**HELP**

**Put User in gather :**

Drop procedure if exists

USER\_ATTENDS\_EVENT

GO

CREATE PROCEDURE USER\_ATTENDS\_EVENT

@userId int

,@Event int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

INSERT INTO GATHERING\_Attendances(GATHERING\_ID,UserID) VALUES (@Event,@userId)

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec USER\_ATTENDS\_EVENT 5,4,@wasSuccessful out;

select @wasSuccessful

**Get Average Age**

Drop procedure if exists

GET\_AVERAGE\_AGE

Go

CREATE PROCEDURE GET\_AVERAGE\_AGE

AS

BEGIN

SELECT AVG(AGE) as AGEE

FROM (

SELECT USERS.UserID,DATEDIFF(year,USERS.Birthday,GETDATE()) AS AGE

FROM USERS ) as tmp

END

GO

exec GET\_AVERAGE\_AGE ;

**Get Photos Based on Album ID :**

GO

Drop procedure if exists

SHOW\_ALBUMS\_PICTURES

GO

CREATE PROCEDURE SHOW\_ALBUMS\_PICTURES

@idOfAlbum int

AS

BEGIN

SELECT pic.\*

FROM Picture pic, Picture\_in\_Album PiA

WHERE PiA.Picture\_Album\_ID =@idOfAlbum ;

END

GO

DECLARE @idOfAlbum int

SET @idOfAlbum=3;

exec SHOW\_ALBUMS\_PICTURES @idOfAlbum;**g.**



**Get Friend Requests**

Drop procedure if exists

GET\_FRIEND\_REQUESTS

Go

CREATE PROCEDURE GET\_FRIEND\_REQUESTS

@idOwner int

AS

BEGIN

SELECT ur.\*

FROM Friend\_Request Fr , USERS ur

WHERE Fr.User\_Who\_Received\_request\_ID =@idOwner AND Fr.ignored=0 AND

Fr. User\_Who\_Sent\_request\_ID = ur.UserID ;

END

GO

declare @idOwner int;

set @idOwner=1;

exec GET\_FRIEND\_REQUESTS @idOwner

**SHOW IGNORED ALSO .**

**Get Friend Requests**

Drop procedure if exists GET\_FRIEND\_REQUESTS\_IGNORED

GO

CREATE PROCEDURE GET\_FRIEND\_REQUESTS\_IGNORED

@idOwner int

AS

BEGIN

SELECT ur.\*

FROM Friend\_Request Fr ,USERS ur

WHERE Fr.User\_Who\_Received\_request\_ID =@idOwner AND Fr.ignored=1 AND

Fr. User\_Who\_Sent\_request\_ID = ur.UserID ;

END

GO

exec GET\_FRIEND\_REQUESTS\_IGNORED 1

**REMOVE FROM IGNORED .**

Drop procedure if exists

REMOVE\_IGNORE

GO

CREATE PROCEDURE REMOVE\_IGNORE

@idOwner int

, @idToRemoveIgn int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

UPDATE Friend\_Request

SET IGNORED=0

WHERE User\_Who\_Received\_request\_ID=@idOwner AND User\_Who\_Sent\_request\_ID=@idToRemoveIgn ;

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec REMOVE\_IGNORE 4,2,@wasSuccessful out;

Send a friend request

Drop procedure if exists

SEND\_A\_FRIEND\_REQUEST

GO

CREATE PROCEDURE SEND\_A\_FRIEND\_REQUEST

@idOwner int

,@idToSent int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

INSERT INTO Friend\_Request(User\_Who\_Sent\_request\_ID,User\_Who\_Received\_request\_ID,ignored) VALUES (@idOwner,@idTosent,0)

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec SEND\_A\_FRIEND\_REQUEST 5,1,@wasSuccessful out;

select @wasSuccessful

**DELETE a friend request .**

Drop procedure if exists

DELETE\_A\_REQUEST

GO

CREATE PROCEDURE DELETE\_A\_REQUEST

@idOwner int

,@idToBeDelete int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

DELETE FROM Friend\_Request

WHERE User\_Who\_Received\_request\_ID=@idOwner AND User\_Who\_Sent\_request\_ID=@idToBeDelete ;

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec DELETE\_A\_REQUEST 5,1,@wasSuccessful out;

select @wasSuccessful

**IGNORE a friend request .**

Drop procedure if exists

IGNORE\_A\_REQUEST

GO

CREATE PROCEDURE IGNORE\_A\_REQUEST

@idOwner int

, @idToIgnore int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

UPDATE Friend\_Request

SET IGNORED=1

WHERE User\_Who\_Received\_request\_ID=@idOwner AND User\_Who\_Sent\_request\_ID=@idToIgnore ;

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec IGNORE\_A\_REQUEST 4,1,@wasSuccessful out;

**ADD afriend request**

Drop procedure if exists

ACCEPT\_A\_REQUEST

GO

CREATE PROCEDURE ACCEPT\_A\_REQUEST

@idOwner int

, @idToBeAccepted int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

DELETE FROM Friend\_Request

WHERE User\_Who\_Received\_request\_ID=@idOwner AND User\_Who\_Sent\_request\_ID=@idToBeAccepted ;

INSERT into Friendship(UserS\_ID, UserR\_ID)

VALUES(@idToBeAccepted, @idOwner);

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec ACCEPT\_A\_REQUEST 4,1,@wasSuccessful out;

**h.**

Διαχείριση Φίλων. Το σύστηµα πρέπει να µπορεί να δώσει µια κατάσταση µε όλους του φίλους του χρήστη. Μέσω αυτής της κατάστασης ο χρήστης µπορεί να διαγράψει κάποιον φίλο. Αυτό όµως δεν εµποδίζει τον διαγραµµένο φίλο να στείλει άλλη αίτηση φιλίας στο µέλλον.

SHOW FRIENDS

GO

Drop procedure if exists

SHOW\_FRIENDS

GO

CREATE PROCEDURE SHOW\_FRIENDS

@idOwner int

AS

BEGIN

SELECT DISTINCT ur.\*

FROM FriendShip FrShip , USERS ur

WHERE FrShip.UserS\_ID =@idOwner AND FrShip.UserR\_ID = ur. UserID ;

END

GO

DECLARE @idOwner int

SET @idOwner=1;

exec SHOW\_FRIENDS @idOwner;

DELETE FRIEND

Drop procedure if exists

DELETE\_FRIENDS

GO

CREATE PROCEDURE DELETE\_FRIENDS

@idOwner int

,@idToBeDeleted int

,@wasSuccessful bit OUT

AS

BEGIN

declare @transactionName varchar(20);

SELECT @transactionName = 'MyTransaction';

BEGIN TRANSACTION @transactionName

DELETE FROM FriendShip

WHERE (UserS\_ID=@idOwner AND UserR\_ID=@idToBeDeleted) ;

IF @@ROWCOUNT = 0

begin

set @wasSuccessful=0

end

else

begin

set @wasSuccessful=1

end

if @@ROWCOUNT>1

begin

ROLLBACK;

end

COMMIT TRANSACTION @transactionName;

END

GO

DECLARE @wasSuccessful bit

exec DELETE\_FRIENDS 1,2,@wasSuccessful out;

SELECT @wasSuccessful

**n.**

**Διαχείριση Συµβάντων. Με αυτή τη λειτουργία ένα χρήστης µπορεί να ψάξει και να απαντήσει σε συµβάντα που έχουν δηλώσει άλλοι χρήστες βάσει πολλαπλών πεδίων (περιοχή, όνοµα, κτλ.) Για αυτή την ερώτηση λάβετε υπόψη τις επιλογές προσωπικού απορρήτου.**

**o.**

**Εύρεση Δηµοφιλέστερων Φίλων. Με αυτή τη λειτουργία ένας χρήστης µπορεί να βρει τον (ή τους) δηµοφιλέστερο/ους φίλους του σε περίπτωση ισοβαθµίας. Δηµοφιλέστερος φίλος ορίζετε κάποιος που έχει τους περισσότερους επικυρωµένους φίλους (όχι απλά προσκλήσεις). Για αυτή την ερώτηση λάβετε υπόψη τις επιλογές προσωπικού απορρήτου.**

VER 2

drop procedure if exists MOST\_POPULAR\_FRIENDS

GO

CREATE PROCEDURE MOST\_POPULAR\_FRIENDS

@id int

AS

BEGIN

IF OBJECT\_ID(N'dbo..#tmp\_users\_friends') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'dbo..#MaxFriends') IS NOT NULL

BEGIN

DROP TABLE #MaxFriends

END

IF OBJECT\_ID(N'dbo..#Friends\_With\_Count') IS NOT NULL

BEGIN

DROP TABLE #Friends\_With\_Count

END

/\* getting all the users friends ID \*/

SELECT F. UserR\_ID AS FriendsID

INTO #tmp\_users\_friends

FROM [FriendShip] AS F

WHERE F.UserS\_ID=@id

/\* Getting the amount of friends each friend has\*/

SELECT #tmp\_users\_friends.FriendsID , COUNT(UserID) AS MYFRIENDS

INTO #Friends\_With\_Count

FROM #tmp\_users\_friends, [FriendShip], [USERS]

WHERE #tmp\_users\_friends.FriendsID = FriendShip.UserR\_ID AND FriendShip.UserS\_ID = USERS.UserID AND USERS.Is\_verified=1

GROUP BY #tmp\_users\_friends.FriendsID

/\* Getting the MAX VALUE\*/

SELECT MAX(MYFRIENDS) AS theNumber

INTO #MaxFriends

FROM #Friends\_With\_Count

/\* Getting the IDS of the users friends that have the MAX number of friends between them making them the most popular\*/

SELECT userr.\*

FROM #Friends\_With\_Count,#MaxFriends , USERS userr

WHERE #Friends\_With\_Count.MYFRIENDS=#MaxFriends.theNumber AND userr.UserID=#Friends\_With\_Count.FriendsID

IF OBJECT\_ID(N'dbo..#tmp\_users\_friends') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'dbo..#MaxFriends') IS NOT NULL

BEGIN

DROP TABLE #MaxFriends

END

IF OBJECT\_ID(N'dbo..#Friends\_With\_Count') IS NOT NULL

BEGIN

DROP TABLE #Friends\_With\_Count

END

END

GO

exec MOST\_POPULAR\_FRIENDS 1

**p**. Εύρεση Κοινών Φίλων. Το ζητούµενο εδώ είναι να βρεθούν ποιοι φίλοι ενός χρήστη έχουν (τουλάχιστον) τους ίδιους φίλους µε τον χρήστη.

drop procedure if exists FRIENDS\_WITH\_ATLEAST\_SAME\_FRIENDS

GO

CREATE PROCEDURE FRIENDS\_WITH\_ATLEAST\_SAME\_FRIENDS

@id int

AS

BEGIN

IF OBJECT\_ID(N'dbo..#tmp\_users\_friends') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'dbo..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

DECLARE @whichFriend as int

DECLARE @CUR as cursor

CREATE TABLE #ResultedUsers(UserID int,Username varchar(50),Pass varchar(50),First\_Name varchar(50),Last\_Name varchar(50),Email varchar(255),Website varchar(1000),Link varchar(1000),Birthday date,Gender bit,Is\_verified bit,Hometown\_LOC\_ID int,Current\_LOC\_ID int)

SELECT Fr.UserR\_ID AS FriendID

INTO #tmp\_users\_friends

FROM [FriendShip] AS Fr

WHERE Fr.UserS\_ID=@id

set @CUR = cursor for

( SELECT Fr.UserR\_ID

FROM [FriendShip] AS Fr

WHERE Fr.UserS\_ID=@id )

OPEN @CUR;

FETCH NEXT FROM @CUR INTO @whichFriend;

WHILE @@FETCH\_STATUS=0

BEGIN

INSERT INTO #ResultedUsers

SELECT users.\*

FROM [USERS] users

WHERE users.UserID = @whichFriend

AND NOT EXISTS(

(SELECT #tmp\_users\_friends.FriendID

FROM #tmp\_users\_friends

WHERE #tmp\_users\_friends.FriendID!=@whichFriend)

EXCEPT

(SELECT Fr.UserR\_ID

FROM [FriendShip] AS Fr

WHERE Fr.UserS\_ID=@whichFriend AND Fr.UserR\_ID!=@id

)

)

fetch next from @CUR into @whichFriend

end;

SELECT \*

FROM #ResultedUsers

DROP TABLE #ResultedUsers

CLOSE @CUR

DEALLOCATE @CUR

IF OBJECT\_ID(N'dbo..#tmp\_users\_friends') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'dbo..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

END

GO

exec FRIENDS\_WITH\_ATLEAST\_SAME\_FRIENDS 1

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

q. Εύρεση Φίλων µε τα Μεγαλύτερα Άλµπουµ. Αναζήτηση για τους άµεσα φίλους του χρήστη που έχουν άλµπουµ µε περισσότερες από X φωτογραφίες. (το Χ είναι παράµετρος που καθορίζεται την ώρα της εκτέλεσης.)

drop procedure if exists FRIENDS\_WITH\_BIGGER\_ALBUM\_THAN\_X

GO

CREATE PROCEDURE FRIENDS\_WITH\_BIGGER\_ALBUM\_THAN\_X

@id int

,@HowBig int

AS

BEGIN

IF OBJECT\_ID(N'ccosti01..#tmp\_users\_friends') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'ccosti01..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

DECLARE @whichFriend as int

DECLARE @CUR as cursor

CREATE TABLE #ResultedUsers(UserID int,Username varchar(50),Pass varchar(50),First\_Name varchar(50),Last\_Name varchar(50),Email varchar(255),Website varchar(1000),Link varchar(1000),Birthday date,Gender bit,Is\_verified bit,Hometown\_LOC\_ID int,Current\_LOC\_ID int)

SELECT Fr.UserR\_ID AS FriendID

INTO #tmp\_users\_friends

FROM [FriendShip] AS Fr

WHERE Fr.UserS\_ID=@id

set @CUR = cursor for

( SELECT Fr.UserR\_ID

FROM [FriendShip] AS Fr

WHERE Fr.UserS\_ID=@id )

OPEN @CUR;

FETCH NEXT FROM @CUR INTO @whichFriend;

WHILE @@FETCH\_STATUS=0

BEGIN

INSERT INTO #ResultedUsers

SELECT users.\*

FROM [USERS] users

WHERE users.UserID = @whichFriend

AND EXISTS(

((SELECT COUNT(Picture\_ID) AS THISMANYPICTURES

FROM Picture\_Album PA , Picture\_in\_Album PiA

WHERE PA.UserID=@whichFriend AND PiA.Picture\_Album\_ID = PA.Picture\_Album\_ID

GROUP BY PiA.Picture\_Album\_ID

HAVING COUNT(Picture\_ID) > @HowBig

)

)

)

fetch next from @CUR into @whichFriend

end;

SELECT \*

FROM #ResultedUsers

DROP TABLE #ResultedUsers

CLOSE @CUR

DEALLOCATE @CUR

IF OBJECT\_ID(N'ccosti01..#tmp\_users\_friends') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'ccosti01..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

END

GO

exec FRIENDS\_WITH\_BIGGER\_ALBUM\_THAN\_X 1,2;

r. Εύρεση Χρηστών µε τα Μεγαλύτερα Άλµπουµ. Η αναζήτηση του ερωτήµατος (q) αλλά για τους χρήστες που ανήκουν στο δίκτυο του χρήστη. Για αυτή την ερώτηση λάβετε υπόψη τις επιλογές προσωπικού απορρήτου.

drop procedure if exists FRIENDS\_NETOWRK\_WITH\_BIGGER\_ALBUM\_THAN\_X

GO

CREATE PROCEDURE FRIENDS\_NETOWRK\_WITH\_BIGGER\_ALBUM\_THAN\_X

@id int

,@HowBig int

AS

BEGIN

IF OBJECT\_ID(N'ccosti01..#temp') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'ccosti01..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

DECLARE @whichFriend as int

DECLARE @CUR as cursor

CREATE TABLE #ResultedUsers(UserID int,Username varchar(50),Pass varchar(50),First\_Name varchar(50),Last\_Name varchar(50),Email varchar(255),Website varchar(1000),Link varchar(1000),Birthday date,Gender bit,Is\_verified bit,Hometown\_LOC\_ID int,Current\_LOC\_ID int)

;WITH lvl1 as

(

SELECT Fr.UserR\_ID as FRIENDS\_ID

FROM [FriendShip] Fr

WHERE Fr.UserS\_ID = 1

)

,

lvl2 as

(

SELECT DISTINCT Fr.UserR\_ID as FRIENDS\_ID

FROM [FriendShip] Fr

WHERE Fr.UserS\_ID IN (SELECT FRIENDS\_ID FROM lvl1)

)

,

lvl3 as

(

SELECT DISTINCT Fr.UserR\_ID as FRIENDS\_ID

FROM [FriendShip] Fr

WHERE Fr.UserS\_ID IN (SELECT FRIENDS\_ID from lvl2)

)

SELECT tmp.FRIENDS\_ID

INTO #temp

FROM

(select \* from lvl1

union

select \* from lvl2

union

select \* from lvl3) AS tmp

set @CUR = cursor for

( SELECT Fr.FRIENDS\_ID

FROM #temp AS Fr

)

OPEN @CUR;

FETCH NEXT FROM @CUR INTO @whichFriend;

WHILE @@FETCH\_STATUS=0

BEGIN

INSERT INTO #ResultedUsers

SELECT users.\*

FROM [USERS] users

WHERE users.UserID = @whichFriend

AND EXISTS(

((SELECT COUNT(Picture\_ID) AS THISMANYPICTURES

FROM Picture\_Album PA , Picture\_in\_Album PiA

WHERE PA.UserID=@whichFriend AND PiA.Picture\_Album\_ID = PA.Picture\_Album\_ID

GROUP BY PiA.Picture\_Album\_ID

HAVING COUNT(Picture\_ID) > @HowBig

)

)

)

fetch next from @CUR into @whichFriend

end;

SELECT \*

FROM #ResultedUsers

DROP TABLE #ResultedUsers

CLOSE @CUR

DEALLOCATE @CUR

IF OBJECT\_ID(N'ccosti01..#temp') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_friends

END

IF OBJECT\_ID(N'ccosti01..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

END

GO

exec FRIENDS\_NETOWRK\_WITH\_BIGGER\_ALBUM\_THAN\_X 1,2;

s. Εύρεση Φίλων µε Κοινά Ενδιαφέροντα. Αυτή η αναζήτηση πρέπει να επιστρέψει τους άµεσα φίλους του χρήστη που έχουν ακριβώς τα ίδια ενδιαφέροντα µε αυτόν.

drop procedure if exists FRIENDS\_WITH\_SAME\_INTERESTS

GO

CREATE PROCEDURE FRIENDS\_WITH\_SAME\_INTERESTS

@id int

AS

BEGIN

IF OBJECT\_ID(N'ccosti01..#tmp\_users\_Interests') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_Interests

END

IF OBJECT\_ID(N'ccosti01..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

DECLARE @whichFriend as int

DECLARE @CUR as cursor

CREATE TABLE #ResultedUsers(UserID int,Username varchar(50),Pass varchar(50),First\_Name varchar(50),Last\_Name varchar(50),Email varchar(255),Website varchar(1000),Link varchar(1000),Birthday date,Gender bit,Is\_verified bit,Hometown\_LOC\_ID int,Current\_LOC\_ID int)

SELECT UI.Interest\_ID AS myInterest

INTO #tmp\_users\_Interests

FROM [User\_Interest] UI

WHERE UI.UserID=@id

set @CUR = cursor for

( SELECT Fr.UserR\_ID

FROM [FriendShip] AS Fr

WHERE Fr.UserS\_ID=@id )

OPEN @CUR;

FETCH NEXT FROM @CUR INTO @whichFriend;

WHILE @@FETCH\_STATUS=0

BEGIN

INSERT INTO #ResultedUsers

SELECT users.\*

FROM [USERS] users

WHERE users.UserID = @whichFriend

AND NOT EXISTS(

(SELECT \*

FROM #tmp\_users\_Interests)

UNION

(SELECT UF.Interest\_ID

FROM User\_Interest UF

WHERE UF.UserID=@whichFriend

)

EXCEPT

(SELECT \*

FROM #tmp\_users\_Interests

)intersect(

SELECT UF.Interest\_ID

FROM User\_Interest UF

WHERE UF.UserID=@whichFriend

)

)

fetch next from @CUR into @whichFriend

end;

SELECT \*

FROM #ResultedUsers

DROP TABLE #ResultedUsers

CLOSE @CUR

DEALLOCATE @CUR

IF OBJECT\_ID(N'ccosti01..#tmp\_users\_Interests') IS NOT NULL

BEGIN

DROP TABLE #tmp\_users\_Interests

END

IF OBJECT\_ID(N'ccosti01..#ResultedUsers') IS NOT NULL

BEGIN

DROP TABLE #ResultedUsers

END

END

GO

exec FRIENDS\_WITH\_SAME\_INTERESTS 1

t. Μη-Δηµοφιλή Συµβάντα. Αυτή η αναζήτηση πρέπει να επιστρέψει τα συµβάντα µε τις ελάχιστες συµµετοχές

drop procedure if exists SHOW\_LEAST\_POPULAR\_GATHERINGS

GO

CREATE PROCEDURE SHOW\_LEAST\_POPULAR\_GATHERINGS

AS

BEGIN

IF OBJECT\_ID(N'ccosti01..#MinNumber') IS NOT NULL

BEGIN

DROP TABLE #MinNumber

END

IF OBJECT\_ID(N'ccosti01..#tmp.ATTENDENTS') IS NOT NULL

BEGIN

DROP TABLE #tmp.ATTENDENTS

END

SELECT GA.GATHERING\_ID AS GatherID,COUNT(UserID) AS ATTENDENTS

into #tmp

FROM [GATHERING\_Attendances] GA

GROUP BY GA.GATHERING\_ID

SELECT MIN(ATTENDENTS) AS VeryLow

INTO #MinNumber

FROM #tmp

SELECT G.\* , #tmp.ATTENDENTS

from [GATHERING] G ,#MinNumber, #tmp

where G.GATHERING\_ID = #tmp.GatherID AND #tmp.ATTENDENTS = #MinNumber.VeryLow

IF OBJECT\_ID(N'ccosti01..#MinNumber') IS NOT NULL

BEGIN

DROP TABLE #MinNumber

END

IF OBJECT\_ID(N'ccosti01..#tmp.ATTENDENTS') IS NOT NULL

BEGIN

DROP TABLE #tmp.ATTENDENTS

END

end;

GO

exec SHOW\_LEAST\_POPULAR\_GATHERINGS