# #DDL

```
create table statuses(
 status varchar(15) not null,
 constraint pk_statuses primary key (status)
);

create table users (
   status varchar(15) not null,
   name varchar(30) not null,
   state varchar(30),
   country varchar(30) not null,
   email varchar(30) not null,
   constraint fk_users_1 foreign key (status) references statuses(status),
   constraint uk_users_1 unique key (email),
   constraint pk_users primary key (name)
);

create table tournaments (
   name varchar(30) not null,
   start_date date not null,
   prize_pool float not null,
   constraint pk_tournaments primary key (name)
);

create table genres (
   name varchar(30) not null,
   constraint pk_genres primary key (name)
);

create table games (
   name varchar(30) not null,
   cost float not null,
   constraint pk_games primary key (name)
);

create table ratings (
   rate_num integer not null,
   rate_name varchar(30) not null,
   constraint pk_ratings primary key (rate_num)
);

create table teams(
   name varchar(30) not null,
   owner varchar(30) not null,
   constraint fk_teams_1 foreign key (owner) references users(name),
   constraint pk_teams primary key (name)
```

```
)

create table communities(
    name varchar(30) not null,
    owner varchar(30) not null,
    constraint fk_communities_1 foreign key (owner) references users(name),
    constraint pk_communities primary key (name)
);

create table events(
    com_name varchar(30) not null,
    event_name varchar(30) not null,
    start_date date not null,
    constraint fk_events_1 foreign key (com_name) references communities(name),
    constraint pk_events primary key (com_name, event_name)
);

create table forums(
    com_name varchar(30) not null,
    title varchar(30) not null,
    constraint fk_forums_1 foreign key (com_name) references communities(name),
    constraint pk_forums primary key (com_name, title)
);

create table community_members(
    com_name varchar(30) not null,
    user varchar(30) not null,
    email varchar(30) not null,
    constraint fk_community_members_1 foreign key (com_name) references
communities(name),
    constraint fk_community_members_2 foreign key (user) references users(name),
    constraint pk_community_members primary key (com_name, user)
);

create table forum_posts(
    com_name varchar(30) not null,
    user varchar(30) not null,
    title varchar(30) not null,
    content varchar(100) not null,
    date_time datetime not null,
    constraint fk_forum_posts_1 foreign key (user, com_name) references
community_members(user, com_name),
    constraint fk_forum_posts_2 foreign key (com_name, title) references
forums(com_name, title),
    constraint pk_forum_posts primary key (com_name, user, title, date_time)
);


create table team_members(
```

```
  team_name varchar(30) not null,
  user varchar(30) not null,
  constraint fk_team_members_1 foreign key (team_name) references teams(name),
  constraint fk_team_members_2 foreign key (user) references users(name),
  constraint pk_team_members primary key (team_name, user)
);

create table tournament_teams(
  team_name varchar(30) not null,
  tournament_name varchar(30) not null,
  constraint fk_tournament_teams_1 foreign key (team_name) references
teams(name),
  constraint fk_tournament_teams_2 foreign key (tournament_name) references
tournaments(name),
  constraint pk_tournament_teams primary key (team_name, tournament_name)
);

create table user_games(
  user_name varchar(30) not null,
  game_name varchar(30) not null,
  constraint fk_user_games_1 foreign key (user_name) references users(name),
  constraint fk_user_games_2 foreign key (game_name) references games(name),
  constraint pk_user_games primary key (user_name, game_name)
);

create table user_genres(
  user_name varchar(30) not null,
  genre_name varchar(30) not null,
  constraint fk_user_genres_1 foreign key (user_name) references users(name),
  constraint fk_user_genres_2 foreign key (genre_name) references genres(name),
  constraint pk_user_genres primary key (user_name, genre_name)
);

create table game_genres(
  game_name varchar(30) not null,
  genre_name varchar(30) not null,
  constraint fk_game_genres_1 foreign key (game_name) references games(name),
  constraint fk_game_genres_2 foreign key (genre_name) references genres(name),
  constraint pk_game_genres primary key (game_name, genre_name)
);

create table game_ratings(
   user_name varchar(30) not null,
   game_name varchar(30) not null,
   rating integer not null,
   comment varchar(100),
   constraint fk_game_ratings_1 foreign key (game_name) references games(name),
   constraint fk_game_ratings_2 foreign key (user_name) references users(name),
```

```
    constraint fk_game_ratings_3 foreign key (rating) references
ratings(rate_num),
    constraint pk_game_ratings primary key (game_name, user_name)
);

create table event_members(
    com_name varchar(30) not null,
    user varchar(30) not null,
    event_name varchar(30) not null,
    constraint fk_event_members_1 foreign key (user, com_name) references
community_members(user, com_name),
    constraint fk_event_members_2 foreign key (com_name, event_name) references
events(com_name, event_name),
    constraint pk_event_members primary key (com_name, user, event_name)
);
```

# #Triggers

#This trigger throws an error whenever a tournament is inserted with a prize_pool that is less than 100.

```
delimiter //
create trigger tournaments_on_insert before insert on tournaments for each row
begin
  if( new.prize_pool < 100 ) then
      signal sqlstate '45000' set message_text = 'A tournament needs a minimum
of $100';
  end if;
end //
delimiter ;
```

#This trigger throws an error whenever a team is inserted for a tournament when the team is already scheduled for a tournament on the same date.

```
delimiter //
create trigger tournament_teams_on_insert before insert on tournament_teams for
each row
begin
  declare t_date date;
  declare result boolean;
  set t_date = (select start_date from tournaments where name =
new.tournament_name);
  set result = exists(
      select 'X'
      from tournament_teams tt inner join tournaments t on tt.tournament_name =
t.name
```

```
      where start_date = t_date and t.name <> new.tournament_name and
tt.team_name = new.team_name
      );
  if( result ) then
      signal sqlstate '45000' set message_text = 'This team is already
participating in a tournament on that day';
  end if;
end //
delimiter ;
```

#This trigger ensures that when a community is inserted, the owner is also inserted as a member of the community.

```
delimiter //
create trigger communities_on_insert after insert on communities for each row
begin
  insert into community_members (com_name, user, email) values (new.name,
new.owner, 'random');
end //
delimiter ;
```

#This trigger ensures that when a community member is deleted, the owner of the community is not deleted

```
delimiter //
create trigger community_members_on_delete before delete on community_members
for each row
begin
  declare c_owner varchar(30);
  set c_owner = (
      select owner
      from communities
      where name = old.com_name
      );
  if ( old.user = c_owner) then
      signal sqlstate '45000' set message_text = 'Cannot delete the owner of a
community.';
  end if;
end //
delimiter ;
```

#This trigger is to enforce the denormalization we chose to implement. Each community member will include the appropriate email for the community member when we insert them.

```
delimiter //
create trigger community_members_on_insert before insert on community_members
for each row
begin
  declare u_email varchar(30);
  set u_email = (
      select email
      from users
```

```
      where name = new.user
      );
  set new.email = u_email;
end //
delimiter ;
```

#This trigger ensures that a comment for a game rating will never include the word bad when it is inserted.

```
delimiter //
create trigger game_ratings_on_insert before insert on game_ratings for each
row
begin
 declare result boolean;
 set result = UPPER(new.comment) REGEXP '\\bBAD\\b';
 if (result) then
     signal sqlstate '45000' set message_text = 'The comment contains the word
"bad".';
 end if;
end //
delimiter ;
```