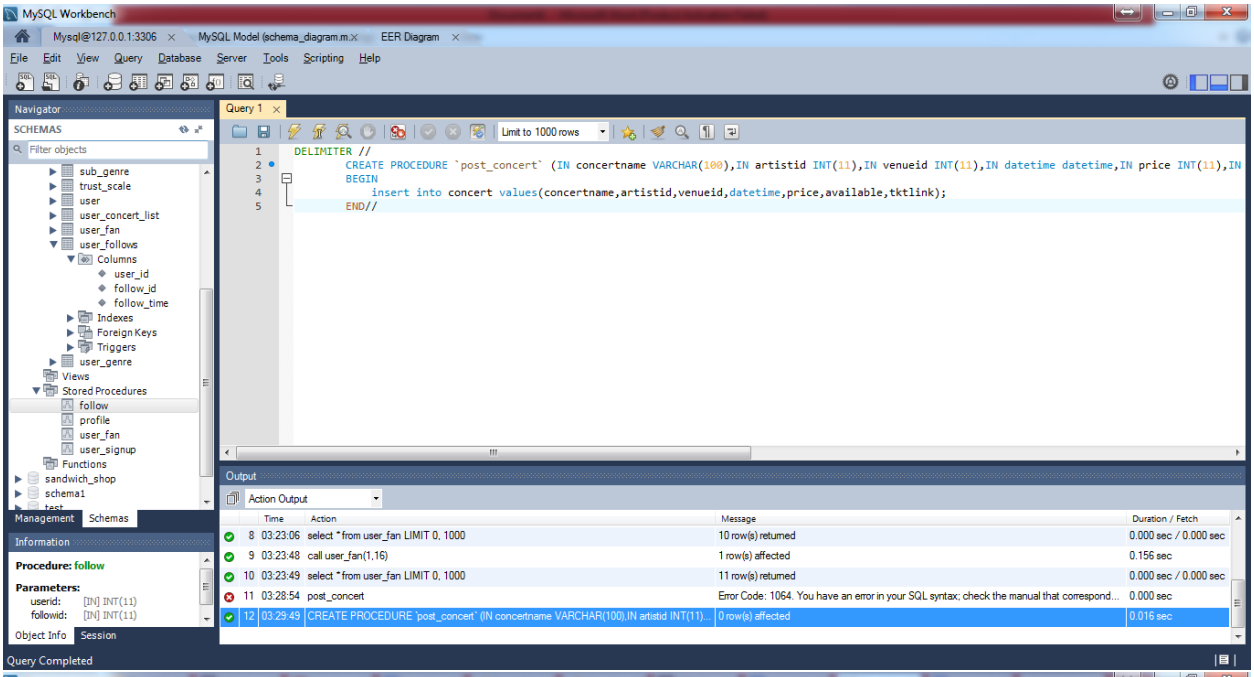


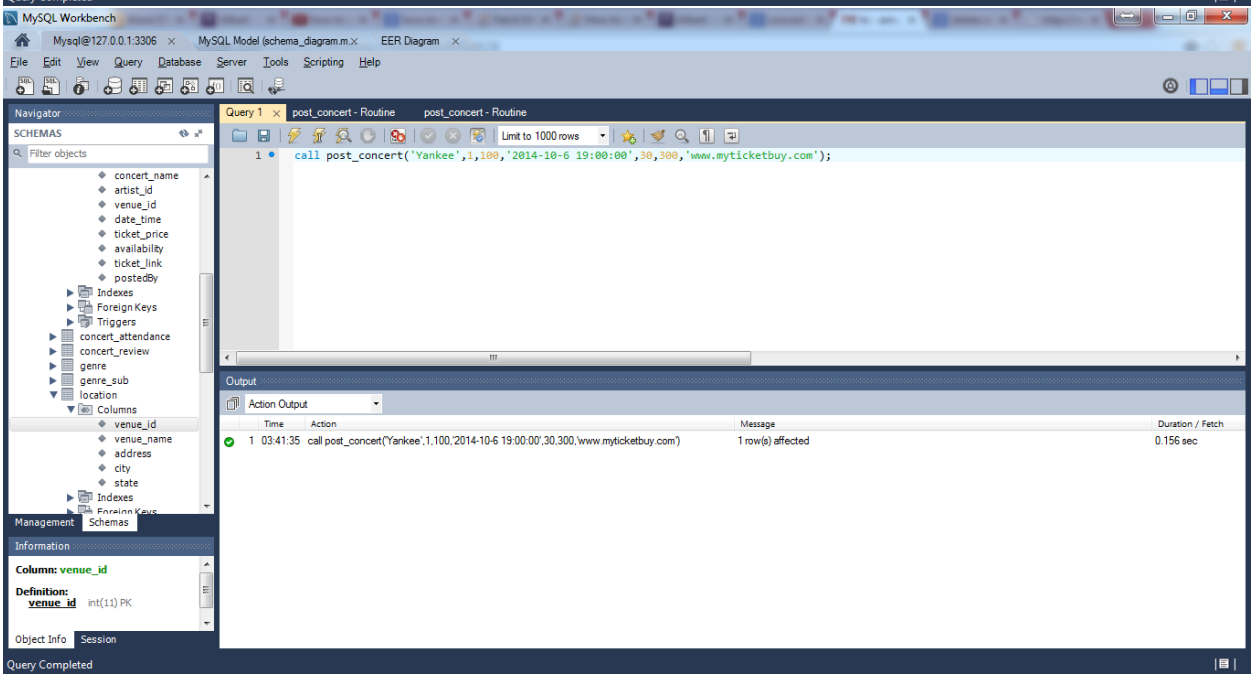


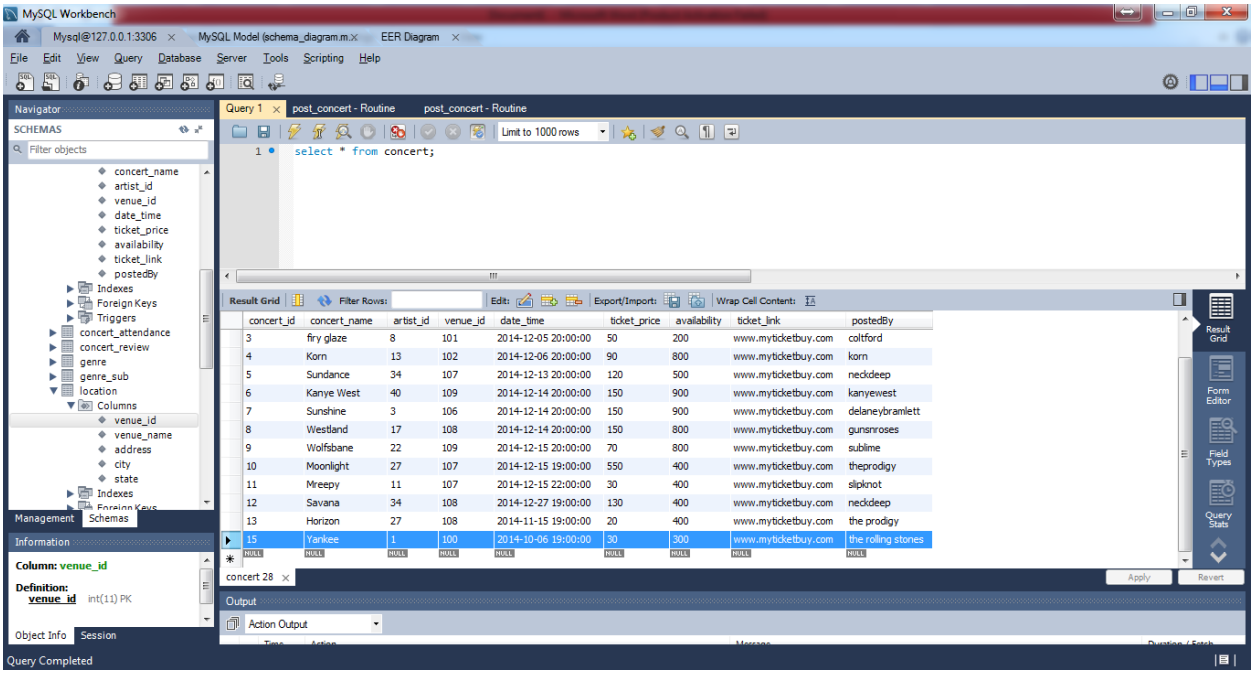




**4.2.1BY BAND POST CONCERT:**

```
DELIMITER //
CREATE PROCEDURE `post_concert` (IN concertname VARCHAR(100),IN artistid INT(11),IN venueid INT(11),IN
date_time datetime,IN price INT(11),IN available INT(11),IN tktlink VARCHAR(200))
BEGIN
Declare artistname varchar(100);
                        select name into artistname from artist where artist_id=artistid;
                        insert into
concert(concert_name,artist_id,venue_id,date_time,ticket_price,availability,ticket_link,postedBy)
values(concertname,artistid,venueid,date_time,price,available,tktlink,artistname);
END//
```

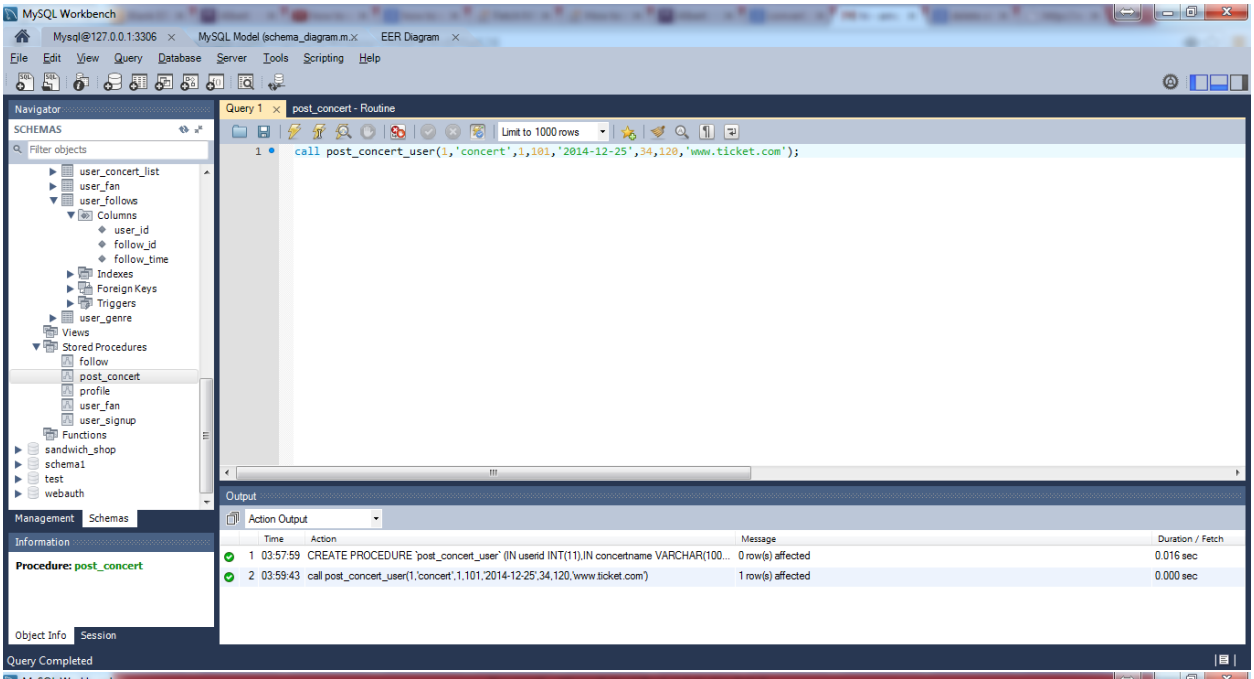

MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sub_genre
trust_scale
user
user_concert_list
user_fan
user_follows
Columns
user_id
follow_id
follow_time
Indexes
Foreign Keys
Triggers
user_genre
Views
Stored Procedures
follow
profile
user_fan
user_signup
Functions
sandwich_shop
schema1
Management Schemas
Information
Procedure: follow
Parameters:
userid: [IN] INT(11)
followid: [IN] INT(11)
Object Info Session
Query 1
Limit to 1000 rows
1 DELIMITER //
2 CREATE PROCEDURE `post_concert` (IN concertname VARCHAR(100),IN artistid INT(11),IN venueid INT(11),IN datetime datetime,IN price INT(11),IN
3 BEGIN
4 insert into concert values(concertname,artistid,venueid,datetime,price,available,tktlink);
5 END//
Output
Action Output
Time Action Message Duration / Fetch
8 03:23:06 select * from user_fan LIMIT 0, 1000 10 row(s) returned 0.000 sec / 0.000 sec
9 03:23:48 call user_fan(1,16) 1 row(s) affected 0.156 sec
10 03:23:49 select * from user_fan LIMIT 0, 1000 11 row(s) returned 0.000 sec / 0.000 sec
11 03:28:54 post_concert Error Code: 1064. You have an error in your SQL syntax; check the manual that correspond... 0.000 sec
12 03:29:49 CREATE PROCEDURE `post_concert` (IN concertname VARCHAR(100),IN artistid INT(11)... 0 row(s) affected 0.016 sec
Query Completed

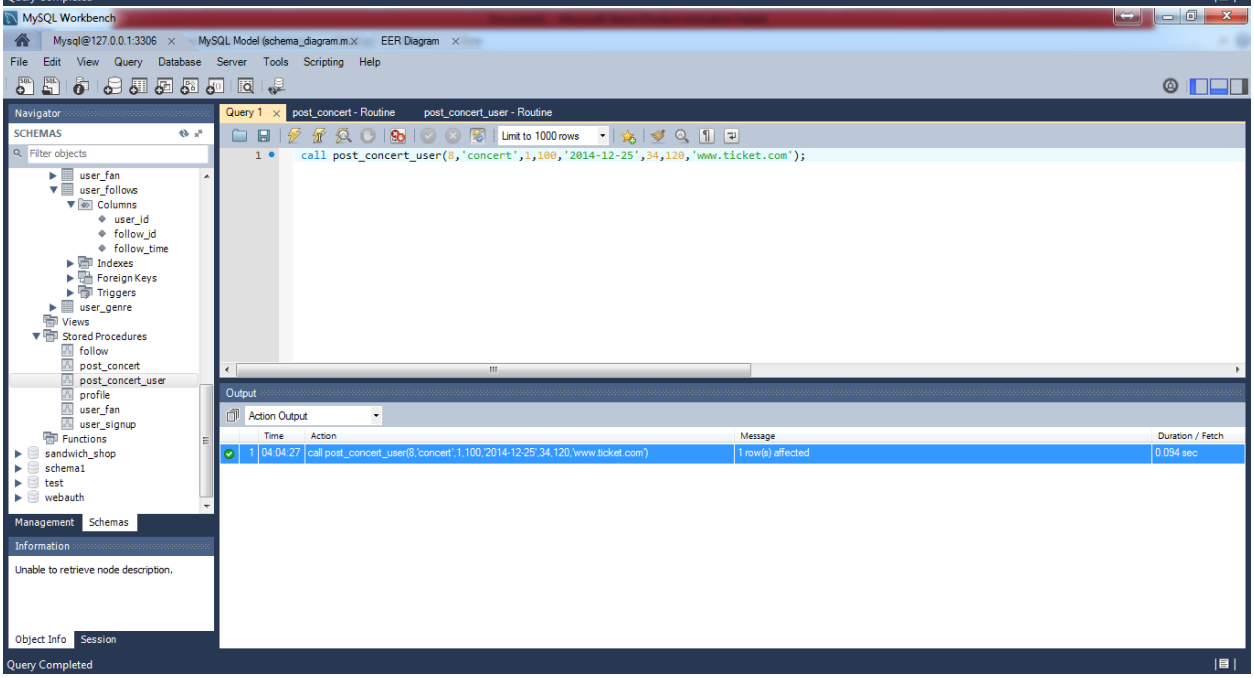

MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
concert_name
artist_id
venue_id
date_time
ticket_price
availability
ticket_link
postedBy
Indexes
Foreign Keys
Triggers
concert_attendance
concert_review
genre
genre_sub
location
Columns
venue_id
venue_name
address
city
state
Indexes
Management Schemas
Information
Column: venue_id
Definition:
venue_id int(11) PK
Object Info Session
Query 1
post_concert - Routine
post_concert - Routine
Limit to 1000 rows
1 call post_concert('Yankee',1,100,'2014-10-6 19:00:00',30,300,'www.myticketbuy.com');
Output
Action Output
Time Action Message Duration / Fetch
1 03:41:35 call post_concert('Yankee',1,100,'2014-10-6 19:00:00',30,300,'www.myticketbuy.com') 1 row(s) affected 0.156 sec
Query Completed

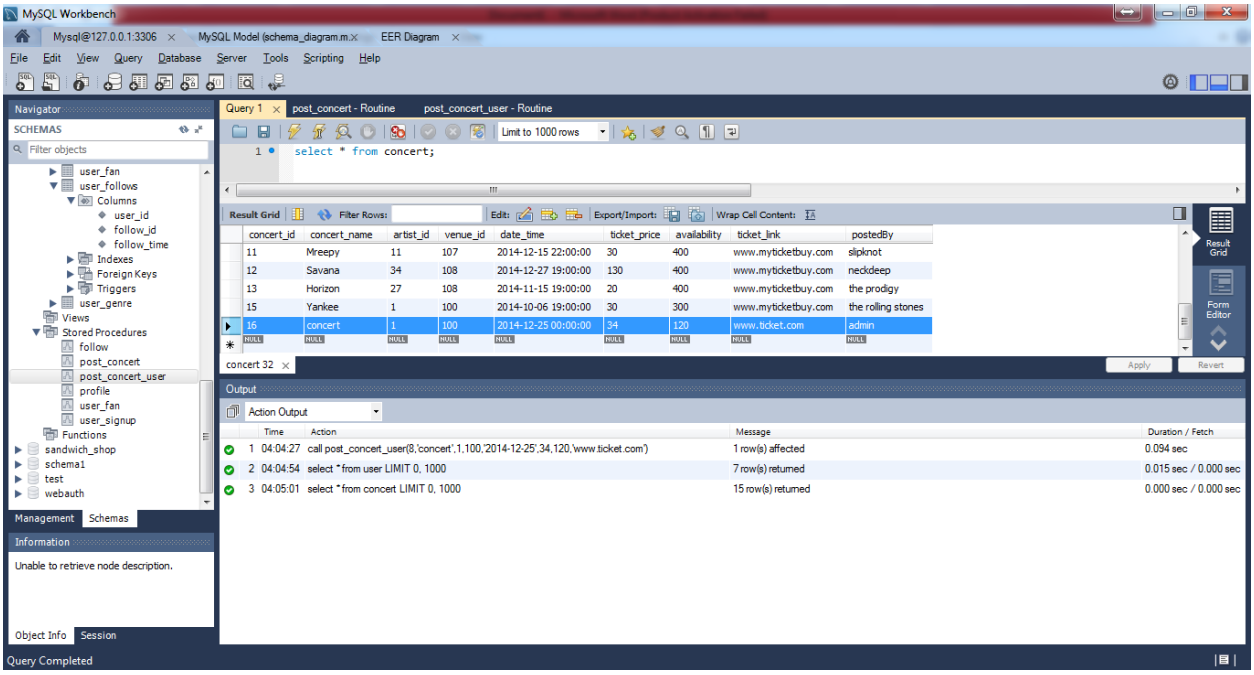



### 4.2.2.BY USER POST CONCERT:

```
DELIMITER //
CREATE PROCEDURE `post_concert_user` (IN userid INT(11),IN concertname VARCHAR(100),IN artistid INT(11),IN
venueid INT(11),IN concert_datetime datetime,IN price INT(11),IN available INT(11),IN tktlink VARCHAR(200))
BEGIN
DECLARE score int(11);
DECLARE newid int(11);
DECLARE username varchar(100);

select trust_score into score from user where user_id=userid;
select username into username from user where user_id=userid;
if score>8 then
insert into concert values(concertname,artistid,venueid,concert_datetime,price,available,tktlink,username);
end if;
END//
```

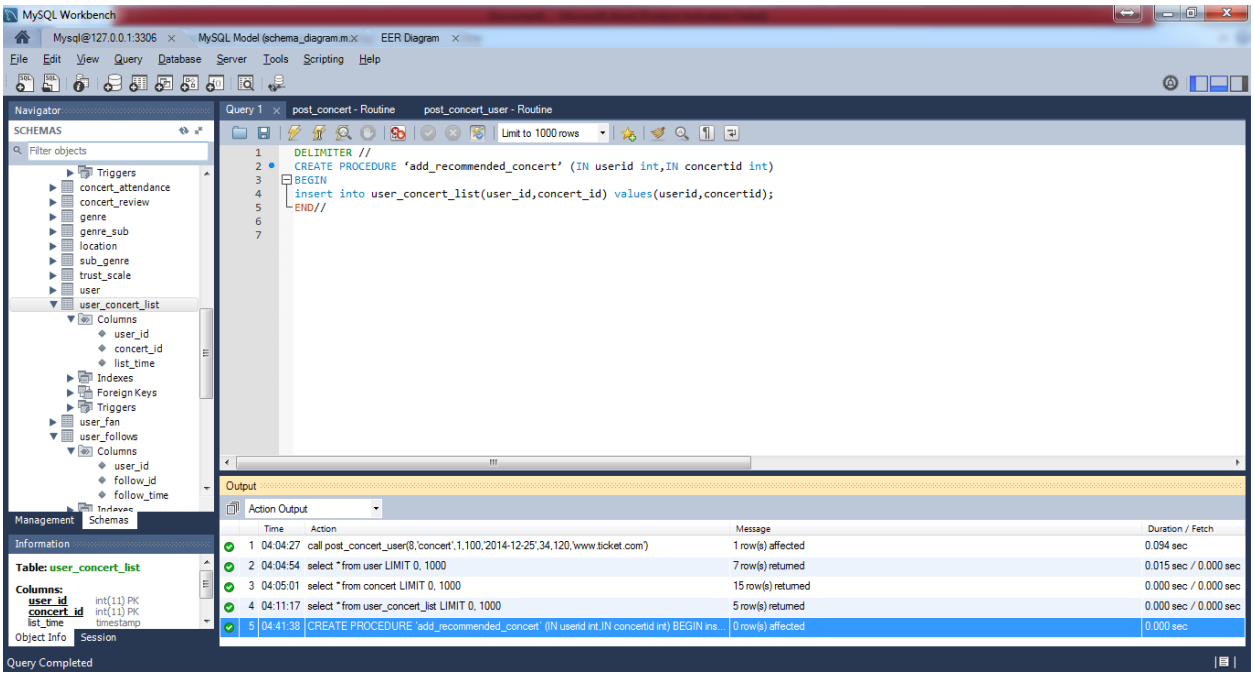

MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
user_concert_list
user_fan
user_follows
Columns
user_id
follow_id
follow_time
Indexes
Foreign Keys
Triggers
user_genre
Views
Stored Procedures
follow
post_concert
profile
user_fan
user_signup
Functions
sandwich_shop
schema1
test
webauth
Management Schemas
Information
Procedure: post_concert
Object Info Session
Query 1
post_concert - Routine
Limit to 1000 rows
1 call post_concert_user(1,'concert',1,101,'2014-12-25',34,120,'www.ticket.com');
Output
Action Output
Time Action Message Duration / Fetch
1 03:57:59 CREATE PROCEDURE `post_concert_user` (IN userid INT(11),IN concertname VARCHAR(100... 0 row(s) affected 0.016 sec
2 03:59:43 call post_concert_user(1,'concert',1,101,'2014-12-25',34,120,'www.ticket.com') 1 row(s) affected 0.000 sec
Query Completed

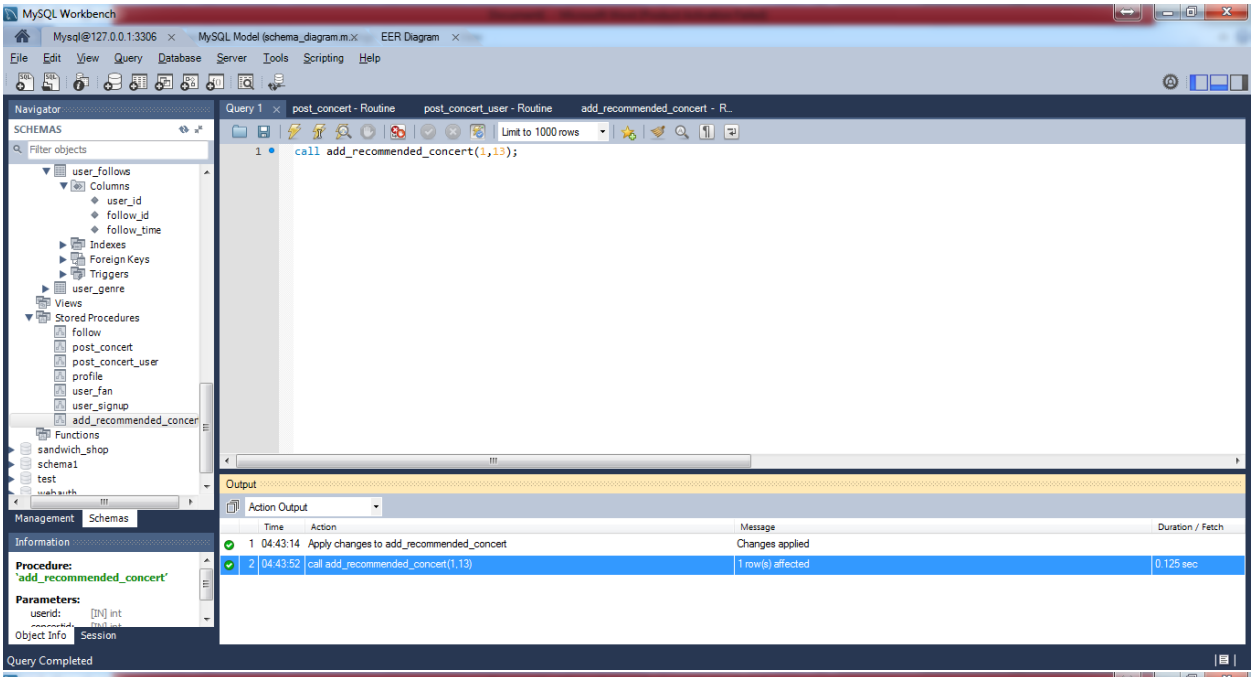

MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
user_fan
user_follows
Columns
user_id
follow_id
follow_time
Indexes
Foreign Keys
Triggers
user_genre
Views
Stored Procedures
follow
post_concert
post_concert_user
profile
user_fan
user_signup
Functions
sandwich_shop
schema1
test
webauth
Management Schemas
Information
Unable to retrieve node description.
Object Info Session
Query 1
post_concert - Routine
post_concert_user - Routine
Limit to 1000 rows
1 call post_concert_user(8,'concert',1,100,'2014-12-25',34,120,'www.ticket.com');
Output
Action Output
Time Action Message Duration / Fetch
1 04:04:27 call post_concert_user(8,'concert',1,100,'2014-12-25',34,120,'www.ticket.com') 1 row(s) affected 0.094 sec
Query Completed

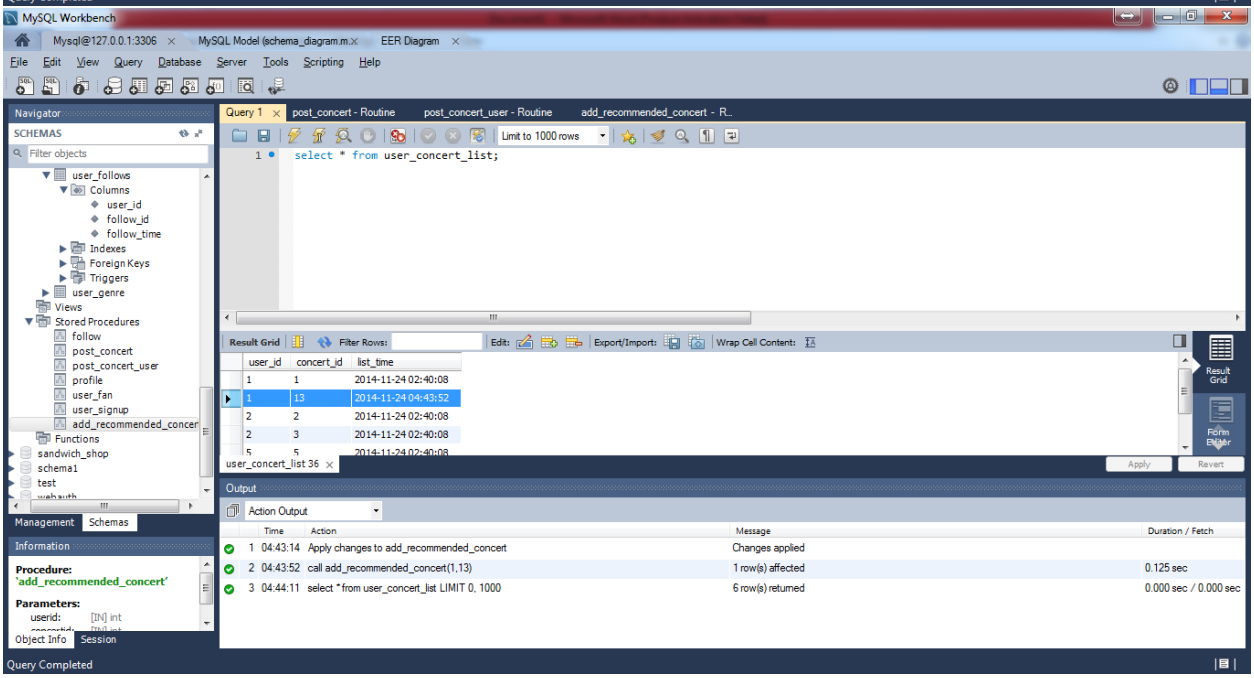



### 4.2.3. ADD RECOMMENDED CONCERT:

DELIMITER //
CREATE PROCEDURE add_recommended_concert (IN userid int,IN concertid int)
BEGIN
Insert into user_concert_list(user_id,concert_id) values(userid,concertid);
END//

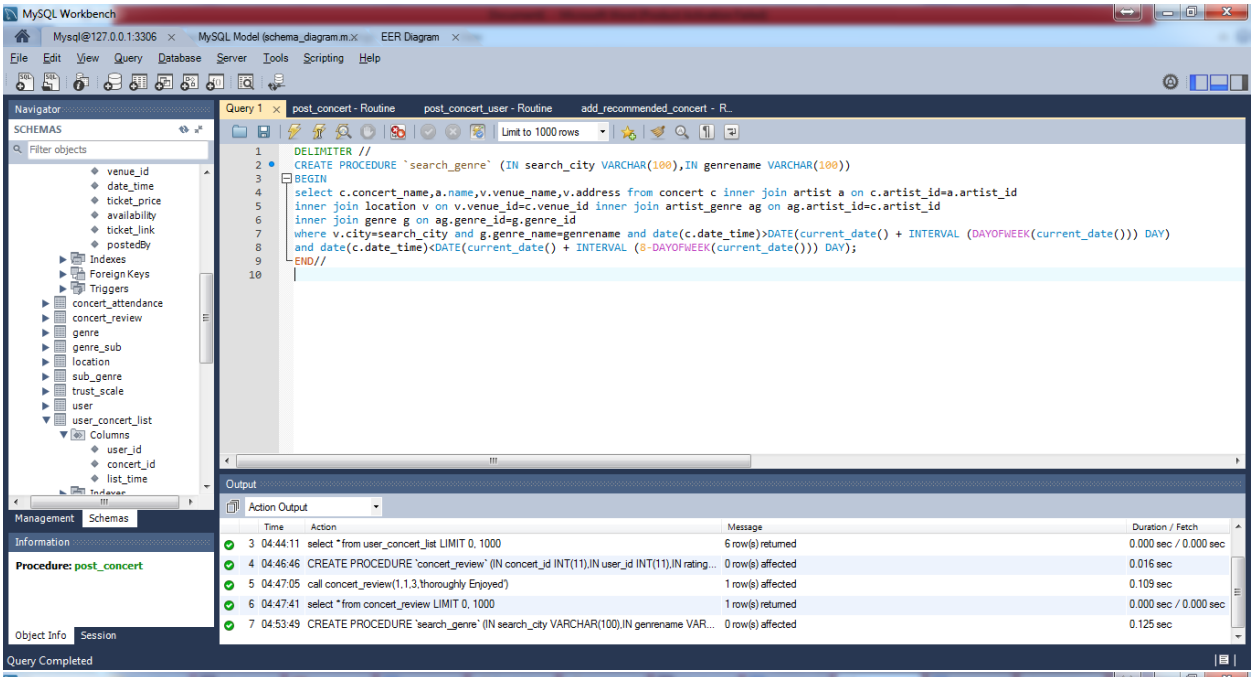


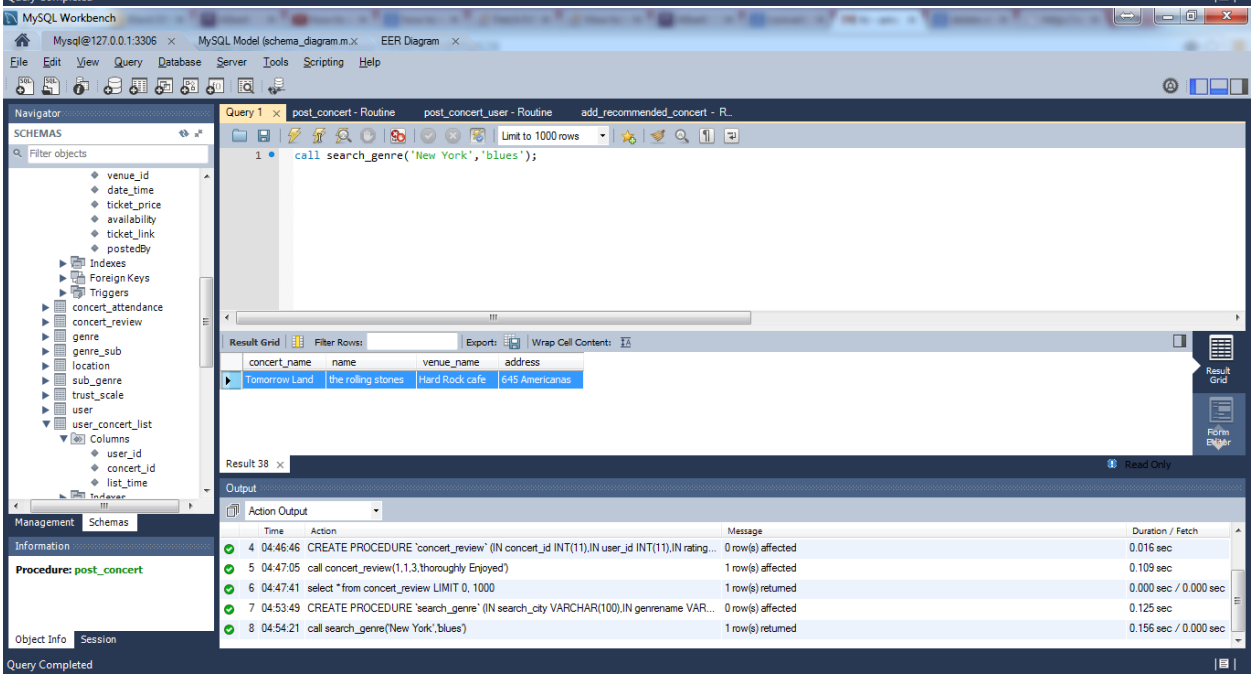



### 4.3.1 TO SEE CONCERTS IN NEXT WEEK

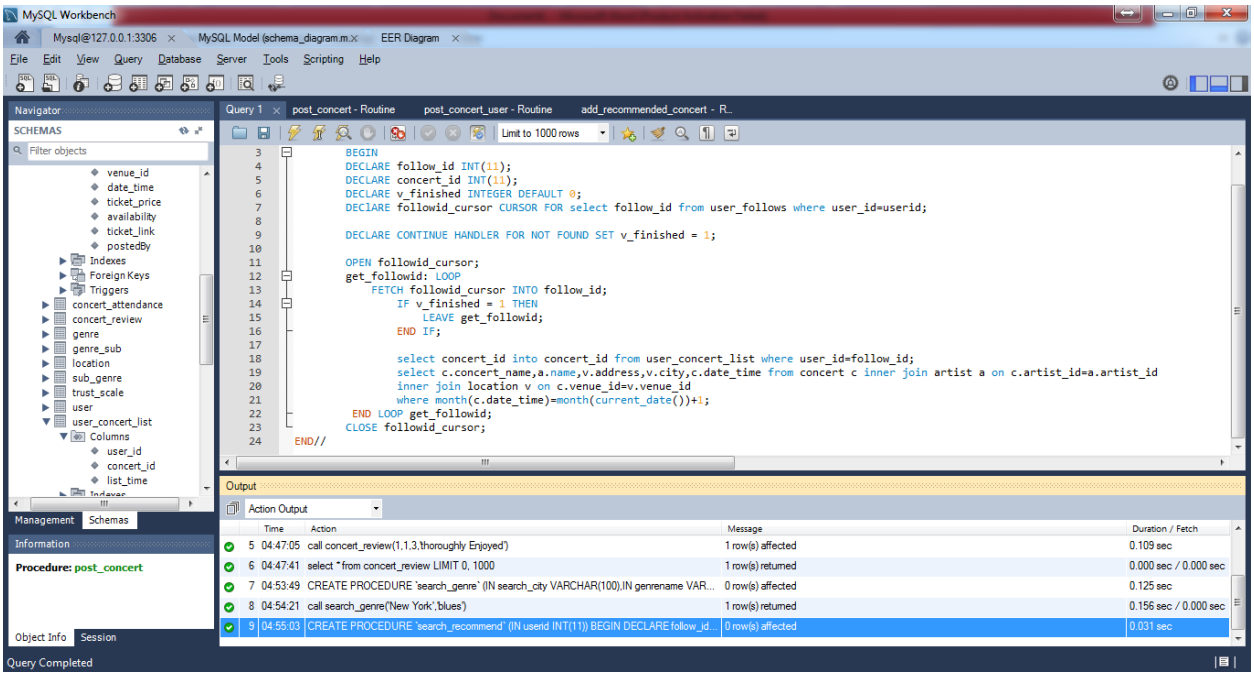
```
DELIMITER //
CREATE PROCEDURE `search_genre` (IN search_city VARCHAR(100),IN genrename VARCHAR(100))
BEGIN
select c.concert_name,a.name,v.venue_name,v.address from concert c inner join artist a on c.artist_id=a.artist_id
inner join location v on v.venue_id=c.venue_id inner join artist_genre ag on ag.artist_id=c.artist_id
inner join genre g on ag.genre_id=g.genre_id
where v.city=search_city and g.genre_name=genrename and date(c.date_time)>DATE(current_date() + INTERVAL
(DAYOFWEEK(current_date())) DAY)
and date(c.date_time)<DATE(current_date() + INTERVAL (8-DAYOFWEEK(current_date())) DAY);
END//
```

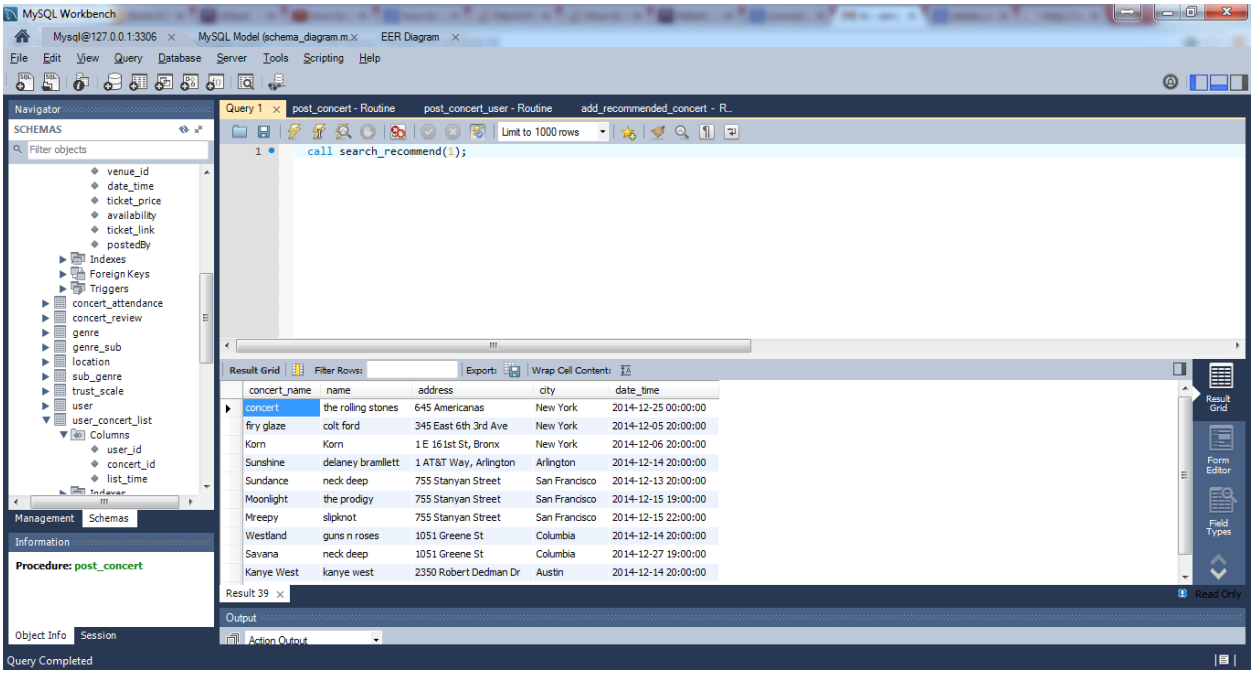

MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
venue_id
date_time
ticket_price
availability
ticket_link
postedBy
Indexes
Foreign Keys
Triggers
concert_attendance
concert_review
genre
genre_sub
location
sub_genre
trust_scale
user
user_concert_list
Columns
user_id
concert_id
list_time
Management Schemas
Information
Procedure: post_concert
Object Info Session
Query 1
post_concert - Routine
post_concert_user - Routine
add_recommended_concert - R...
Limit to 1000 rows
1 DELIMITER //
2 CREATE PROCEDURE `search_genre` (IN search_city VARCHAR(100),IN genrename VARCHAR(100))
3 BEGIN
4 select c.concert_name,a.name,v.venue_name,v.address from concert c inner join artist a on c.artist_id=a.artist_id
5 inner join location v on v.venue_id=c.venue_id inner join artist_genre ag on ag.artist_id=c.artist_id
6 inner join genre g on ag.genre_id=g.genre_id
7 where v.city=search_city and g.genre_name=genrename and date(c.date_time)>DATE(current_date() + INTERVAL (DAYOFWEEK(current_date())) DAY)
8 and date(c.date_time)<DATE(current_date() + INTERVAL (8-DAYOFWEEK(current_date())) DAY);
9 END//
10
Output
Action Output
Time Action Message Duration / Fetch
3 04:44:11 select * from user_concert_list LIMIT 0, 1000 6 row(s) returned 0.000 sec / 0.000 sec
4 04:46:46 CREATE PROCEDURE `concert_review` (IN concert_id INT(11),IN user_id INT(11),IN rating... 0 row(s) affected 0.016 sec
5 04:47:05 call concert_review(1,1,3,'thoroughly Enjoyed') 1 row(s) affected 0.109 sec
6 04:47:41 select * from concert_review LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
7 04:53:49 CREATE PROCEDURE `search_genre` (IN search_city VARCHAR(100),IN genrename VAR... 0 row(s) affected 0.125 sec
Query Completed


MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
venue_id
date_time
ticket_price
availability
ticket_link
postedBy
Indexes
Foreign Keys
Triggers
concert_attendance
concert_review
genre
genre_sub
location
sub_genre
trust_scale
user
user_concert_list
Columns
user_id
concert_id
list_time
Management Schemas
Information
Procedure: post_concert
Object Info Session
Query 1
post_concert - Routine
post_concert_user - Routine
add_recommended_concert - R...
Limit to 1000 rows
1 call search_genre('New York','blues');
Result Grid Filter Rows: Export: Wrap Cell Content:
concert_name name venue_name address
Tomorrow Land the rolling stones Hard Rock cafe 645 Americanas
Result Grid
Form Editor
Result 38
Read Only
Output
Action Output
Time Action Message Duration / Fetch
4 04:46:46 CREATE PROCEDURE `concert_review` (IN concert_id INT(11),IN user_id INT(11),IN rating... 0 row(s) affected 0.016 sec
5 04:47:05 call concert_review(1,1,3,'thoroughly Enjoyed') 1 row(s) affected 0.109 sec
6 04:47:41 select * from concert_review LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
7 04:53:49 CREATE PROCEDURE `search_genre` (IN search_city VARCHAR(100),IN genrename VAR... 0 row(s) affected 0.125 sec
8 04:54:21 call search_genre('New York','blues') 1 row(s) returned 0.156 sec / 0.000 sec
Query Completed

### 4.3.2 SEE ALL CONCERTS RECOMMENDED BY PEOPLE THEY FOLLOW DATING NEXT MONTH

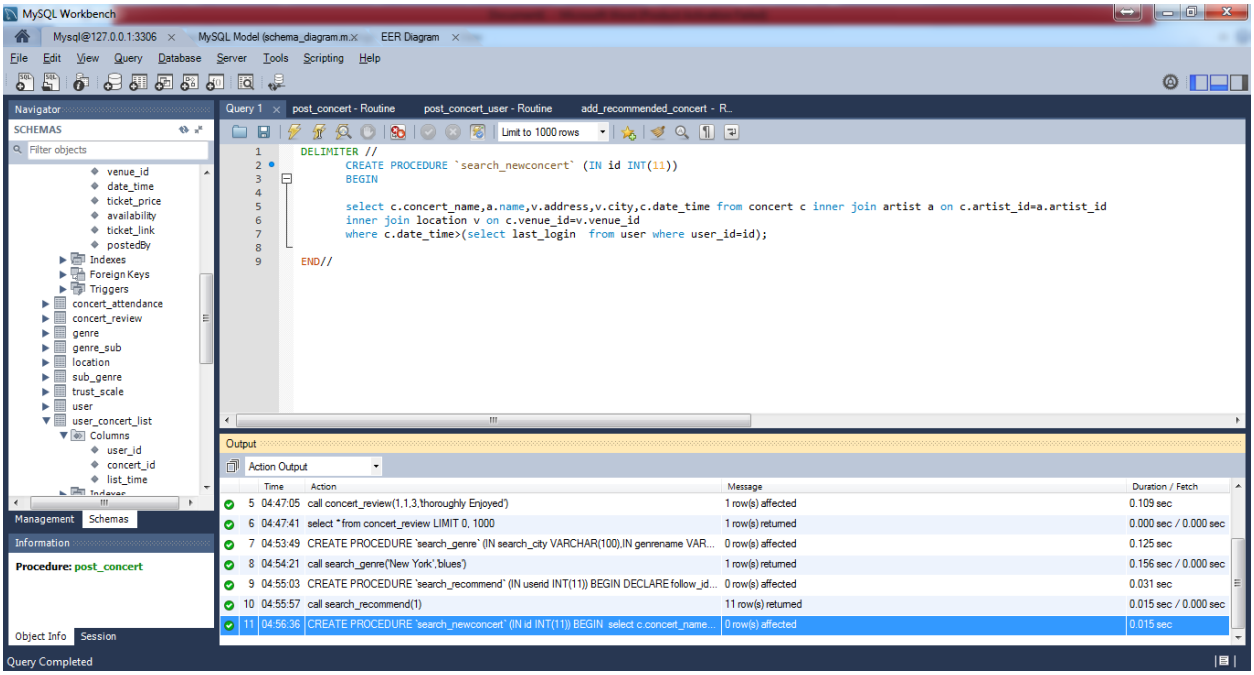
```
DELIMITER //
CREATE PROCEDURE `search_recommend` (IN userid INT(11))
BEGIN
DECLARE follow_id INT(11);
DECLARE concert_id INT(11);
DECLARE v_finished INTEGER DEFAULT 0;
DECIARE followid_cursor CURSOR FOR select follow_id from user_follows where user_id=userid;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
OPEN followid_cursor;
get_followid: LOOP
FETCH followid_cursor INTO follow_id;
IF v_finished = 1 THEN
LEAVE get_followid;
END IF;
select concert_id into concert_id from user_concert_list where user_id=follow_id;
select c.concert_name,a.name,v.address,v.city,c.date_time from concert c inner join artist a on c.artist_id=a.artist_id
inner join location v on c.venue_id=v.venue_id
where month(c.date_time)=month(current_date())+1;
 END LOOP get_followid;
CLOSE followid_cursor;
END//
```

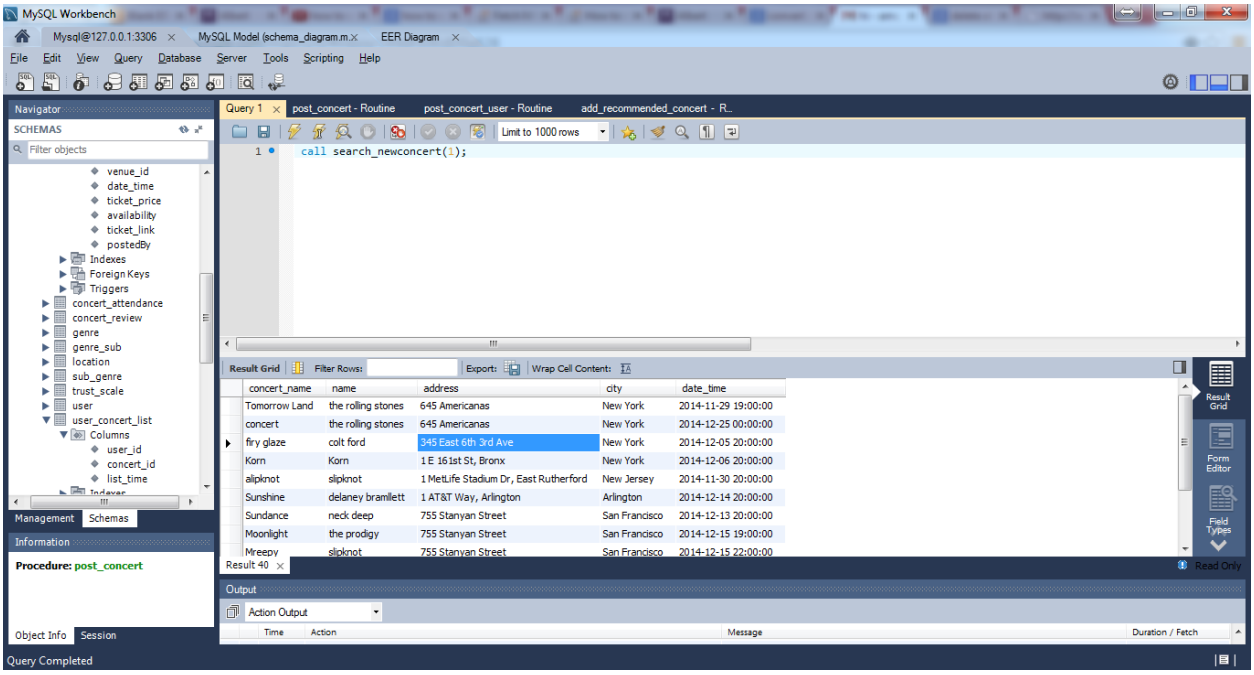





### 4.3.3 SEE ALL POSTED CONCERTS SINCE LAST TIME USER LOGGED IN

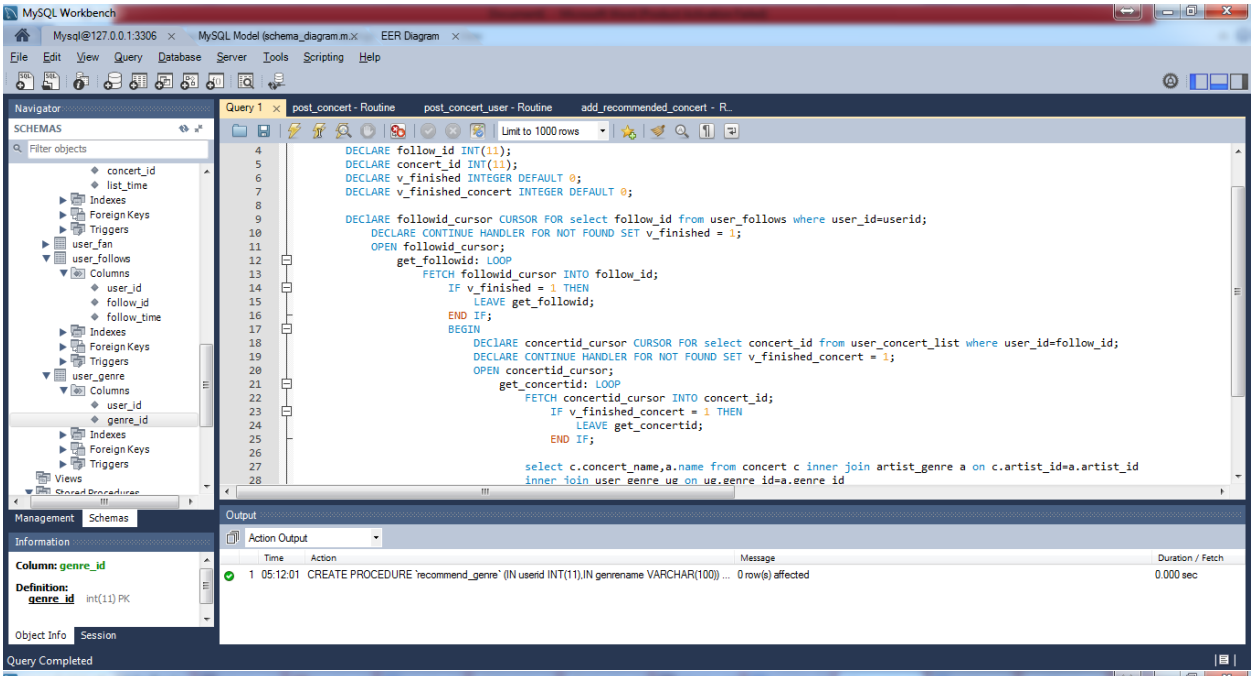
```
DELIMITER //
CREATE PROCEDURE `search_newconcert` (IN id INT(11))
BEGIN
select c.concert_name,a.name,v.address,v.city,c.date_time from concert c inner join artist a on c.artist_id=a.artist_id
inner join location v on c.venue_id=v.venue_id
where c.date_time>(select last_login  from user where user_id=id);
END//
```

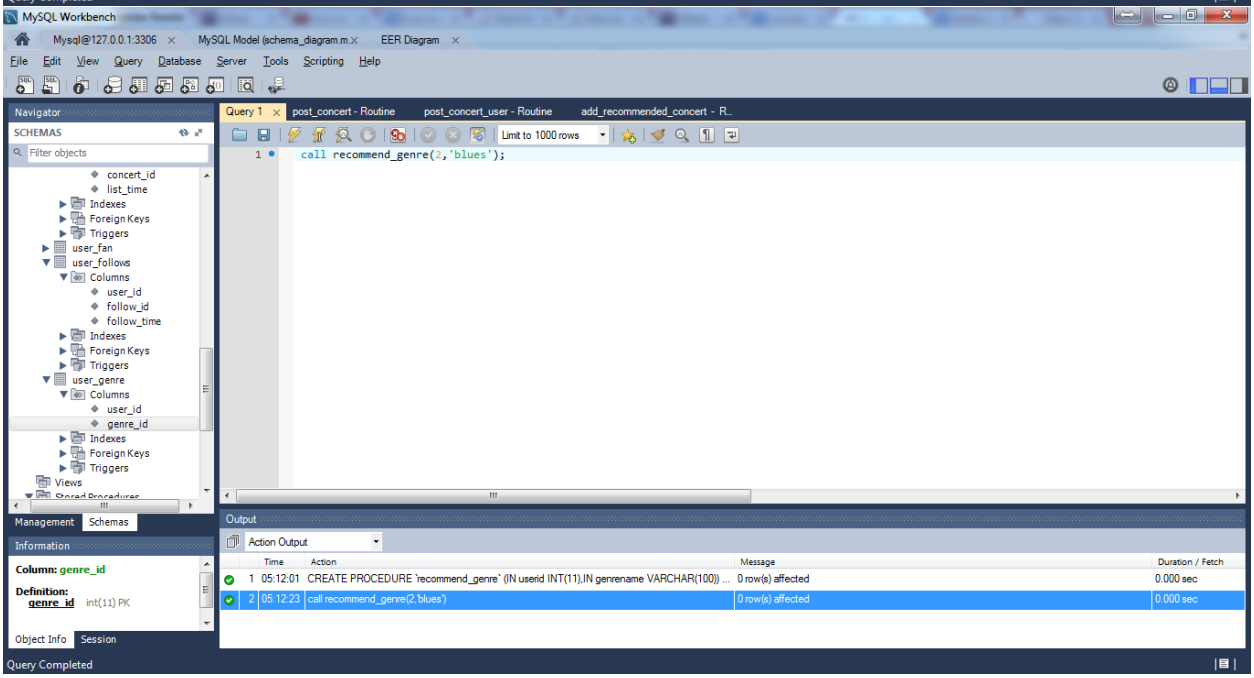





### 4.4.1 RECOMMEND CONCERTS TO USER AS PER GENRES HE LIKES WHICH EXIST IN MANY RECOMMENDED LISTS BY USERS

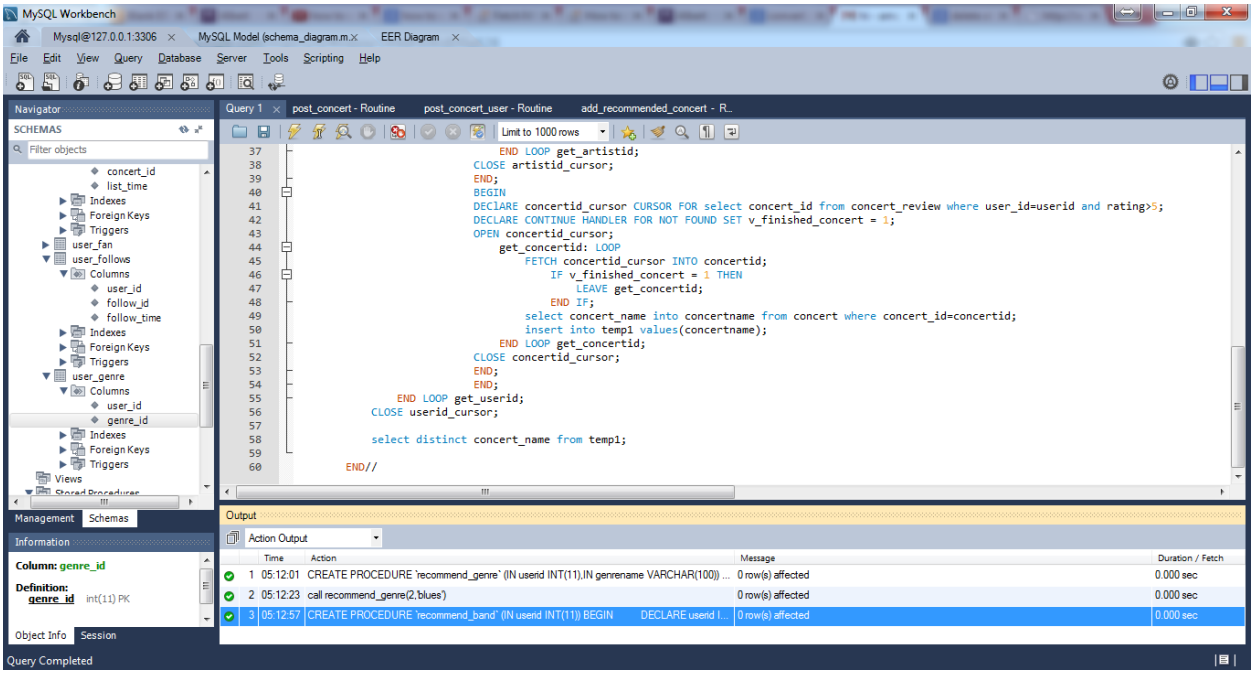
```
 DELIMITER //
CREATE PROCEDURE `recommend_genre` (IN userid INT(11),IN genrename VARCHAR(100))
BEGIN
DECLARE follow_id INT(11);
DECLARE concert_id INT(11);
DECLARE v_finished INTEGER DEFAULT 0;
DECLARE v_finished_concert INTEGER DEFAULT 0;
DECIARE followid_cursor CURSOR FOR select follow_id from user_follows where user_id=userid;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
OPEN followid_cursor;
get_followid: LOOP
FETCH followid_cursor INTO follow_id;
IF v_finished = 1 THEN
LEAVE get_followid;
END IF;
                BEGIN
DECIARE concertid_cursor CURSOR FOR select concert_id from user_concert_list where user_id=follow_id;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished_concert = 1;
OPEN concertid_cursor;
get_concertid: LOOP
FETCH concertid_cursor INTO concert_id;
IF v_finished_concert = 1 THEN
LEAVE get_concertid;
END IF;
select c.concert_name,a.name from concert c inner join artist_genre a on c.artist_id=a.artist_id
inner join user_genre ug on ug.genre_id=a.genre_id
                        where ug.user_id=userid;
END LOOP get_concertid;
CLOSE concertid_cursor;
                    END;
END LOOP get_followid;
CLOSE followid_cursor;
 END//
```


MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
concert_id
list_time
Indexes
Foreign Keys
Triggers
user_fan
user_follows
Columns
user_id
follow_id
follow_time
Indexes
Foreign Keys
Triggers
user_genre
Columns
user_id
genre_id
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Management
Schemas
Information
Column: genre_id
Definition:
genre_id int(11) PK
Object Info
Session
Query 1
post_concert - Routine
post_concert_user - Routine
add_recommended_concert - R...
Limit to 1000 rows
DECLARE follow_id INT(11);
DECLARE concert_id INT(11);
DECLARE v_finished INTEGER DEFAULT 0;
DECLARE v_finished_concert INTEGER DEFAULT 0;
DECLARE followid_cursor CURSOR FOR select follow_id from user_follows where user_id=userid;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
OPEN followid_cursor;
get_followid: LOOP
FETCH followid_cursor INTO follow_id;
IF v_finished = 1 THEN
LEAVE get_followid;
END IF;
BEGIN
DECLARE concertid_cursor CURSOR FOR select concert_id from user_concert_list where user_id=follow_id;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished_concert = 1;
OPEN concertid_cursor;
get_concertid: LOOP
FETCH concertid_cursor INTO concert_id;
IF v_finished_concert = 1 THEN
LEAVE get_concertid;
END IF;
select c.concert_name,a.name from concert c inner join artist_genre a on c.artist_id=a.artist_id
inner join user_genre ug on ug.genre_id=a.genre_id
Output
Action Output
Time Action Message Duration / Fetch
1 05:12:01 CREATE PROCEDURE `recommend_genre` (IN userid INT(11),IN genrename VARCHAR(100)) ... 0 row(s) affected 0.000 sec
Query Completed

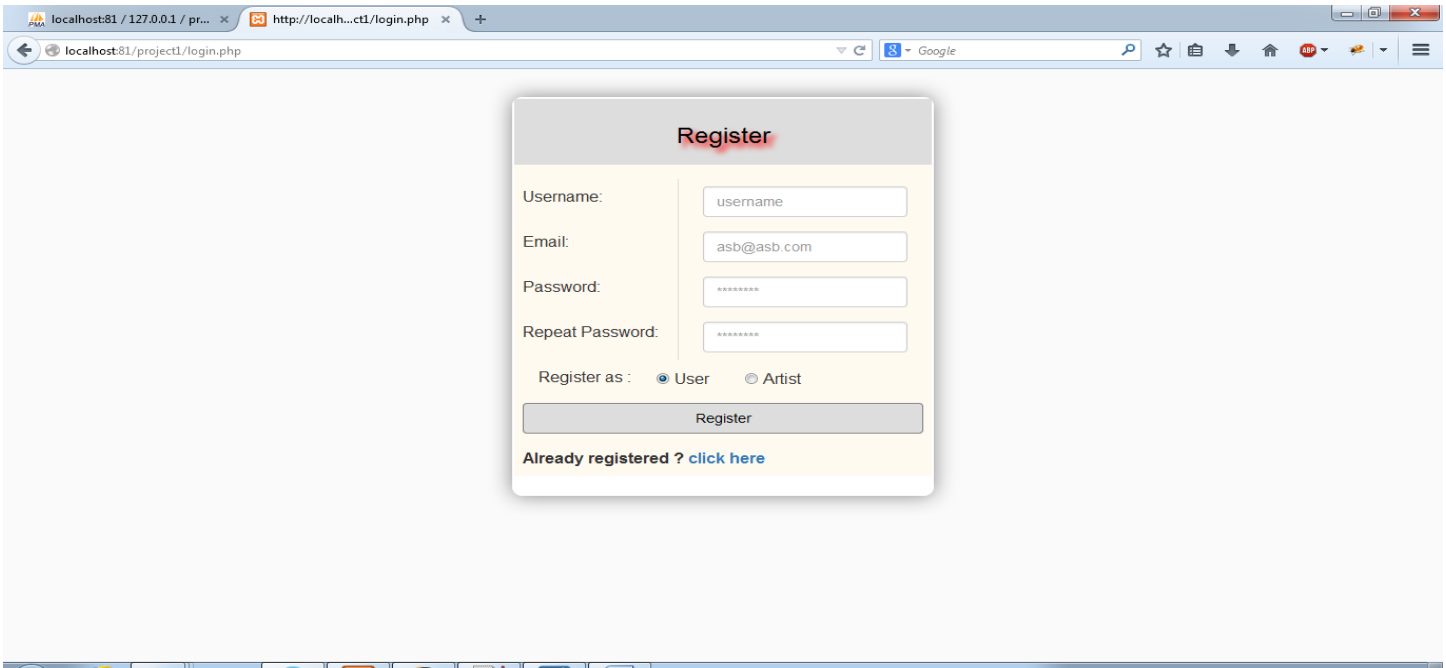

MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
concert_id
list_time
Indexes
Foreign Keys
Triggers
user_fan
user_follows
Columns
user_id
follow_id
follow_time
Indexes
Foreign Keys
Triggers
user_genre
Columns
user_id
genre_id
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Management
Schemas
Information
Column: genre_id
Definition:
genre_id int(11) PK
Object Info
Session
Query 1
post_concert - Routine
post_concert_user - Routine
add_recommended_concert - R...
Limit to 1000 rows
call recommend_genre(2,'blues');
Output
Action Output
Time Action Message Duration / Fetch
1 05:12:01 CREATE PROCEDURE `recommend_genre` (IN userid INT(11),IN genrename VARCHAR(100)) ... 0 row(s) affected 0.000 sec
2 05:12:23 call recommend_genre(2,'blues') 0 row(s) affected 0.000 sec
Query Completed

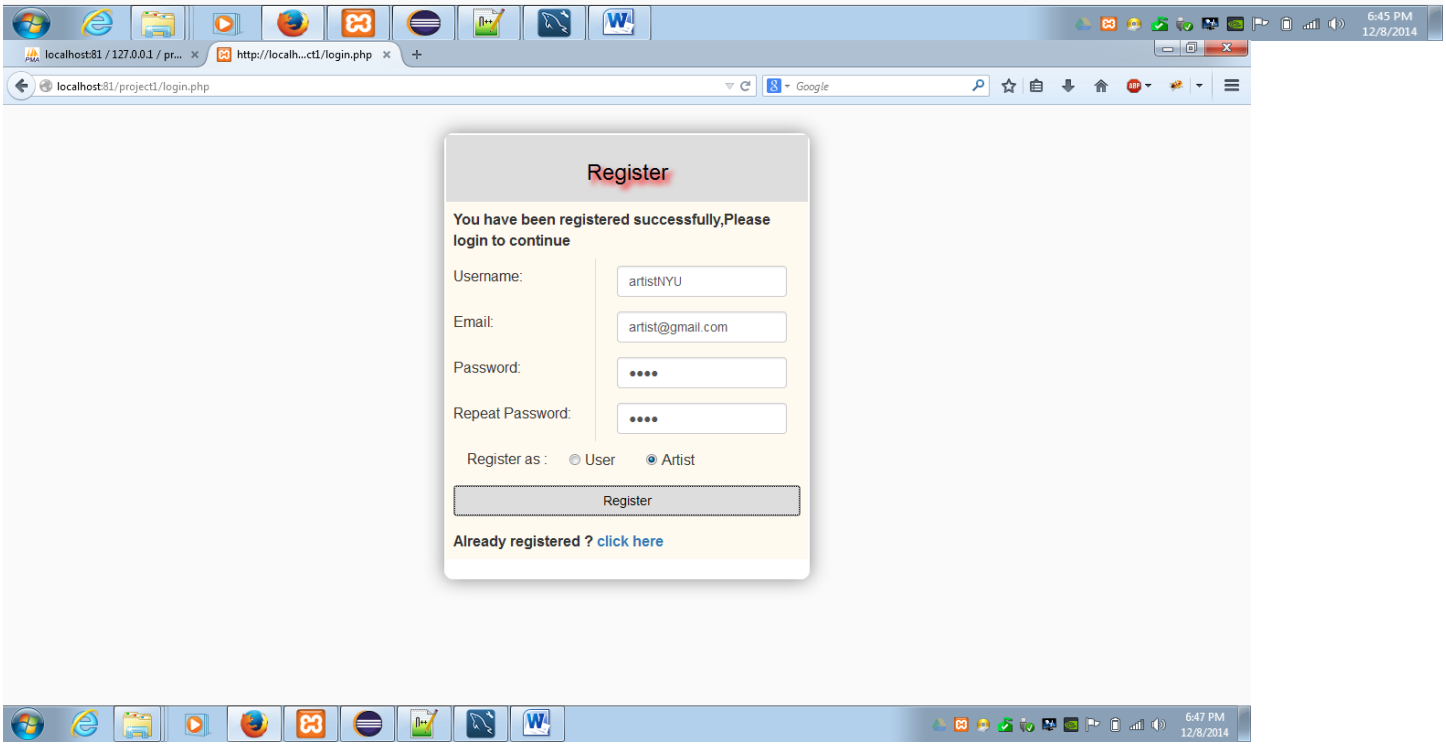
### 4.4.2 RECOMMEND BANDS THAT ARE HIGHLY LIKED BY USERS WITH SIMILAR TASTE

```
DELIMITER //
CREATE PROCEDURE `recommend_band` (IN userid INT(11))
BEGIN

DECLARE userid INT(11);
DECLARE concertid INT(11);
DECLARE artistid INT(11);
DECLARE concertname VARCHAR(100);
DECLARE v_finished INTEGER DEFAULT 0;
DECLARE v_finished_user INTEGER DEFAULT 0;
DECLARE v_finished_concert INTEGER DEFAULT 0;

DEClARE userid_cursor CURSOR FOR select  distinct u1.user_id, u2.user_id from user_genre u1,user_genre u2
where u1.genre_id=u2.genre_id and u1.user_id<>u2.user_id order by u1.user_id;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
OPEN userid_cursor;
get_userid: LOOP
FETCH userid_cursor INTO userid;
IF v_finished = 1 THEN
LEAVE get_userid;
END IF;
                BEGIN
                create temporary table temp1(concert_name varchar(100));
BEGIN
DEClARE artistid_cursor CURSOR FOR select artist_id from user_fan where user_id=userid;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished_user = 1;
OPEN artistid_cursor;
get_artistid: LOOP
FETCH artistid_cursor INTO artistid;
IF v_finished_user = 1 THEN
LEAVE get_artistid;
END IF;
select concert_name into concertname from concert where artist_id=artistid;
insert into temp1 values(concertname);
END LOOP get_artistid;
CLOSE artistid_cursor;
END;
BEGIN
DEClARE concertid_cursor CURSOR FOR select concert_id from concert_review where user_id=userid and rating>5;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished_concert = 1;
OPEN concertid_cursor;
get_concertid: LOOP
FETCH concertid_cursor INTO concertid;
IF v_finished_concert = 1 THEN
LEAVE get_concertid;
END IF;
select concert_name into concertname from concert where concert_id=concertid;
insert into temp1 values(concertname);
END LOOP get_concertid;
CLOSE concertid_cursor;
                END;
                END;
END LOOP get_userid;
CLOSE userid_cursor;
select distinct concert_name from temp1;
END//
```


MySQL Workbench
Mysql@127.0.0.1:3306
MySQL Model (schema_diagram.m...
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
concert_id
list_time
Indexes
Foreign Keys
Triggers
user_fan
user_follows
Columns
user_id
follow_id
follow_time
Indexes
Foreign Keys
Triggers
user_genre
Columns
user_id
genre_id
Indexes
Foreign Keys
Triggers
Views
Management Schemas
Information
Column: genre_id
Definition:
genre_id int(11) PK
Object Info Session
Query 1
post_concert - Routine
post_concert_user - Routine
add_recommended_concert - R...
Limit to 1000 rows
END LOOP get_artistid;
CLOSE artistid_cursor;
END;
BEGIN
DECLARE concertid_cursor CURSOR FOR select concert_id from concert_review where user_id=userid and rating>5;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished_concert = 1;
OPEN concertid_cursor;
get_concertid: LOOP
FETCH concertid_cursor INTO concertid;
IF v_finished_concert = 1 THEN
LEAVE get_concertid;
END IF;
select concert_name into concertname from concert where concert_id=concertid;
insert into temp1 values(concertname);
END LOOP get_concertid;
CLOSE concertid_cursor;
END;
END;
END LOOP get_userid;
CLOSE userid_cursor;
select distinct concert_name from temp1;
END//
Output
Action Output
Time Action Message Duration / Fetch
1 05:12:01 CREATE PROCEDURE `recommend_genre` (IN userid INT(11),IN genrename VARCHAR(100)) ... 0 row(s) affected 0.000 sec
2 05:12:23 call recommend_genre(2,'blues') 0 row(s) affected 0.000 sec
3 05:12:57 CREATE PROCEDURE `recommend_band` (IN userid INT(11)) BEGIN DECLARE userid I... 0 row(s) affected 0.000 sec
Query Completed

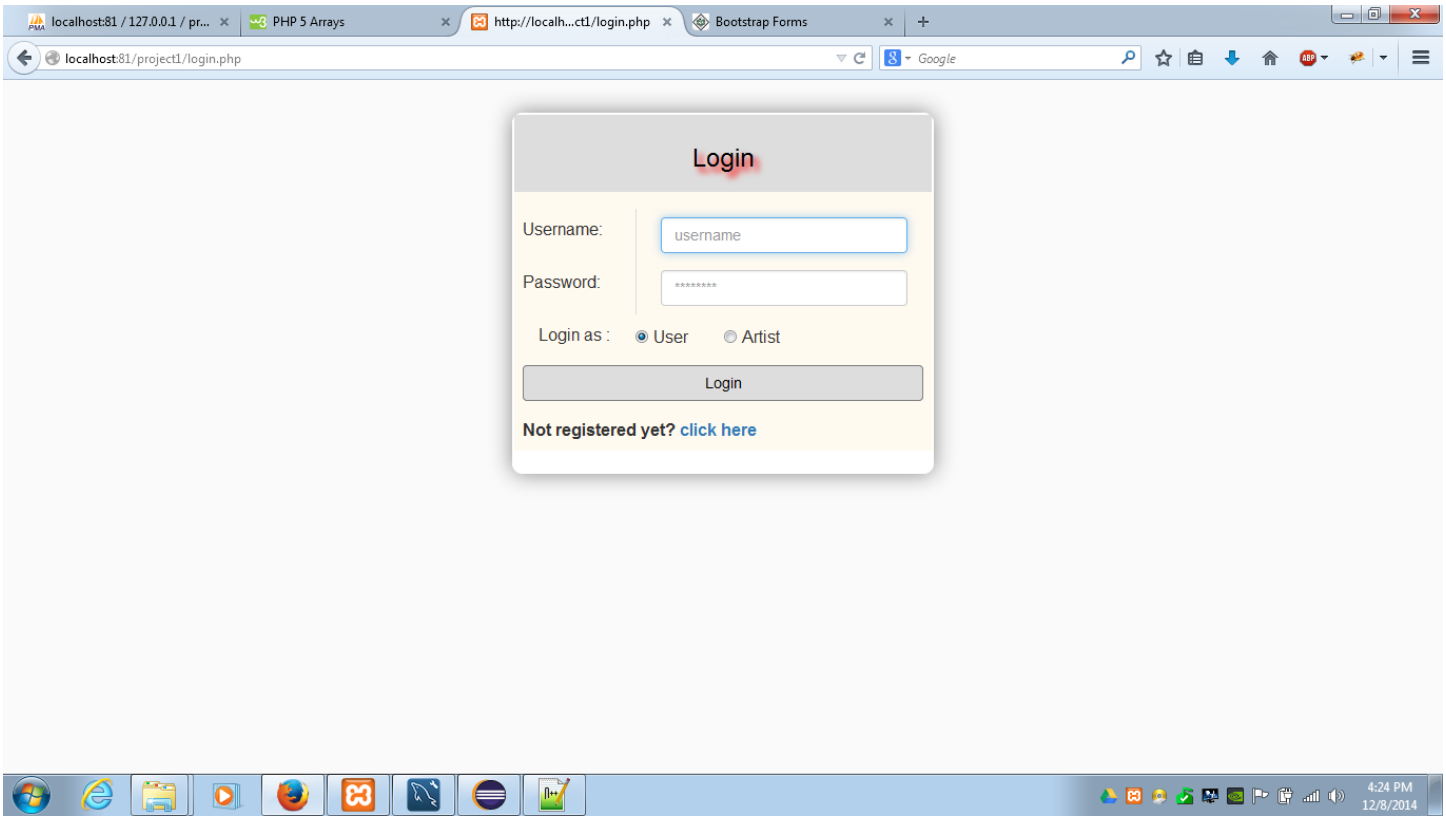
## 5) UI FLOW
## REGISTRATION



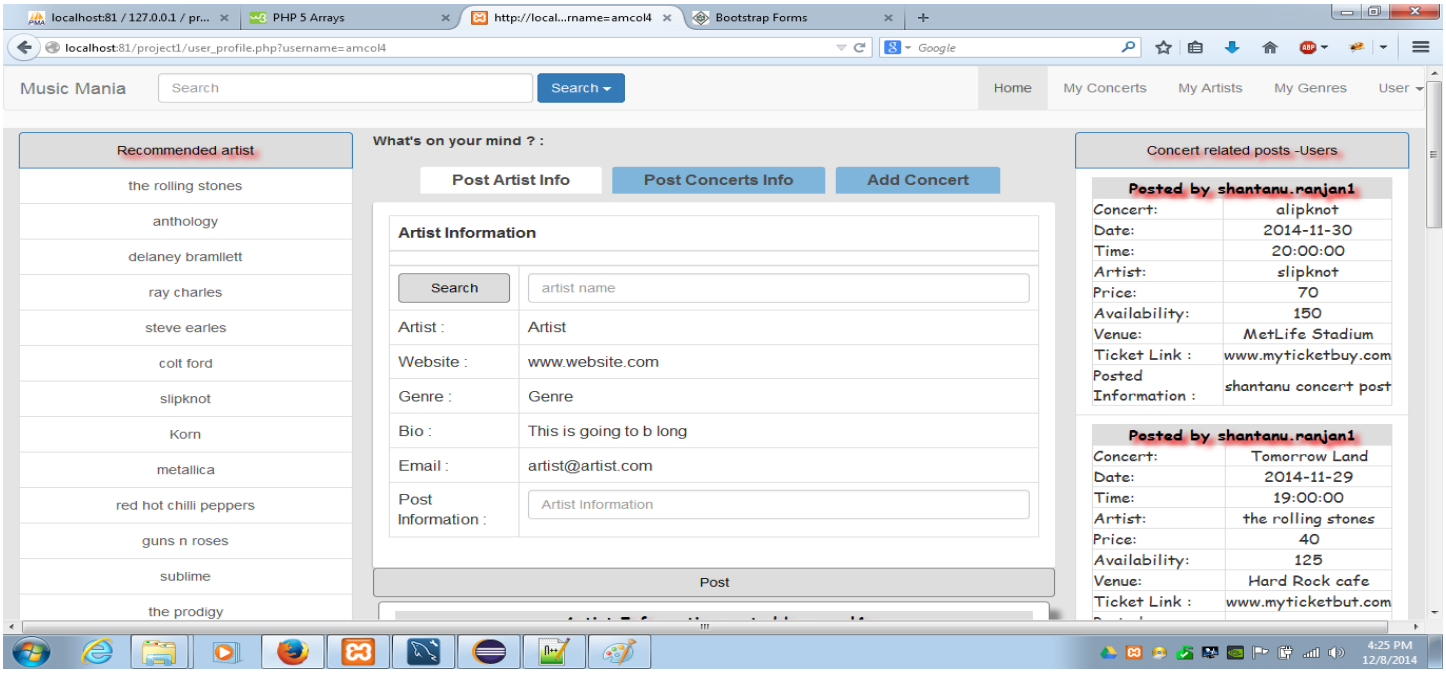

The system allows users/artist to register themselves on the system.Once successfully registered he could view his profile by logging into the system.

**LOGIN**



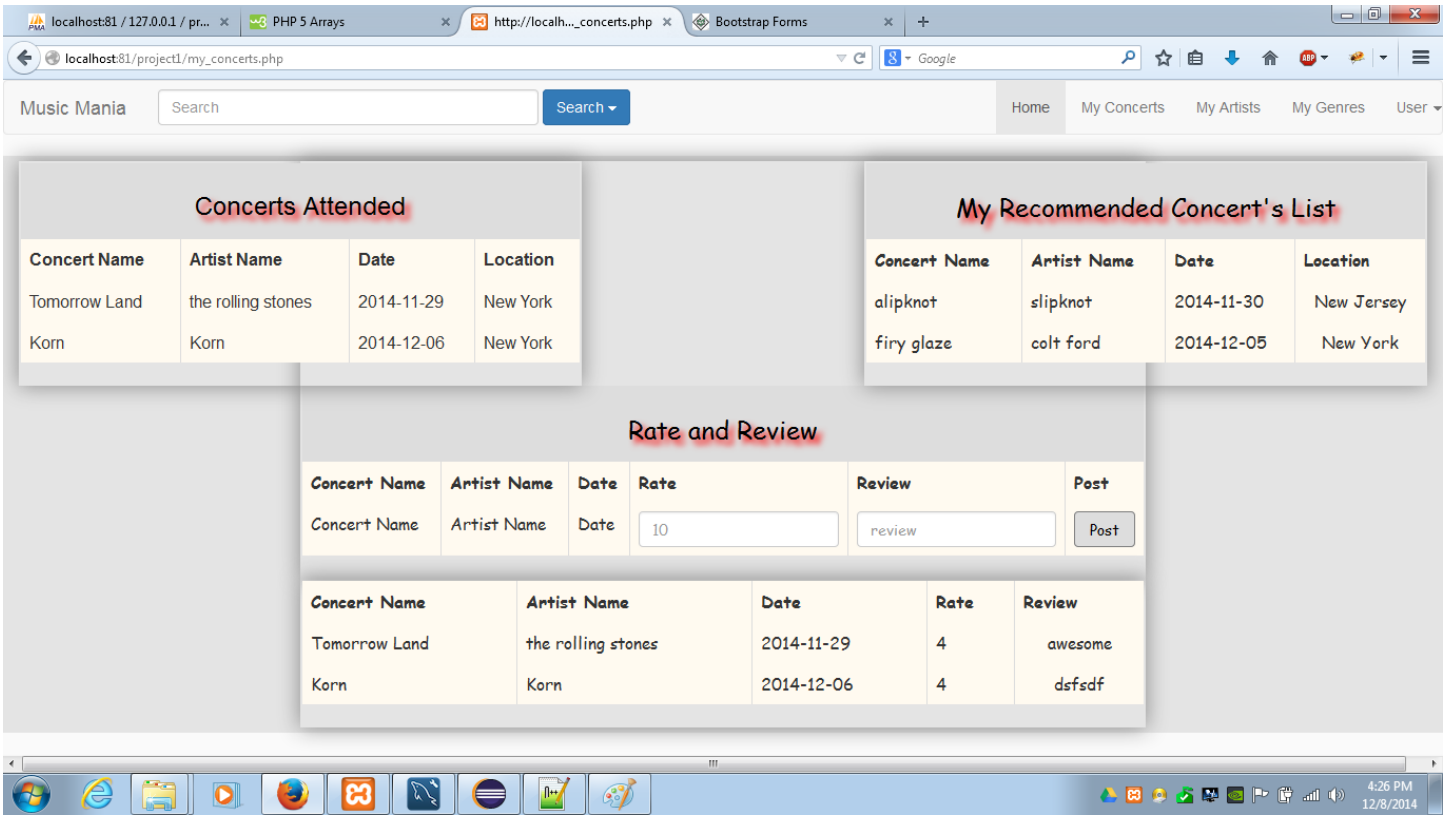

The user/artist could login from this page by entering their username and password.If the credentials does'nt match then an error message is shown.Once successfully logged in he is redirected to create profile page where he could provide his basic information,otherwise he is directed to profile page of user/artist.

**USER PROFILE**



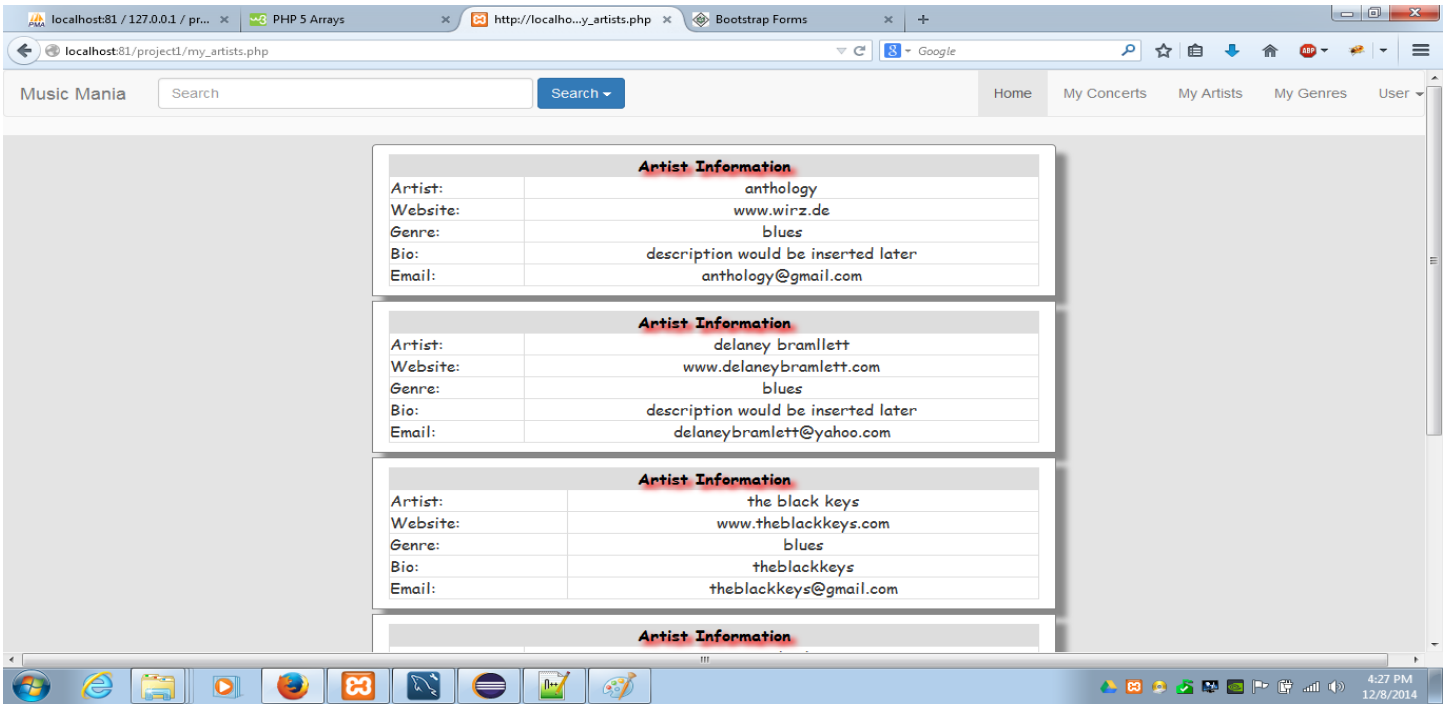
On this page the user could view his recommended artist list,concert information posted by other users /artists to whom he follows/fan.He could post artist information which would be visible to other users.He could also post concerted related information and add new concert to database.

**MY CONCERT**



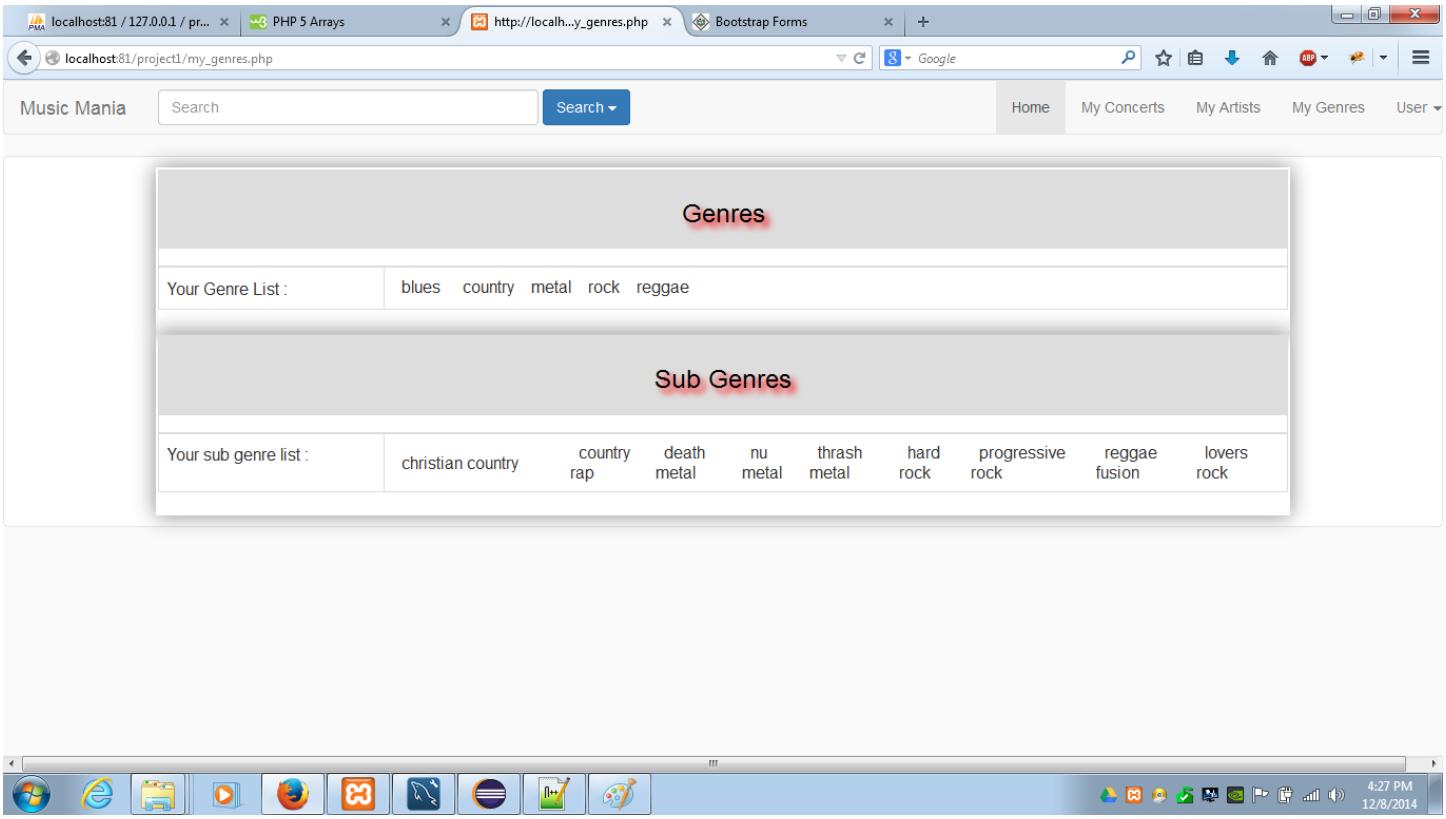

The user could see a list of concerts that he has already attended.He could also view his recommended list of concerts.The user could provide rating and review to visited concerts.

**MY ARTIST**



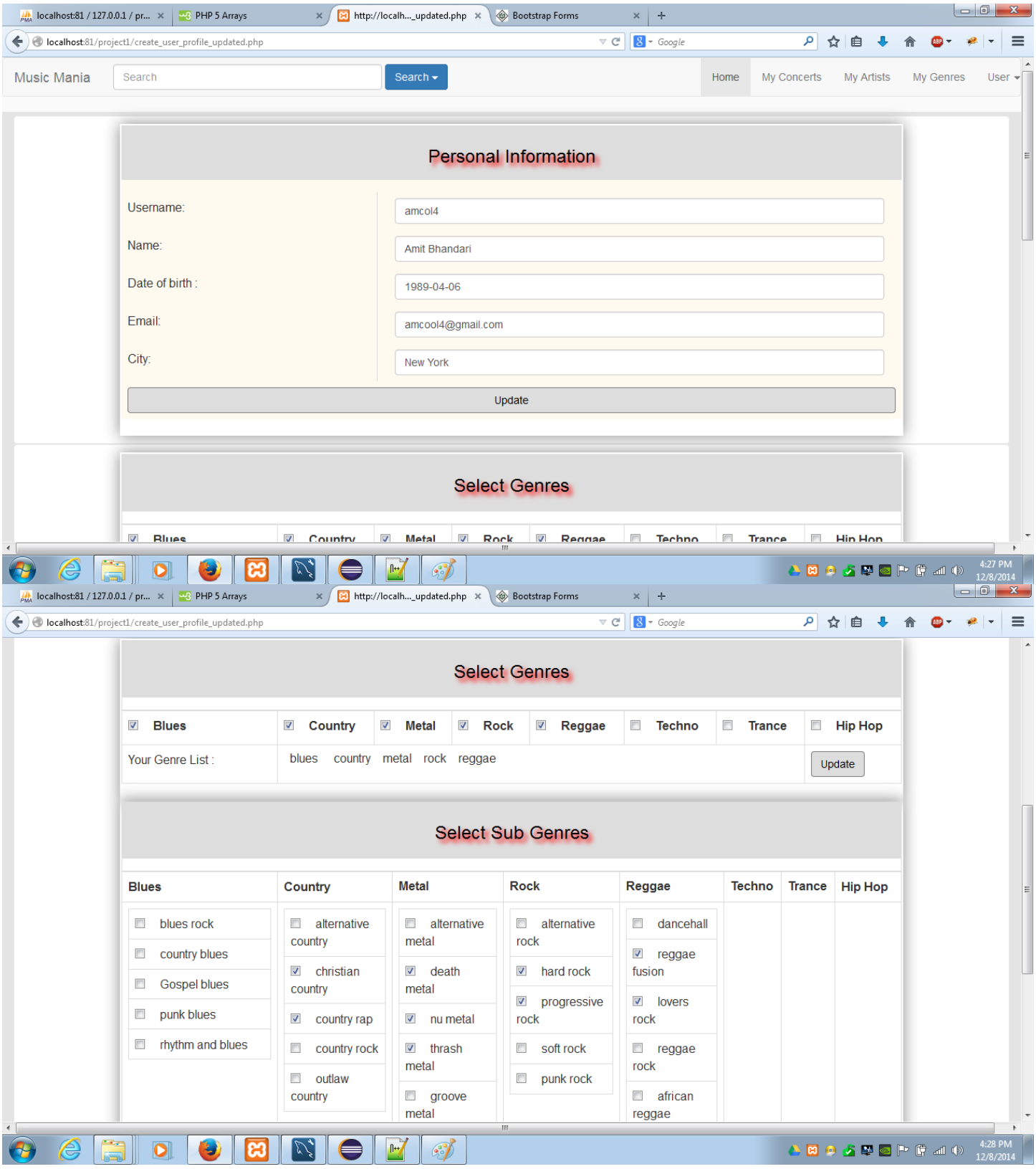

The user could view the list of artist to whom he follows and could visit their profile based on artist selection.

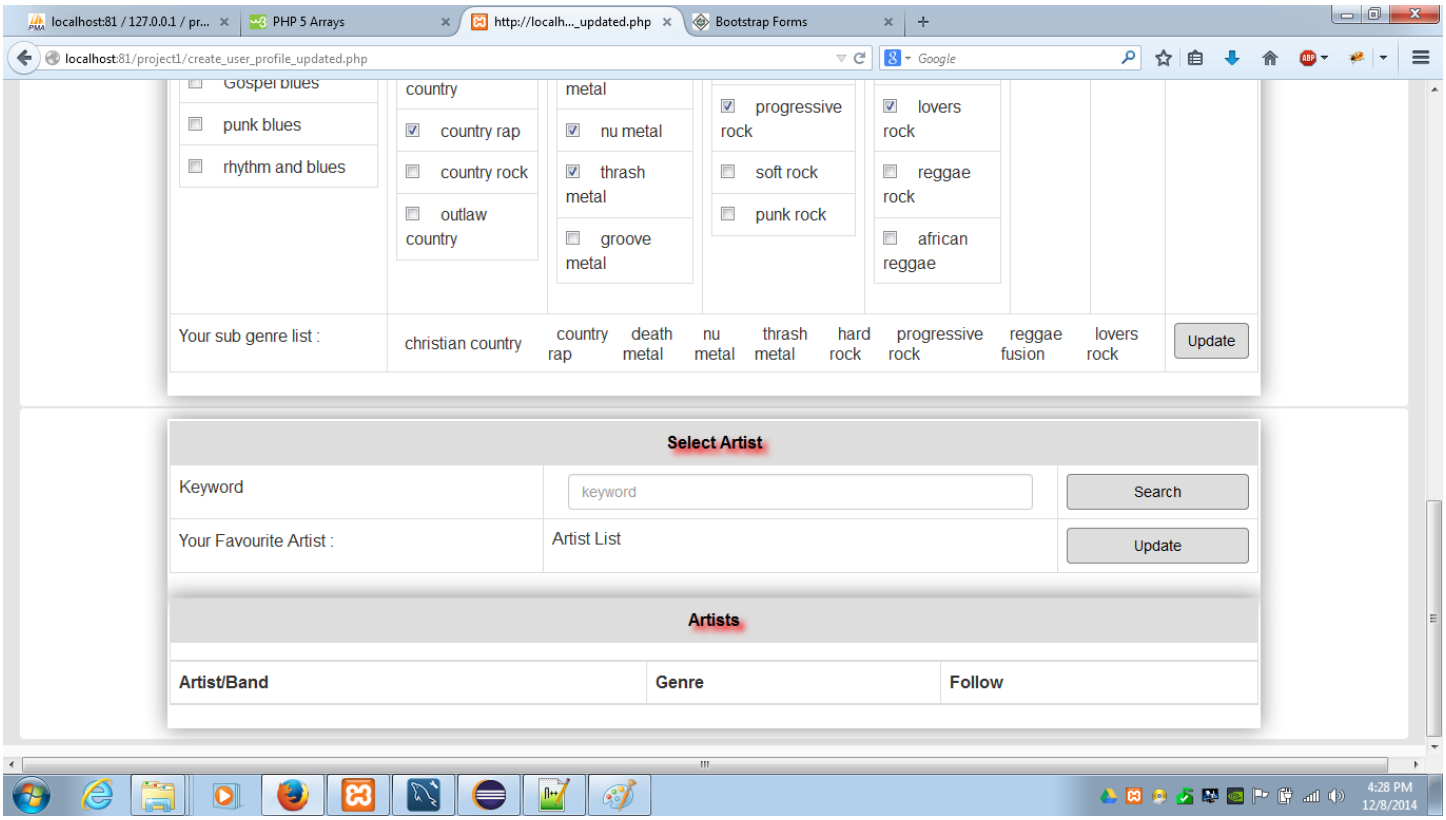
**MY GENRE**




The user could see a list of genre and sub genre that he likes.

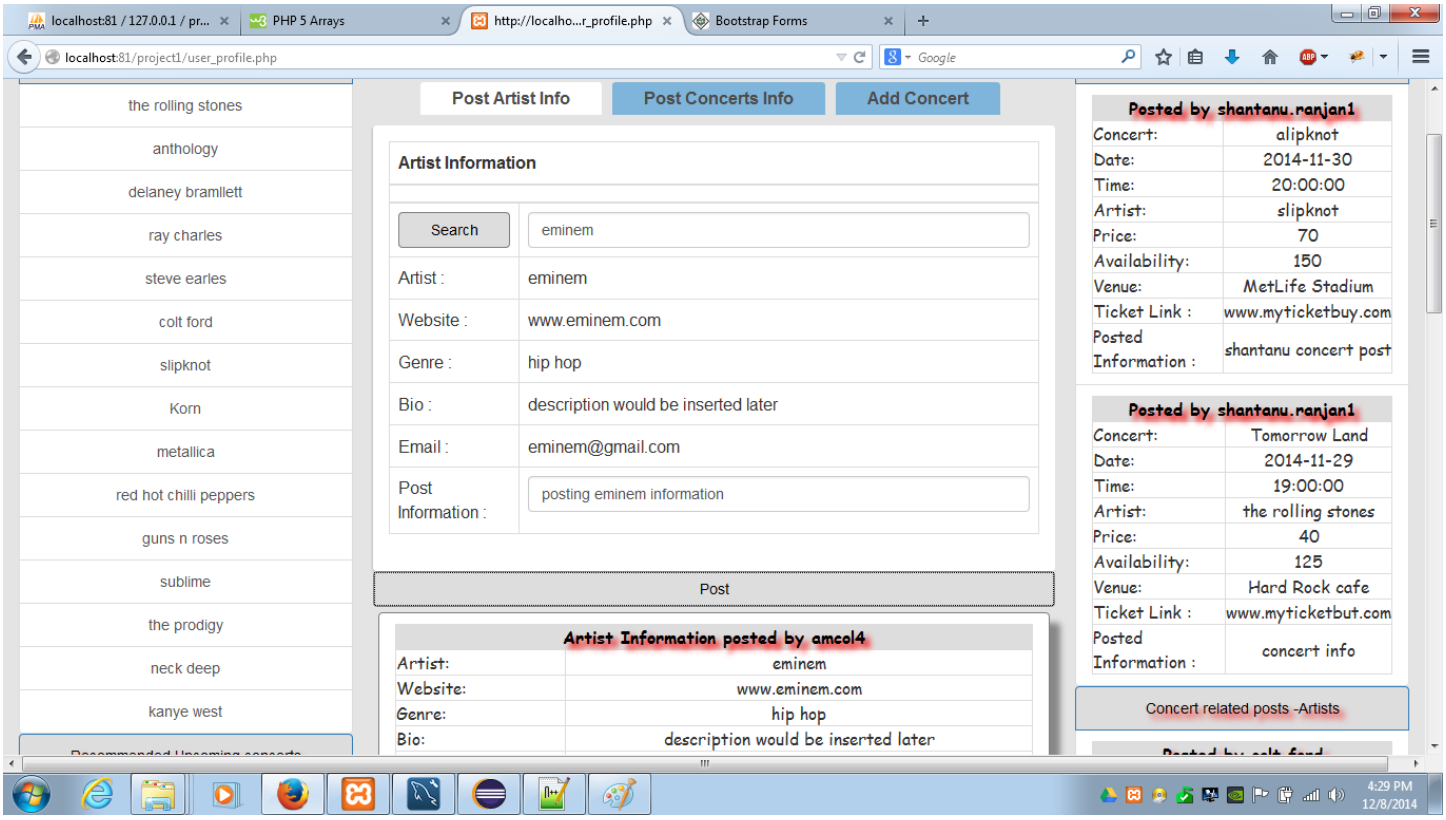
## UPDATE PROFILE





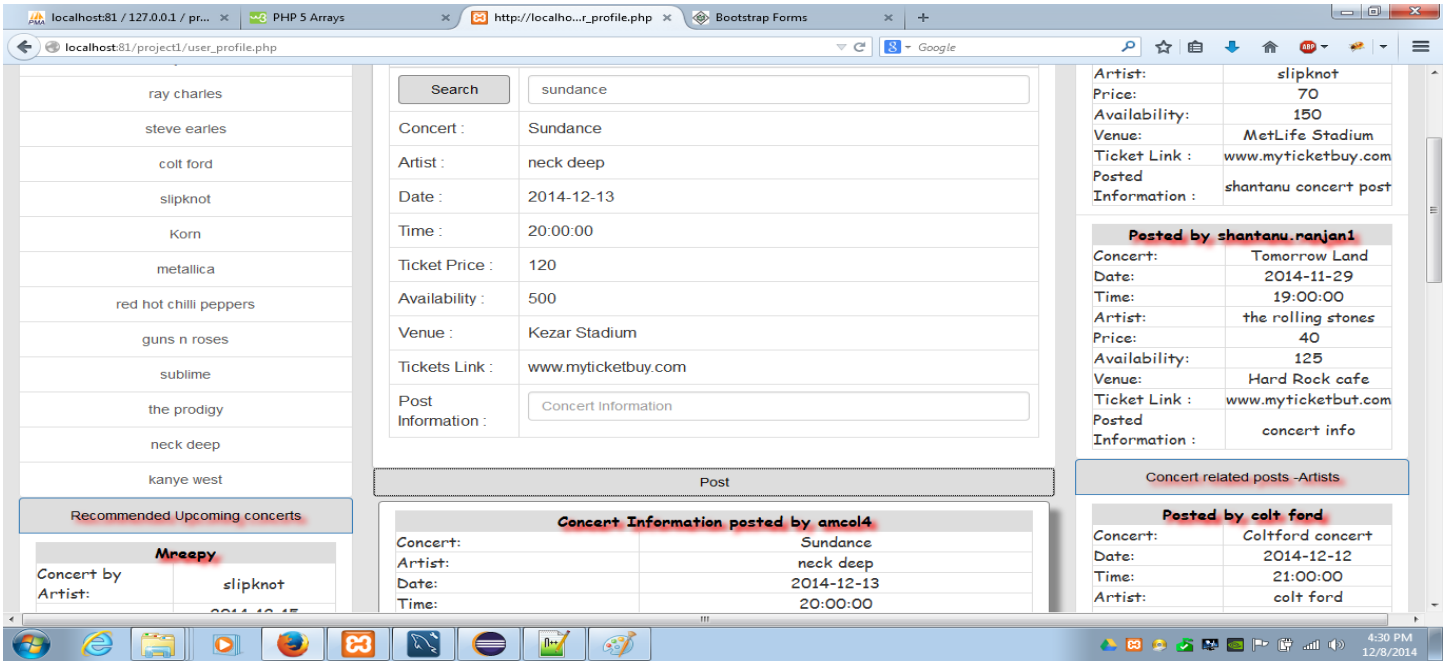

The user could update his profile once looged in.He could also like/unlike genre and could also follow Artist.

**POST ARTIST INFORMATION**



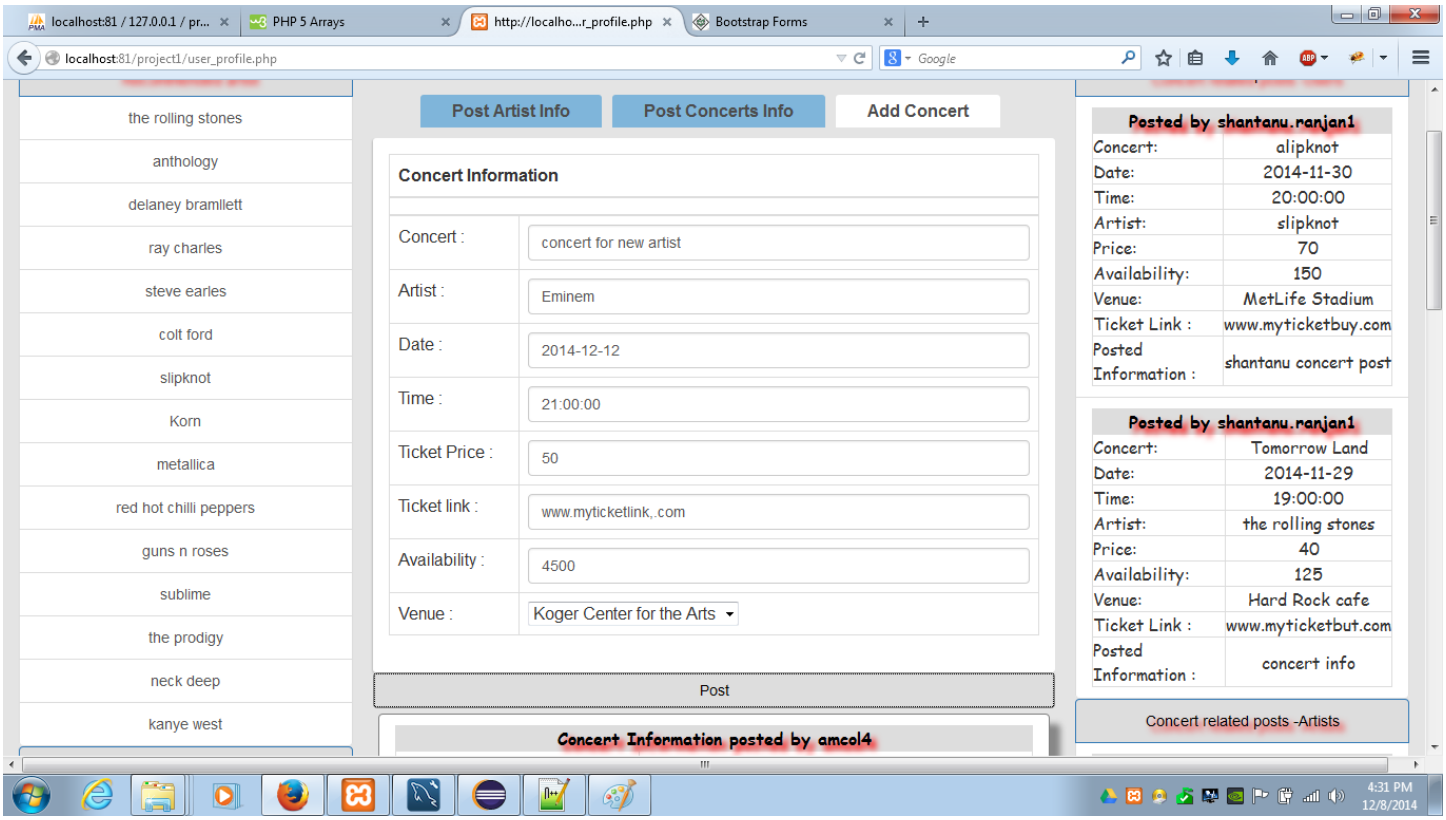
The user could post artist information searching about him.This information could then be posted under public posts and would be visible to other users.

**POST CONCERT INFORMATION**



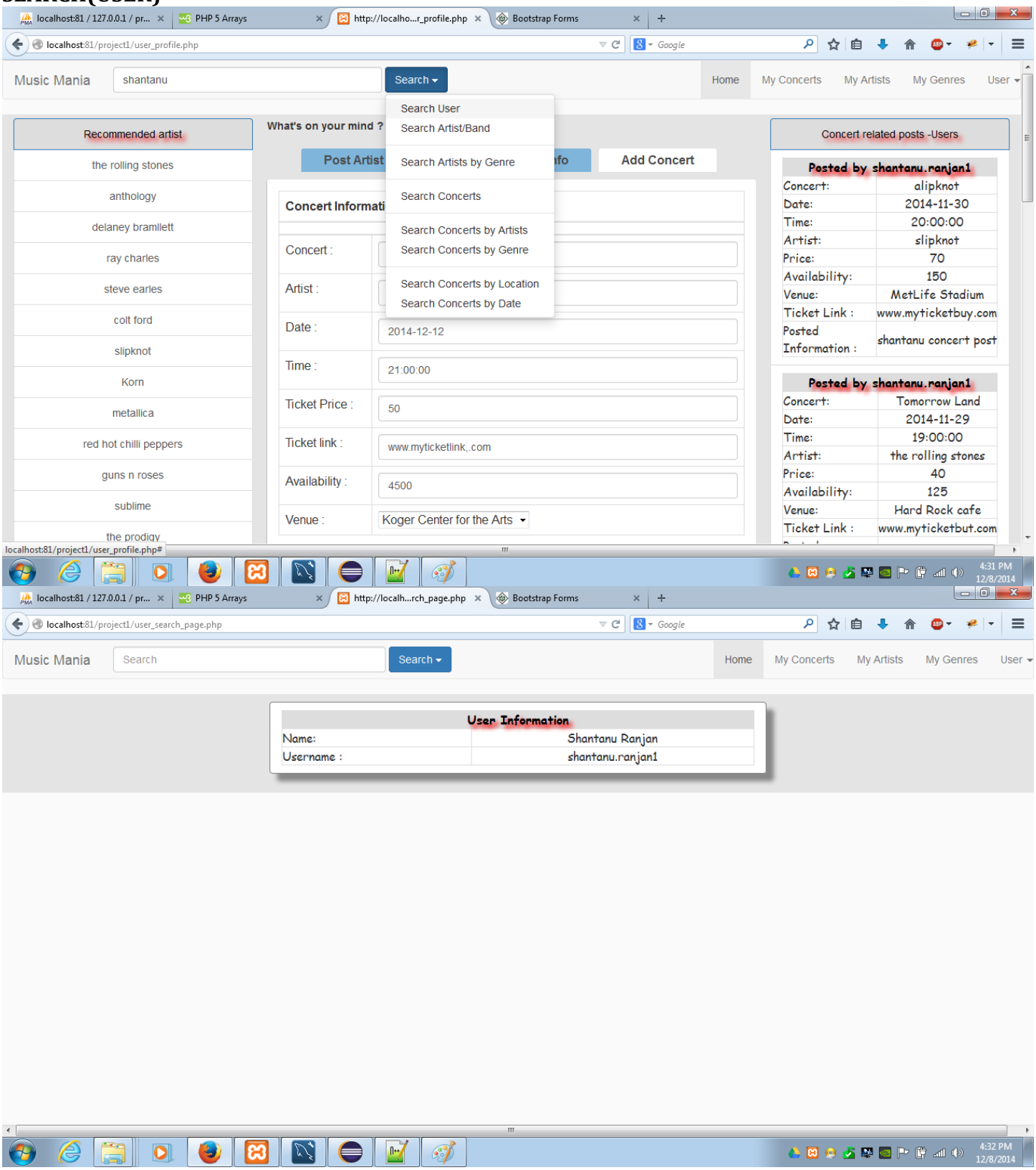

The user could post information about concert under public post and would be visible to other users.

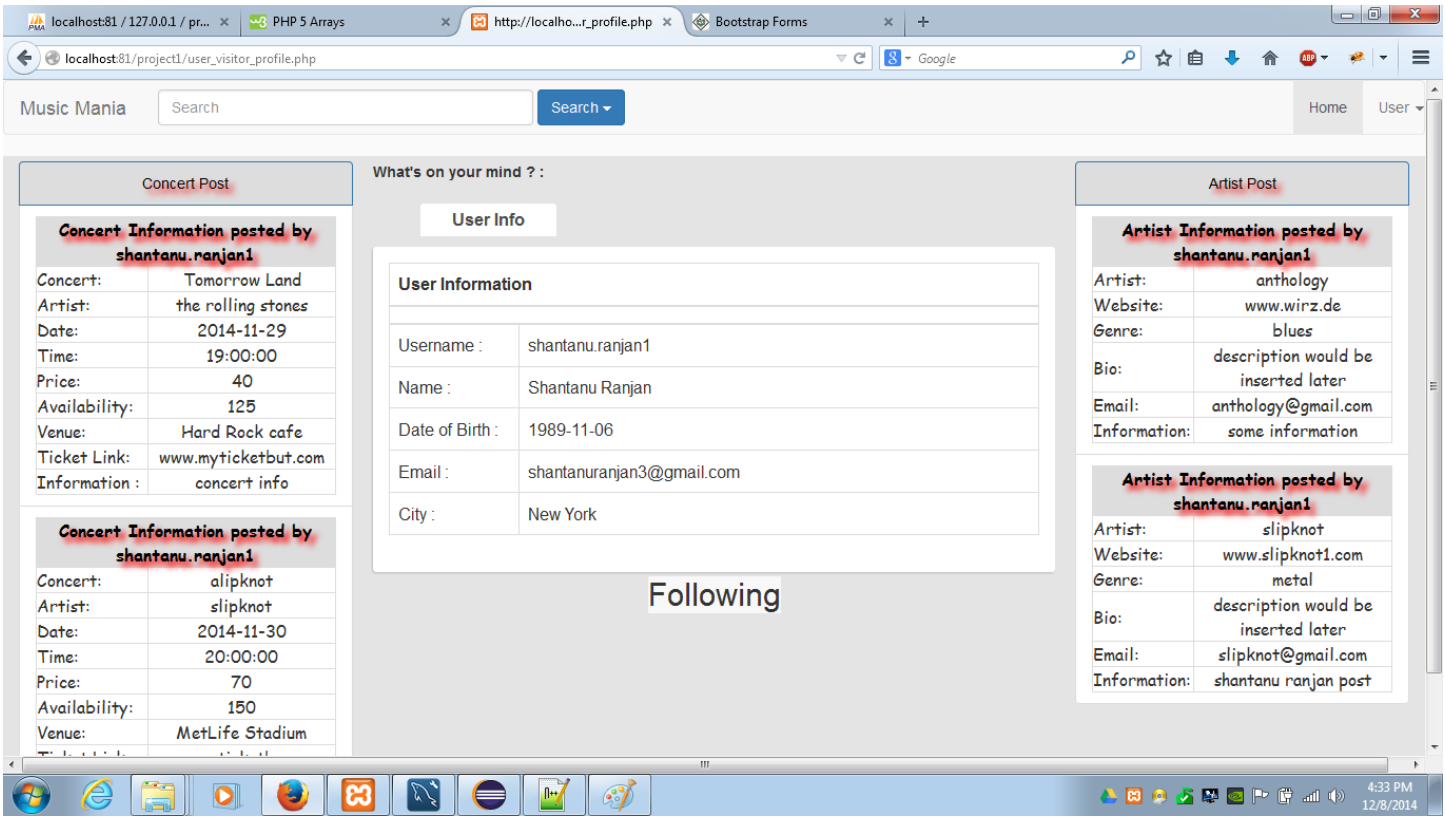
**ADD NEW CONCERT**



The user could add information about upcoming concert and would be visible to other users.

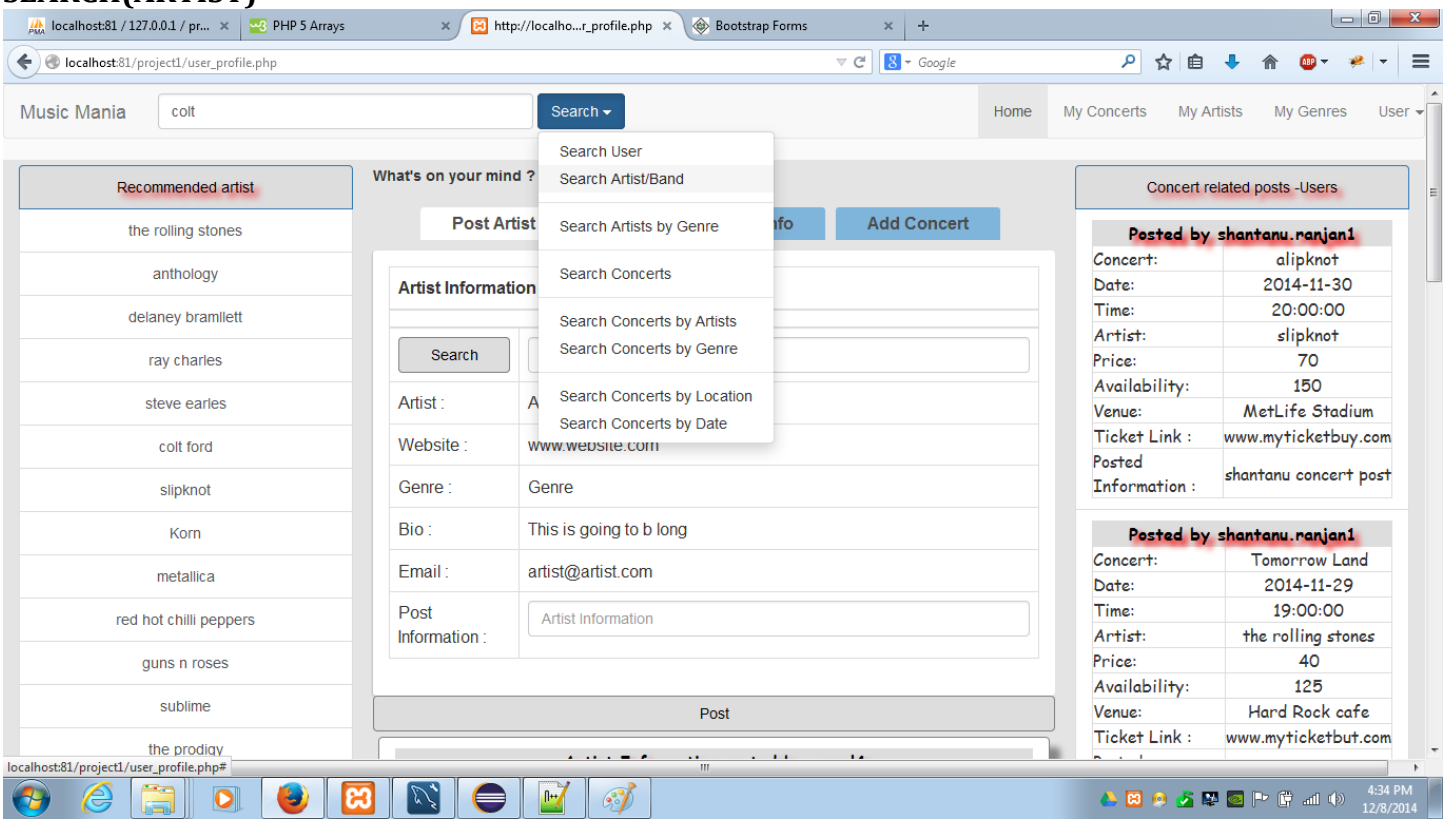
**SEARCH(USER)**

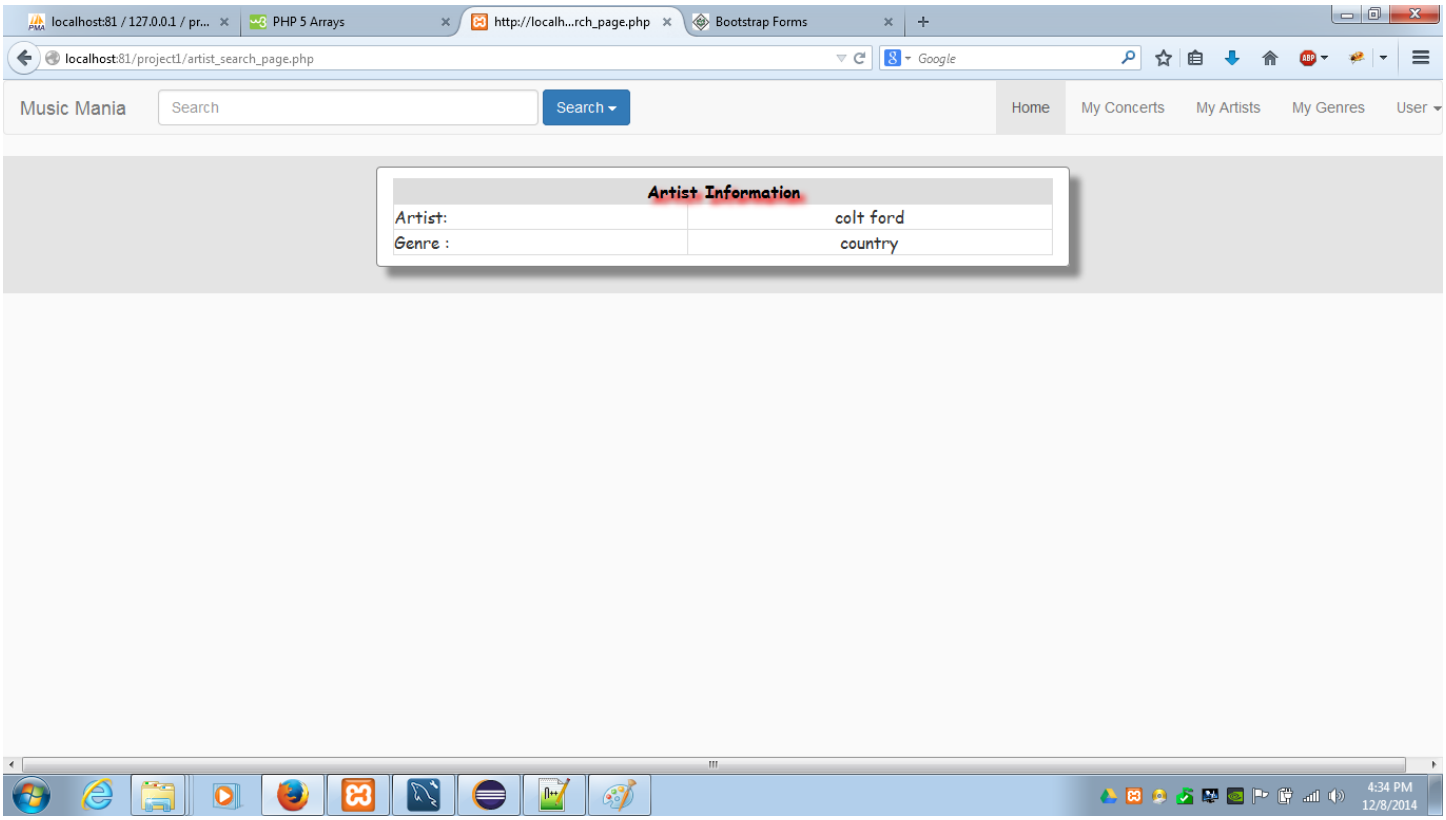


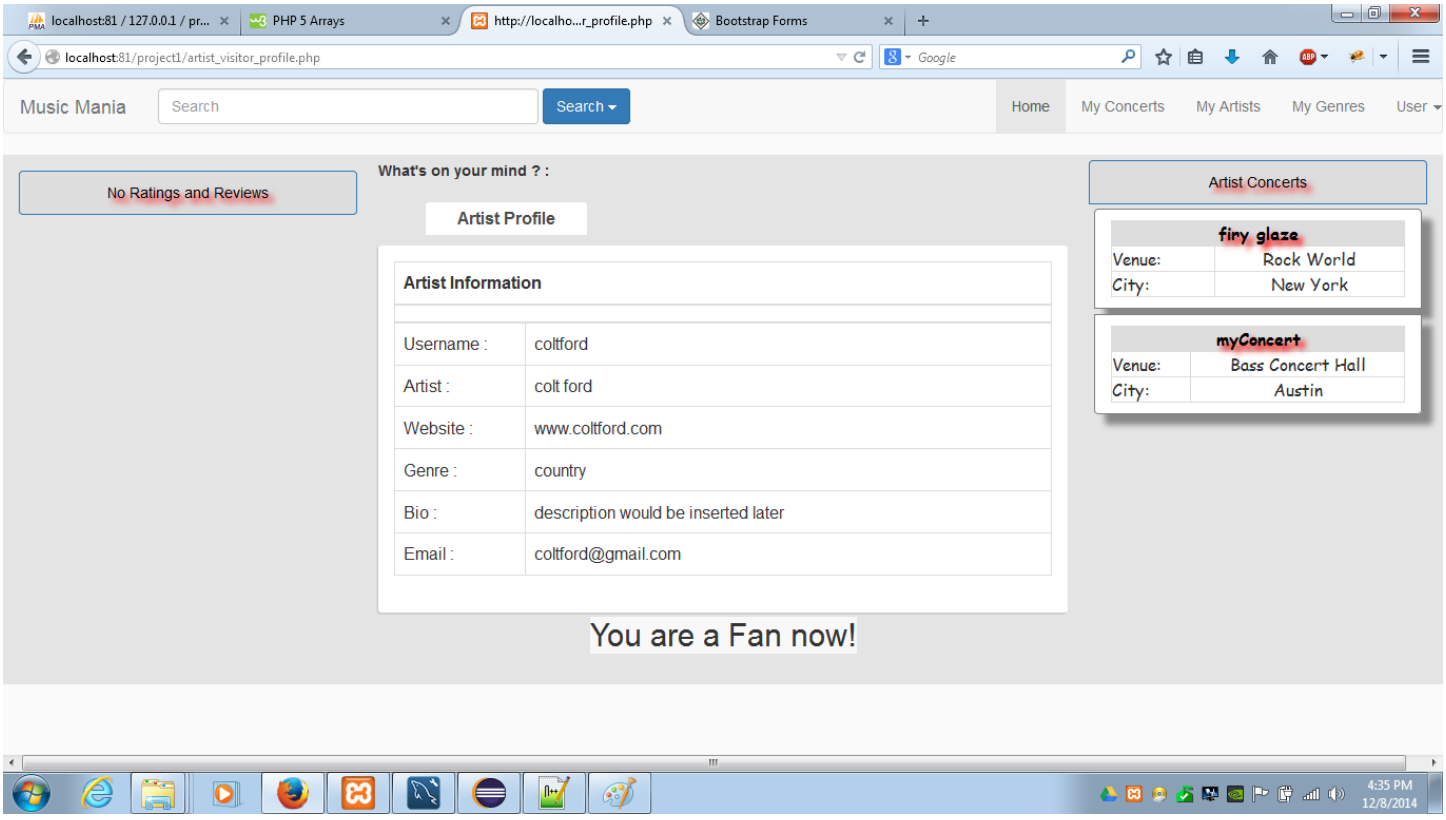



The user could search information about other users and could then visit their profile.

**SEARCH(ARTIST)**

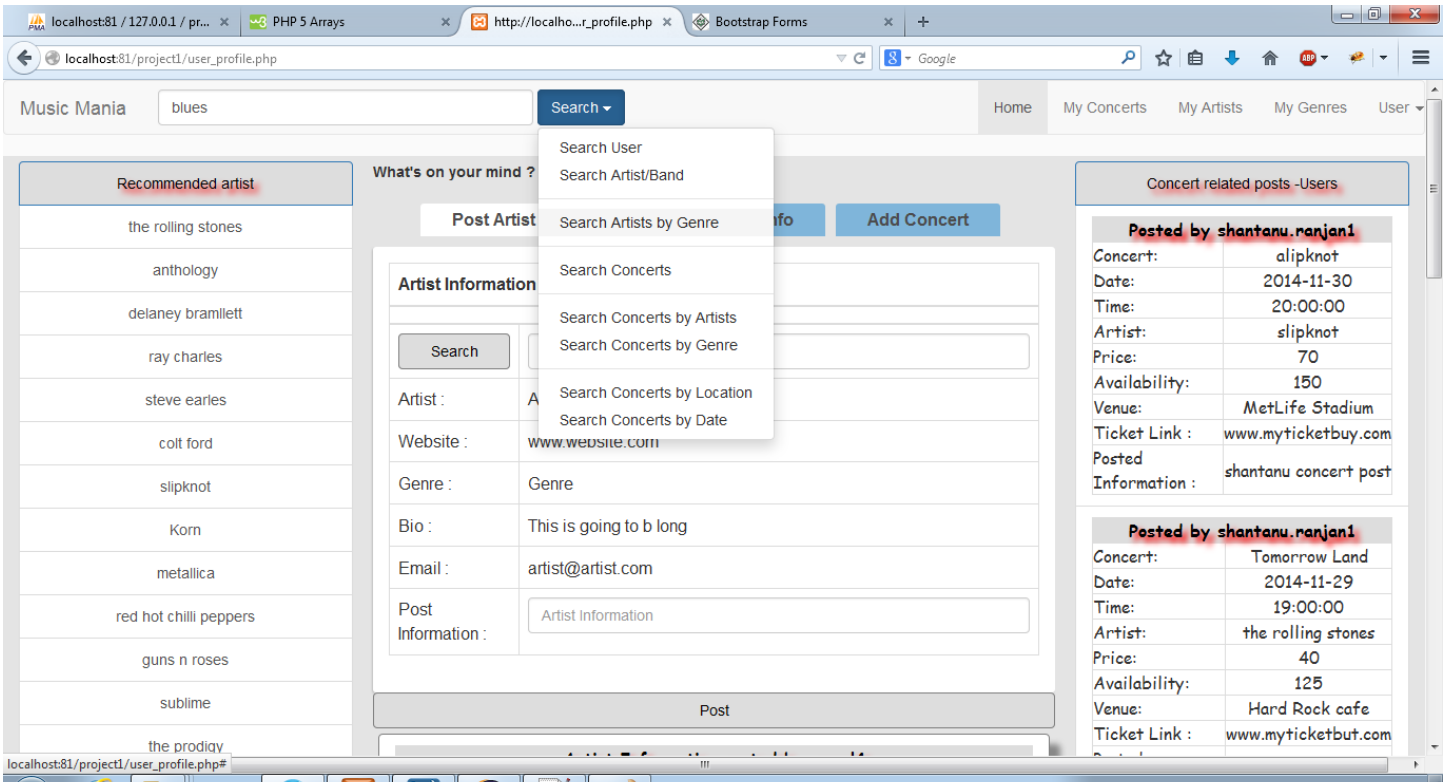


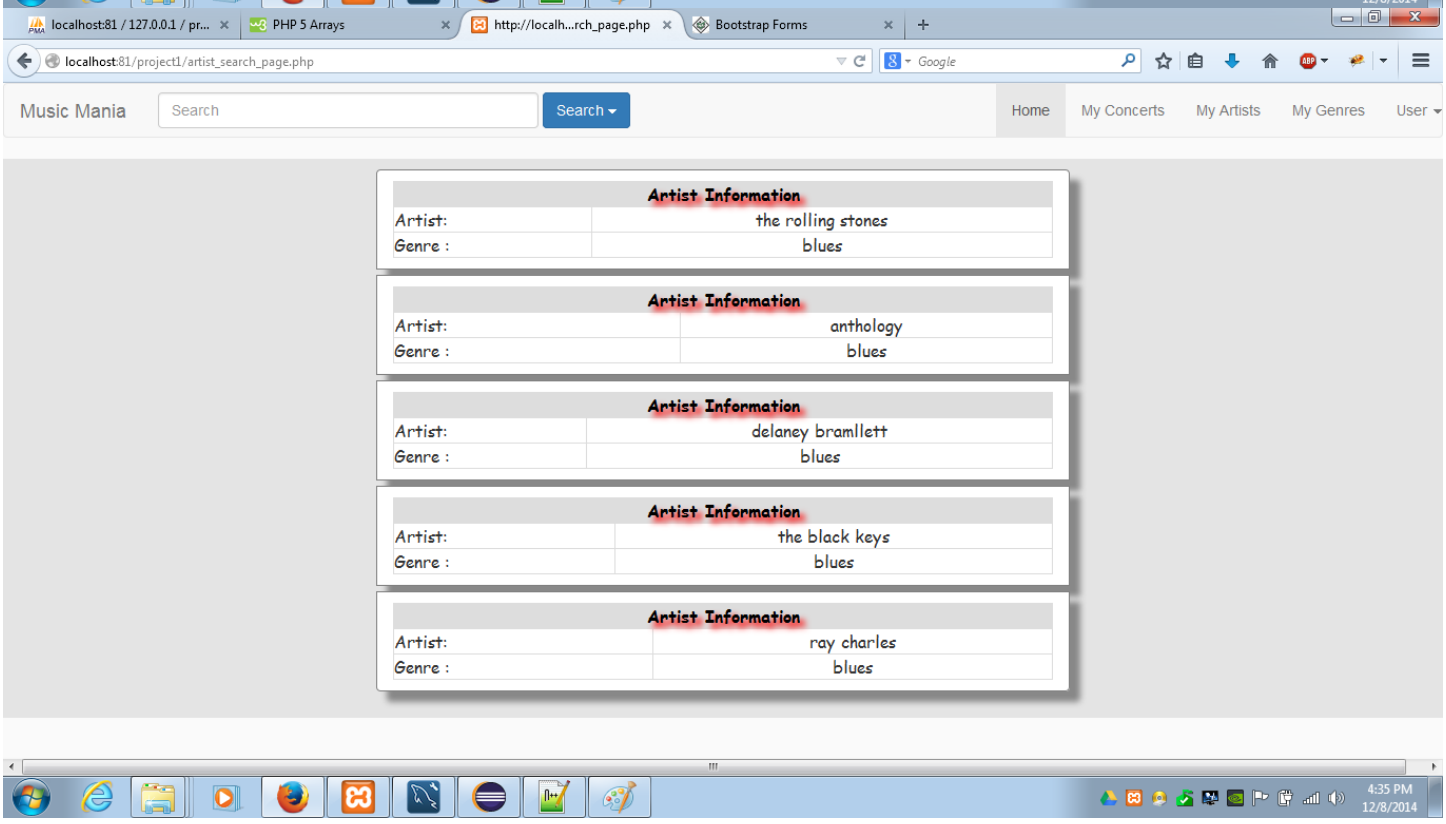


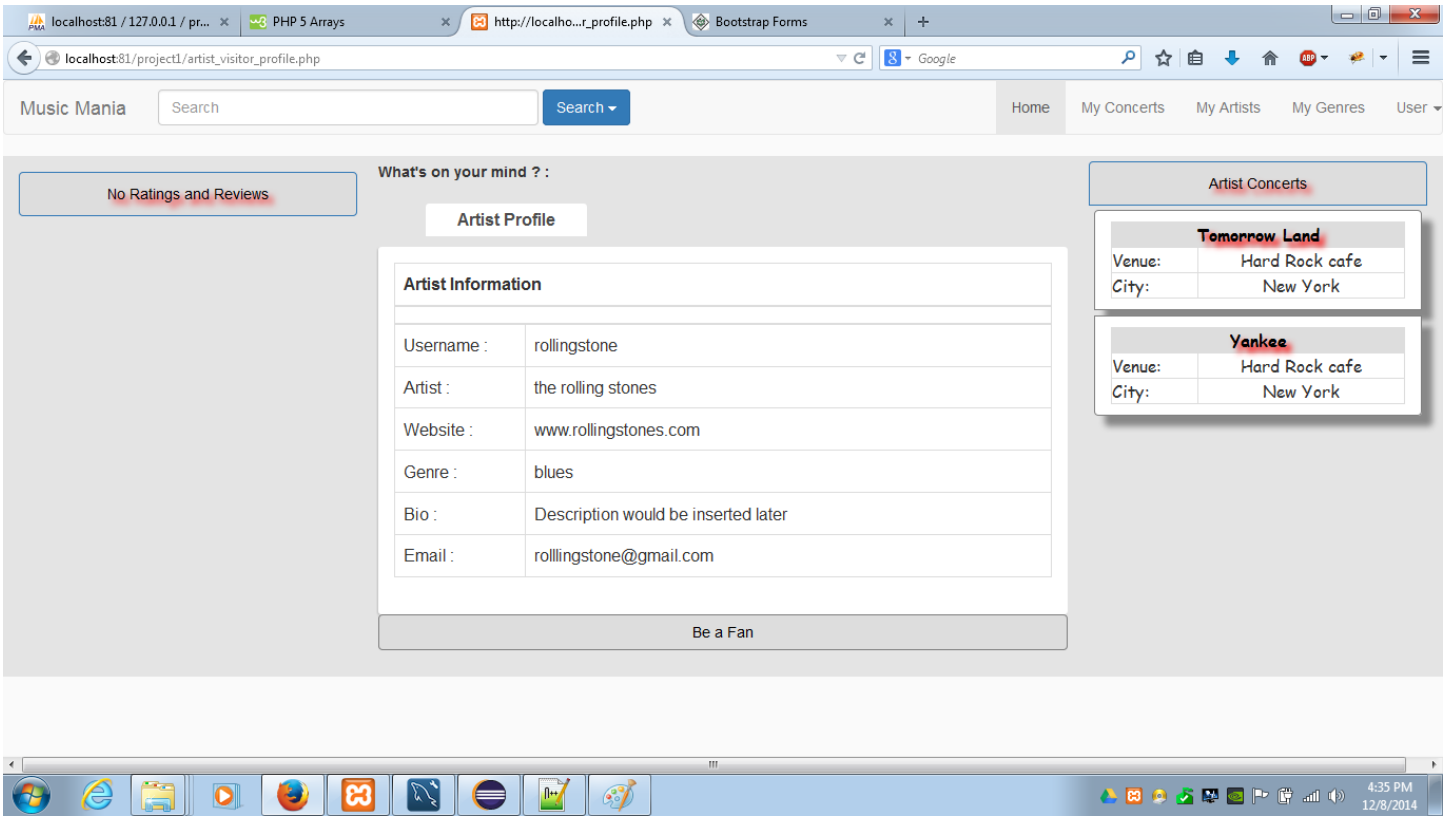




The user could search information about artist/band and could visit their profile.

## SEARCH ARTIST BY GENRE

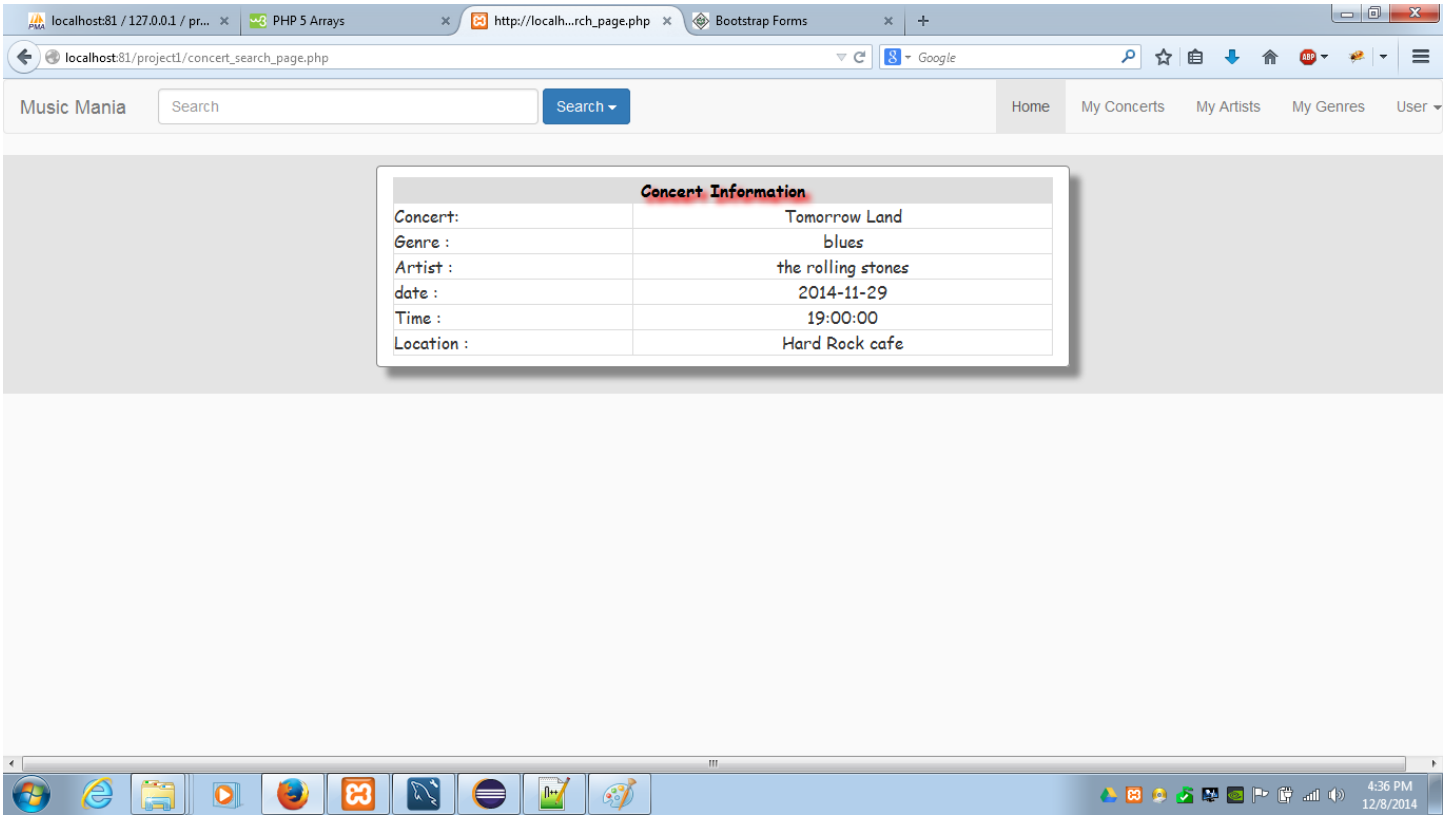




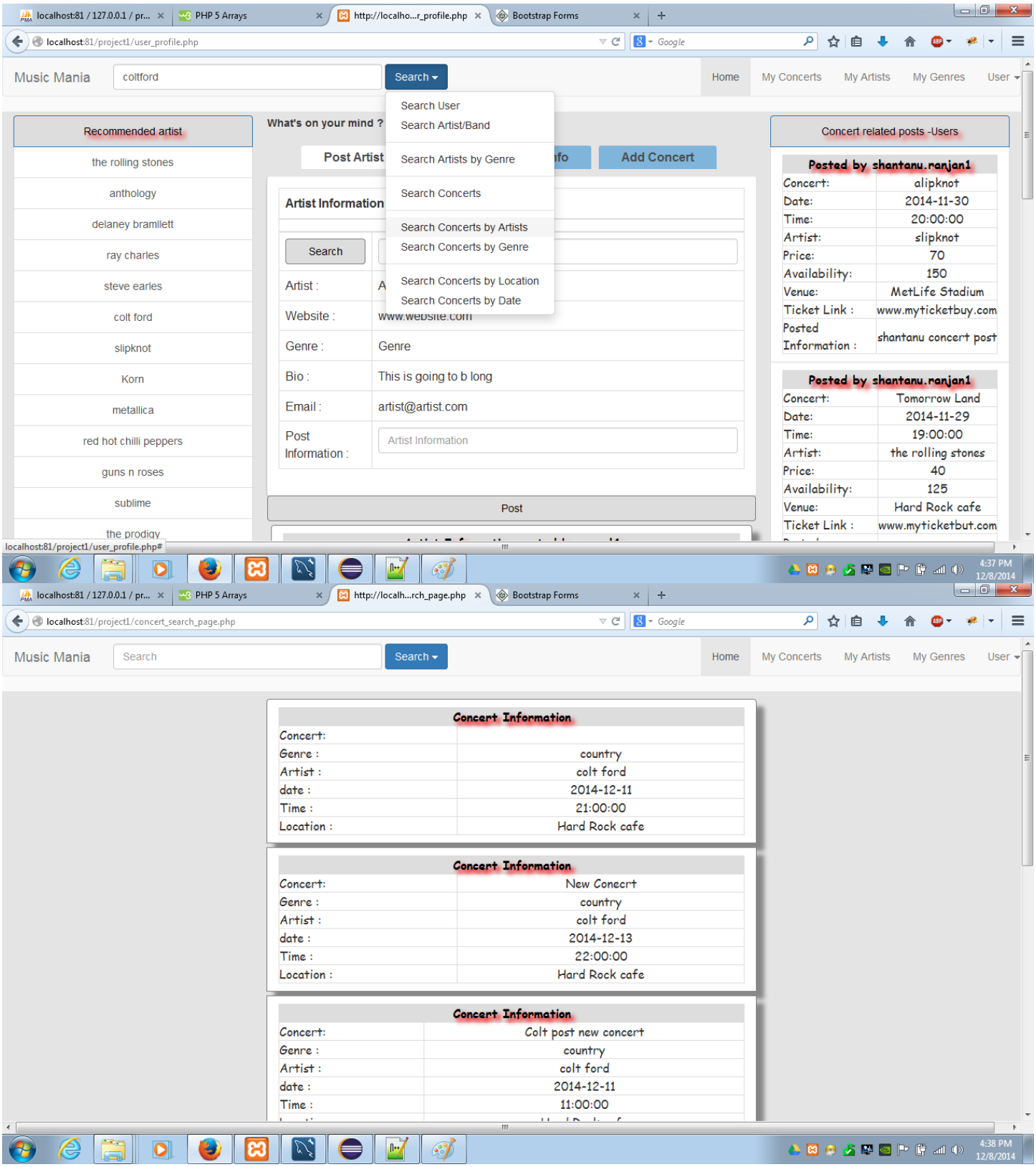

The user could search artist based on genre as shown in screenshot.He is directed to user search page where a list of artist gets populated and then upon clicking on artist ,he is redirected to artist visitor profile page.

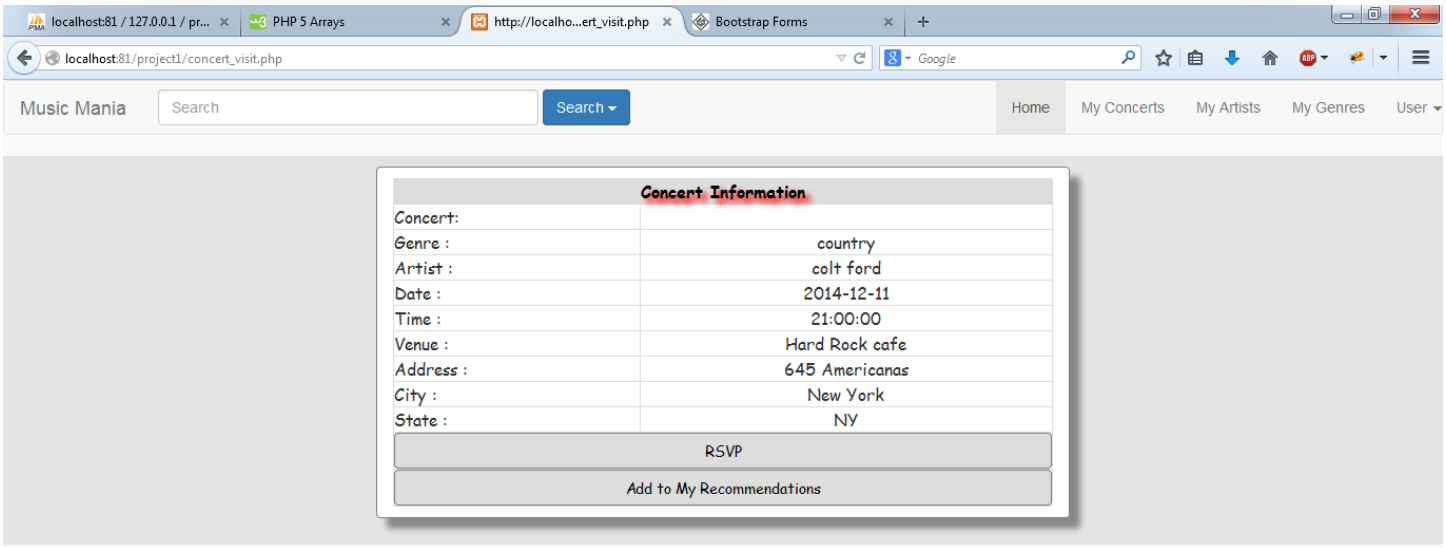
**SEARCH CONCERTS**

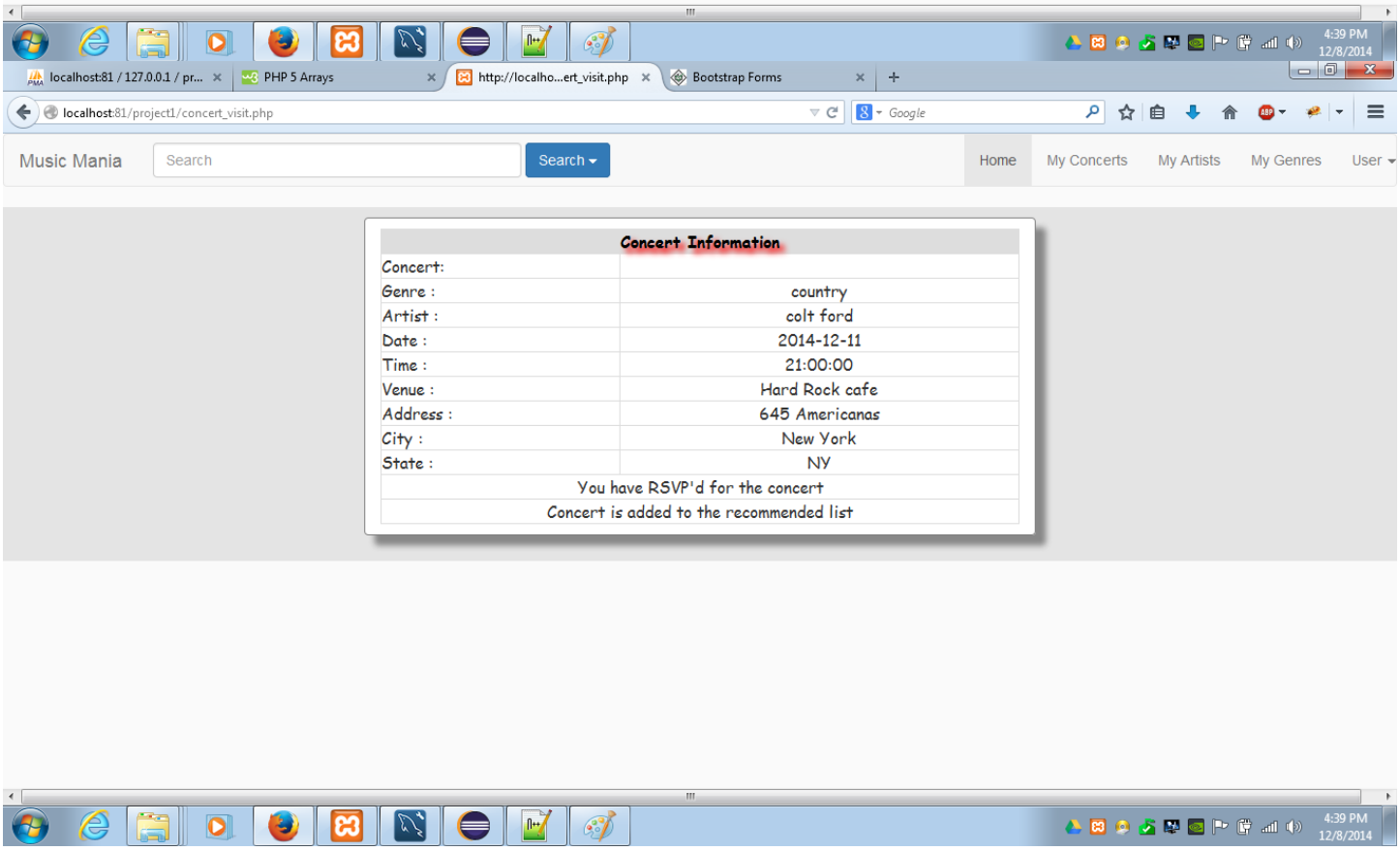


The user could search information about any particular concert.

**SEARCH CONCERT BY ARTISTS**

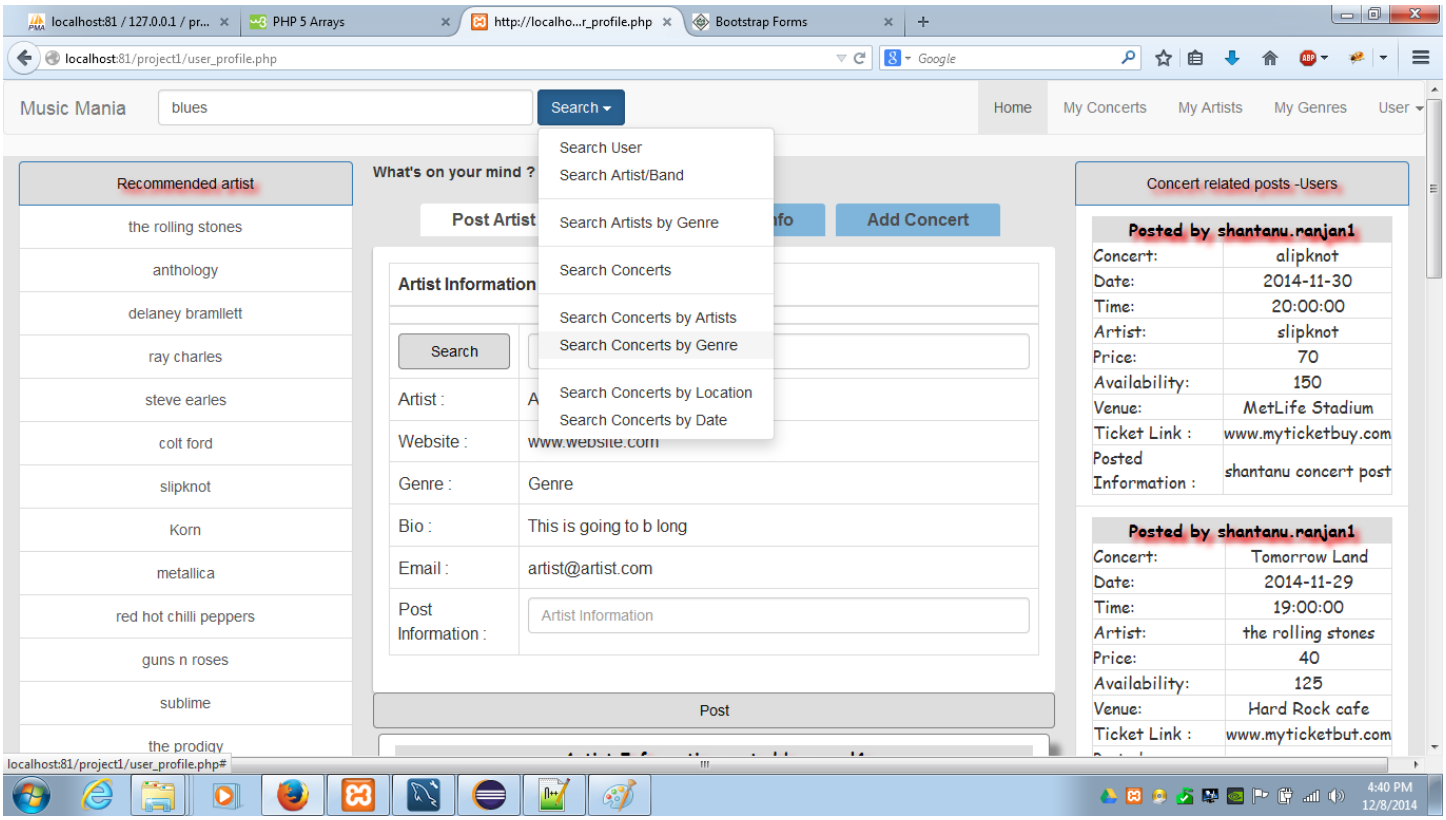


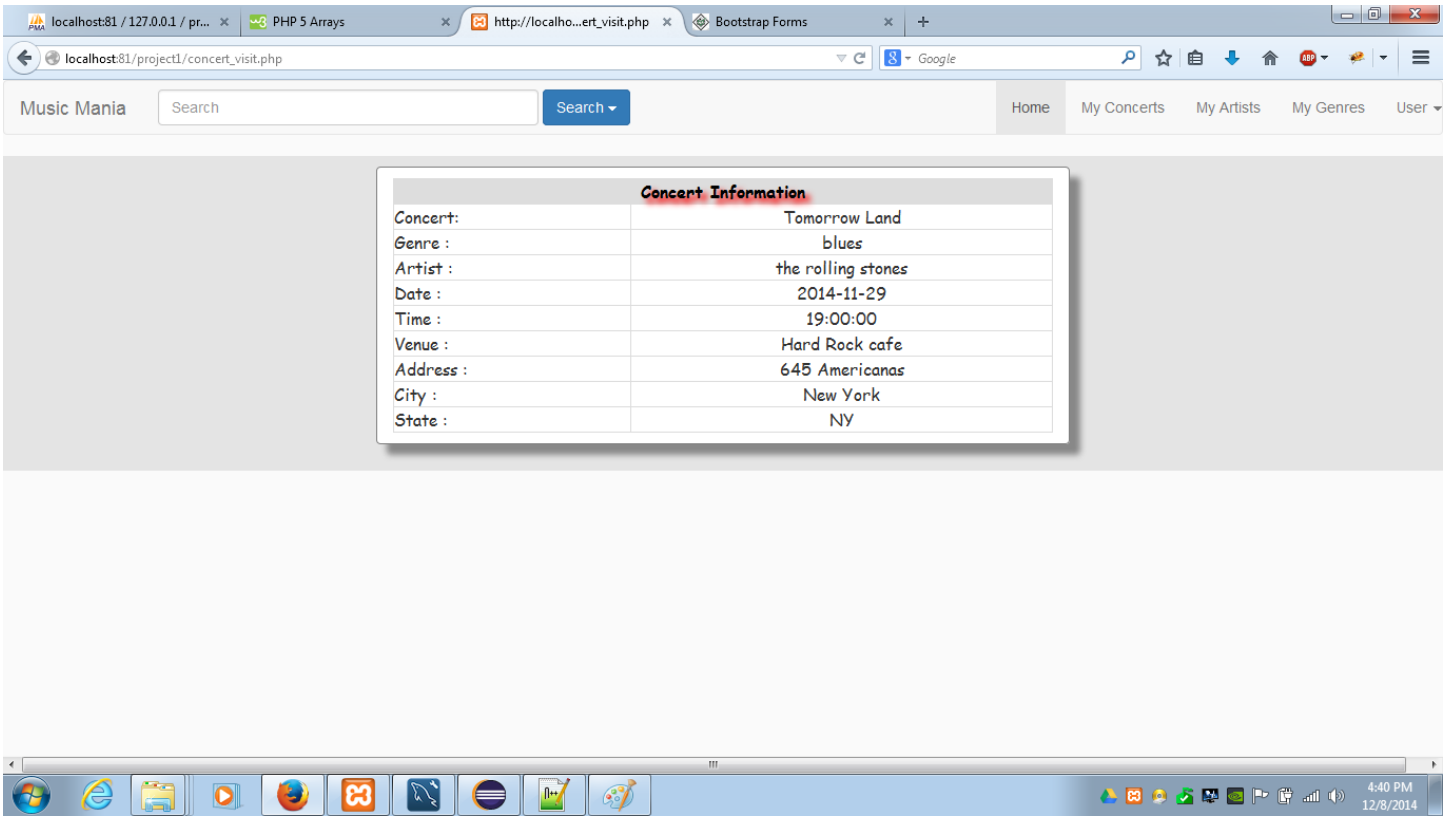



The user could search concert by artist names.A list of concert appears on concert search page and upon clicking one of them he is redirected to concert visit page where he could RSVP for the concert and could also add that concert to my recommendation list.

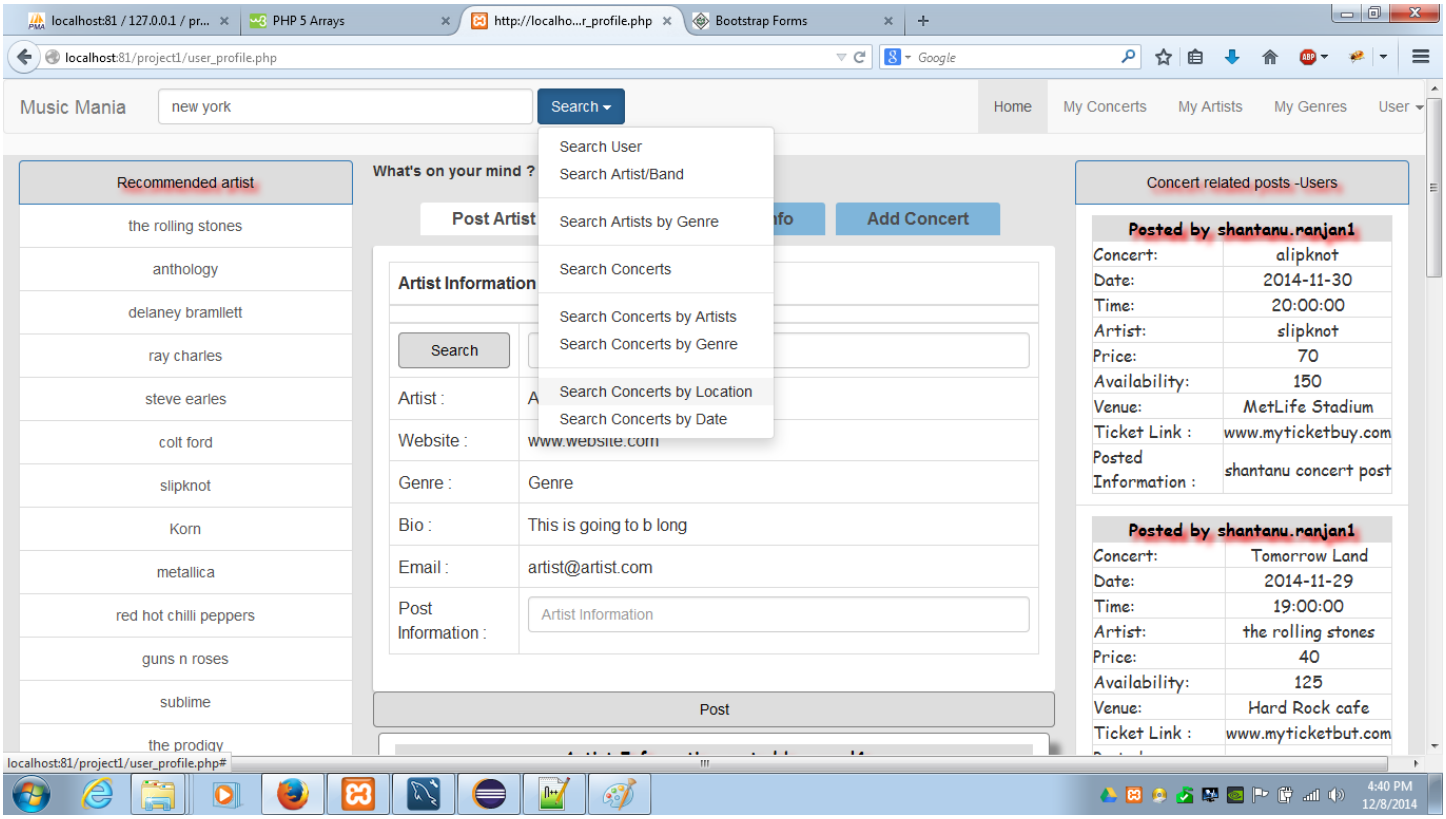
## SEARCH CONCERT BY GENRE

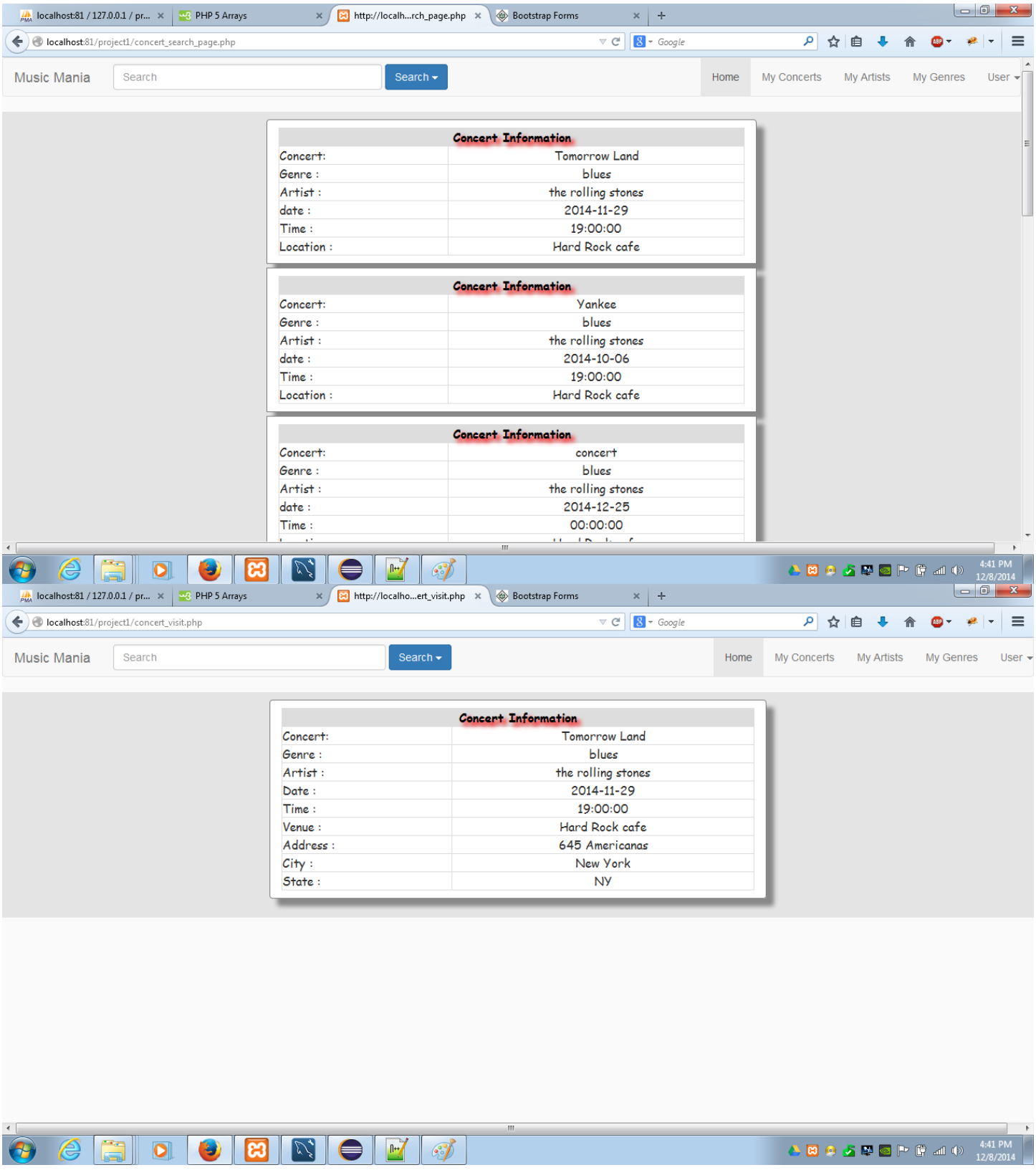



The user could search about concert based on different genres.

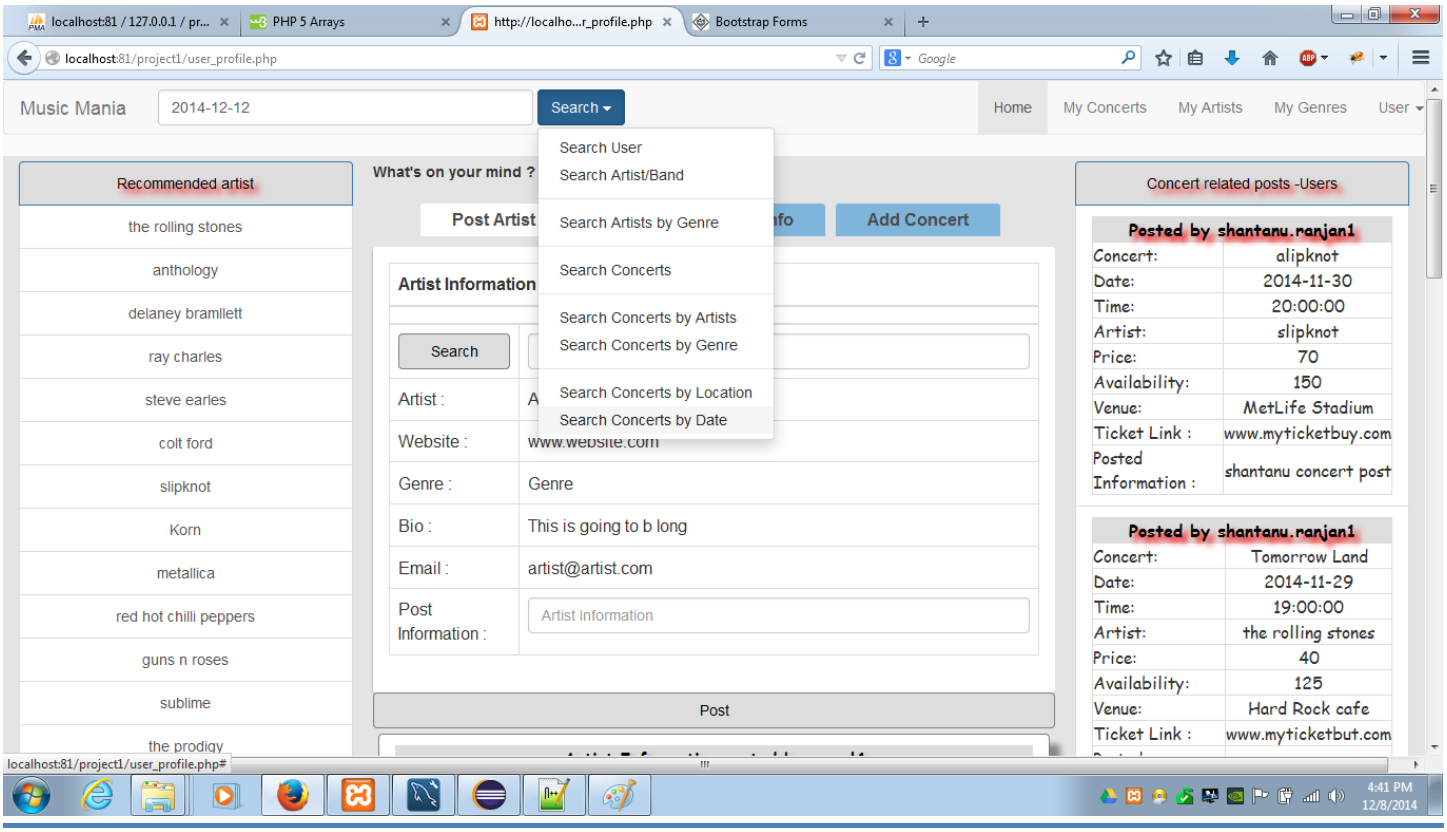
## SEARCH CONCERT BY LOCATION

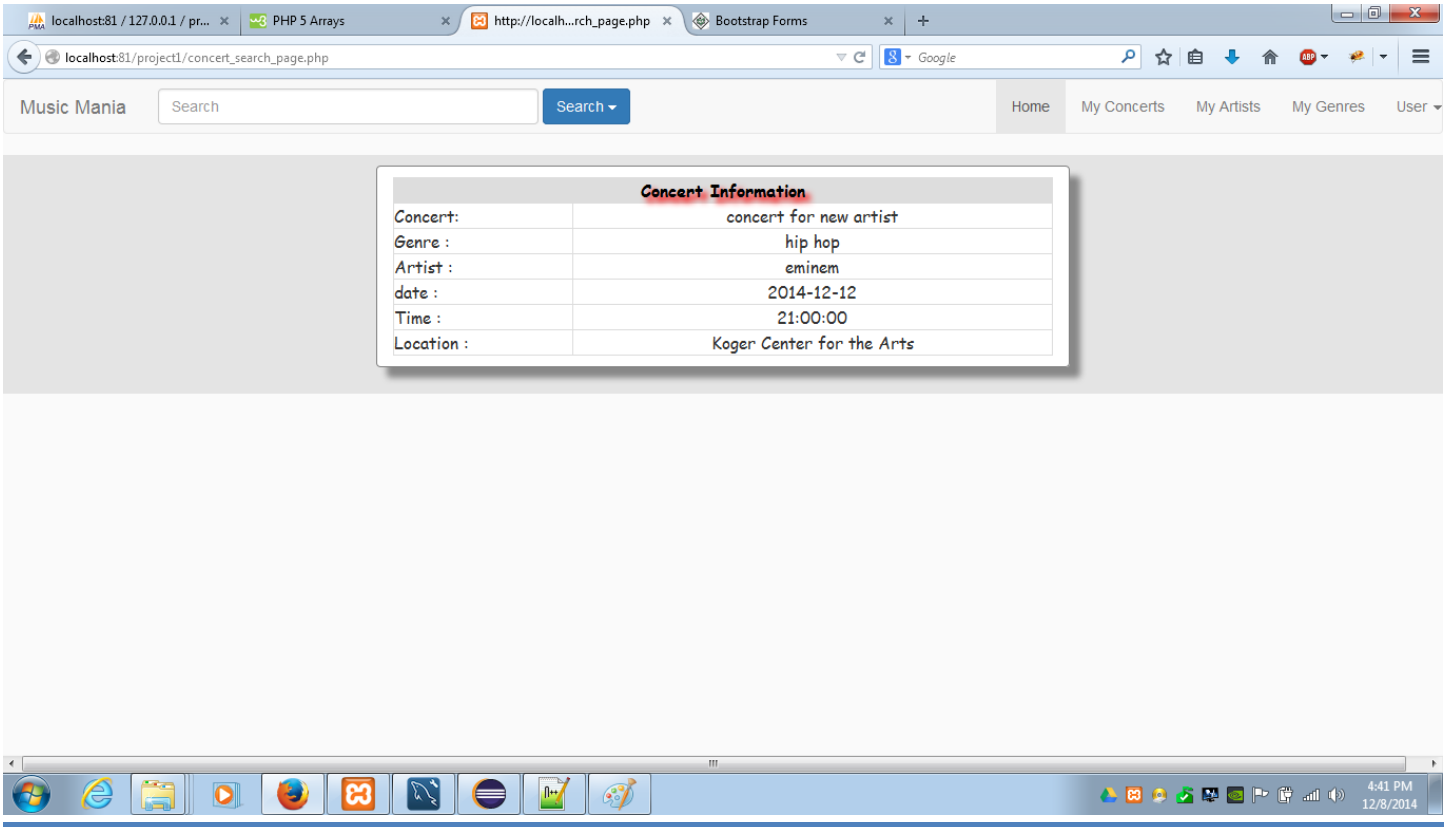





The user could search about concert based on different location and could view their information.

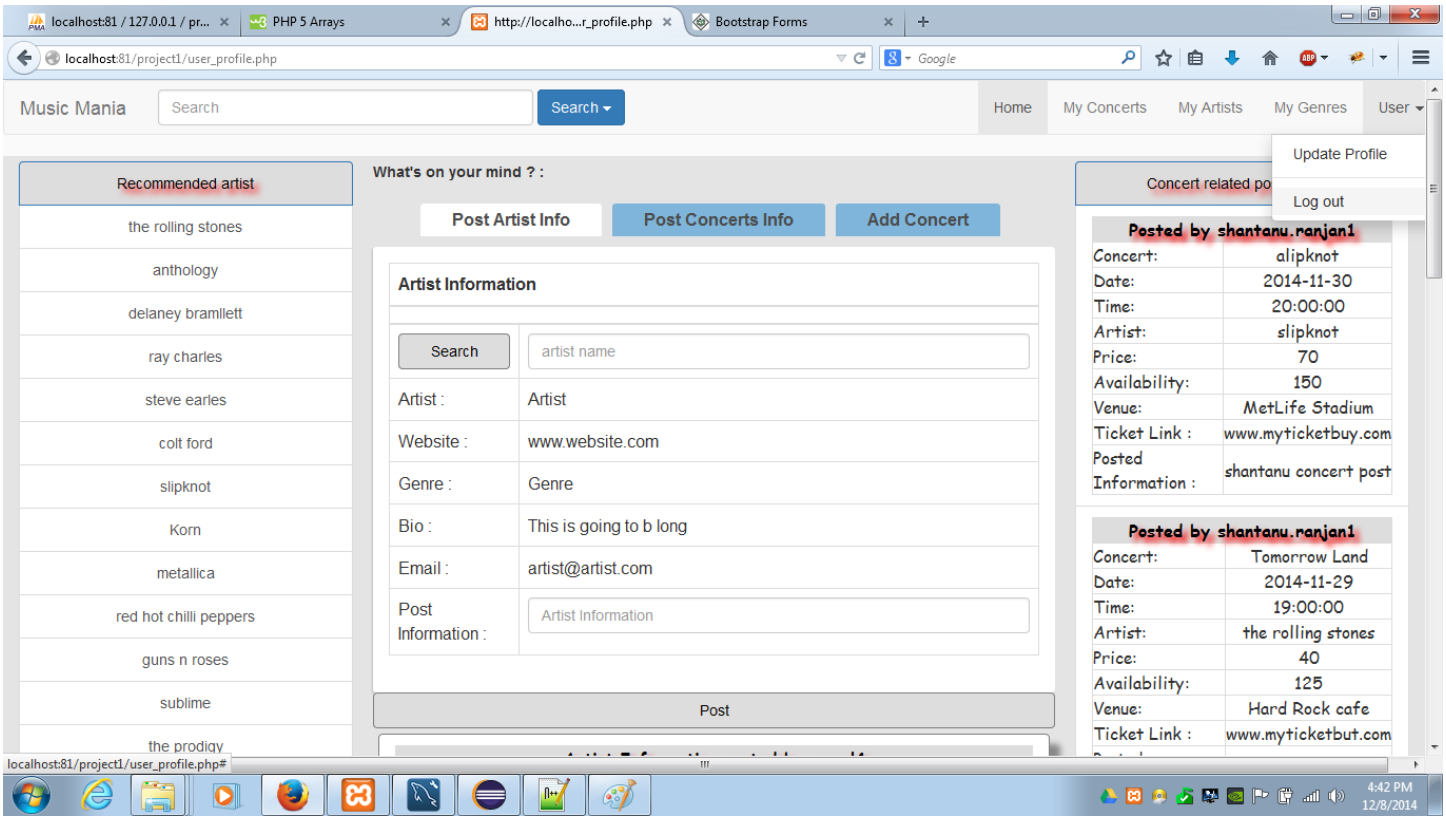
**SEARCH CONCERT BY DATE**



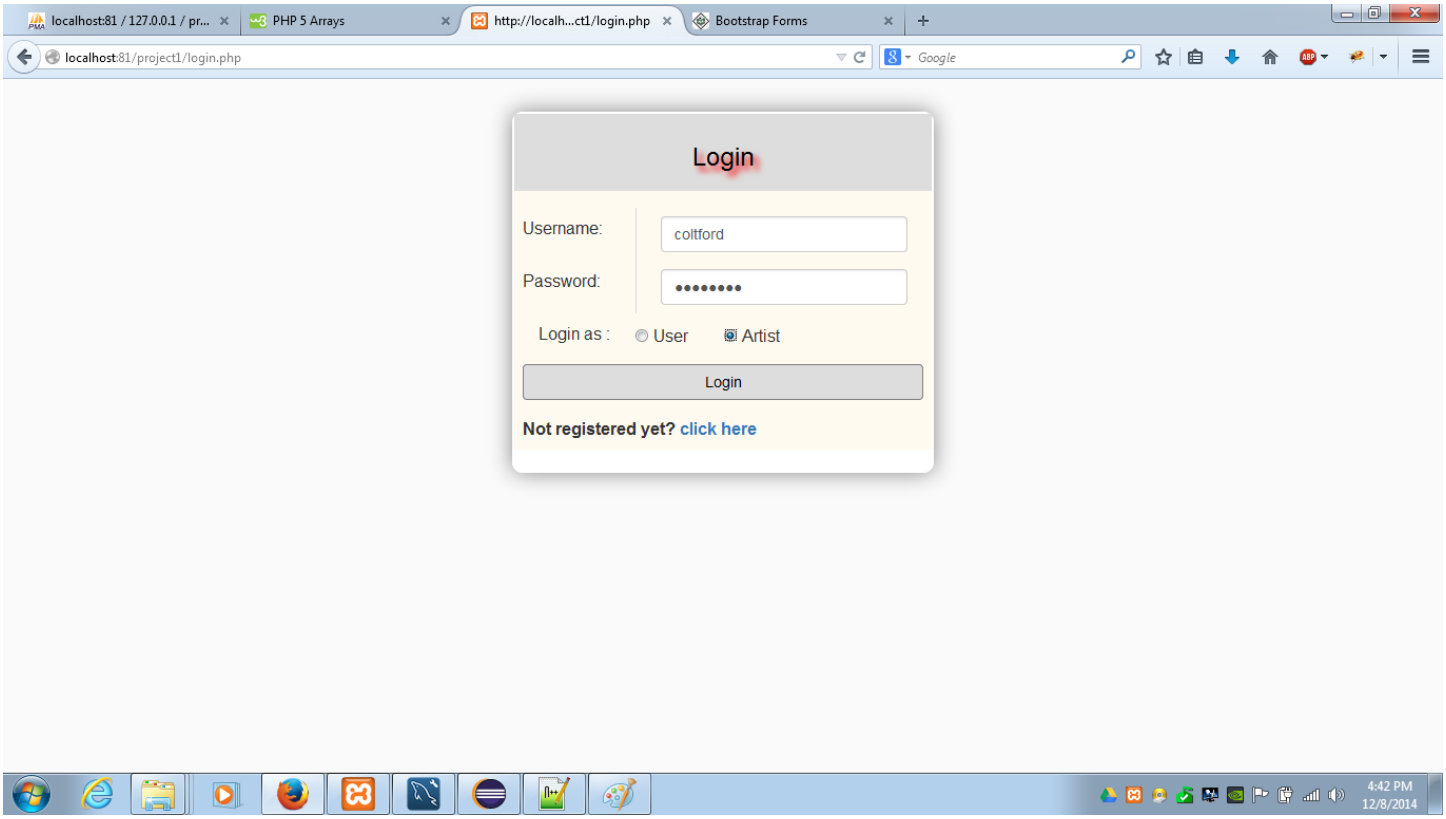

The user could search about concert based on particular date.

**LOGOUT**

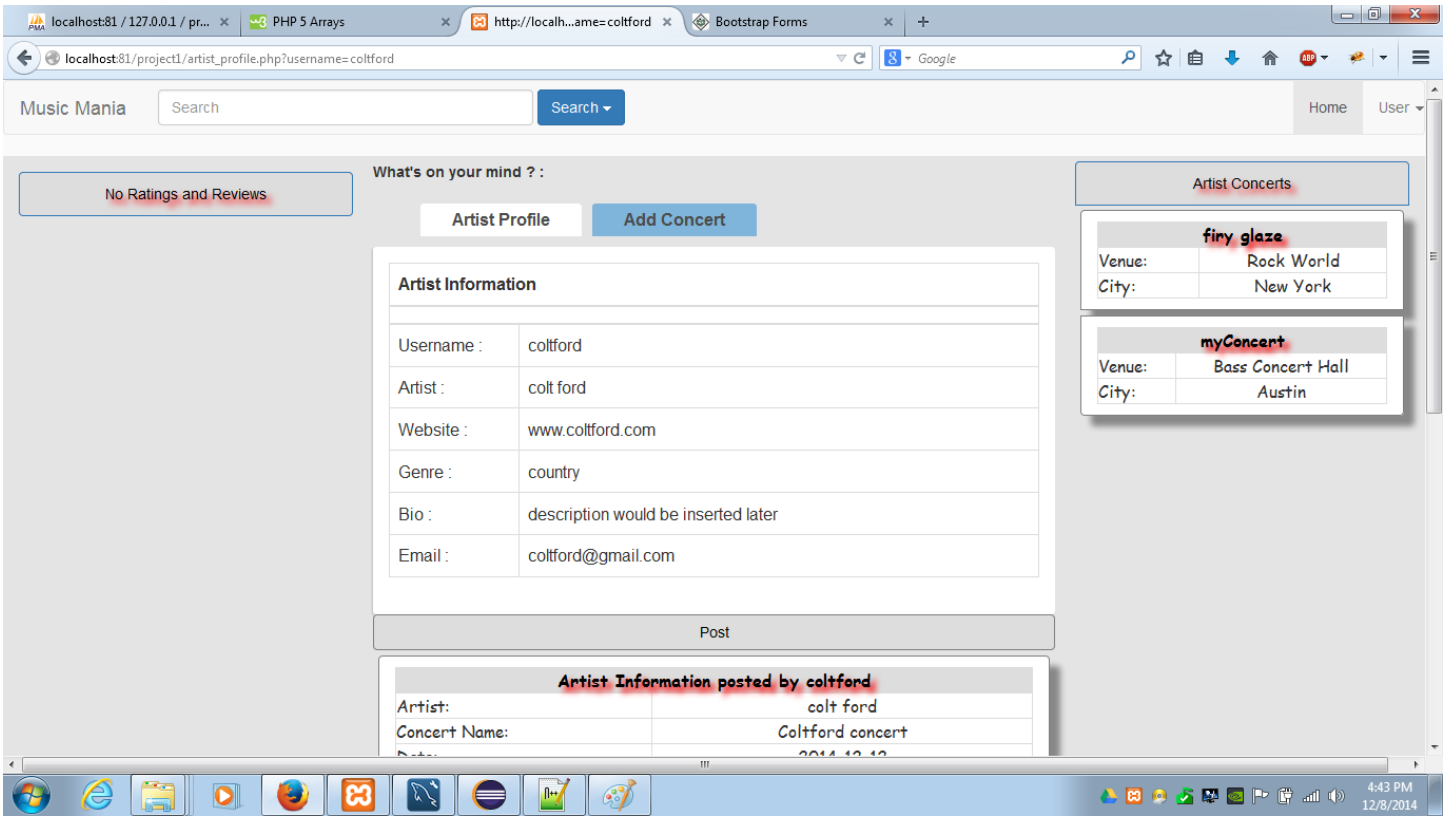



The user could logout from his profile page and would be redirected to login page.Once the user is logged out the session which is storing the information about user is destroyed.\

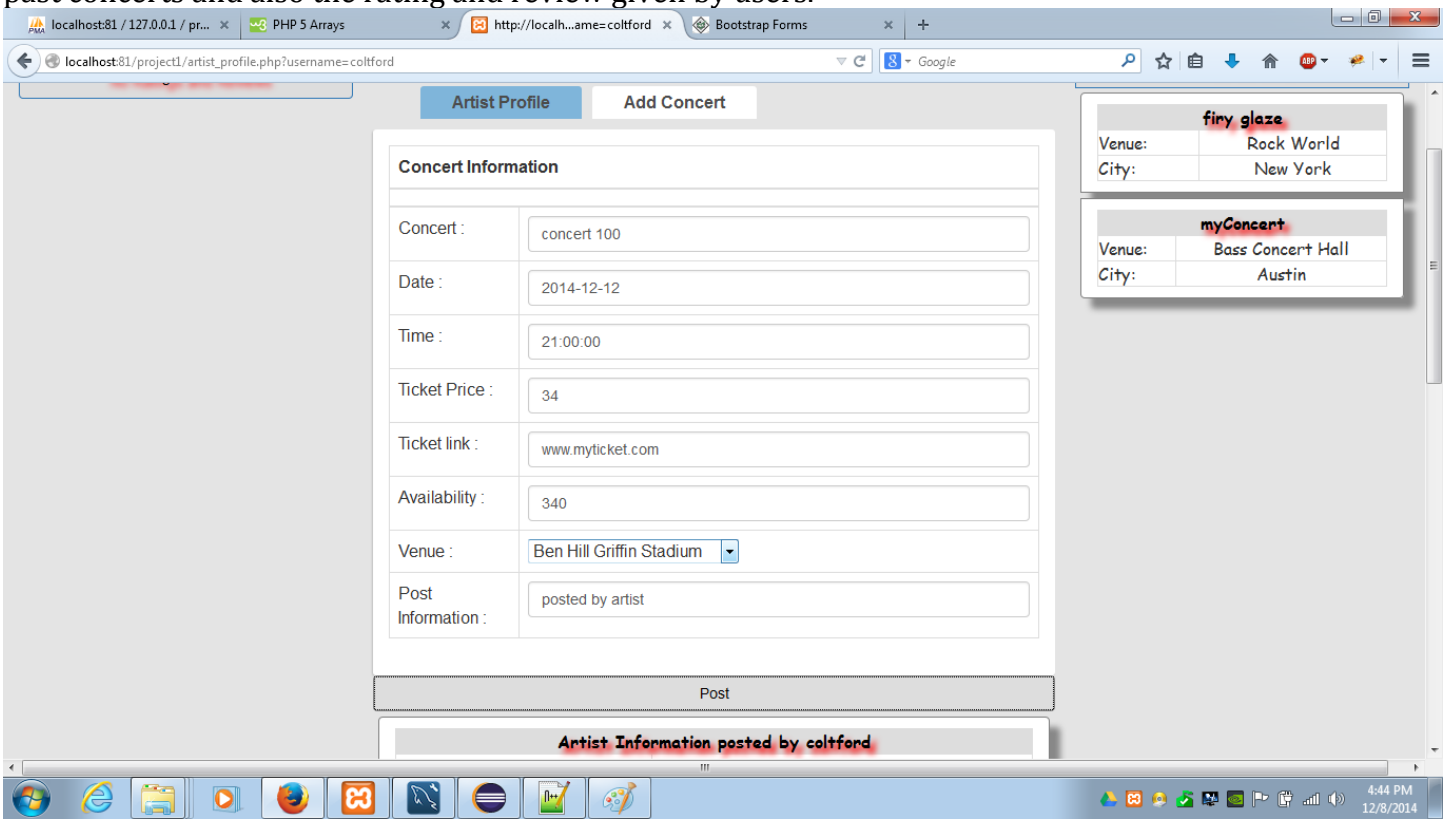
**ARTIST LOGIN**



**ARTIST PROFILE**




Once successfully logged in the artist could view his profile and post information about his upcoming concerts which would then be visible to other users who are fan of him/her.The artist could also view his past concerts and also the rating and review given by users.



## 6)MODIFICATIONS

To implement the above UI we have modified, added extra stored procedures and tables to the database as explained in part1.The following are the stored procedures that we have used:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `search_artist`(IN `keyword` VARCHAR(100))
   NO SQL
select a.artist_id,a.name,a.website,a.email,a.description,g.genre_name from artist a,artist_genre ag,genre g
where a.name = keyword
and ag.artist_id = a.artist_id
and ag.genre_id =g.genre_id$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_artist_profile`(IN artistid int(11))
BEGIN
select a.artist_id,a.name,a.website,a.email,a.description,a.username,g.genre_name from artist a inner join
artist_genre ag on a.artist_id=ag.artist_id inner join genre g on
g.genre_id=ag.genre_id
where a.artist_id=artistid;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_concert`(IN `concertname` VARCHAR(100))
   NO SQL
select c.concert_id,c.concert_name,a.name,date(c.date_time)as date,time(c.date_time) as
time,c.ticket_price,c.availability,c.ticket_link,v.venue_name
from concert c,artist a,location v
where c.venue_id = v.venue_id and c.artist_id = a.artist_id
and c.concert_name=concertname$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_genre`(IN search_city VARCHAR(100),IN
genrename VARCHAR(100))
BEGIN
select c.concert_name,a.name,v.venue_name,v.address from concert c inner join artist a on
c.artist_id=a.artist_id
inner join location v on v.venue_id=c.venue_id inner join artist_genre ag on ag.artist_id=c.artist_id
inner join genre g on ag.genre_id=g.genre_id
where v.city=search_city and g.genre_name=genrename and date(c.date_time)>DATE(current_date() +
INTERVAL (DAYOFWEEK(current_date())) DAY)
and date(c.date_time)<DATE(current_date() + INTERVAL (8-DAYOFWEEK(current_date())) DAY);
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_newconcert`(IN id INT(11))
BEGIN

        select c.concert_name,a.name,v.address,v.city,c.date_time from concert c inner join artist a
on c.artist_id=a.artist_id
        inner join location v on c.venue_id=v.venue_id
        where c.date_time>(select last_login  from user where user_id=id);
```

```
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_recommend`(IN userid INT(11))
BEGIN
		DECLARE follow_id INT(11);
		DECLARE concert_id INT(11);
		DECLARE v_finished INTEGER DEFAULT 0;
		DEClARE followid_cursor CURSOR FOR select follow_id from user_follows where
user_id=userid;

		DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

		OPEN followid_cursor;
		get_followid: LOOP
			FETCH followid_cursor INTO follow_id;
				IF v_finished = 1 THEN
					LEAVE get_followid;
				END IF;

				select concert_id into concert_id from user_concert_list where
user_id=follow_id;
				select c.concert_name,a.name,v.address,v.city,c.date_time from concert c
inner join artist a on c.artist_id=a.artist_id
				inner join location v on c.venue_id=v.venue_id
				where month(c.date_time)=month(current_date())+1;
		 END LOOP get_followid;
		CLOSE followid_cursor;
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_user`(IN `userid` INT)
   NO SQL
select * from user where user_id = userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_concert_posts`(IN userid INT(11))
BEGIN
select artist_name,concert_name,date,time,venue,price,availability,link,username,description from
user_post_concert
where user_id in(select follow_id from user_follows where user_id=userid);

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_fan`(IN userid INT(11),IN followid INT(11))
BEGIN
insert into user_fan(user_id,artist_id) values(userid,followid);
END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_genre`(IN `userid` INT)
   NO SQL
```

```
select distinct g.genre_name from genre g,user_genre ug where ug.genre_id =  g.genre_id and ug.user_id =
userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_personal_information`(IN `userid` INT)
   NO SQL
select * from user where user_id=userid$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_post_artist`(IN userid INT(11))
BEGIN
select user_id,artist_name,url,email,bio,description,username,posted_time,genre_name from
user_post_artist
where user_id=userid order by posted_time desc;

END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_post_concert`(IN `userid` INT)
   NO SQL
select * from user_post_concert where user_id =userid order by posted_time desc$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_signup`(IN username VARCHAR(100),IN pass
varchar(100))
BEGIN
                        INSERT INTO user(username,password) VALUES (username,pass);
                END$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `user_sub_genre`(IN `userid` INT)
   NO SQL
select distinct s.sub_genre_name from sub_genre s , user_sub_genre ug where ug.sub_genre_id =
s.sub_genre_id and ug.user_id=userid$$

DELIMITER ;
```