Player base management system

A Database Systems Project

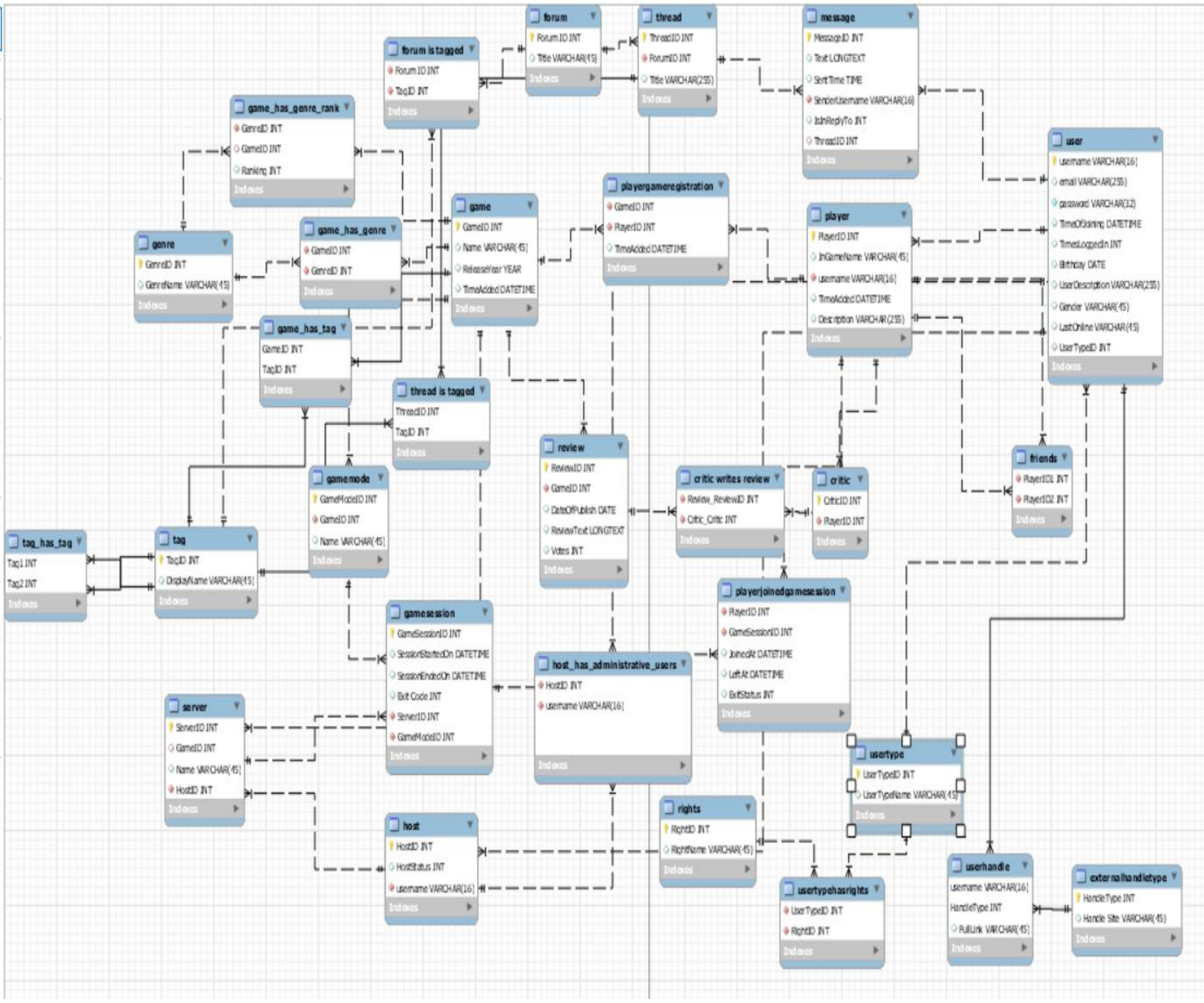
By

Muhammad Muneeb

CMS: 296133

Class: BSCS9 C

ERD:


forum
ForumID INT
Title VARCHAR(45)
thread
ThreadID INT
ForumID INT
Title VARCHAR(255)
message
MessageID INT
Text LONGTEXT
SentTime TIME
SenderUsername VARCHAR(16)
IsInReplyTo INT
ThreadID INT
forum istagged
ForumID INT
TagID INT
game_has_genre_rank
GenreID INT
GameID INT
Ranking INT
user
username VARCHAR(16)
email VARCHAR(255)
password VARCHAR(12)
TimeOfJoining DATETIME
TimesLoggedIn INT
Birthday DATE
UserDescription VARCHAR(255)
Gender VARCHAR(45)
LastOnline VARCHAR(45)
UserTypeID INT
playergameregistration
GameID INT
PlayerID INT
TimeAdded DATETIME
game
GameID INT
Name VARCHAR(45)
ReleaseYear YEAR
TimeAdded DATETIME
player
PlayerID INT
InGameName VARCHAR(45)
username VARCHAR(16)
TimeAdded DATETIME
Description VARCHAR(255)
genre
GenreID INT
GenreName VARCHAR(45)
game_has_genre
GameID INT
GenreID INT
game_has_tag
GameID INT
TagID INT
thread is tagged
ThreadID INT
TagID INT
review
ReviewID INT
GameID INT
DateOfPublish DATE
ReviewText LONGTEXT
Votes INT
critic writes review
Review_ReviewID INT
Critic_Critic INT
critic
CriticID INT
PlayerID INT
friends
PlayerID1 INT
PlayerID2 INT
tag_has_tag
Tag1 INT
Tag2 INT
tag
TagID INT
DisplayName VARCHAR(45)
gamemode
GameModeID INT
GameID INT
Name VARCHAR(45)
playerjoinedgamesession
PlayerID INT
GameSessionID INT
JoinedAt DATETIME
LeftAt DATETIME
ExitStatus INT
gamesession
GameSessionID INT
SessionStartedOn DATETIME
SessionEndedOn DATETIME
Exit Code INT
ServerID INT
GameModeID INT
host_has_administrative_users
HostID INT
username VARCHAR(16)
server
ServerID INT
GameID INT
Name VARCHAR(45)
HostID INT
usertype
UserTypeID INT
UserTypeName VARCHAR(45)
rights
RightID INT
RightName VARCHAR(45)
host
HostID INT
HostStatus INT
username VARCHAR(16)
usertypehasrights
UserTypeID INT
RightID INT
userhandle
username VARCHAR(16)
HandleType INT
FullLink VARCHAR(45)
externalhandletype
HandleType INT
Handle Site VARCHAR(45)
Indexes

Query Criteria:

A. 5 views using different kind of joins

B. 3 triggers

C. 3 queries each of create, update, read, delete

D. 6 queries using aggregate functions with at least 3 using group by

E. 3 Subqueries each for select, from and where

Query 1:

```
CREATE TABLE Right (
  RightID INT NOT NULL AUTO_INCREMENT,
  RightName VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (RightID))
```

- Creates a table named Right in the default database.

Criteria Fulfilled:

- Query of create (C).

Query 2:

```
CREATE TABLE usertype (
  UserTypeID INT NOT NULL AUTO_INCREMENT,
  UserTypeName VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (UserTypeID))
```

- Creates a table named UserType in the default db.

Criteria Fulfilled:

- Query of create (C).

Query 3:

```
CREATE TABLE usertypehasright (
  UserTypeID INT NOT NULL,
  RightID INT NOT NULL,
  CONSTRAINT rightuthr
    FOREIGN KEY (RightID)
    REFERENCES rights (RightID) ON DELETE CASCADE  ON UPDATE CASCADE,
  CONSTRAINT usertype
    FOREIGN KEY (UserTypeID)
    REFERENCES usertype (UserTypeID) ON DELETE CASCADE ON UPDATE CASCADE
    )
```

- Creates a table named UserTypeHasRight in the default db, creating a m:n relationship between UserType and Right.

Criteria Fulfilled:

- Query of create (C).

Query 4:

SET @target = 1;

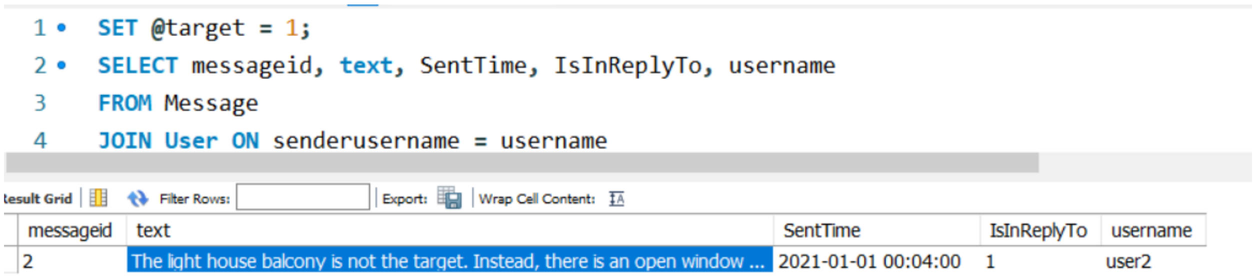
SELECT *

FROM Message

WHERE message.IsInReplyTo = @target

Order By SentTime ASC;

Screenshot:



```
SET @target = 1;
SELECT messageid, text, SentTime, IsInReplyTo, username
FROM Message
JOIN User ON senderusername = username
```

messageid	text	SentTime	IsInReplyTo	username
2	The light house balcony is not the target. Instead, there is an open window ...	2021-01-01 00:04:00	1	user2

- Gets all messages that address the message set in @target, irrespective of thread and forum, and orders them by SentTime ascendingly.

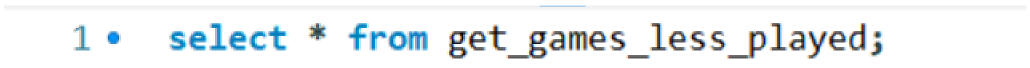
Criteria Fulfilled:

- Read (C)

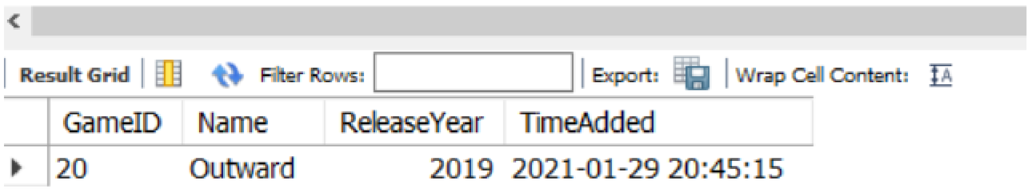
Query 5:

```
CREATE VIEW get_games_less_played AS
SELECT * FROM game
WHERE GameID IN(
	SELECT GameID FROM GameMode WHERE GameModeID IN(
		SELECT GameModeID FROM GameSession WHERE GameSessionID IN(
			SELECT GameSessionID as player_counts FROM
playerjoinedgamesession
				GROUP BY GameSessionID
				HAVING count(PlayerID) < (
					SELECT AVG(player_counts) FROM (
						SELECT GameSessionID, count(PlayerID) as
player_counts FROM playerjoinedgamesession
						GROUP BY GameSessionID) as counts
					)


		)
	)
)
```

Screenshot:



```
select * from get_games_less_played;
```



GameID	Name	ReleaseYear	TimeAdded
20	Outward	2019	2021-01-29 20:45:15

- Gets all games for which the number of players joining gamesessions held is less than the average number of players joining gamesessions held per game.

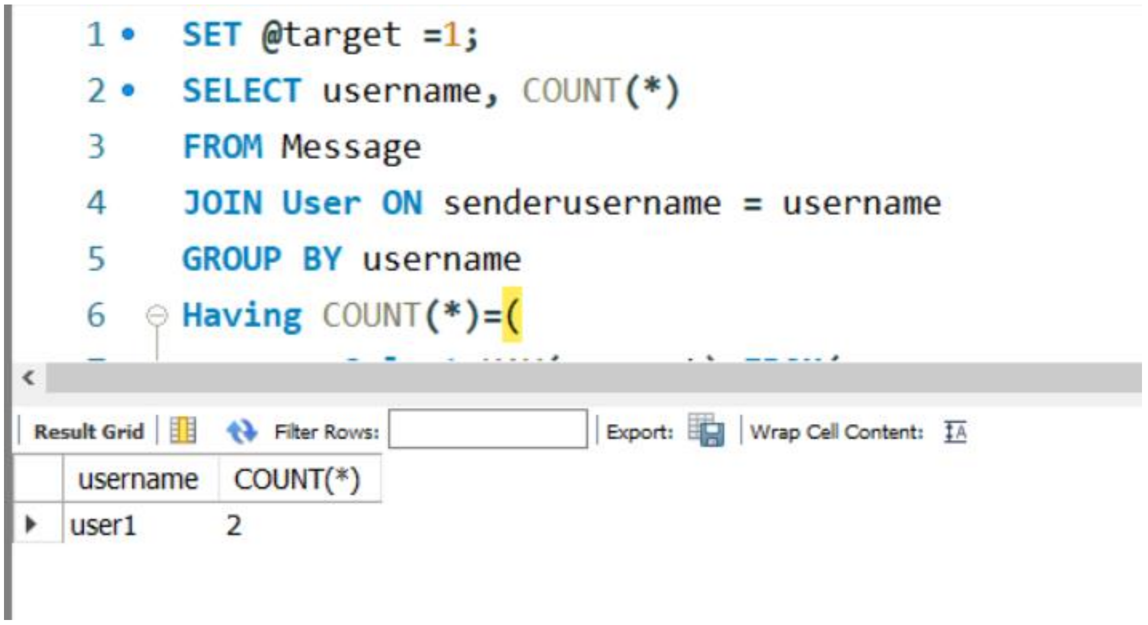
Criteria Fulfilled:

- Read (C)
- View (A)
- Query with Aggregate Function and Group by(D)
- Subqueries (E)

Query 6:

```
SET @target =1;
SELECT username, COUNT(*)
FROM Message
JOIN User ON senderusername = username
GROUP BY username
Having COUNT(*)=(
             Select MAX(u_count) FROM(
            SELECT Count(messageid) as u_count
            FROM Message
            JOIN User ON senderusername = username
            WHERE message.Threadid IN (
                SELECT threadid FROM Thread WHERE Forumid = @target
            )
            Group By Username
            ) as counts_of_messages
);
```

Screenshot:



```
SET @target =1;
SELECT username, COUNT(*)
FROM Message
JOIN User ON senderusername = username
GROUP BY username
Having COUNT(*)=(
```

username	COUNT(*)
user1	2

- Gets the user that is the most active in the forum with ID @target.

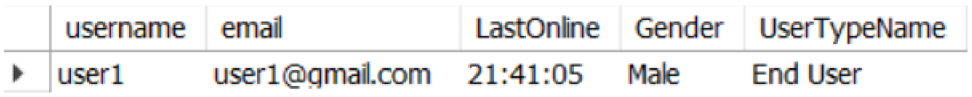
Criteria Fulfilled:

- View (A)
- Query with Aggregate Function and Group by(D)
- Subqueries (E)

Query 7:

```
SET @target = 1;
SELECT username, email,LastOnline,Gender, UserTypeName FROM User
JOIN UserType USING (UserTypeid)
WHERE User.username = (
SELECT username FROM Player WHERE Playerid = @target
);
```

Screenshot:



	username	email	LastOnline	Gender	UserTypeName
▶	user1	user1@gmail.com	21:41:05	Male	End User

- Gets the user that registered player with playerID same as the target.

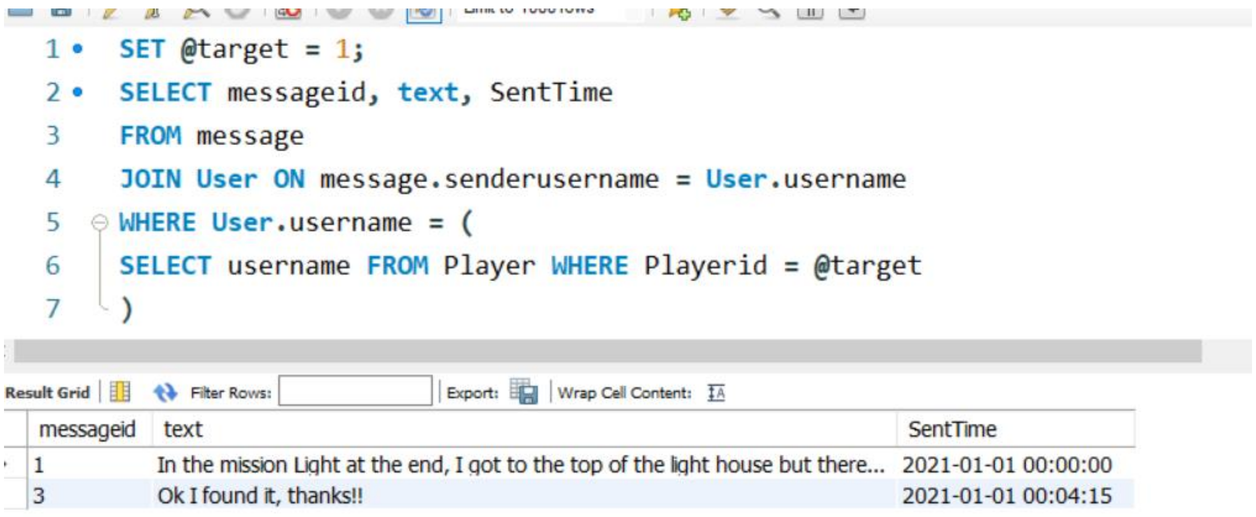
Criteria Fulfilled:

- Subqueries (E)

Query 8:

```
SET @target = 1;
SELECT messageid, text, SentTime
FROM message
JOIN User ON message.senderusername = User.username
WHERE User.username = (
SELECT usernameFROM Player WHERE Playerid = @target
);
Order By SentTime ASC;
```

Screenshot:



```
SET @target = 1;
SELECT messageid, text, SentTime
FROM message
JOIN User ON message.senderusername = User.username
WHERE User.username = (
SELECT username FROM Player WHERE Playerid = @target
)
```

messageid	text	SentTime
1	In the mission Light at the end, I got to the top of the light house but there...	2021-01-01 00:00:00
3	Ok I found it, thanks!!	2021-01-01 00:04:15

- From a playerID (target), reads all the messages the parent user sent.

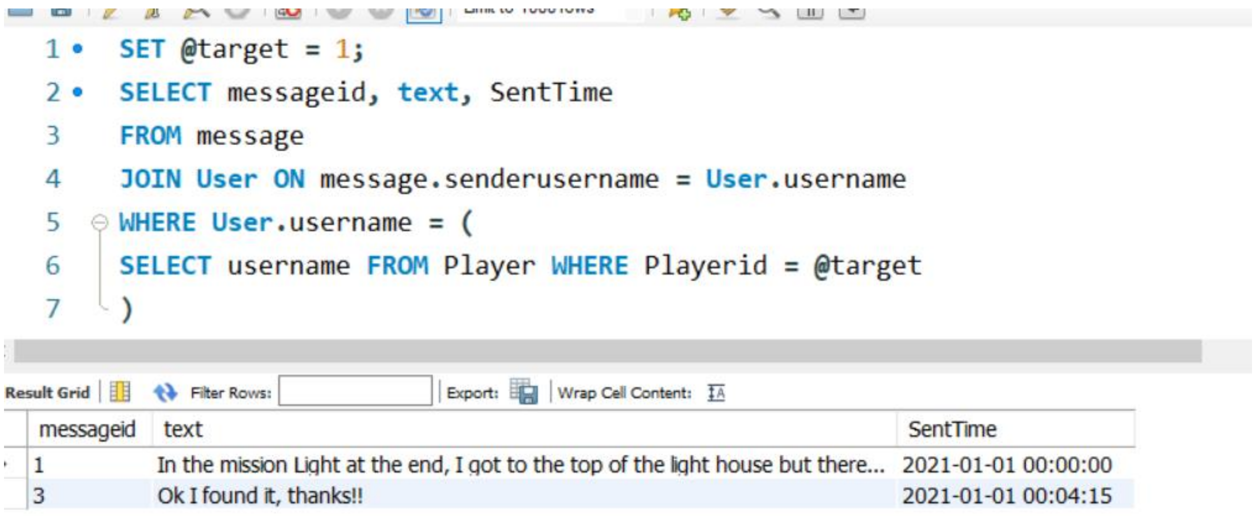
Criteria Fulfilled:

- Read (A)
- Subqueries (E)

Query 9:

```
set @target = 1;
SELECT PlayerID, InGameName FROM Player
WHERE Player.Playerid IN (
# Bug in mysql, cant operate on union unless it is packed as a derived table
SELECT * FROM (
      SELECT p1.PlayerID1 FROM Friends p1 WHERE PlayerID2 = @target
      UNION
      SELECT p2.PlayerID2 FROM Friends p2 WHERE PlayerID1 = @target)
       as Friends_of_PlayerID);
```

Screenshot:



```
SET @target = 1;
SELECT messageid, text, SentTime
FROM message
JOIN User ON message.senderusername = User.username
WHERE User.username = (
SELECT username FROM Player WHERE Playerid = @target
)
```

messageid	text	SentTime
1	In the mission Light at the end, I got to the top of the light house but there...	2021-01-01 00:00:00
3	Ok I found it, thanks!!	2021-01-01 00:04:15

- Gets all the friends of the player with PlayerID set as target.

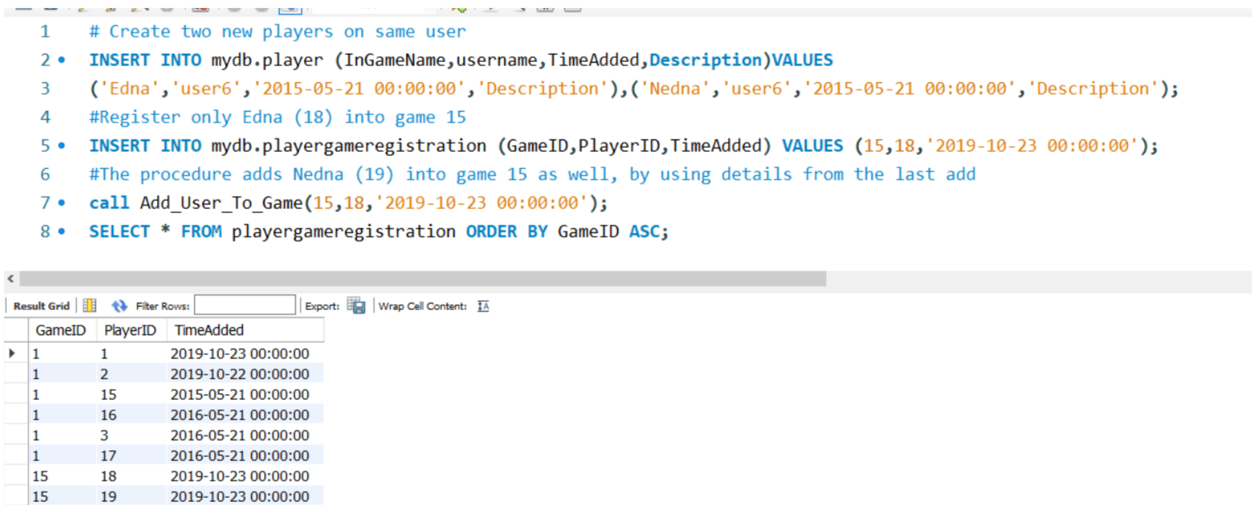
Criteria Fulfilled:

- Subquery in FROM (E)

Query 10:

```
DELIMITER $$


CREATE PROCEDURE Add_User_To_Game(
            IN newPlayerID INT,
   IN newGameID INT,
   IN newTimeAdded DateTime
)
BEGIN
   IF (NOT EXISTS (
SELECT PlayerID FROM PlayerGameRegistration
WHERE GameID = newGameID  AND
PlayerID  = ANY
            (SELECT PlayerID FROM Player WHERE  PlayerID != newPlayerID AND username = (
                        SELECT DISTINCT username FROM Player WHERE PlayerID = newPlayerID
                        )
            )
)
)THEN
INSERT IGNORE INTO PlayerGameRegistration (GameID, PlayerID, TimeAdded)
            SELECT newGameID, (SELECT PlayerID FROM Player WHERE PlayerID != newPlayerID AND
username IN (
       SELECT username FROMPlayer WHERE PlayerID = newPlayerID
     )), newTimeAdded;
            END IF;
END$$


DELIMITER ;
```

Screenshot:



```
# Create two new players on same user
INSERT INTO mydb.player (InGameName,username,TimeAdded,Description)VALUES
('Edna','user6','2015-05-21 00:00:00','Description'),('Nedna','user6','2015-05-21 00:00:00','Description');
#Register only Edna (18) into game 15
INSERT INTO mydb.playergameregistration (GameID,PlayerID,TimeAdded) VALUES (15,18,'2019-10-23 00:00:00');
#The procedure adds Nedna (19) into game 15 as well, by using details from the last add
call Add_User_To_Game(15,18,'2019-10-23 00:00:00');
SELECT * FROM playergameregistration ORDER BY GameID ASC;
```

GameID	PlayerID	TimeAdded
1	1	2019-10-23 00:00:00
1	2	2019-10-22 00:00:00
1	15	2015-05-21 00:00:00
1	16	2016-05-21 00:00:00
1	3	2016-05-21 00:00:00
1	17	2016-05-21 00:00:00
15	18	2019-10-23 00:00:00
15	19	2019-10-23 00:00:00

- A procedure that registers all players with the same username into a game, if one of them is added.
- It is meant as a workaround to the restriction of mySQL that stops triggers on PlayerGameRegistration from editing itself.

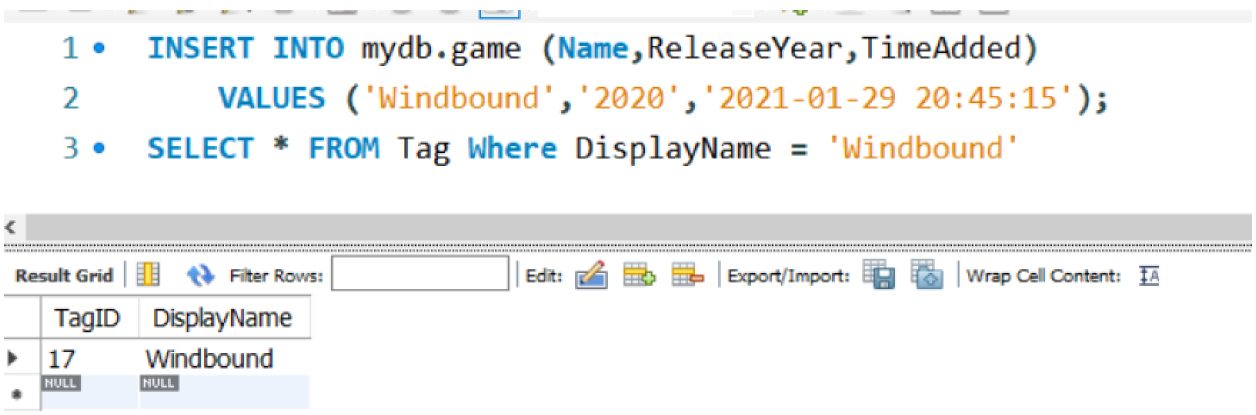
Criteria Fulfilled:

- Subquery in Select (E)

Query 11:

```
DELIMITER %%
CREATE TRIGGER ADD_GAME_TAG_DELETE_GAME_REQUEST
AFTER INSERT ON Game
FOR EACH ROW
BEGIN
IF(NOT EXISTS (SELECT * FROM Tag WHERE DisplayName = new.Name)) THEN
        INSERT INTO Tag VALUES (new.Name);
  END IF ;

IF(EXISTS ( SELECT * FROM Thread WHERE Thread.Title = new.Name AND ThreadID IN
        (SELECT ThreadID FROM ThreadisTagged WHERE TagID = (
                SELECT TagID FROM Tag WHERE DisplayName = 'Request')))) THEN
        DELETE FROM Thread Where Thread.Title = new.Name;
        END IF;
END %%
DELIMITER ;
```

Screenshot:



```
INSERT INTO mydb.game (Name,ReleaseYear,TimeAdded)
    VALUES ('Windbound','2020','2021-01-29 20:45:15');
SELECT * FROM Tag Where DisplayName = 'Windbound'
```

TagID	DisplayName
17	Windbound
NULL	NULL

- Upon Addition of a game into db if the game, adds a tag with the same name as the game.
- Further, removes the thread that requests the addition of the game (if any).

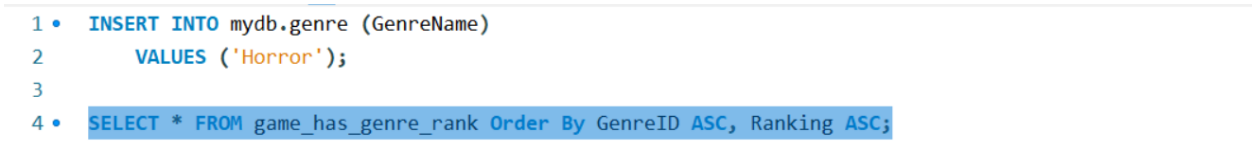
Criteria Fulfilled:

- Delete (C)
- Trigger(B)

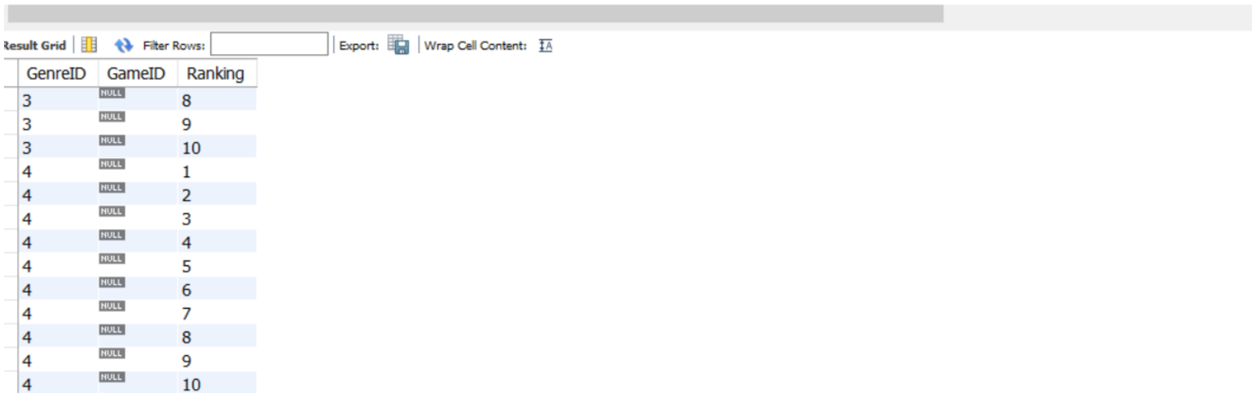
Query 12:

```
DELIMITER %%

CREATE TRIGGER ADD_GENRE_RANK

AFTER INSERT ON Genre

FOR EACH ROW

BEGIN


INSERT INTO Game_Has_Genre_Rank (GenreID, GameID, Ranking) VALUES

(new.GenreID, NULL, 1),

(new.GenreID, NULL, 2),

(new.GenreID, NULL, 3),

(new.GenreID, NULL, 4),

(new.GenreID, NULL, 5),

(new.GenreID, NULL, 6),

(new.GenreID, NULL, 7),

(new.GenreID, NULL, 8),

(new.GenreID, NULL, 9),

(new.GenreID, NULL, 10);



END%%

DELIMITER ;
```

Screenshot:



```
INSERT INTO mydb.genre (GenreName)
    VALUES ('Horror');

SELECT * FROM game_has_genre_rank Order By GenreID ASC, Ranking ASC;
```



GenreID	GameID	Ranking
3	NULL	8
3	NULL	9
3	NULL	10
4	NULL	1
4	NULL	2
4	NULL	3
4	NULL	4
4	NULL	5
4	NULL	6
4	NULL	7
4	NULL	8
4	NULL	9
4	NULL	10

- Upon insertion of a genre, creates placeholder rankings for that genre

Criteria Fulfilled:

- Trigger (B)

Query 13:

CREATE VIEW ALL_USERTYPES AS

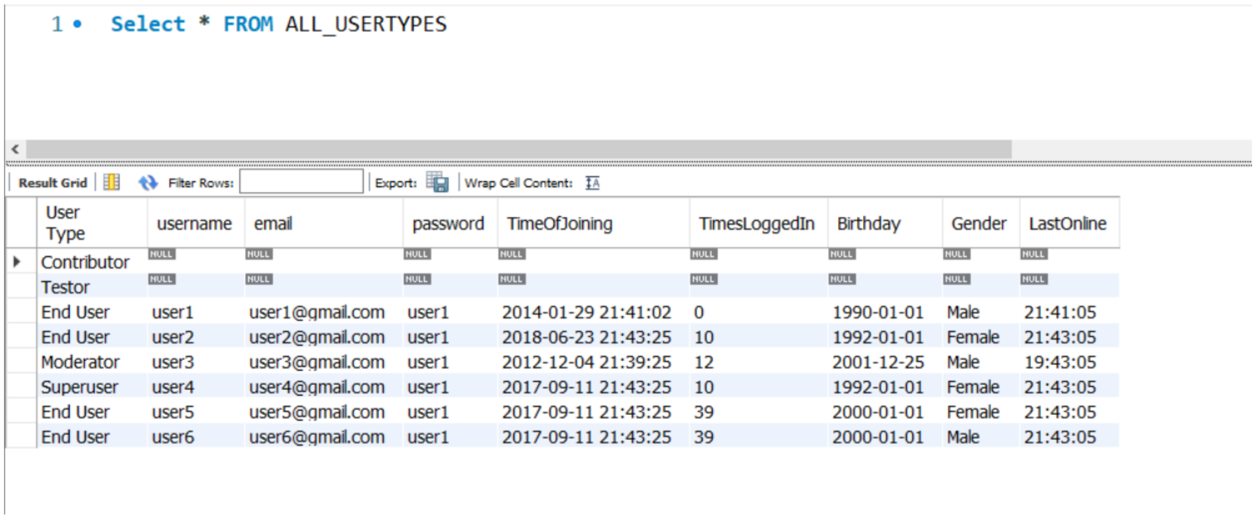
SELECT UserTypeName AS 'User Type', username,email,password,TimeOfJoining,TimesLoggedIn,Birthday,Gender, LastOnline

FROM user

RIGHT JOIN usertype USING (UserTypeID)

ORDER BY 'User Type' ASC, Username ASC;

Screenshot:



```
Select * FROM ALL_USERTYPES
```

User Type	username	email	password	TimeOfJoining	TimesLoggedIn	Birthday	Gender	LastOnline
Contributor	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
Testor	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
End User	user1	user1@gmail.com	user1	2014-01-29 21:41:02	0	1990-01-01	Male	21:41:05
End User	user2	user2@gmail.com	user1	2018-06-23 21:43:25	10	1992-01-01	Female	21:43:05
Moderator	user3	user3@gmail.com	user1	2012-12-04 21:39:25	12	2001-12-25	Male	19:43:05
Superuser	user4	user4@gmail.com	user1	2017-09-11 21:43:25	10	1992-01-01	Female	21:43:05
End User	user5	user5@gmail.com	user1	2017-09-11 21:43:25	39	2000-01-01	Female	21:43:05
End User	user6	user6@gmail.com	user1	2017-09-11 21:43:25	39	2000-01-01	Male	21:43:05

- Gets all usertypes and the users that are associated with them, using right join.

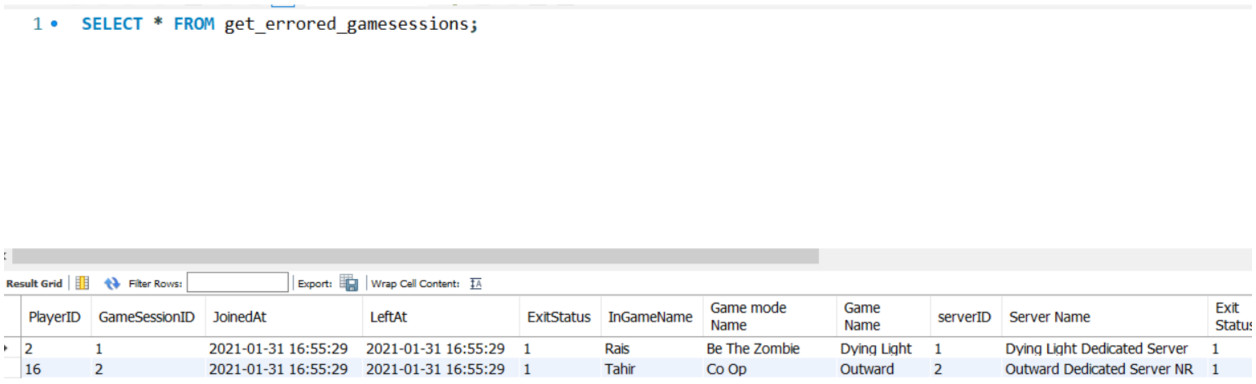
Criteria Fulfilled:

- View using right join (A)

Query 14:

```
CREATE VIEW ERRORED_GAMESESSIONS AS

SELECT playerjoinedgamesession.*, Player.InGameName,

 GameMode.Name AS 'Game mode Name', Game.Name as 'Game Name',

 Server.serverID,server.Name as 'Server Name',

ExitStatus as 'Exit Status' from playerjoinedgamesession

JOIN Player USING (PlayerID)

JOIN gamesession ON playerjoinedgamesession.GameSessionID = gamesession.GameSessionID

JOIN Server ON gamesession.ServerID = Server.ServerID

JOIN GameMode ON gamesession.GameModeID = gamemode.GameModeID

JOIN Game ON gamemode.GameID = Game.GameID

WHERE ExitStatus <> 0
```

Screenshot:



```
SELECT * FROM get_errored_gamesessions;
```

PlayerID	GameSessionID	JoinedAt	LeftAt	ExitStatus	InGameName	Game mode Name	Game Name	serverID	Server Name	Exit Status
2	1	2021-01-31 16:55:29	2021-01-31 16:55:29	1	Rais	Be The Zombie	Dying Light	1	Dying Light Dedicated Server	1
16	2	2021-01-31 16:55:29	2021-01-31 16:55:29	1	Tahir	Co Op	Outward	2	Outward Dedicated Server NR	1

- Gets the details of all failed attempts of players joining game sessions

Criteria Fulfilled:

- View (A)

Query 15:

CREATE VIEW GET_INACTIVE_USERS AS

SELECT username,COUNT(playerjoinedgamesession.playerID) FROM user

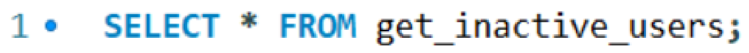
LEFT JOIN Player USING (username)

LEFT JOIN playerjoinedgamesession USING (playerID)

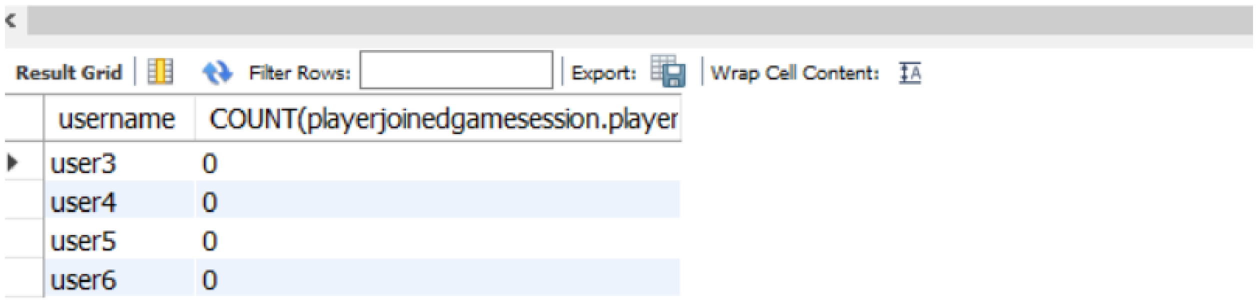
GROUP BY username

HAVING COUNT(playerjoinedgamesession.playerID) = 0;

Screenshot:



```
SELECT * FROM get_inactive_users;
```



username	COUNT(playerjoinedgamesession.player
user3	0
user4	0
user5	0
user6	0

- Gets all the users who haven't yet joined a gamesession.

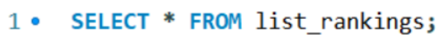
Criteria Fulfilled:

- View (A)
- Aggregate Function (C)

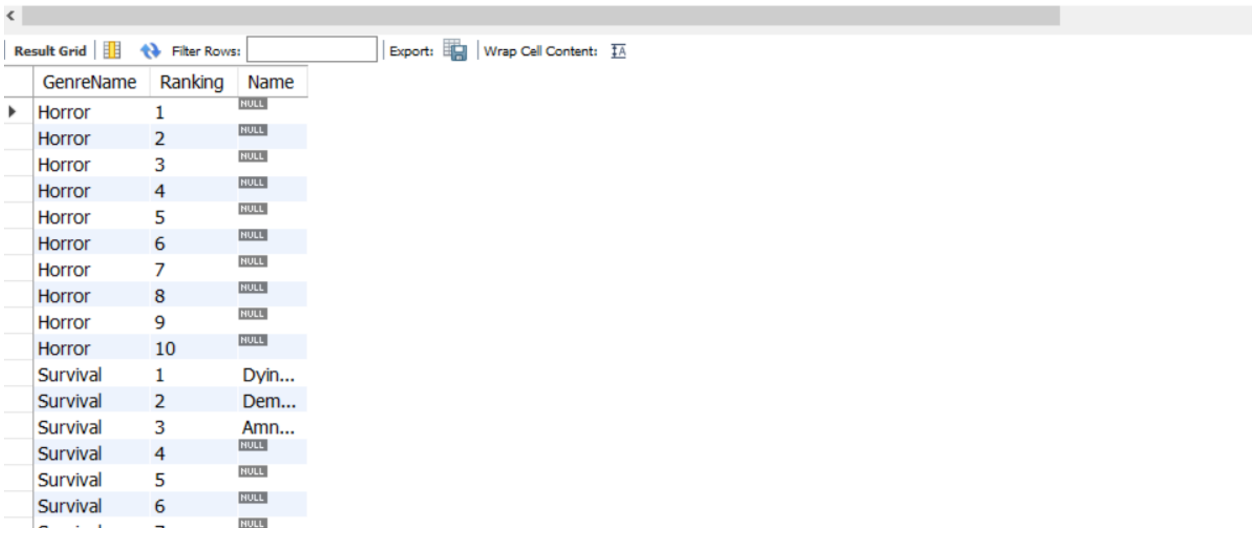
Query 16:

```
CREATE VIEW LIST_RANKINGS AS

SELECT GenreName, Ranking, Game.Name

FROM game_has_genre_rank

INNER JOIN Genre USING (GenreID)

LEFT JOIN Game USING (GameID)

ORDER BY GenreName ASC, Ranking ASC;
```

Screenshot:



```
SELECT * FROM list_rankings;
```



GenreName	Ranking	Name
Horror	1	NULL
Horror	2	NULL
Horror	3	NULL
Horror	4	NULL
Horror	5	NULL
Horror	6	NULL
Horror	7	NULL
Horror	8	NULL
Horror	9	NULL
Horror	10	NULL
Survival	1	Dyin...
Survival	2	Dem...
Survival	3	Amn...
Survival	4	NULL
Survival	5	NULL
Survival	6	NULL

- Neatly lists rankings with relevant information.

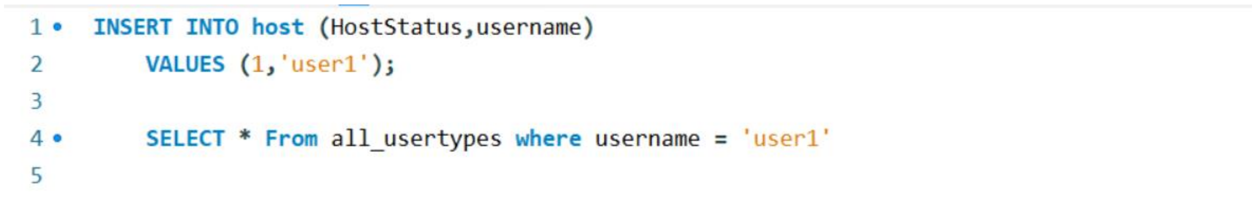
Criteria Fulfilled:

- View (A)

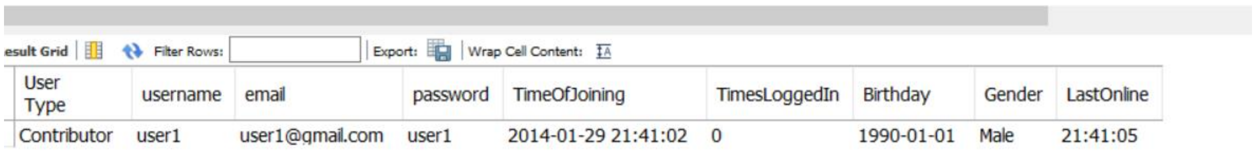
Query 17:

```
DELIMITER %%
CREATE TRIGGER PROMOTE_END_USER
AFTER INSERT ON host FOR EACH ROW
BEGIN
IF (EXISTS (SELECT username from User WHERE User.username = new.username AND UserTypeID = (
                                        Select UserTypeID FROM UserType WHERE UserTypeName = 'End
User'))) THEN
          UPDATE User SET UserTypeID = (
                    Select UserTypeID FROM UserType WHERE UserTypeName = 'Contributor'
          ) WHERE username = new.username;
  END IF;


END%%

DELIMITER ;
```

Screenshot:



```
INSERT INTO host (HostStatus,username)
    VALUES (1,'user1');

    SELECT * From all_usertypes where username = 'user1'
```



User Type	username	email	password	TimeOfJoining	TimesLoggedIn	Birthday	Gender	LastOnline
Contributor	user1	user1@gmail.com	user1	2014-01-29 21:41:02	0	1990-01-01	Male	21:41:05

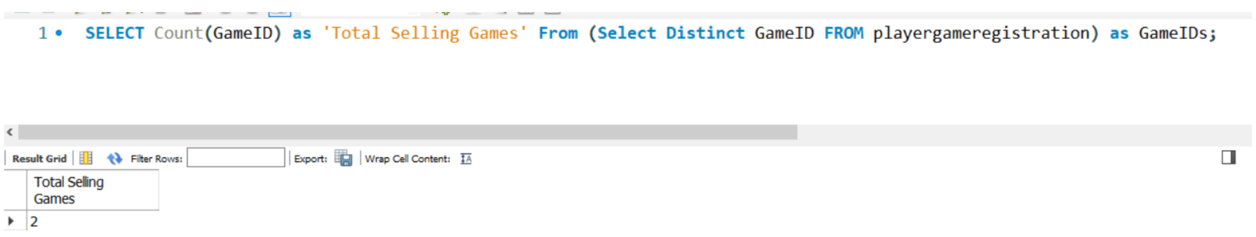
- When an end user creates a host, promotes them to Contributor.

Criteria Fulfilled:

- Update (A)

Query 18:

SELECT Count(GameID) as 'Total Selling Games' From (Select Distinct GameID FROM playergameregistration) as GameIDs;Screenshot:



```
SELECT Count(GameID) as 'Total Selling Games' From (Select Distinct GameID FROM playergameregistration) as GameIDs;
```

Total Selling Games
2

- Count all games that have sold at least a copy.

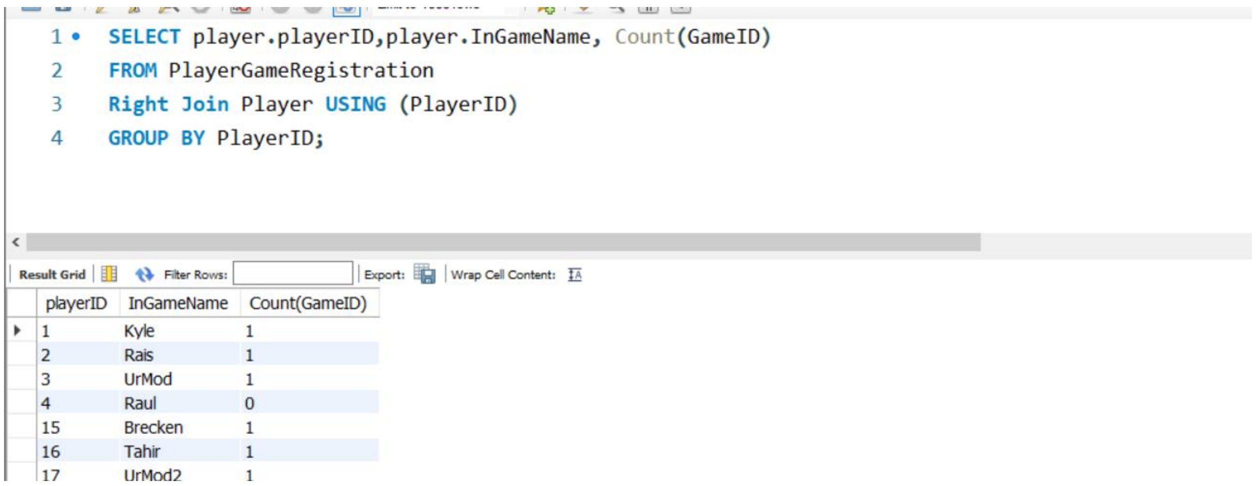
Criteria Fulfilled:

- Aggregate Function(D)
- Subquery in FROM (E)

Query 19:

```
SELECT player.playerID,player.InGameName, Count(GameID)
FROM PlayerGameRegistration
Right Join Player USING (PlayerID)
GROUP BY PlayerID;
```

Screenshot:



```
SELECT player.playerID,player.InGameName, Count(GameID)
FROM PlayerGameRegistration
Right Join Player USING (PlayerID)
GROUP BY PlayerID;
```

playerID	InGameName	Count(GameID)
1	Kyle	1
2	Rais	1
3	UrMod	1
4	Raul	0
15	Brecken	1
16	Tahir	1
17	UrMod2	1

- Get the amount of games each player is registered in

Criteria Fulfilled:

- Aggregate Function(D)

Query 20:

SELECT GameID, Name FROM(

SELECT DISTINCT GameID, Game.Name, Count(username) as UserCount FROM PlayerGameRegistration

JOIN Player USING (playerID)

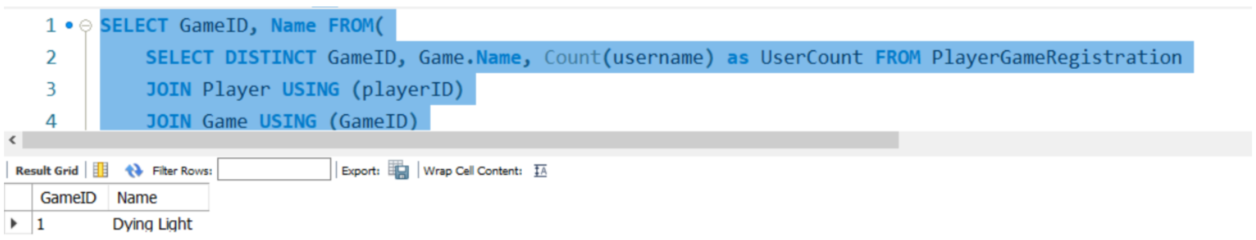
JOIN Game USING (GameID)

GROUP BY GameID

) as GamesSoldCopies

HAVING UserCount = MAX(UserCount);

Screenshot:



```
SELECT GameID, Name FROM(
    SELECT DISTINCT GameID, Game.Name, Count(username) as UserCount FROM PlayerGameRegistration
    JOIN Player USING (playerID)
    JOIN Game USING (GameID)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

GameID	Name
1	Dying Light

- Get the game that has the most users registered to it.

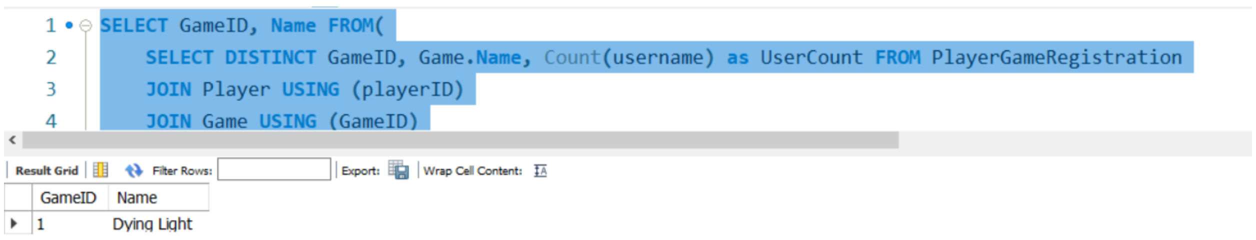
Criteria Fulfilled:

- Aggregate Function (D)
- Subquery in FROM (E)

Query 21:

SELECT CriticID,
 InGameName,
 SUM(totalvotes) FROM
criticWritesReview
JOIN (SELECT ReviewID, SUM(votes) as totalvotes FROM Review GROUP BY ReviewID) as SumOfReviews USING (ReviewID)
JOIN Critic USING (CriticID)
JOIN Player USING (PlayerID)
GROUP BY criticWritesReview.CriticID

Screenshot:



```
SELECT GameID, Name FROM(
    SELECT DISTINCT GameID, Game.Name, Count(username) as UserCount FROM PlayerGameRegistration
    JOIN Player USING (playerID)
    JOIN Game USING (GameID)
```

GameID	Name
1	Dying Light

- Get the critic names and the sum of the votes their reviews have received

Criteria Fulfilled:

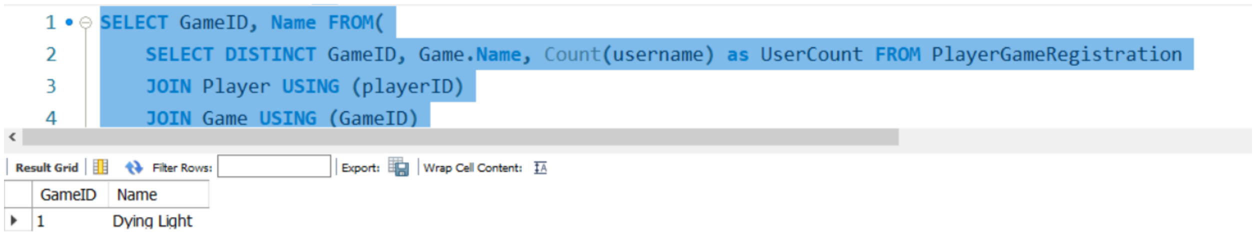
- Aggregate Function (D)
- Subquery in FROM (E)

Query 22:

SELECT PlayerID,
 InGameName,
 (
 SELECT AVG(Game_Count) FROM (
 SELECT PlayerID,Count(GameID) as Game_Count FROM playergameregistration
 GROUP BY PlayerID
) as GameCounts
) as 'Average Games Per Player'

  From Player

Screenshot:



```
SELECT GameID, Name FROM(
    SELECT DISTINCT GameID, Game.Name, Count(username) as UserCount FROM PlayerGameRegistration
    JOIN Player USING (playerID)
    JOIN Game USING (GameID)
```

GameID	Name
1	Dying Light

- Get the critic names and the sum of the votes their reviews have received

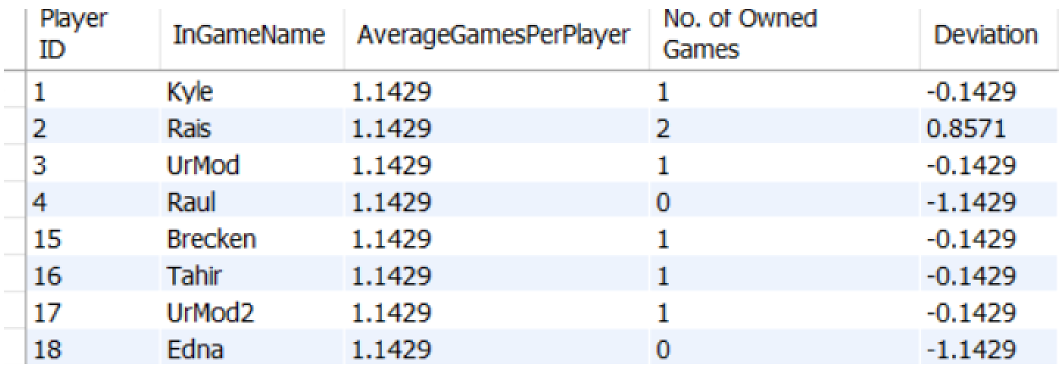
Criteria Fulfilled:

- Aggregate Function (D)
- Subquery in FROM   (E)

Query 23:

```
SELECT PlayerID as `Player ID`,
 InGameName,
 (
 SELECT  AVG(Game_Count) FROM (
 SELECT PlayerID,Count(GameID) as Game_Count FROM playergameregistration
 GROUP BY PlayerID
  ) as GameCounts
  ) as AveragePlayerGamesPerPlayer,

(SELECT Count(GameID) as Game_Count FROM playergameregistration WHERE PlayerID = `Player ID`)
as 'No. of Owned Games',
  #A faster solution to this would be a self join or using this as a derived table, but this is for
#demonstration
  ((
  (SELECT  AVG(Game_Count) FROM (
 SELECT PlayerID,Count(GameID) as Game_Count FROM playergameregistration
 GROUP BY PlayerID
  ) as GameCounts
  )-
  (SELECT Count(GameID) as Game_Count FROM playergameregistration WHERE PlayerID = `Player ID`)
   )*-1) as Deviation


  From Player
```

Screenshot:



Player ID	InGameName	AverageGamesPerPlayer	No. of Owned Games	Deviation
1	Kyle	1.1429	1	-0.1429
2	Rais	1.1429	2	0.8571
3	UrMod	1.1429	1	-0.1429
4	Raul	1.1429	0	-1.1429
15	Brecken	1.1429	1	-0.1429
16	Tahir	1.1429	1	-0.1429
17	UrMod2	1.1429	1	-0.1429
18	Edna	1.1429	0	-1.1429

- Compare the games owned by a player with games owned by players on average

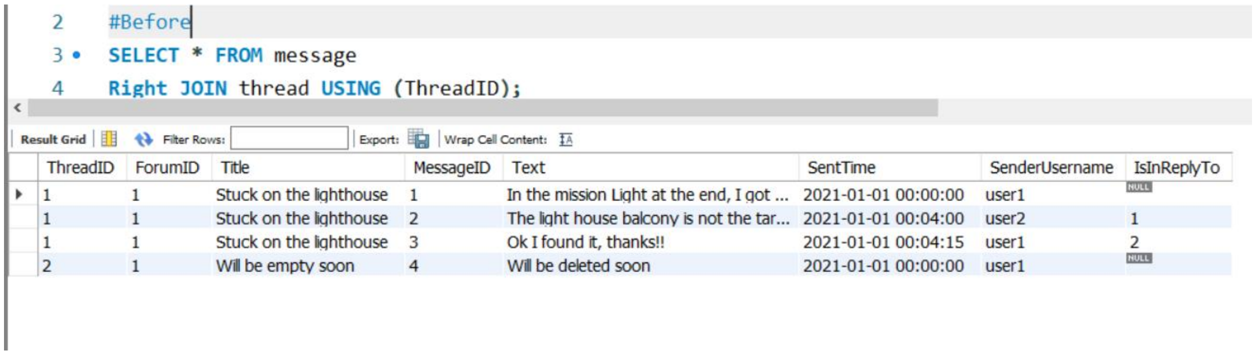
Criteria Fulfilled:

- Aggregate Function (D)
- Subquery in SELECT (E)

Query 24:

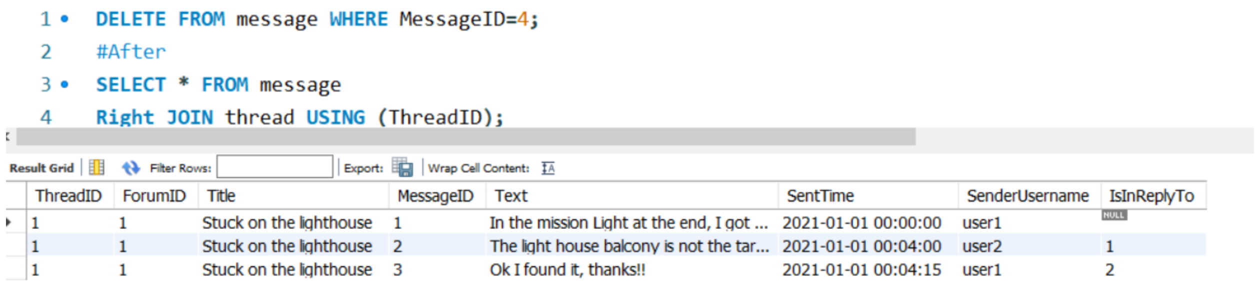
```
DELIMITER %%
CREATE TRIGGER DELETE_EMPTY_THREAD
AFTER DELETE ON message FOR EACH ROW
BEGIN
IF (EXISTS (
SELECT remaining FROM (
SELECT COUNT(messageID) as remaining FROM message WHERE ThreadID = old.ThreadID
) as Wrap
Where remaining =0
)
) THEN
        DELETE FROM Thread Where ThreadID = old.ThreadID;
   END IF;


END%%
DELIMITER ;
```

Screenshot:



```
#Before
SELECT * FROM message
Right JOIN thread USING (ThreadID);
```

ThreadID	ForumID	Title	MessageID	Text	SentTime	SenderUsername	IsInReplyTo
1	1	Stuck on the lighthouse	1	In the mission Light at the end, I got ...	2021-01-01 00:00:00	user1	NULL
1	1	Stuck on the lighthouse	2	The light house balcony is not the tar...	2021-01-01 00:04:00	user2	1
1	1	Stuck on the lighthouse	3	Ok I found it, thanks!!	2021-01-01 00:04:15	user1	2
2	1	Will be empty soon	4	Will be deleted soon	2021-01-01 00:00:00	user1	NULL



```
DELETE FROM message WHERE MessageID=4;
#After
SELECT * FROM message
Right JOIN thread USING (ThreadID);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ThreadID	ForumID	Title	MessageID	Text	SentTime	SenderUsername	IsInReplyTo
1	1	Stuck on the lighthouse	1	In the mission Light at the end, I got ...	2021-01-01 00:00:00	user1	NULL
1	1	Stuck on the lighthouse	2	The light house balcony is not the tar...	2021-01-01 00:04:00	user2	1
1	1	Stuck on the lighthouse	3	Ok I found it, thanks!!	2021-01-01 00:04:15	user1	2

- Upon deletion of the last message in the thread, deletes the thread.

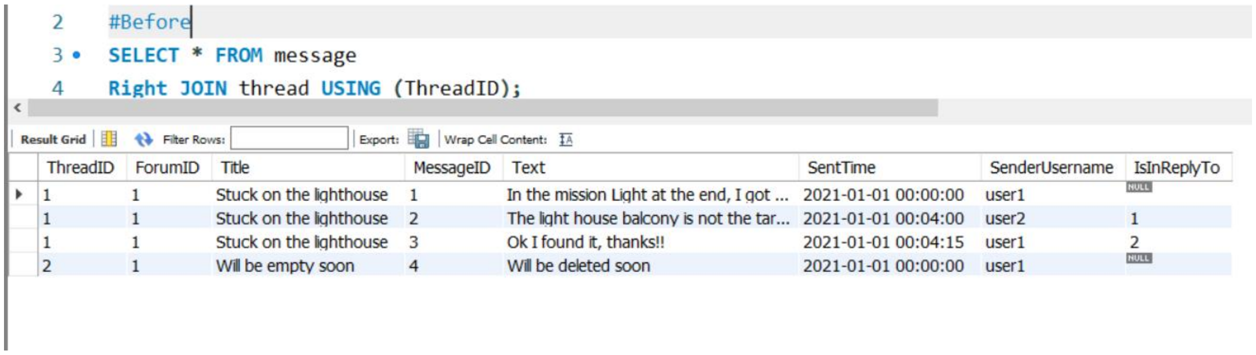
Criteria Fulfilled:

- Aggregate Function (D)
- Delete (C)

Query 25:

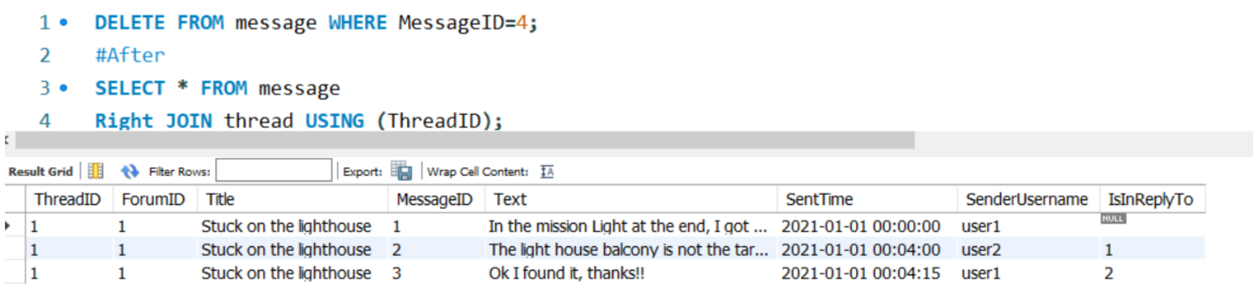
```
DELIMITER %%
CREATE TRIGGER DELETE_EMPTY_THREAD
AFTER DELETE ON message FOR EACH ROW
BEGIN
IF (EXISTS (
SELECT remaining FROM (
SELECT COUNT(messageID) as remaining FROM message WHERE ThreadID = old.ThreadID
) as Wrap
Where remaining =0
)
) THEN
        DELETE FROM Thread Where ThreadID = old.ThreadID;
  END IF;


END%%
DELIMITER ;
```

Screenshot:



```
#Before
SELECT * FROM message
Right JOIN thread USING (ThreadID);
```

ThreadID	ForumID	Title	MessageID	Text	SentTime	SenderUsername	IsInReplyTo
1	1	Stuck on the lighthouse	1	In the mission Light at the end, I got ...	2021-01-01 00:00:00	user1	NULL
1	1	Stuck on the lighthouse	2	The light house balcony is not the tar...	2021-01-01 00:04:00	user2	1
1	1	Stuck on the lighthouse	3	Ok I found it, thanks!!	2021-01-01 00:04:15	user1	2
2	1	Will be empty soon	4	Will be deleted soon	2021-01-01 00:00:00	user1	NULL



```
DELETE FROM message WHERE MessageID=4;
#After
SELECT * FROM message
Right JOIN thread USING (ThreadID);
```

ThreadID	ForumID	Title	MessageID	Text	SentTime	SenderUsername	IsInReplyTo
1	1	Stuck on the lighthouse	1	In the mission Light at the end, I got ...	2021-01-01 00:00:00	user1	NULL
1	1	Stuck on the lighthouse	2	The light house balcony is not the tar...	2021-01-01 00:04:00	user2	1
1	1	Stuck on the lighthouse	3	Ok I found it, thanks!!	2021-01-01 00:04:15	user1	2

- Upon deletion of the last message in the thread, deletes the thread.

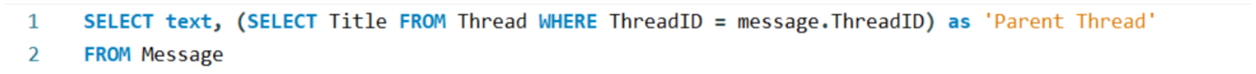
Criteria Fulfilled:

- Aggregate Function (D)
- Delete (C)

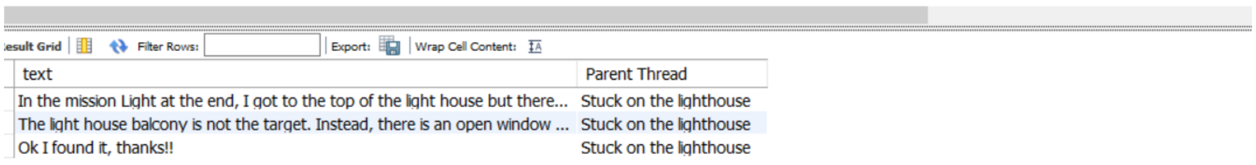
Query 26:

SELECT text, (SELECT Title FROM Thread WHERE ThreadID = message.ThreadID) as 'Parent Thread'
FROM Message

Screenshot:



```
SELECT text, (SELECT Title FROM Thread WHERE ThreadID = message.ThreadID) as 'Parent Thread'
FROM Message
```



text	Parent Thread
In the mission Light at the end, I got to the top of the light house but there...	Stuck on the lighthouse
The light house balcony is not the target. Instead, there is an open window ...	Stuck on the lighthouse
Ok I found it, thanks!!	Stuck on the lighthouse

- Shows all messages along with the title of the thread they belong to

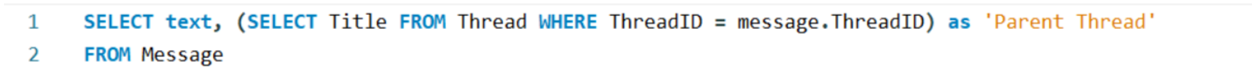
Criteria Fulfilled:

- Subquery in SELECT (E)

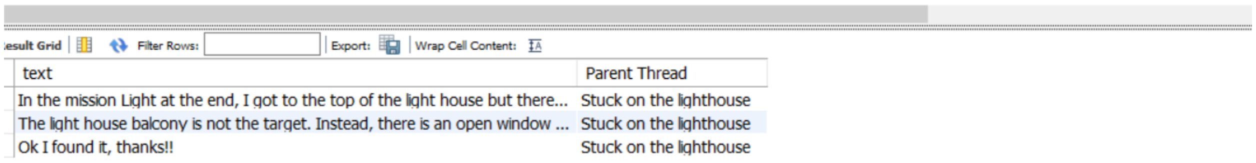
Query 27:

SELECT text, (SELECT Title FROM Thread WHERE ThreadID = message.ThreadID) as 'Parent Thread'
FROM Message

Screenshot:



```sql
SELECT text, (SELECT Title FROM Thread WHERE ThreadID = message.ThreadID) as 'Parent Thread'
FROM Message
```



text	Parent Thread
In the mission Light at the end, I got to the top of the light house but there...	Stuck on the lighthouse
The light house balcony is not the target. Instead, there is an open window ...	Stuck on the lighthouse
Ok I found it, thanks!!	Stuck on the lighthouse

- Shows all messages along with the title of the thread they belong to

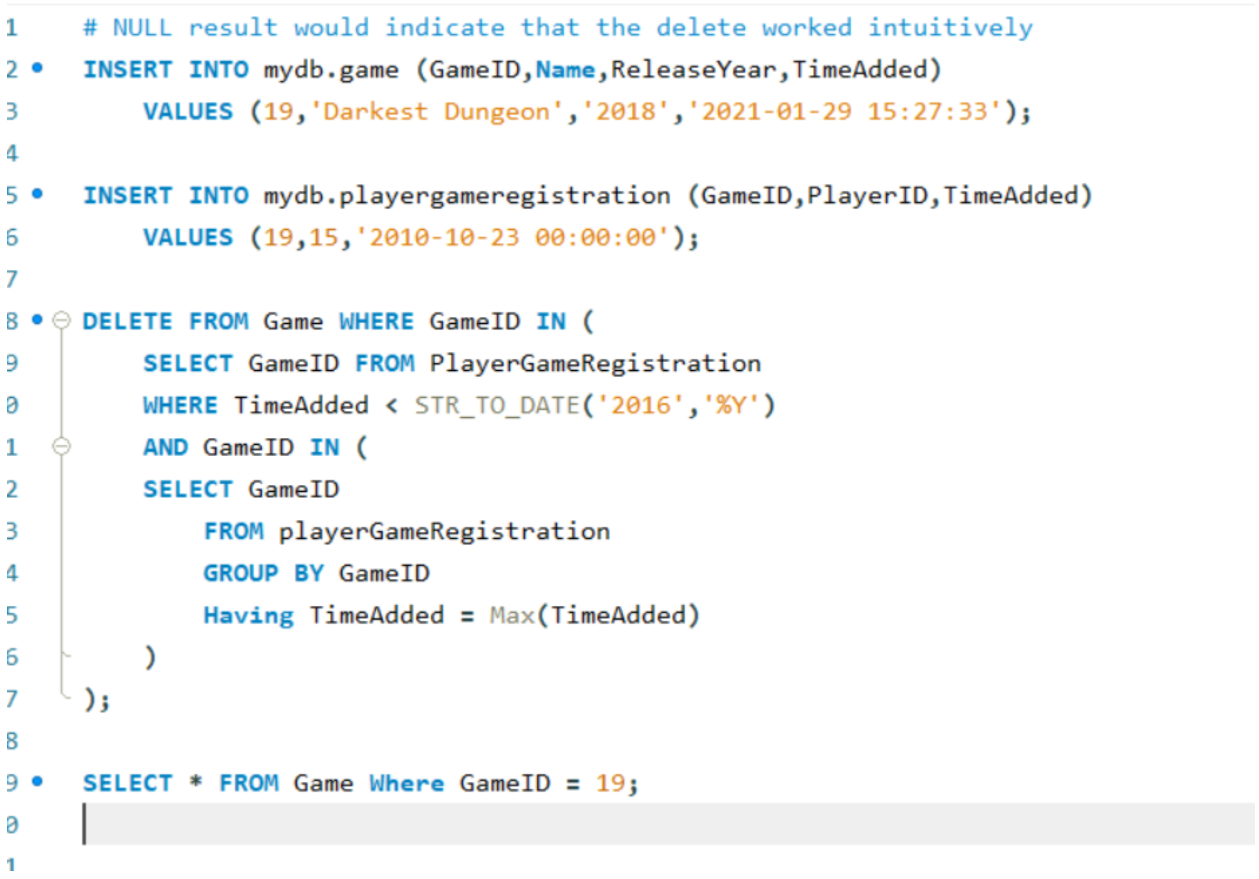
Criteria Fulfilled:

- Subquery in SELECT (E)

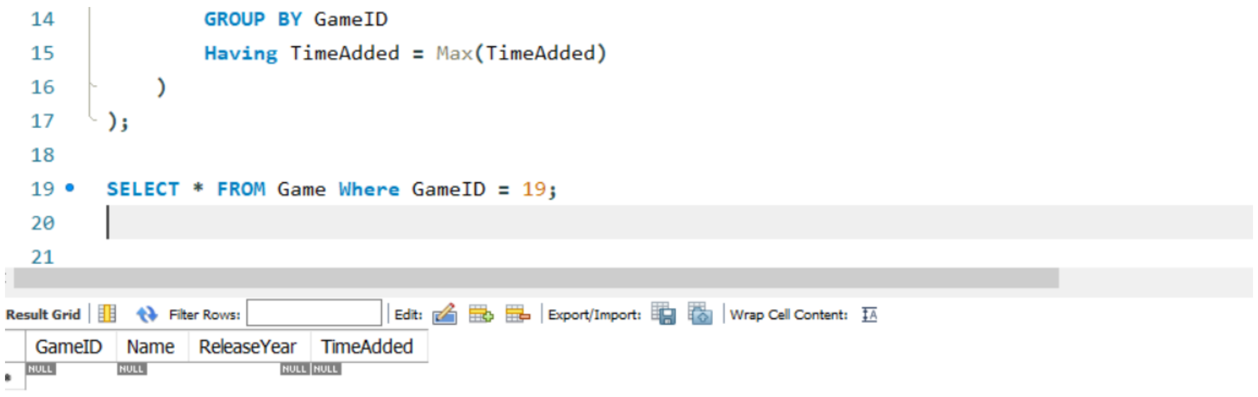
Query 28:

```
DELETE FROM Game WHERE GameID IN (
            SELECT GameID FROM PlayerGameRegistration
    WHERE TimeAdded < STR_TO_DATE('2016','%Y')
    AND GameID IN (
    SELECT GameID
                        FROM playerGameRegistration

                        GROUP BY GameID
                        Having TimeAdded = Max(TimeAdded)
    )
);
```

Screenshot:



```
# NULL result would indicate that the delete worked intuitively
INSERT INTO mydb.game (GameID,Name,ReleaseYear,TimeAdded)
    VALUES (19,'Darkest Dungeon','2018','2021-01-29 15:27:33');

INSERT INTO mydb.playergameregistration (GameID,PlayerID,TimeAdded)
    VALUES (19,15,'2010-10-23 00:00:00');

DELETE FROM Game WHERE GameID IN (
    SELECT GameID FROM PlayerGameRegistration
    WHERE TimeAdded < STR_TO_DATE('2016','%Y')
    AND GameID IN (
    SELECT GameID
        FROM playerGameRegistration
        GROUP BY GameID
        Having TimeAdded = Max(TimeAdded)
    )
);

SELECT * FROM Game Where GameID = 19;
```



```
            GROUP BY GameID
            Having TimeAdded = Max(TimeAdded)
        )
);

SELECT * FROM Game Where GameID = 19;

```

```
GameID  Name  ReleaseYear  TimeAdded
NULL    NULL  NULL         NULL
```

- Deletes all games for which the last registration was made before 2016

Criteria Fulfilled:

- DELETE (C)