# Problem Description

A startup company is working on a new online community for fans of live music. Fans who are interested in going to concerts can then sign up, become fans of bands, and also follow other users who might post information about new bands and upcoming concerts. Design should be able to support the following four aspects:

* The system should store information about bands (artists), concerts by these bands, and the places where the concerts take place.
* Users should be able to sign up and specify their musical tastes and which bands they like, and the system will then be able to inform them of relevant upcoming concerts. The users should then be able to specify that they plan to go to the concert, and to give it a rating afterwards.
* Users should also be able to post additional content to help other users find good concerts; in particular, users should be able to make lists of recommended upcoming concerts, and users may also post information about other concerts that are missing from the system.
* The system should help users to find upcoming concerts that might be of interest to them, even if the user is not (yet) a fan of the band or type of music, by incorporating some (naive) recommender system technology and search mechanisms into the system.

# Goal

The Main goal is to design and develop a Website for fans of live music. The online network that offers information about bands and concerts. Initially fans who are interested have to first sign up onto the website, feed in their profile information and then become fans of artists and post information about new bands, their taste and upcoming concerts. Bands can also post their information about concerts and their schedule so that they can reach out to their fans beforehand.

## Software Requirements

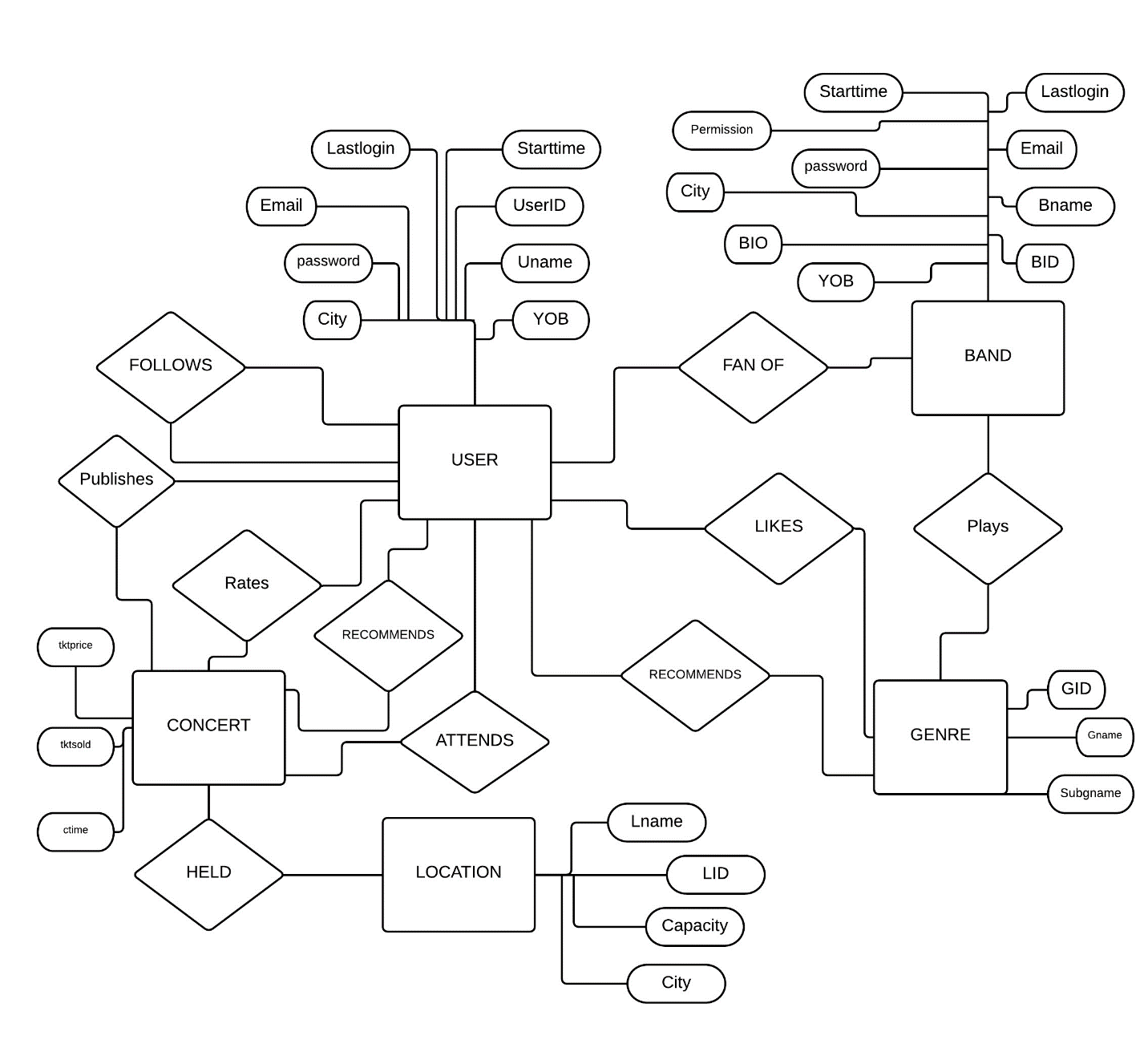
Database Managing System: MySQL

Tool to access the database: PHP

Programming paradigm: PHP

Tool used to edit PHP, JavaScript and HTML : Sublime Text Editor, XAMPP,MYSQL WORKBENCH.

# Architecture and Database Design

This paragraph describes the information architecture for the database system. The data model consists of an Entity Relationship Diagram (ERD) and relational Schemas.

We have chosen a separate entities User and Band Table because we need to identify band from users.If we combined both the entities it would be difficult to segregate bands and provide permission to band by an admin since both have almost same attributes. Band only has 2 additional Bio description and permission.

Below shows the entity and their relationships and key constraints in them.

**User🡪 Attends🡪 Concert**

User can attend many concerts and User contains attributes describing him and attend relation contains attributes describing which concert user attended and time at which he attended.

**User🡪Recommends🡪Concert**

User can recommend other users a list of concerts which others may be interested in. Not all Users can recommend concerts only users with certain trust level [which is calculated dynamically based on rate time and number of previous rated concert] can recommend other users about upcoming concerts. Recommend relation contains recommended user, recommended concert and time.

**User🡪Rates🡪Concert**

User can provide rating to other concerts and can also review other concerts. Thus we have rates relation which contains a rate time which indicates time when user rated a concert through which we can dynamically calculate the trust level.

**User🡪Publishes🡪Concert**

User can publish a concert which he liked if he meets the trust level. He can post concerts and provide reviews about the concerts he attended.

**User-🡪Follows🡪Users**

Users can follow other users he likes. Thus we have a follow relationship where we have the followerid and following ID.

**User🡪Fan of🡪Band**

Apart from following other Users, Users can follow bands of his choice and taste.We have Userid and Band ID in user band table which indicates which user is a fan of which band and when did he like the band.

**User🡪Likes🡪Genre**

We can publish posts to users home page based on which Genre user likes and which band he follows the most. Hence we have User Likes Genre relation which contains Genre Id and subgenre name so that it can be used to find out which type genre user prefers which can be basically used as a filter while posting.

**User🡪Recommends🡪 Genre**

Apart from posting list of concerts he likes. User can also post list of Genre he prefers which can be used to determine list of bands and concerts he liked and rated.

**Band🡪Plays🡪Genre**

Band can play different Genre so we have a relation Band plays a genre where we have band ID and Genre ID to determine which genre is played by which Band.

**Concert🡪Held🡪Location**

Each Concert is held at a particular venue. In the location table we have location ID ,City and number of seats it can accommodate. Thus we have ticket sold in the concert table which is subtracted with capacity to determine the vacant seats left.

# Database Design

**USER**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Userid | Username | Email | Year of Birth | City | Password | Start time | Last Login |

**Primary Key**: Userid

**Band**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BID | Bname | Email | Year of Birth | City | Password | Start time | Last Login | Bio | Permission |

**Primary Key** : BID

**Location**

|  |  |  |  |
| --- | --- | --- | --- |
| LID | Lname | City | Capacity |

**Primary Key** : LID

**Genre**

|  |  |  |
| --- | --- | --- |
| GID | Gname | Subgname |

**Primary Key**: GID

**Concert**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CID | BID | GID | Venue | Tktsold | Ctime | tprice |

**Primary Key** CID

Foreign Key Venue references location LID

Foreign Key BID references Band BID

Foreign Key GID references Genre GID

**User Published**

|  |  |
| --- | --- |
| UserID | CID |

Foreign Key UserID references User UserID

Foreign Key CID references Concert CID

**Attends**

|  |  |  |
| --- | --- | --- |
| UserID | CID | AttendTime |

Foreign Key userid references User userid

Foreign key CID references concert CID.

**Genre\_User**

|  |  |  |
| --- | --- | --- |
| UserID | GID | Liketime |

Foreign Key userid references User userid

Foreign key GID references Genre GID.

**User\_Band**

|  |  |  |
| --- | --- | --- |
| UserID | BID | Liketime |

Foreign Key Userid references User userid

Foreign key BID references Band BID.

**Genre\_Band**

|  |  |
| --- | --- |
| BID | GID |

Foreign Key GID references Genre GID

Foreign key BID references Band BID.

**User\_Rating**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | CID | Rating | Review | Rtime |

Foreign Key Userid references User userid

Foreign key CID references Concert CID.

**Followers**

|  |  |  |
| --- | --- | --- |
| Followerid | FollowingID | Ftime |

Foreign Key FollowerID references User userid.

Foreign Key FollowingID references User userid

**Recommendation concert**

|  |  |  |
| --- | --- | --- |
| UserID | CID | Rec\_time |

Foreign Key Userid references User userid

Foreign Key CID references Concert CID.

**Recommendation Genre**

|  |  |  |
| --- | --- | --- |
| UserID | GID | Rec\_time |

Foreign Key Userid references User userid

Foreign Key GID references Genre GID.

**RSVP CONCERT**

|  |  |
| --- | --- |
| UserID | CID |

Foreign Key Userid references User userid

Foreign Key CID references Concert ( CID).

# Problem Domain

Based on Schema provided we are creating tables in our database

**table creation**

create table user(

userid varchar(20) primary key,

uname varchar(20),

email varchar(30),

yob year,

city varchar(20),

pwd varchar(20),

starttime datetime,

lastlogin datetime

);

create table band(

bid varchar(20) primary key,

bname varchar(20),

email varchar(30),

yob year,

city varchar(20),

pwd varchar(20),

starttime datetime,

lastlogin datetime,

bio varchar(100),

permission varchar(20)

);

create table location(

lid varchar(20) primary key,

lname varchar(20),

city varchar(20),

capacity int

);

create table genre

(

gid varchar(20),

gname varchar(20),

subgname varchar(20),

primary key(gid)

);

create table genre\_band(

bid varchar(20),

gid varchar(20),

foreign key(bid) references band(bid),

foreign key(gid) references genre(gid)

);

create table concert

(

cid varchar(20),

bid varchar(20),

gid varchar(20),

venue varchar(20),

tktsold int,

tprice int,

ctime datetime,

posttime datetime,

foreign key(venue) references location(lid),

foreign key(bid) references band(bid),

foreign key(gid) references genre(gid),

primary key(cid)

);

create table user\_publish

(

cid varchar(20),

userid varchar(20),

publishtime varchar(20),

foreign key(userid) references user(userid),

foreign key(cid) references concert(cid)

);

create table rsvp\_concert

(

userid varchar(20),

cid varchar(20),

foreign key(userid) references user(userid),

foreign key(cid) references concert(cid)

);

create table genre\_user

(

userid varchar(20),

gid varchar(20),

liketime datetime,

foreign key(userid) references user(userid),

foreign key(gid) references genre(gid)

);

create table user\_band

(

userid varchar(20),

bid varchar(20),

liketime datetime,

foreign key(userid) references user(userid),

foreign key(bid) references band(bid)

);

create table user\_rating

(

userid varchar(20),

cid varchar(20),

rating int,

review varchar(30),

rtime datetime,

foreign key(userid) references user(userid),

foreign key(cid) references concert(cid)

);

create table followers

(

followerid varchar(20),

followingid varchar(20),

ftime datetime,

foreign key(followerid) references user(userid),

foreign key(followingid) references user(userid)

);

create table rec\_concert

(

userid varchar(20),

cid varchar(20),

rec\_time datetime,

foreign key(userid) references user(userid),

foreign key(cid) references concert(cid)

);

create table rec\_genre

(

userid varchar(20),

gid varchar(20),

rec\_time datetime,

foreign key(userid) references user(userid),

foreign key(gid) references genre(gid)

);

**INSERTION OF DATA**

# data insertion

insert into user values("user1","Amit","a@123.com","1990","manglore","pwd1",20141110000000,20141119000000);

insert into user values("user2","chaithra","c@123.com","1993","hyderabad","pwd2",20141120000000,20141121000000);

insert into user values("user3","db","d@123.com","1970","newyork","pwd3",19801110000000,20141119000000);

insert into band values("band1","rockers","r@123.com","1990","newyork","bpwd1",19801111000000,20141119000000,"we rock","no");

insert into band values("band2","shakers","s@123.com","1991","california","bpwd2",19891012000000,20141119000000,"we shake","no");

insert into band values("band3","crackers","ck@123.com","1992","florida","bpwd3",19891012000000,20141119000000,"we shake","no");

insert into location values("loc1","location1","brooklyn","100");

insert into location values("loc2","location2","manhattan","200");

insert into location values("loc3","location3","queens","300");

insert into genre values("g1","jazz","free jazz");

insert into genre values("g2","jazz","cool jazz");

insert into genre values("g3","jazz","bebob");

insert into genre values("g4","hiphop","free hiphop");

insert into genre values("g5","hiphop","cool hiphop");

insert into genre values("g6","hiphop","bebob");

insert into genre\_band values("band1","g1");

insert into genre\_band values("band2","g2");

insert into genre\_band values("band3","g3");

insert into genre\_band values("band1","g4");

insert into genre\_band values("band2","g5");

insert into genre\_band values("band3","g6");

insert into concert values("concert1","band1","g1","loc1",0,10,20141124080000,20141126080000);

insert into concert values("concert2","band2","g2","loc2",0,20,20141125080000,20141127080000);

insert into concert values("concert3","band3","g3","loc3",0,30,20141126080000,20141128080000);

insert into attends values("user1","concert1",20141124080000);

insert into attends values("user2","concert2",20141125080000);

insert into attends values("user3","concert3",20141126080000);

insert into genre\_user values("user1","g1",20141110000000);

insert into genre\_user values("user2","g2",20141120000000);

insert into genre\_user values("user3","g3",19801110000000);

insert into user\_band values("user1","band1",20141110000000);

insert into user\_band values("user2","band2",20141120000000);

insert into user\_band values("user3","band3",19801110000000);

insert into user\_rating values("user1","concert1",7,"good",20141124080000);

insert into user\_rating values("user2","concert2",3,"bad",20141125080000);

insert into user\_rating values("user3","concert3",5,"average",20141126080000);

insert into user\_rating values("user3","concert2",9,"excellent",20141231080000);

insert into user\_rating values("user3","concert1",1,"poor",20141231080000);

insert into followers values("user1","user2",20141124080000);

insert into followers values("user2","user3",20141125080000);

insert into followers values("user3","user1",20141126080000);

insert into rec\_concert values("user1","concert2",20141121080000);

insert into rec\_concert values("user2","concert3",20141122080000);

insert into rec\_concert values("user3","concert1",20141123080000);

insert into rec\_concert values("user1","concert1",20141124080000);

insert into rec\_concert values("user2","concert2",20141125080000);

insert into rec\_concert values("user3","concert3",20141126080000);

insert into rec\_genre values("user1","g1",20141121080000);

insert into rec\_genre values("user2","g2",20141122080000);

insert into rec\_genre values("user3","g3",20141123080000);

insert into rec\_genre values("user1","g4",20141124080000);

insert into rec\_genre values("user2","g5",20141125080000);

insert into rec\_genre values("user3","g6",20141126080000);

insert into user\_publish values("concert1","user1",20141123010000);

insert into user\_publish values("concert2","user2",20141126010000);

**SQL Query for Signup:**

**DELIMITER $$**

**create procedure signup(in Userid varchar(20),in uname varchar(20),in email varchar(30),**

**in yob year(4),in city varchar(20),in pwd varchar(20),in starttime datetime,in lastlogin datetime,**

**inout flag int)**

**begin**

**declare count int default 0;**

**select count(\*) into count from user where user.userid= Userid;**

**if count<1 then**

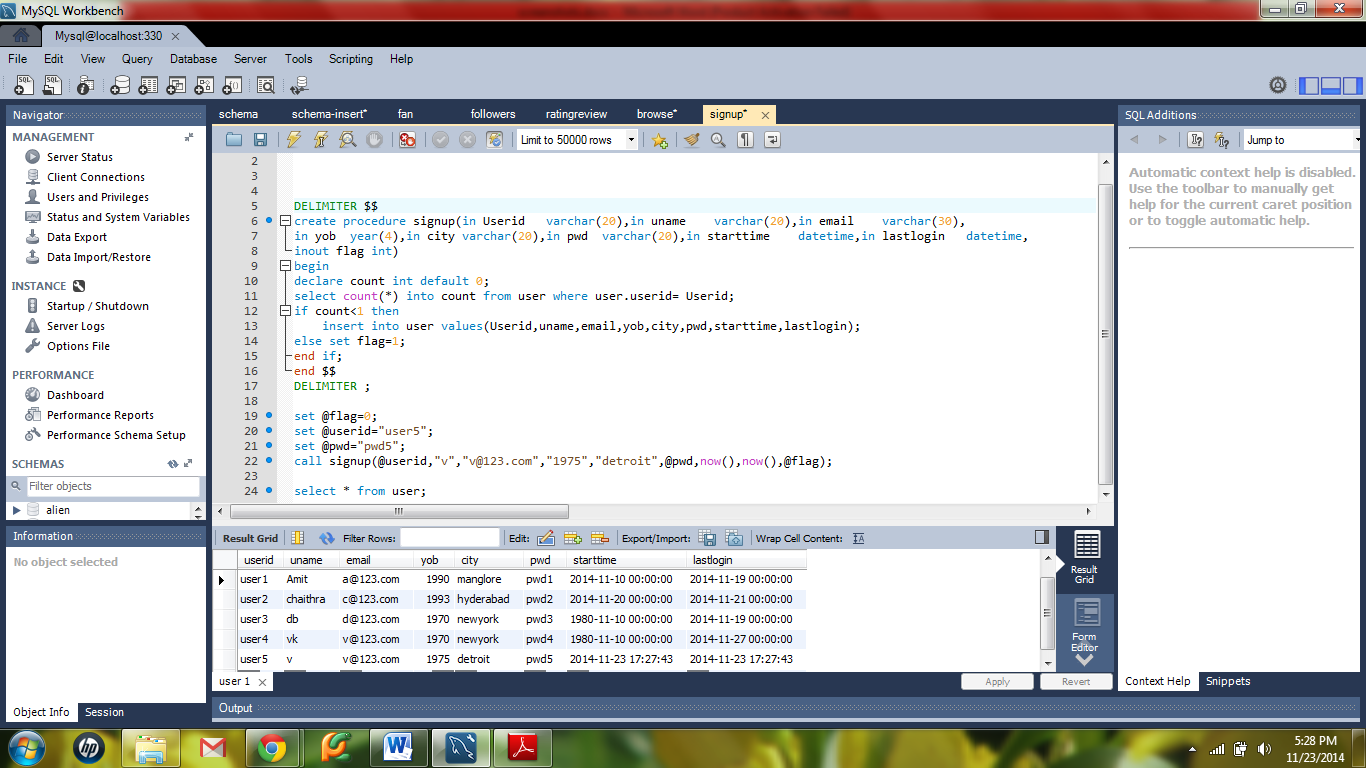
**insert into user values(Userid,uname,email,yob,city,pwd,starttime,lastlogin);**

**else set flag=1;**

**end if;**

**end $$**

**DELIMITER ;**



**Create or edit personal profile:**

**DELIMITER $$**

**create procedure create\_or\_edit\_personal(**

**in User varchar(20),**

**in username varchar(20),**

**in yr year(4),**

**in ucity varchar(20),**

**in pswd varchar(20),**

**inout update\_user\_flag int)**

**begin**

**if update\_user\_flag=1 then**

**update user set uname = username , yob = yr, city = ucity, pwd = pswd where userid=User;**

**end if;**

**end $$**

**DELIMITER ;**



**Create or edit interests:**

**DELIMITER $$**

**create procedure create\_or\_edit\_interests(**

**in User varchar(20),**

**in G varchar(20),**

**inout update\_user\_flag int)**

**begin**

**declare count int default 0;**

**select count(\*) into count from genre\_user g where g.userid = User and g.gid=G;**

**if update\_user\_flag=0 and count<1 then**

**set update\_user\_flag=2;**

**insert into genre\_user values(User,G,now());**

**elseif update\_user\_flag=1 then**

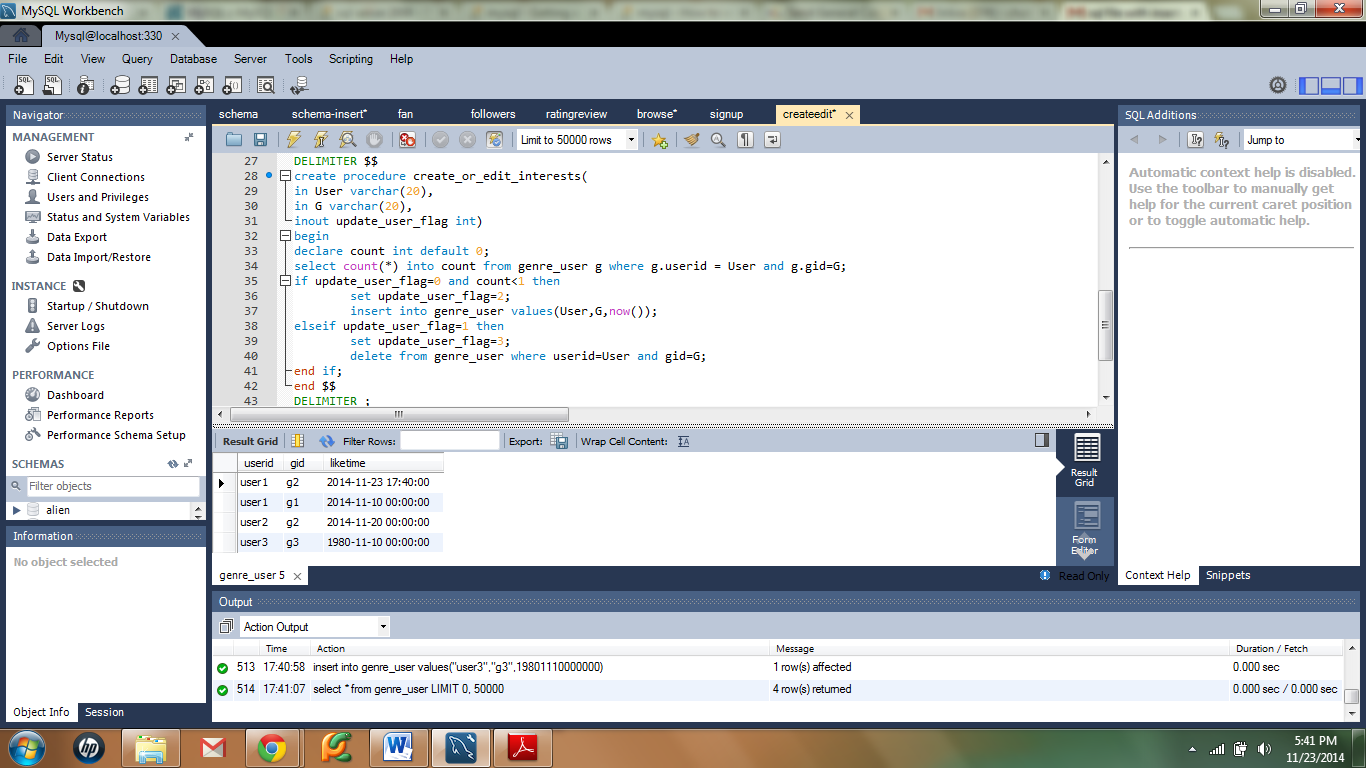
**set update\_user\_flag=3;**

**delete from genre\_user where userid=User and gid=G;**

**end if;**

**end $$**

**DELIMITER ;**



**To become a fan of a band:**

**DELIMITER $$**

**CREATE PROCEDURE fan (in user varchar(20),in band varchar(20),inout flag int)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from user\_band u where u.userid=user and u.bid=band;**

**if count<1 and flag=0 then**

**insert into user\_band values(user,band,now());**

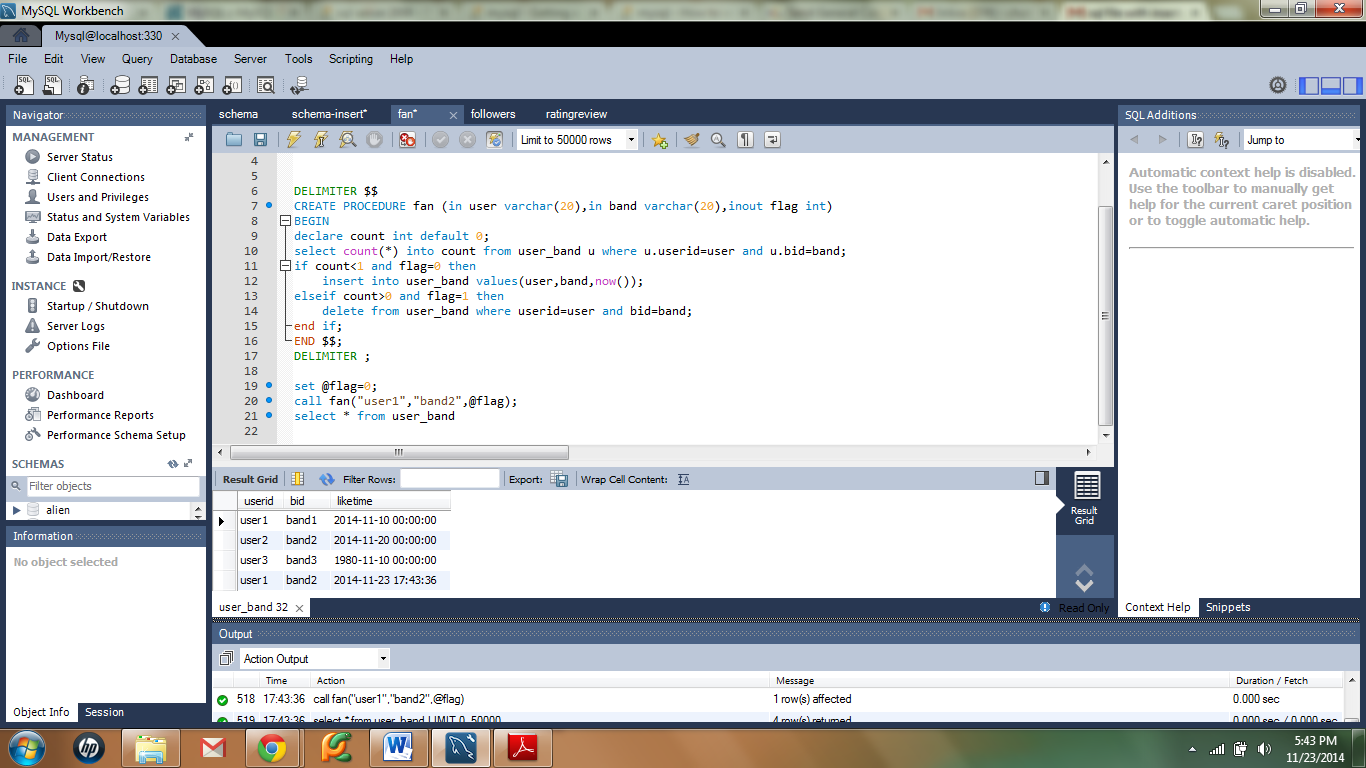
**elseif count>0 and flag=1 then**

**delete from user\_band where userid=user and bid=band;**

**end if;**

**END $$;**

**DELIMITER ;**



**To follow another user:**

**DELIMITER $$**

**CREATE PROCEDURE add\_or\_remove\_follower(In follower varchar(20), in following varchar(20), inout flag int)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from followers f where f.followerid=follower and f.followingid=following;**

**if count<1 and flag = 0 then**

**insert into followers values(follower,following,now());**

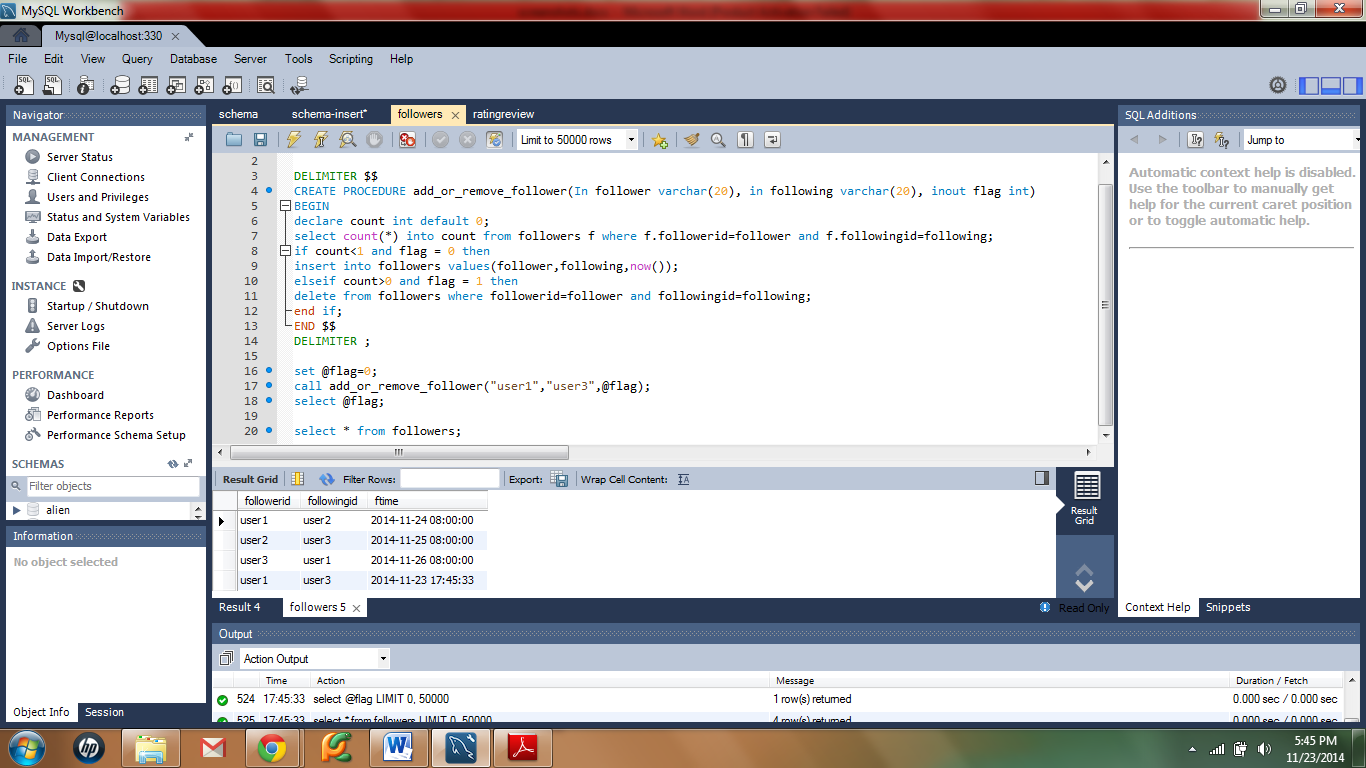
**elseif count>0 and flag = 1 then**

**delete from followers where followerid=follower and followingid=following;**

**end if;**

**END $$**

**DELIMITER ;**



**To rate and review a concert:**

**DELIMITER $$**

**CREATE PROCEDURE add\_or\_remove\_rating(In user varchar(20), in concert varchar(20), in rate int, comments varchar(30),inout flag int)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from user\_rating r where r.userid=user and r.cid=concert;**

**if count<1 and flag = 0 then**

**insert into user\_rating values(user,concert,rate,comments,now());**

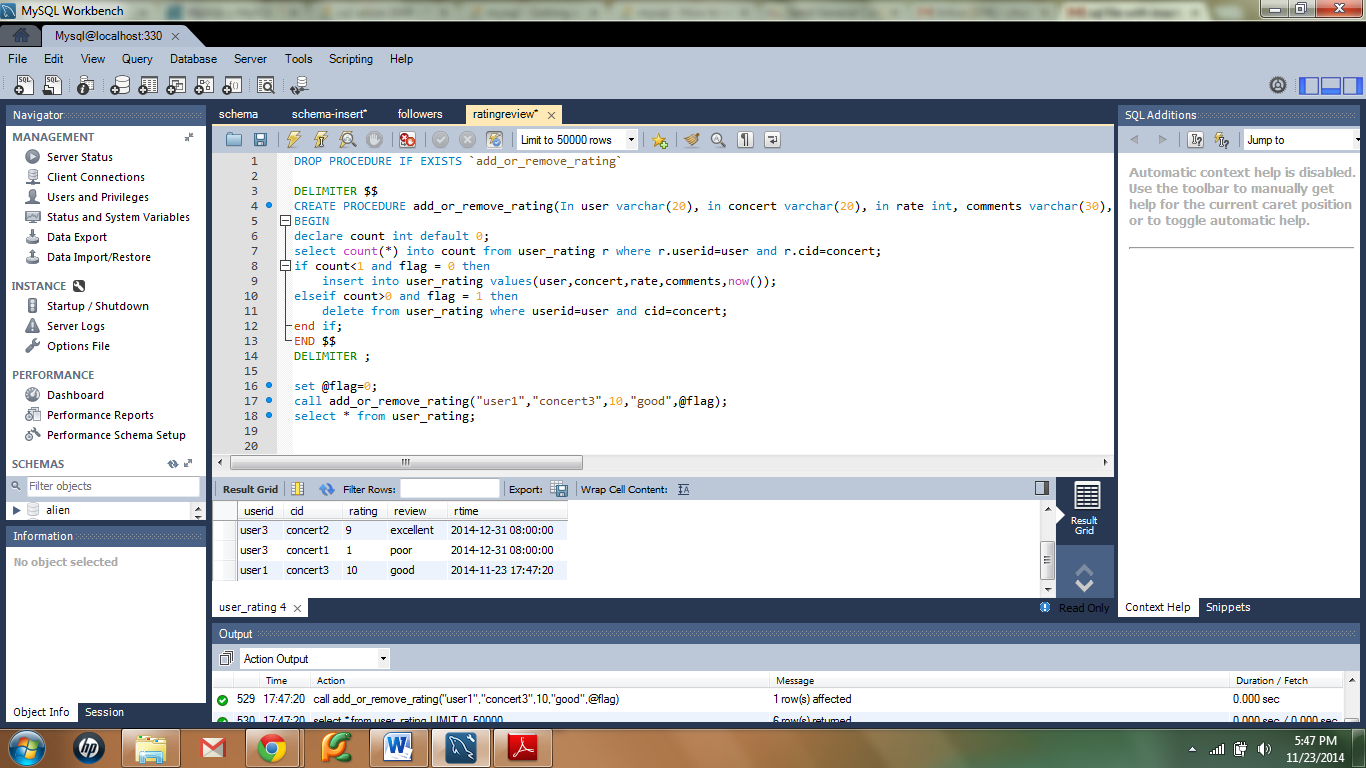
**elseif count>0 and flag = 1 then**

**delete from user\_rating where userid=user and cid=concert;**

**end if;**

**END $$**

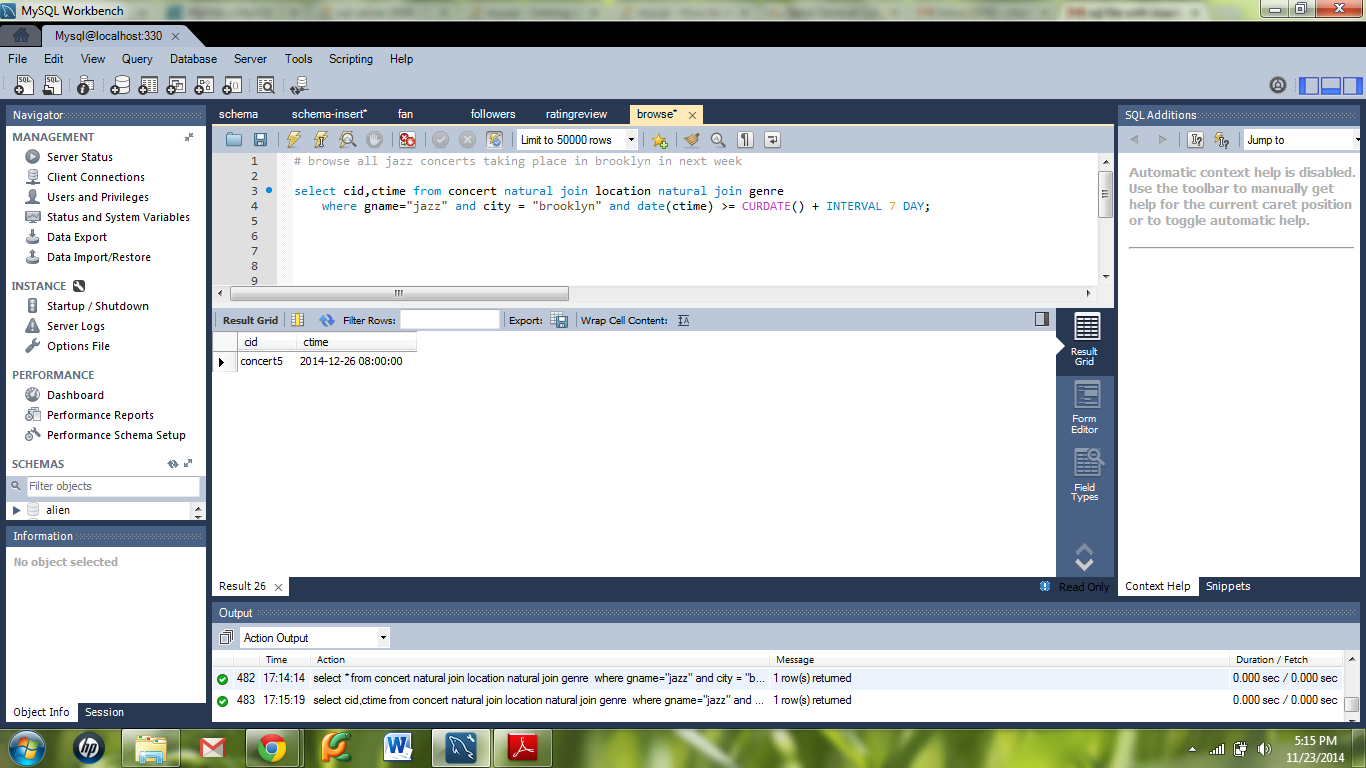
**DELIMITER ;**



**Browse/search queries:**

select cid,ctime from concert natural join location natural join genre

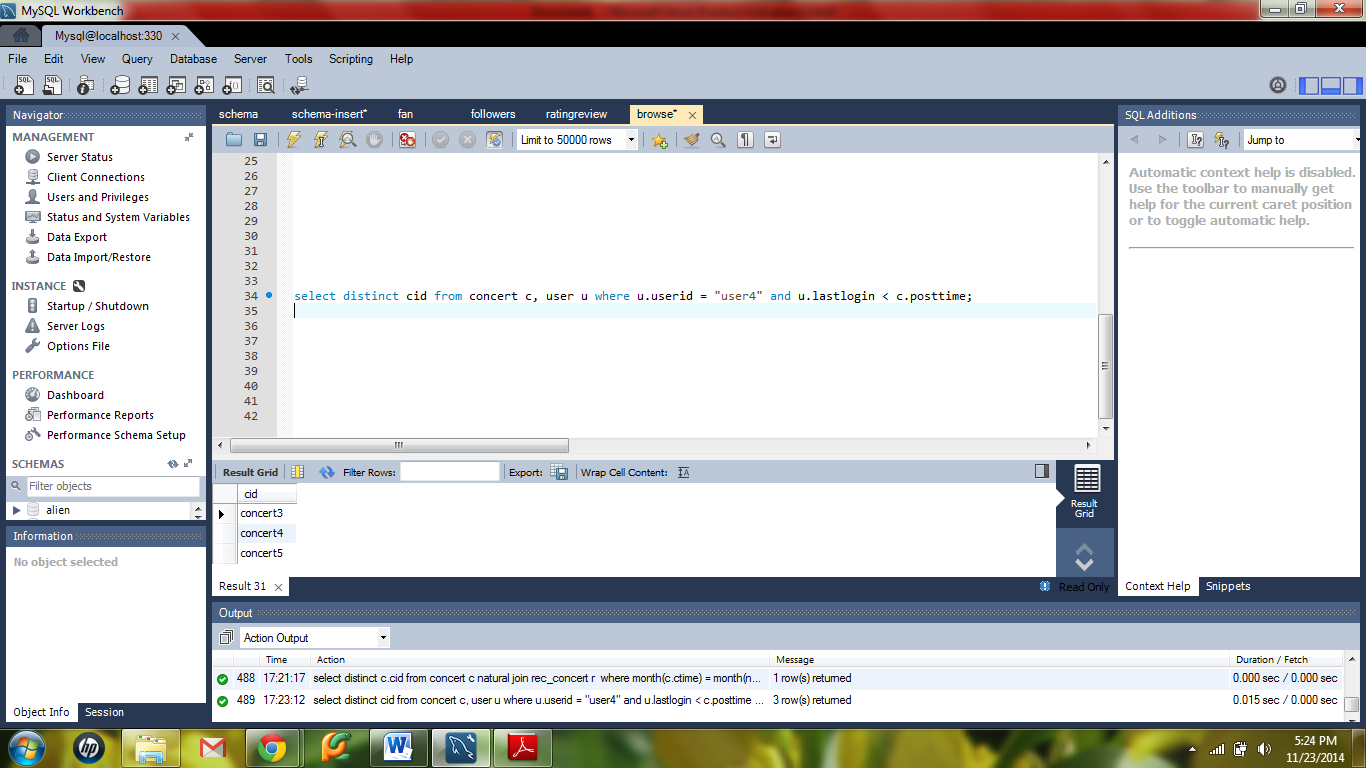
where gname="jazz" and city = "brooklyn" and date(ctime) >= CURDATE() + INTERVAL 7 DAY;



select distinct c.cid from concert c natural join rec\_concert r where month(c.ctime) = month(now())+1 and r.userid in(select followingid from followers where followerid="user1");



select distinct cid from concert c, user u where u.userid = "user4" and u.lastlogin < c.posttime;



**bands can use to post new concerts**

DELIMITER $$

CREATE PROCEDURE insertBand(In cid varchar(20) ,In bid varchar(20),In gid varchar(20),

In venue varchar(20), in tktsold int, in tprice int, in ctime datetime,in publishtime datetime)

BEGIN

Set publishtime=now();

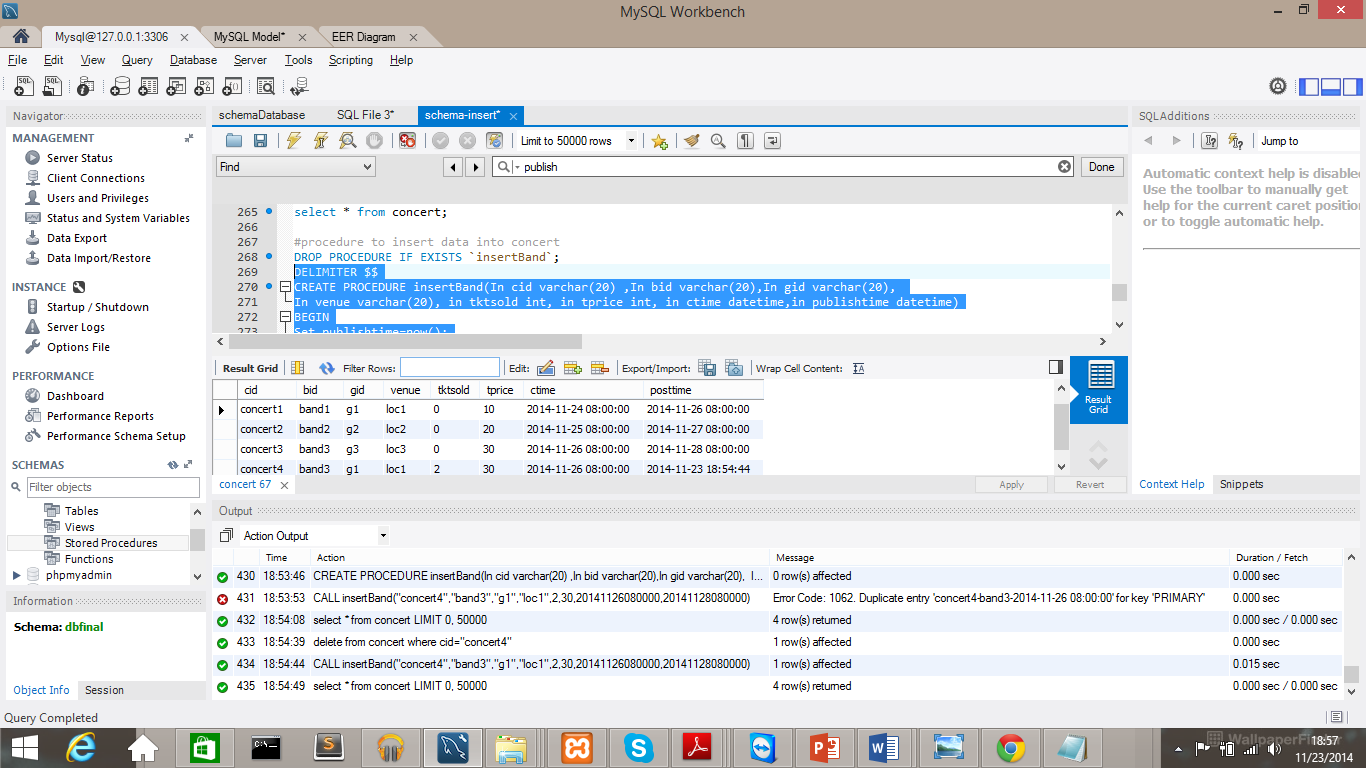
insert into concert(cid,bid,gid,venue,tktsold,tprice,ctime,posttime)

values(cid,bid,gid,venue,tktsold,tprice,ctime,publishtime);

END $$

DELIMITER ;

CALL insertBand("concert4","band3","g1","loc1",2,30,20141126080000,20141128080000);



users can use to post user data (with a check on the user’s trust level)

**DELIMITER $$**

**CREATE PROCEDURE insert\_user\_concert(In cid varchar(20) ,In bid varchar(20),In gid varchar(20),**

**In venue varchar(20), in tktsold int, in tprice int, in ctime datetime,in publishtime datetime)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from ( select u.userid , DATEDIFF(r.rtime,u.starttime) as trust,count(\*) as ratecount**

**from user u natural join user\_rating r group by u.userid) X where X.trust > 100 and X.ratecount>2**

**group by X.userid;**

**if count>0 then**

**Set publishtime=now();**

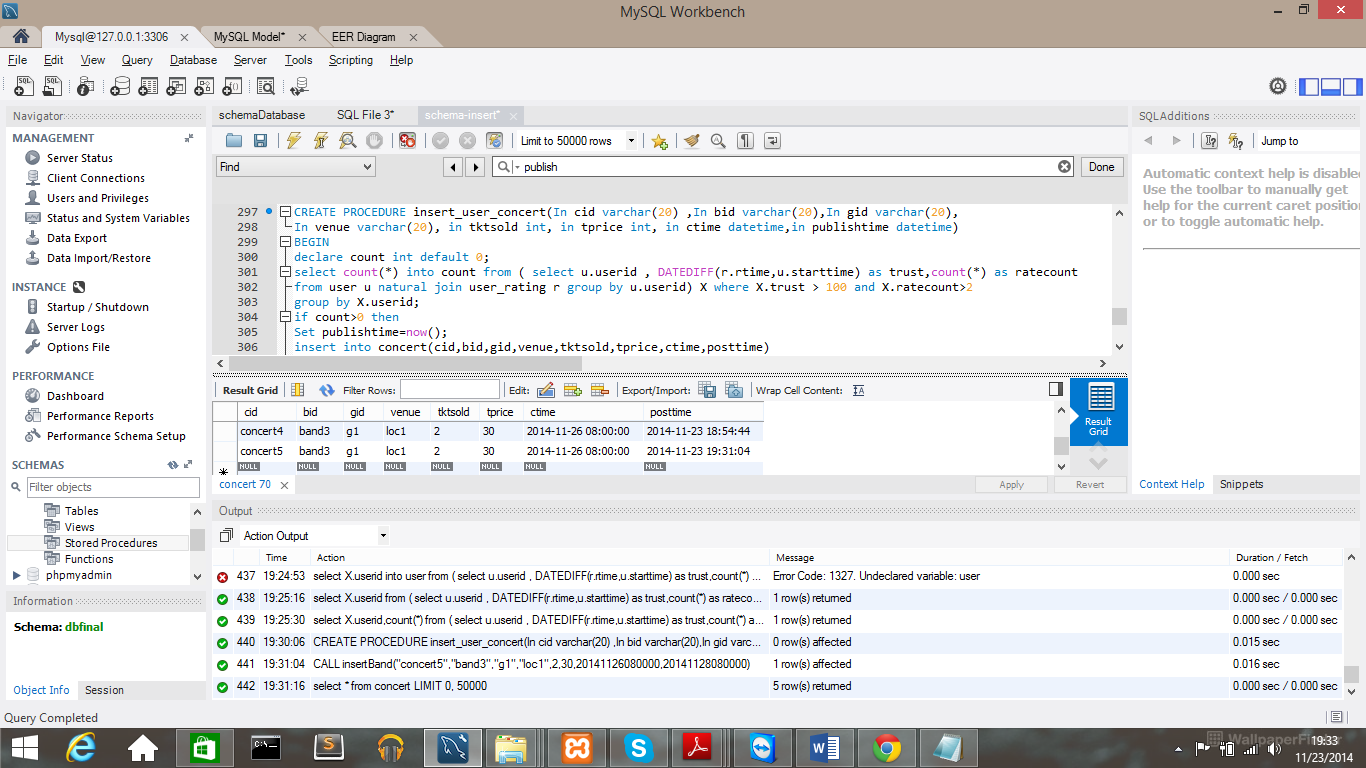
**insert into concert(cid,bid,gid,venue,tktsold,tprice,ctime,posttime)**

**values(cid,bid,gid,venue,tktsold,tprice,ctime,publishtime);**

**end if;**

**END $$**

**CALL insertBand("concert5","band3","g1","loc1",2,30,20141126080000,20141128080000);**



**list of recommended concerts**

**DROP PROCEDURE IF EXISTS `recommendConcert`**

**DELIMITER $$**

**CREATE PROCEDURE recommendConcert(In userID varchar(20))**

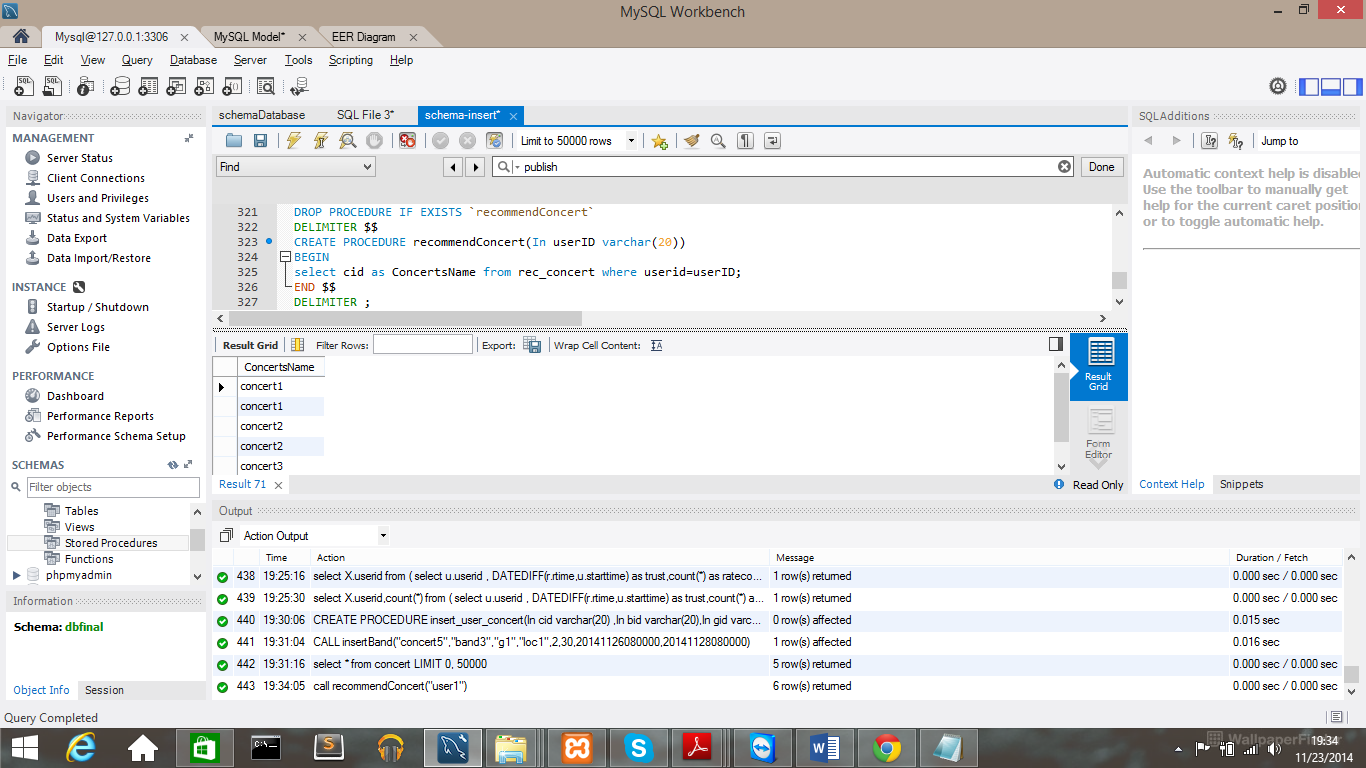
**BEGIN**

**select cid as ConcertsName from rec\_concert where userid=userID;**

**END $$**

**DELIMITER ;**

**call recommendConcert("user1");**



**add concerts to list**

**DROP PROCEDURE IF EXISTS `addtoList`**

**DELIMITER $$**

**CREATE PROCEDURE addtoList(In userID varchar(20), in cid varchar(20),in rec\_time varchar(20))**

**BEGIN**

**Set rec\_time=now();**

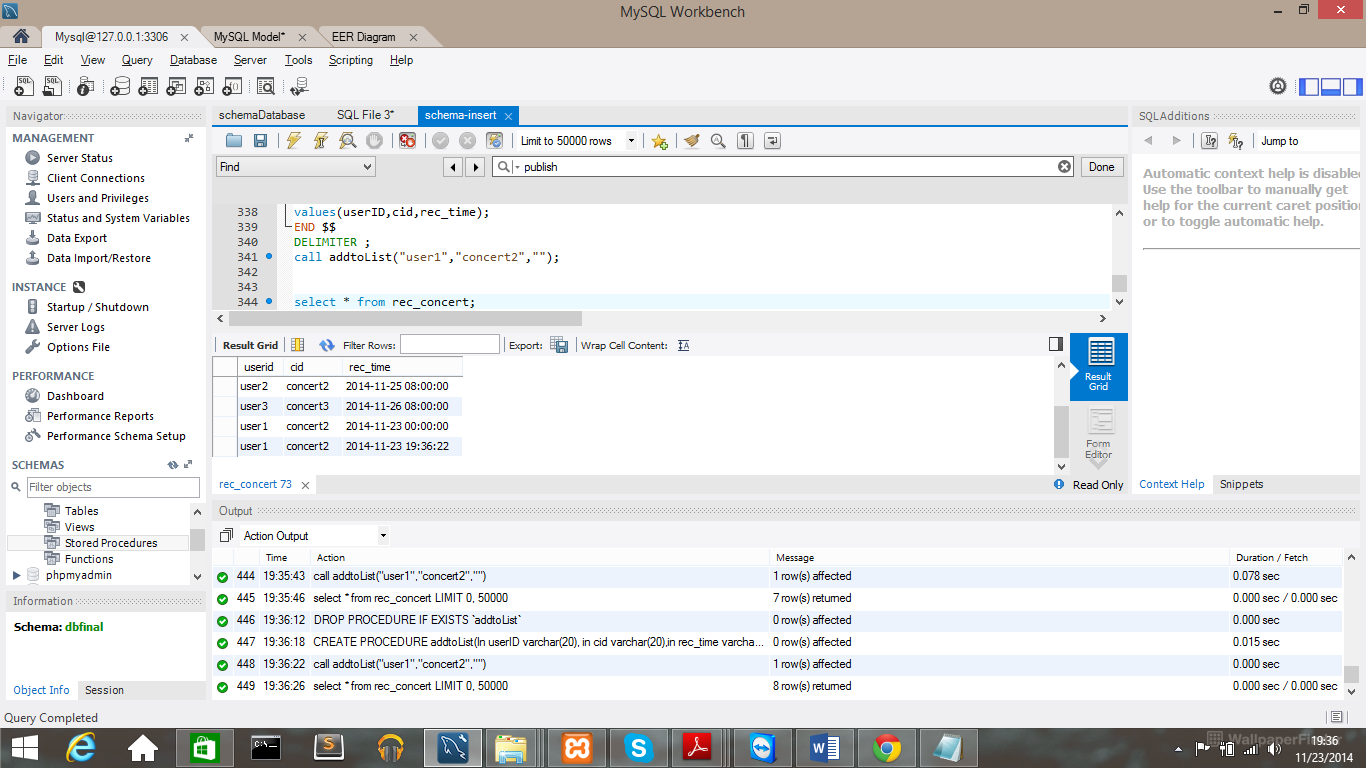
**insert into rec\_concert(userid,cid,rec\_time)**

**values(userID,cid,rec\_time);**

**END $$**

**DELIMITER ;**

**call addtoList("user1","concert2","");**



**Recommend Concerts based on user likes which genre (System Recommends)**

**drop procedure if exists concertrecommendGenre;**

**DELIMITER $$**

**create procedure concertrecommendGenre(in usernumber varchar(20))**

**begin**

**select cid as recommendedConcert from concert natural join user where gid not in**

**(**

**select gid from user natural join genre\_user**

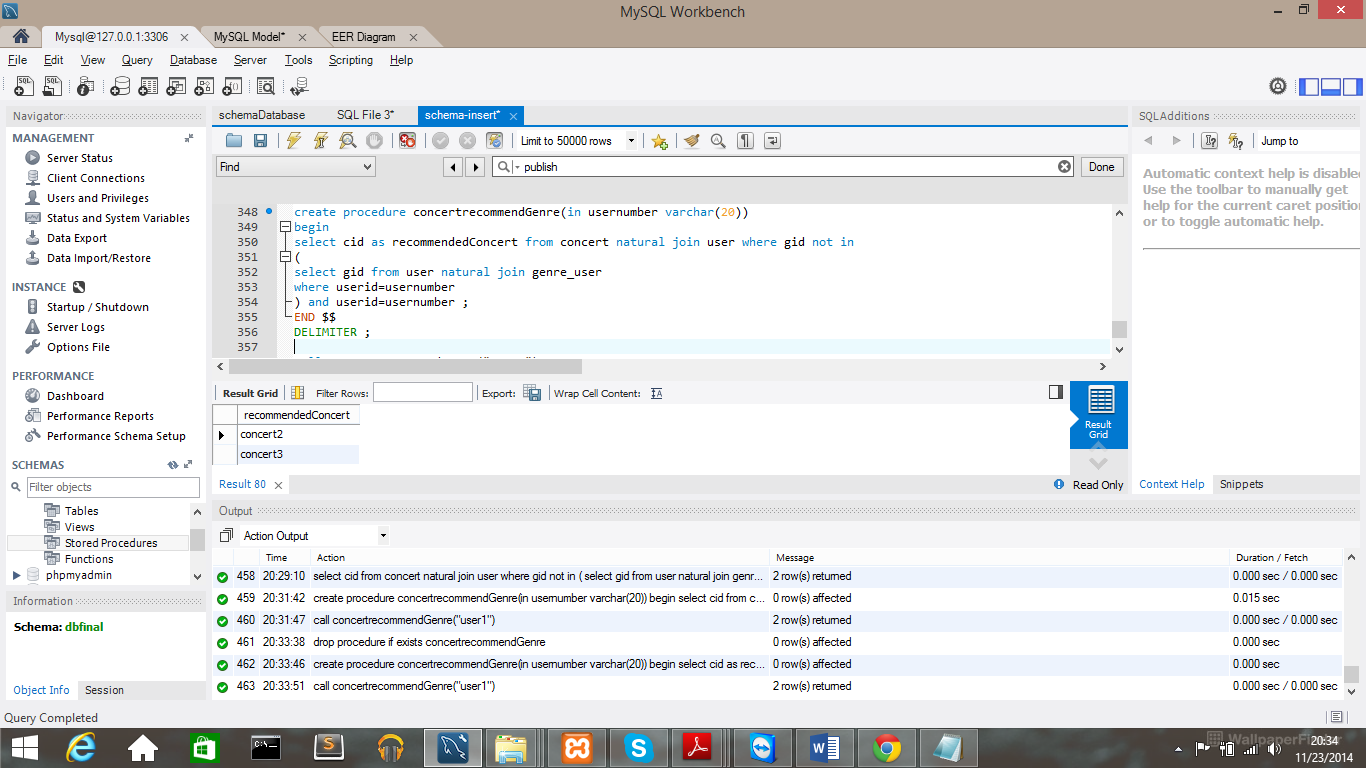
**where userid=usernumber**

**) and userid=usernumber ;**

**END $$**

**DELIMITER ;**

**call concertrecommendGenre("user1");**



Recommend concerts based on the band other users like

**DELIMITER $$**

**create procedure concertrecommendBand(in usernumber varchar(20))**

**begin**

**select cid from concert, (**

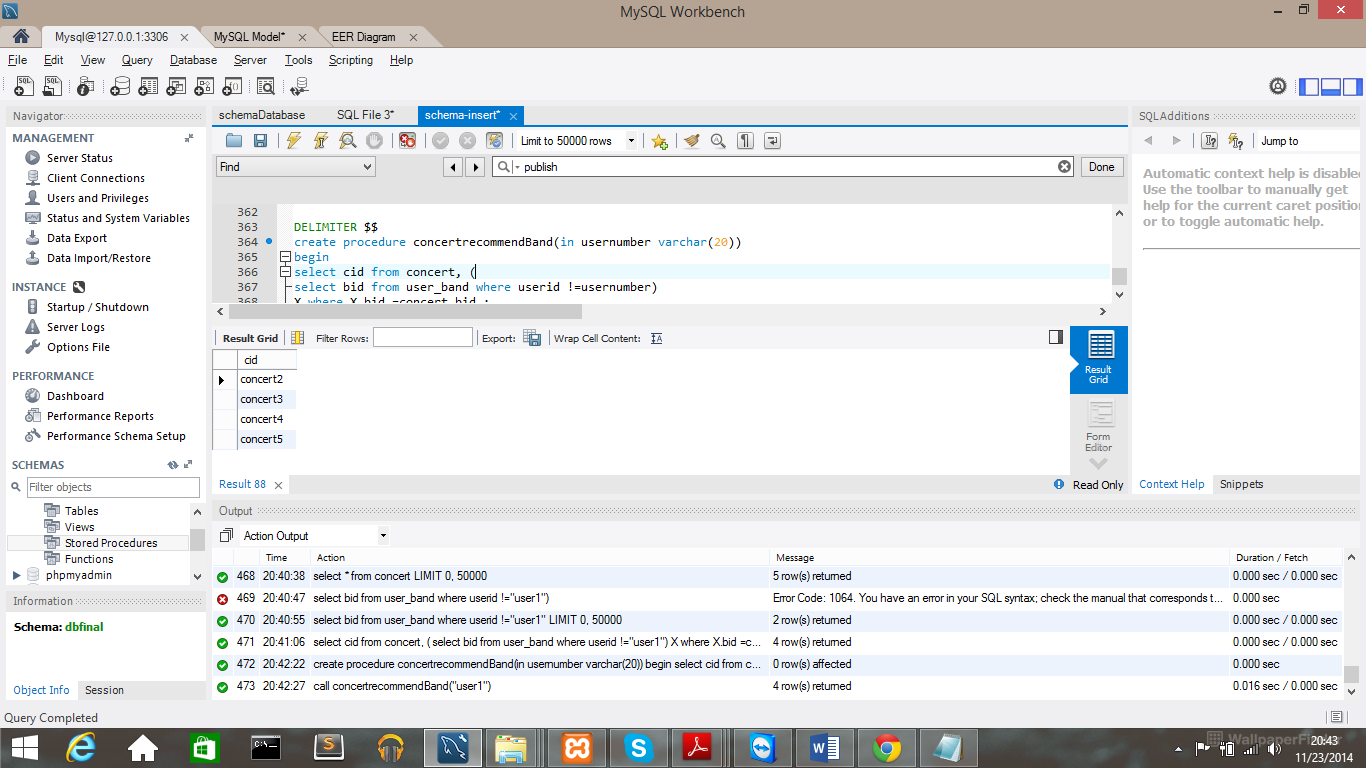
**select bid from user\_band where userid !=usernumber)**

**X where X.bid =concert.bid ;**

**END $$**

**DELIMITER ;**

**call concertrecommendBand("user1");**



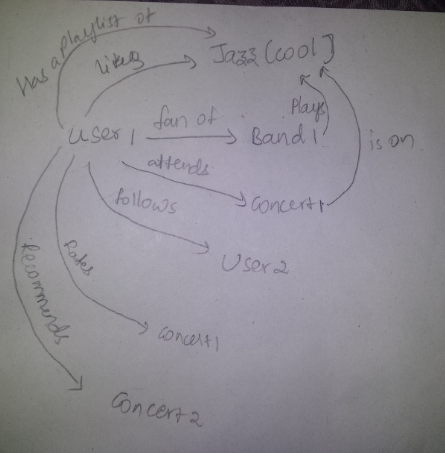
**Query for recommending bands from genre liked by the user:**

**select distinct bid,bname from band natural join genre natural join genre\_band natural join genre\_user**

**where genre\_user.gid="g1" and genre\_user.userid="user1" group by genre.gid;**



# Test Data and its Map

****

# Additional Stored Procedures

**DELIMITER $$**

**create procedure signup(in uid varchar(20),in user varchar(20),in mail varchar(30),**

**in yb year(4),in cty varchar(20),in pswd varchar(20),in strt datetime,in lstlogin datetime,**

**inout flag int)**

**begin**

**declare count int default 0;**

**select count(\*) into count from user where user.userid= uid;**

**if count<1 then**

**set flag=2;**

**insert into user values(uid,user,mail,yb,cty,pswd,strt,lstlogin);**

**else set flag=1;**

**end if;**

**end $$**

**DELIMITER ;**

**DELIMITER $$**

**create procedure bsignup(in bnd varchar(20),in bndname varchar(20),in mail varchar(30),**

**in yb year(4),in cty varchar(20),in pswd varchar(20),in strt datetime,in lstlogin datetime, in about varchar(100),**

**inout flag int)**

**begin**

**declare count int default 0;**

**select count(\*) into count from band where band.bid= bnd;**

**if count<1 then**

**set flag=2;**

**insert into band values(bnd,bndname,mail,yb,cty,pswd,strt,lstlogin,about,"no");**

**else set flag=1;**

**end if;**

**end $$**

**DELIMITER ;**

**DELIMITER $$**

**create procedure create\_or\_edit\_personal(**

**in user varchar(20),**

**in username varchar(20),**

**in yr year(4),**

**in cty varchar(20),**

**in pswd varchar(20),**

**inout update\_user\_flag int)**

**begin**

**if update\_user\_flag=1 then**

**update user set uname = username, yob=yr, city = cty, pwd = pswd where userid=user;**

**end if;**

**end $$**

**DELIMITER ;**

**set @flag=1;**

**call create\_or\_edit\_personal("user5","Venu",1975,"Detroit","pwd5",@flag);**

**select \* from user;**

**DELIMITER $$**

**create procedure create\_or\_edit\_interests(**

**in User varchar(20),**

**in G varchar(20),**

**inout update\_user\_flag int)**

**begin**

**declare count int default 0;**

**select count(\*) into count from genre\_user g where g.userid = User and g.gid=G;**

**if update\_user\_flag=0 and count<1 then**

**set update\_user\_flag=2;**

**insert into genre\_user values(User,G,now());**

**elseif update\_user\_flag=1 then**

**set update\_user\_flag=3;**

**delete from genre\_user where userid=User and gid=G;**

**end if;**

**end $$**

**DELIMITER ;**

**DELIMITER $$**

**create procedure bedit\_personal(**

**in user varchar(20),**

**in username varchar(20),**

**in yr year(4),**

**in cty varchar(20),**

**in pswd varchar(20),**

**in about varchar(100),**

**inout update\_user\_flag int)**

**begin**

**if update\_user\_flag=1 then**

**update band set bname = username, yob=yr, city = cty, pwd = pswd, bio = about where bid=user;**

**end if;**

**end $$**

**DELIMITER ;**

**DELIMITER $$**

**create procedure bedit\_interests(**

**in User varchar(20),**

**in G varchar(20),**

**inout update\_user\_flag int)**

**begin**

**declare count int default 0;**

**select count(\*) into count from genre\_band g where g.bid = User and g.gid=G;**

**if update\_user\_flag=0 and count<1 then**

**set update\_user\_flag=2;**

**insert into genre\_band values(User,G);**

**elseif update\_user\_flag=1 then**

**set update\_user\_flag=3;**

**delete from genre\_band where userid=User and gid=G;**

**end if;**

**end $$**

**DELIMITER ;**

**DELIMITER $$**

**CREATE PROCEDURE fan (in user varchar(20),in band varchar(20),inout flag int)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from user\_band u where u.userid=user and u.bid=band;**

**if count<1 and flag=0 then**

**insert into user\_band values(user,band,now());**

**elseif count>0 and flag=1 then**

**delete from user\_band where userid=user and bid=band;**

**end if;**

**END $$;**

**DELIMITER ;**

**DELIMITER $$**

**CREATE PROCEDURE add\_or\_remove\_follower(In follower varchar(20), in following varchar(20), inout flag int)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from followers f where f.followerid=follower and f.followingid=following;**

**if count<1 and flag = 0 then**

**insert into followers values(follower,following,now());**

**elseif count>0 and flag = 1 then**

**delete from followers where followerid=follower and followingid=following;**

**end if;**

**END $$**

**DELIMITER ;**

**DELIMITER $$**

**create procedure news\_feed(**

**in user varchar(20))**

**begin**

**select distinct \* from concert natural join genre where (concert.bid in (select bid from user\_band where user\_band.userid = user)) or (concert.gid in (select gid from genre\_user where genre\_user.userid = user))**

**or (concert.cid in (select cid from user\_publish u where u.userid = user)) or (select cid from user\_publish u2 where u2.userid in (select followingid from followers where followerid = user));**

**end $$**

**DELIMITER ;**

**DELIMITER $$**

**CREATE PROCEDURE add\_or\_remove\_rating(In user varchar(20), in concert varchar(20), in rate int, comments varchar(30),inout flag int)**

**BEGIN**

**declare count int default 0;**

**select count(\*) into count from user\_rating r where r.userid=user and r.cid=concert;**

**if count<1 and flag = 0 then**

**insert into user\_rating values(user,concert,rate,comments,now());**

**elseif count>0 and flag = 1 then**

**delete from user\_rating where userid=user and cid=concert;**

**end if;**

**END $$**

**DELIMITER ;**

**DELIMITER $$**

**create procedure rsvp(**

**in user varchar(20),**

**in concert varchar(20))**

**begin**

**declare count int default 0;**

**select count(\*) into count from rsvp\_concert r where r.userid=user and r.cid=concert;**

**if count<1 then**

**insert into rsvp\_concert values(user,concert);**

**end if;**

**end $$**

**DELIMITER ;**

# Functionalities Implemented:

* Login for bands and users
* Signup for bands and users
* Create and edit profile for bands and users
* News feed about upcoming concerts
* System recommendations to follow other users and bands
* Search using keywords
* Publish Concerts by Trusted Users and Bands.
* RSVP
* Rate and review concerts
* User specific System recommendations for concerts.
* Connecting homepage to social networking sites Facebook, twitter and YouTube.
* Connecting to Gmail.

# Project Screenshots

