MPCS 53001 Database

Final Project - Step 3

GameIQ (Games Information Query) Database

Team 6

Group Members:
Yiyi Wu
Jiayue Chen
Fengshi Teng
Donglizhen Shi

The **GameIQ** project aims to build a comprehensive and interactive database system that integrates game-related information from various platforms and sources. Our goal is to provide a unified interface for querying, analyzing, and understanding game distribution, player behaviors, and game metadata across multiple gaming environments.

Database Requirements/Specifications:

Dataset:
Steam Game datasets:
https://www.kaggle.com/datasets/fronkongames/steam-games-dataset?select=games.csv
https://www.kaggle.com/datasets/nikdavis/steam-store-games
Steam Review Dataset: https://zenodo.org/records/1000885
https://www.kaggle.com/datasets/andrewmvd/steam-reviews
Video Games Dataset API: https://rawg.io/apidocs
PS5 Game Dataset:
https://www.kaggle.com/datasets/kanchana1990/ps5-game-data-2000-titles-explored
Xbox Game Dataset: https://www.kaggle.com/datasets/shivamb/all-xbox-one-games

Entity Set:
Game:

- GameID
- Name
- LanguageSupport
- Release_Date
- Country
- Tags
- Price
- SupportedPlatform
- Total_Revenue
- Units_sold
- Description
- Required_age
- PublisherID
- DeveloperID
- Genre

Achievements (weak entity):

- *Achievements_Name*
- Achievements_Description

Publisher:

- Publisher_ID

Publisher_Name
Founded_Year
Contact_Email
Country
Address
Website
Description

Developer:

Developer_ID
Developer_Name
Founded_Year
Contact_Email
Country
Address
Website
Description

Platform:

Platform_Id
Name
Manufacturer
TotalGameOwned
Website

Player:

PlayerID
Player_UserName
Player_Email
Player_Region
Player_JoinDate
Player_Level
Player_PlayTime
Player_GamesOwned

Relationship Sets:

(1) "Has" (Game - Achievement):
Each game can have multiple achievements that players can earn through gameplay. If a game is removed from the database, all associated achievements must also be deleted.

(2) "Unlock" (Player - Achievement):
Each player can earn multiple achievements by playing different games. The gain relationship must record the timestamp when a player earns each achievement.

(3) "Play" (Player - Game - Platform):
Each player can play multiple games, and each game can be played by the same player on different platforms. Each play relationship must record total playing time and the last played date.

(4) "Purchase" (Player - Game - Platform):
Each player can purchase different games, and each transaction relationship must record the price, the transaction time, and the transaction ID.

(5) "Review" (Player - Game - Platform):
Each player can comment on different games on multiple platforms. The review relationship must record the review time and comment context.

(6) "Support" (Game - Platform):
Each game may be used on multiple platforms, and each platform can hold different games. The issuing relationship must record the game issued time on the platform and its esrb_rating on the platform.

(7) "Publish" (Game - Publisher):
Each game is produced by one producer, and each producer may produce multiple games. The production relationship must record the production time.

(8) "Develope" (Game - Developer):
Each game is produced by one producer, and each producer may produce multiple games. The production relationship must record the production time.

(9) "Use" (Player - Platform):
Each player can register on multiple platforms and each platform can have different players. The use relationship must record the registration time and the total time the player has spent on the platform.

Database Queries:

Queries on Game:

1. List top n games by rating on certain platform (Game, Platform)
2. List all games of certain genre ordered by given attributes. List all games produced by a specific producer along with their release date and category. (Game, Producer)
3. List all games produced by a specific publisher / developer along with their release date and genre, ordered by release date. (Player,developer/publisher)
4. List all games released after certain year, along with their support system and the producers who made them. (Game, Producer)
5. List average ratings / total units sold of each genre, ordered by them in descending order.

Queries on Platform:

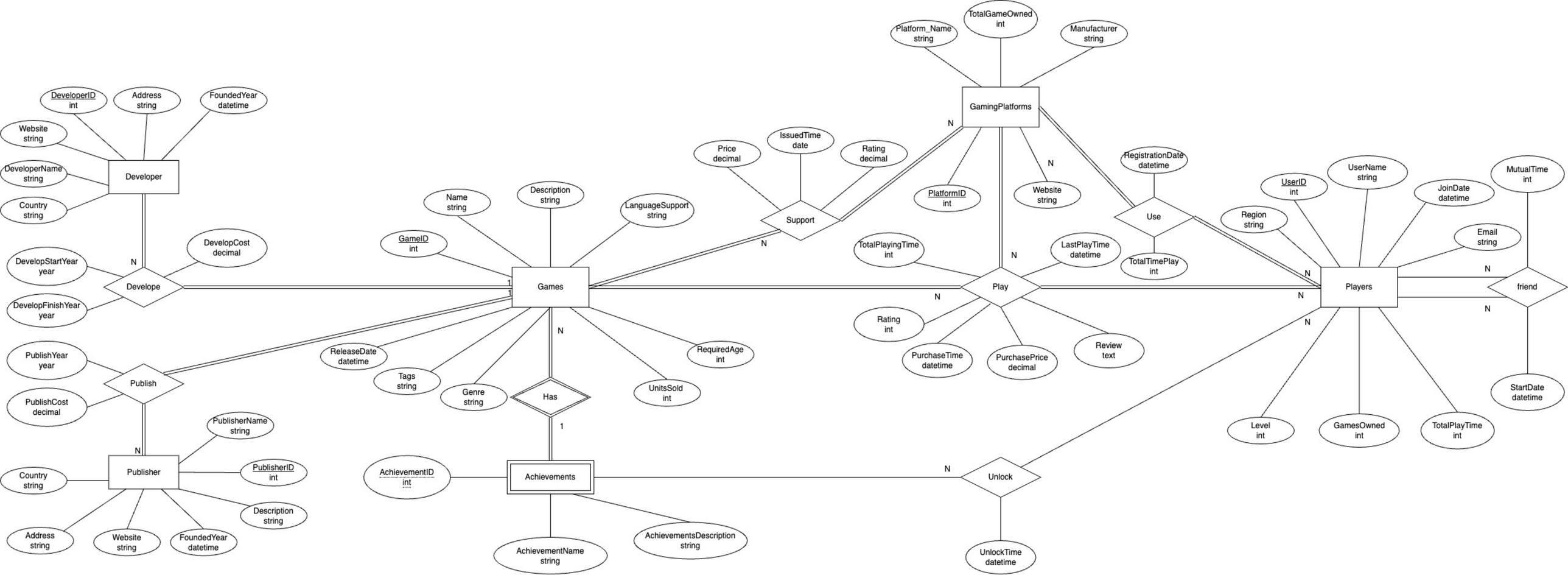
6. Find all games that are exclusive to a platform. (Game, Platform)
7. List the Number of users who spent more than X hours on a platform (User, Platform)
8. List revenue of a platform during a year. (Platform)

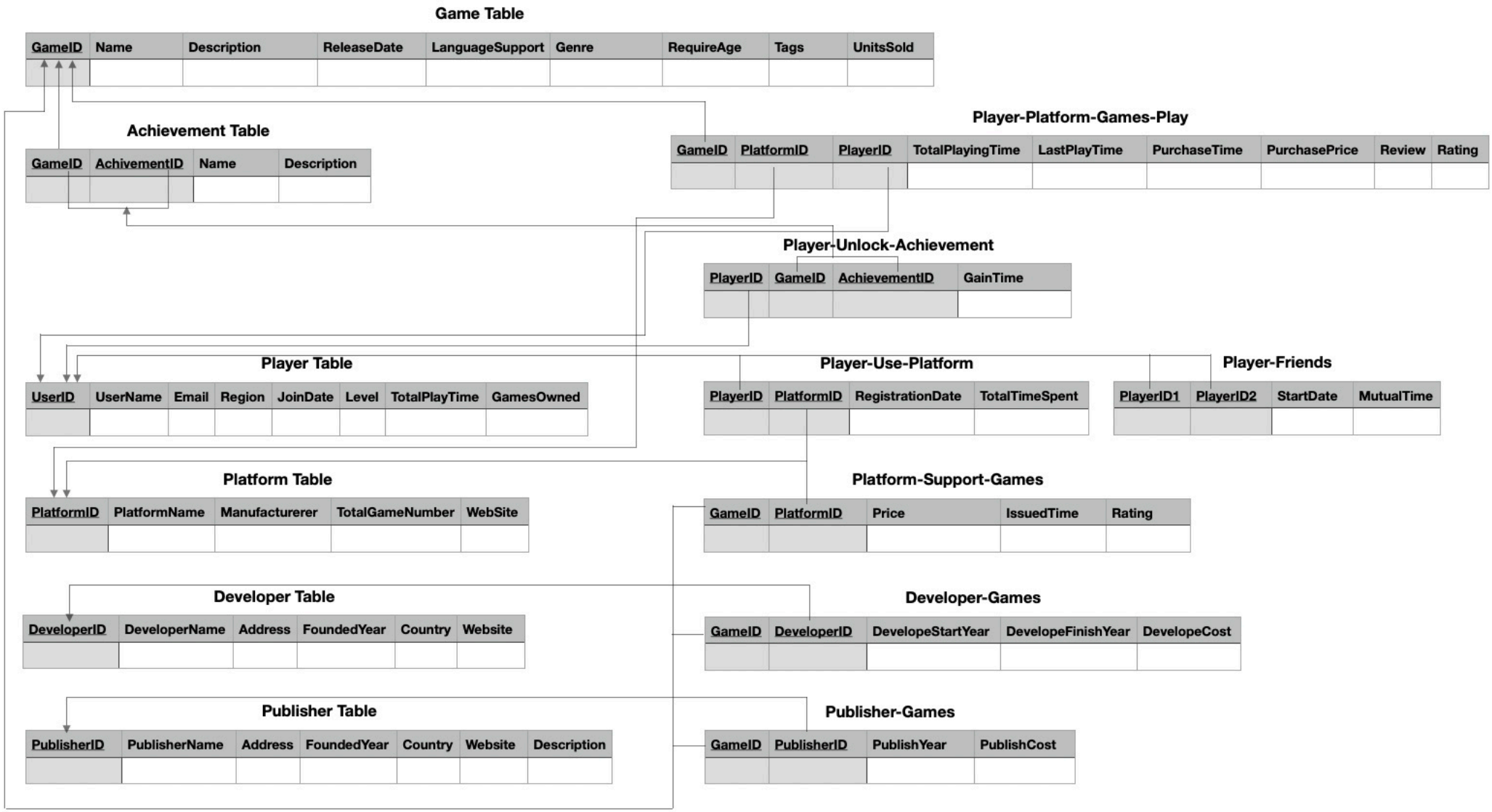
Queries on Users:

9. Find the top N players by playtime of a game. (Player, Game)
10. Display players ranked by achievements in a specific game . (Player, Game)
11. List total money spent by a player.
12. List all purchases of a player.
13. List friends with mutual time above threshold.

Queries on Publisher/Developer:

14. List top N developers/publishers by revenue during a time period.
15. List developers/publishers by number of high-rated games.
16. List developers/publishers by number of platform-compatible games.


DeveloperID int
Address string
FoundedYear datetime
Website string
DeveloperName string
Country string
Developer
DevelopStartYear year
DevelopCost decimal
Develope
DevelopFinishYear year
GameID int
Name string
Description string
LanguageSupport string
Games
PublishYear year
Publish
PublishCost decimal
ReleaseDate datetime
Tags string
Genre string
Has
UnitsSold int
RequiredAge int
PublisherName string
Publisher
PublisherID int
Country string
Address string
Website string
FoundedYear datetime
Description string
AchievementID int
Achievements
AchievementName string
AchievementsDescription string
Price decimal
IssuedTime date
Rating decimal
Support
Platform_Name string
TotalGameOwned int
Manufacturer string
GamingPlatforms
PlatformID int
Website string
RegistrationDate datetime
Use
TotalTimePlay int
TotalPlayingTime int
LastPlayTime datetime
Play
Rating int
PurchaseTime datetime
PurchasePrice decimal
Review text
Unlock
UnlockTime datetime
UserID int
UserName string
JoinDate datetime
MutualTime int
Region string
Email string
Players
friend
StartDate datetime
Level int
GamesOwned int
TotalPlayTime int
N
1

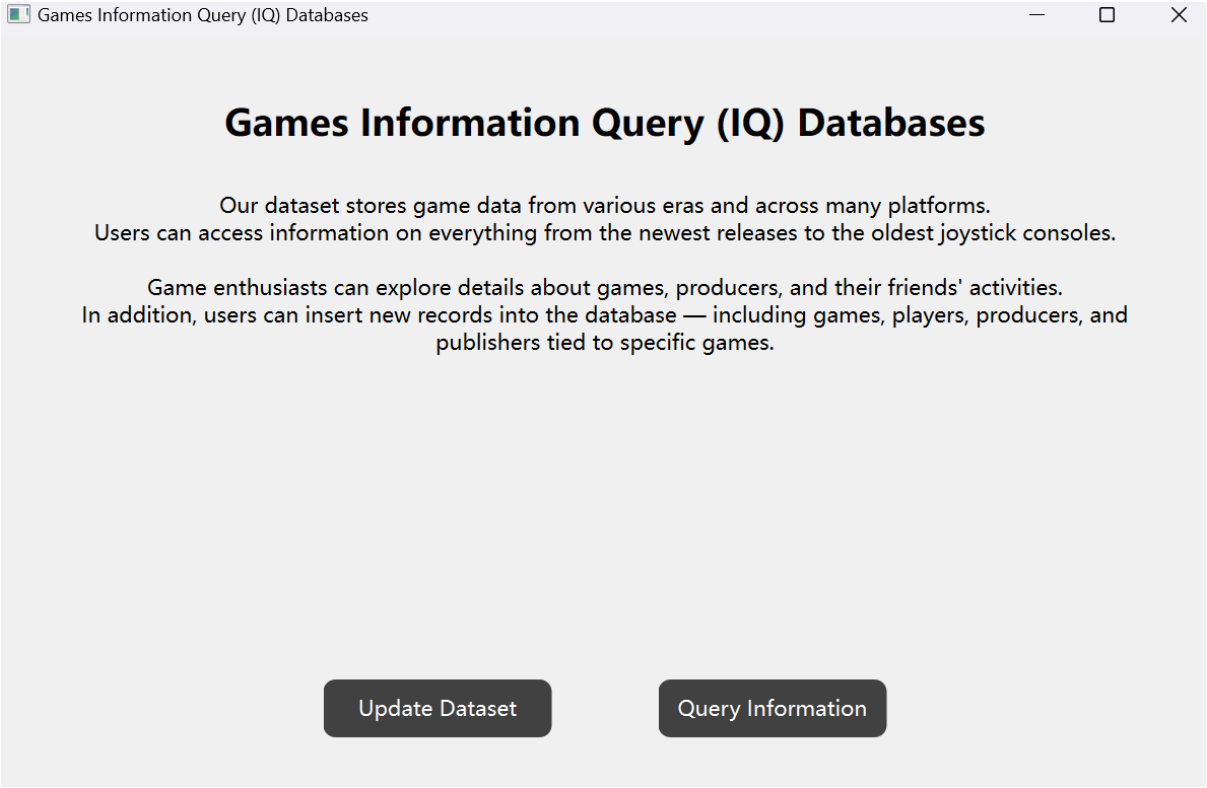

Game Table
GameID Name Description ReleaseDate LanguageSupport Genre RequireAge Tags UnitsSold
Achievement Table
GameID AchivementID Name Description
Player-Platform-Games-Play
GameID PlatformID PlayerID TotalPlayingTime LastPlayTime PurchaseTime PurchasePrice Review Rating
Player-Unlock-Achievement
PlayerID GameID AchievementID GainTime
Player Table
UserID UserName Email Region JoinDate Level TotalPlayTime GamesOwned
Player-Use-Platform
PlayerID PlatformID RegistrationDate TotalTimeSpent
Player-Friends
PlayerID1 PlayerID2 StartDate MutualTime
Platform Table
PlatformID PlatformName Manufacturerer TotalGameNumber WebSite
Platform-Support-Games
GameID PlatformID Price IssuedTime Rating
Developer Table
DeveloperID DeveloperName Address FoundedYear Country Website
Developer-Games
GameID DeveloperID DevelopeStartYear DevelopeFinishYear DevelopeCost
Publisher Table
PublisherID PublisherName Address FoundedYear Country Website Description
Publisher-Games
GameID PublisherID PublishYear PublishCost

Dataset:
Steam Game datasets:
https://www.kaggle.com/datasets/fronkongames/steam-games-dataset?select=games.csv
Game(Name, Requeired_Age, Release_date, Description, LanguageSupport, UnitsSold)
https://www.kaggle.com/datasets/nikdavis/steam-store-games
Game(Name, PulisherID, DeveloperID, Genres, SupportedPlatform, Price, Required_age)
Steam Review Dataset: https://zenodo.org/records/1000885
https://www.kaggle.com/datasets/andrewmvd/steam-reviews
Review
Video Games Dataset API: https://rawg.io/apidocs
Game(Developer, Publisher, SupportedPlatform, tag, Genres)
PS5 Game Dataset:
https://www.kaggle.com/datasets/kanchana1990/ps5-game-data-2000-titles-explored
Game(Name, Publisher, Release_date)
Xbox Game Dataset: https://www.kaggle.com/datasets/shivamb/all-xbox-one-games
Game(Name, Publisher, Developer, SupportedPlatform, Genres)

Initialize data:

```
PS D:\Uchi\course\DB\project_group6\project> python UI.py
✅ MySQL connected: <mysql.connector.connection.MySQLConnectio
n object at 0x000001CF0CBD67F0>
Error: 1050 (42S01): Table 'games' already exists
Inserting into table: Games
206 records inserted into 'Games'.
Inserting into table: Player
100 records inserted into 'Player'.
Inserting into table: Platform
14 records inserted into 'Platform'.
Inserting into table: Developer
75 records inserted into 'Developer'.
Inserting into table: Publisher
29 records inserted into 'Publisher'.
Inserting into table: Achievement
```

User Interface:



Queries

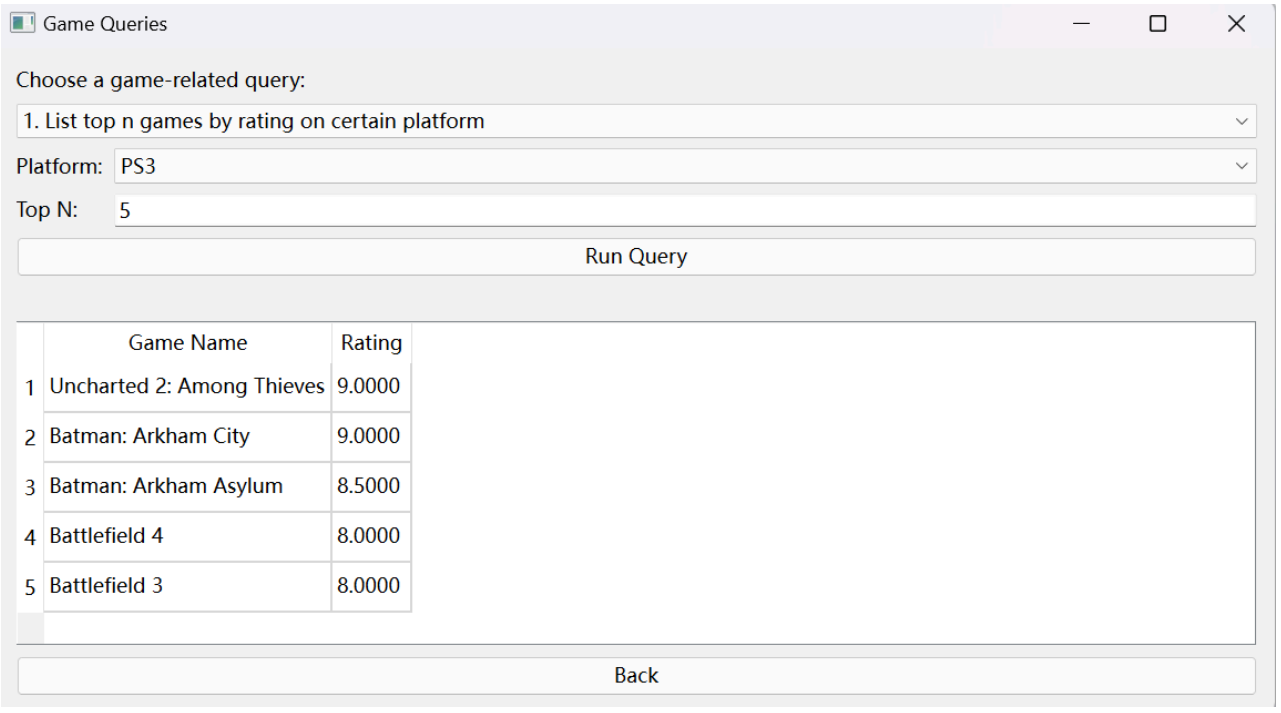
Games

1 list top n games by rating on certain platform

query attributes: platform top n

results:game name and average user rating

```
SELECT G.Name, AVG(PPGP.Rating) AS Rating
FROM Games AS G
INNER JOIN Player_Platform_Games_Play AS PPGP
  ON G.GameID = PPGP.GameID
INNER JOIN Platform AS P
  ON PPGP.PlatformID = P.PlatformID
WHERE P.PlatformName = {platform}
GROUP BY G.GameID
ORDER BY Rating DESC
LIMIT {Top N}
```



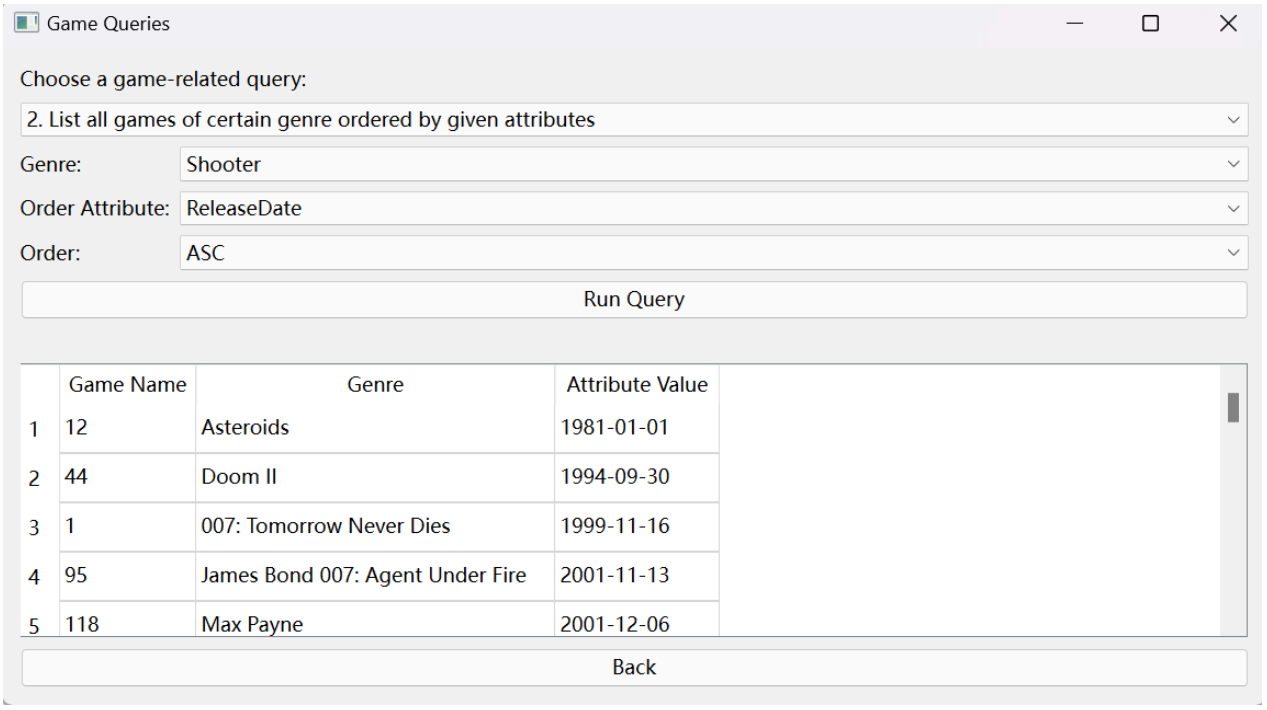
	Game Name	Rating
1	Uncharted 2: Among Thieves	9.0000
2	Batman: Arkham City	9.0000
3	Batman: Arkham Asylum	8.5000
4	Battlefield 4	8.0000
5	Battlefield 3	8.0000

2 list all games of certain genre ordered by given attributes

query attributes: genre / selected attributes

results: id name

```
SELECT GameID, Name, {order_attri}
FROM Games
WHERE Genre = {Genre}
ORDER BY {order_attri} {order}
```



3 List all games produced by a specific publisher / developer along with their release date and genre, ordered by released date

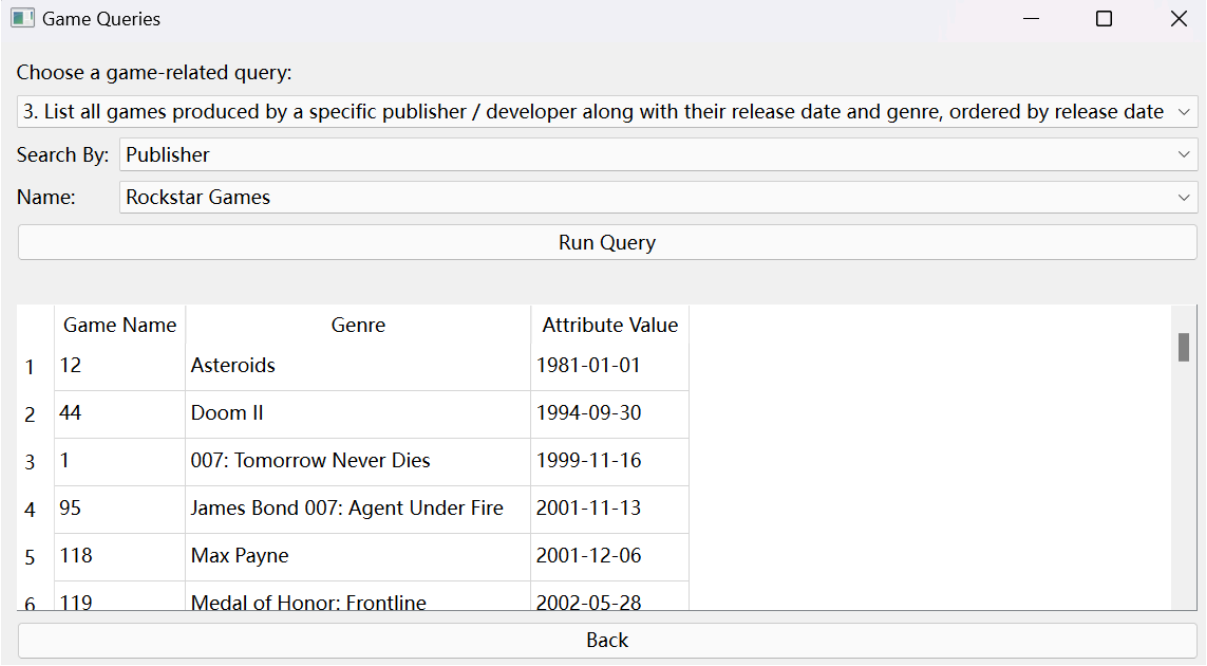
query attributes: publisher developer

results: id name release_date genre

```
SELECT G.GameID, G.ReleaseDate
FROM Games AS G
INNER JOIN Publisher_Games AS PG
  ON G.GameID = PG.GameID
INNER JOIN Publisher AS P
  ON PG.PublisherID = P.PublisherID
WHERE P.PublisherName = {Name}
ORDER BY G.ReleaseDate ASC
```

or

```
SELECT G.GameID, G.ReleaseDate
FROM Games AS G
INNER JOIN Developer_Games AS DG
  ON G.GameID = DG.GameID
INNER JOIN Developer AS D
  ON DG.DeveloperID = D.DeveloperID
WHERE D.DeveloperName ={Name}
ORDER BY G.ReleaseDate ASC
```



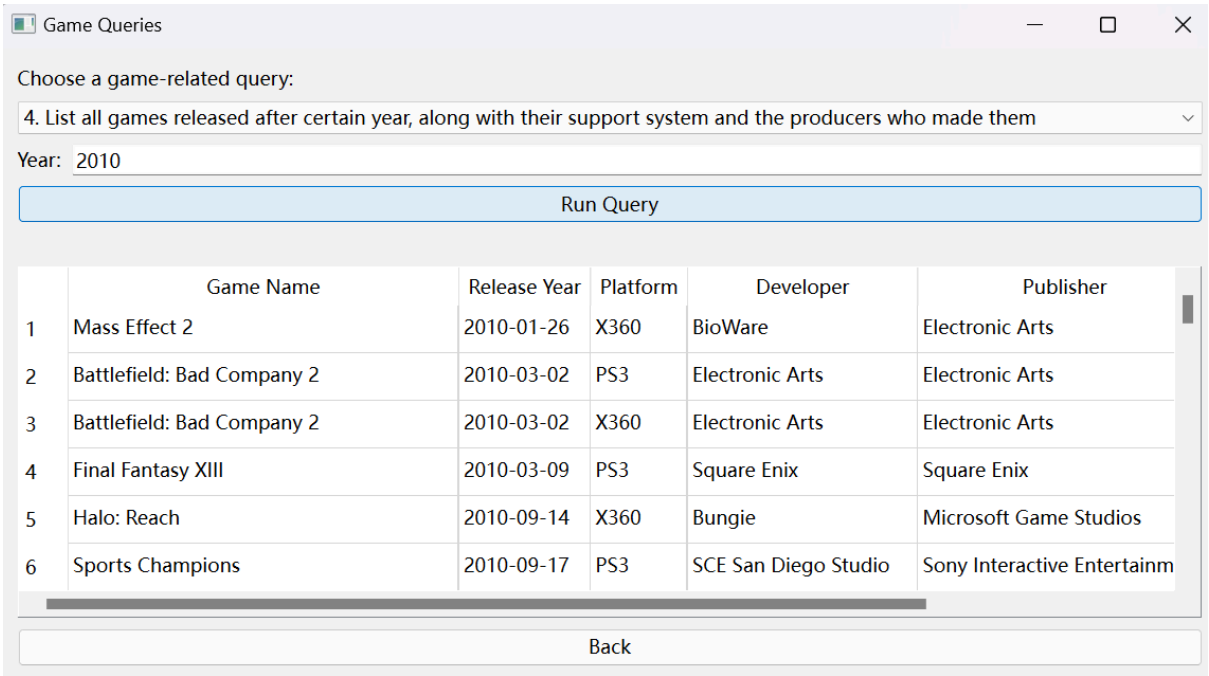
	Game Name	Genre	Attribute Value
1	12	Asteroids	1981-01-01
2	44	Doom II	1994-09-30
3	1	007: Tomorrow Never Dies	1999-11-16
4	95	James Bond 007: Agent Under Fire	2001-11-13
5	118	Max Payne	2001-12-06
6	119	Medal of Honor: Frontline	2002-05-28

4 List all games released after certain year, along with their support system and the producers who
made them

query attributes: year

results: id name support_platform publisher developer

```
SELECT G.Name,
       G.ReleaseDate,
       PF.PlatformName,
       D.DeveloperName,
       P.PublisherName
FROM Games AS G
INNER JOIN Platform_Support_Games AS PSG
  ON G.GameID = PSG.GameID
INNER JOIN Platform AS PF
  ON PSG.PlatformID = PF.PlatformID
INNER JOIN Developer_Games AS DG
  ON G.GameID = DG.GameID
INNER JOIN Developer AS D
  ON DG.DeveloperID = D.DeveloperID
INNER JOIN Publisher_Games AS PG
  ON G.GameID = PG.GameID
INNER JOIN Publisher AS P
  ON PG.PublisherID = P.PublisherID
WHERE G.ReleaseDate >= {Year}
ORDER BY G.ReleaseDate ASC
```



	Game Name	Release Year	Platform	Developer	Publisher
1	Mass Effect 2	2010-01-26	X360	BioWare	Electronic Arts
2	Battlefield: Bad Company 2	2010-03-02	PS3	Electronic Arts	Electronic Arts
3	Battlefield: Bad Company 2	2010-03-02	X360	Electronic Arts	Electronic Arts
4	Final Fantasy XIII	2010-03-09	PS3	Square Enix	Square Enix
5	Halo: Reach	2010-09-14	X360	Bungie	Microsoft Game Studios
6	Sports Champions	2010-09-17	PS3	SCE San Diego Studio	Sony Interactive Entertainm

5 List average ratings/ total units sold of each genre, ordered by them in descending order

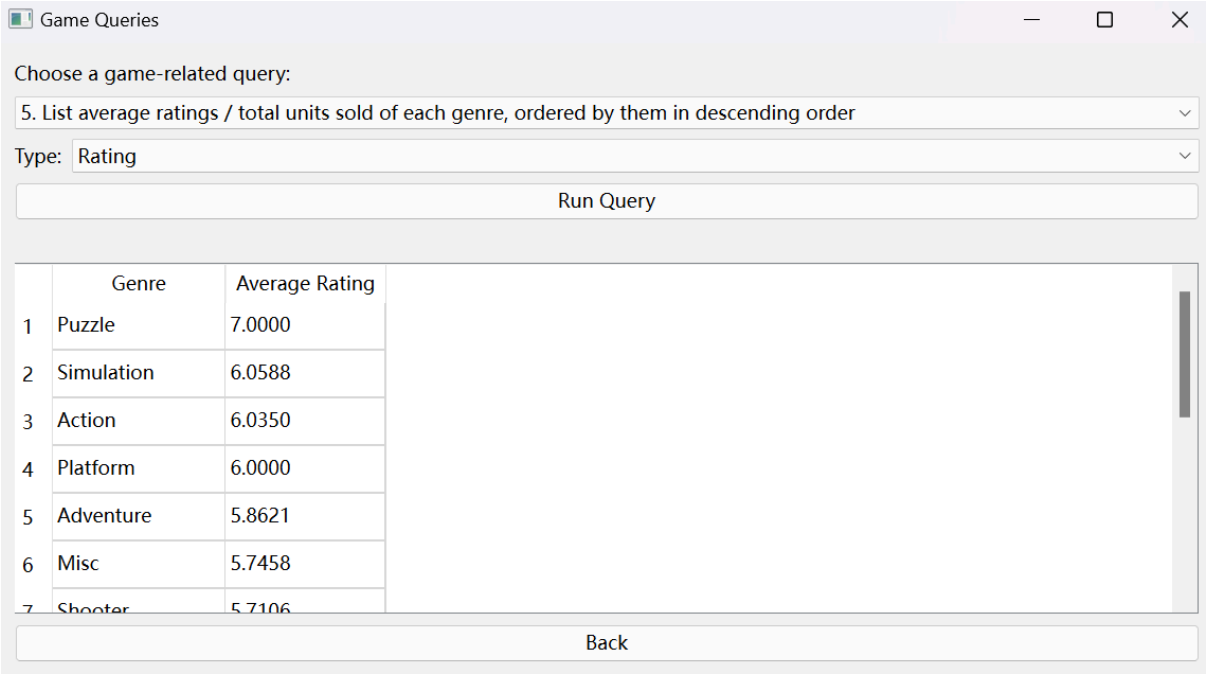
query attributes: average_rating/ total_units

results: genre average_rating

```
SELECT G.Genre,
      AVG(PPGP.Rating) AS AverageRating
FROM Games AS G
INNER JOIN Player_Platform_Games_Play AS PPGP
  ON G.GameID = PPGP.GameID
WHERE G.Genre IS NOT NULL
GROUP BY G.Genre
ORDER BY AverageRating DESC
```

or

```
SELECT Genre,
      SUM(UnitsSold) AS TotalUnitsSold
FROM Games
WHERE Genre IS NOT NULL
GROUP BY Genre
ORDER BY TotalUnitsSold DESC
```



	Genre	Average Rating
1	Puzzle	7.0000
2	Simulation	6.0588
3	Action	6.0350
4	Platform	6.0000
5	Adventure	5.8621
6	Misc	5.7458
7	Shooter	5.7106

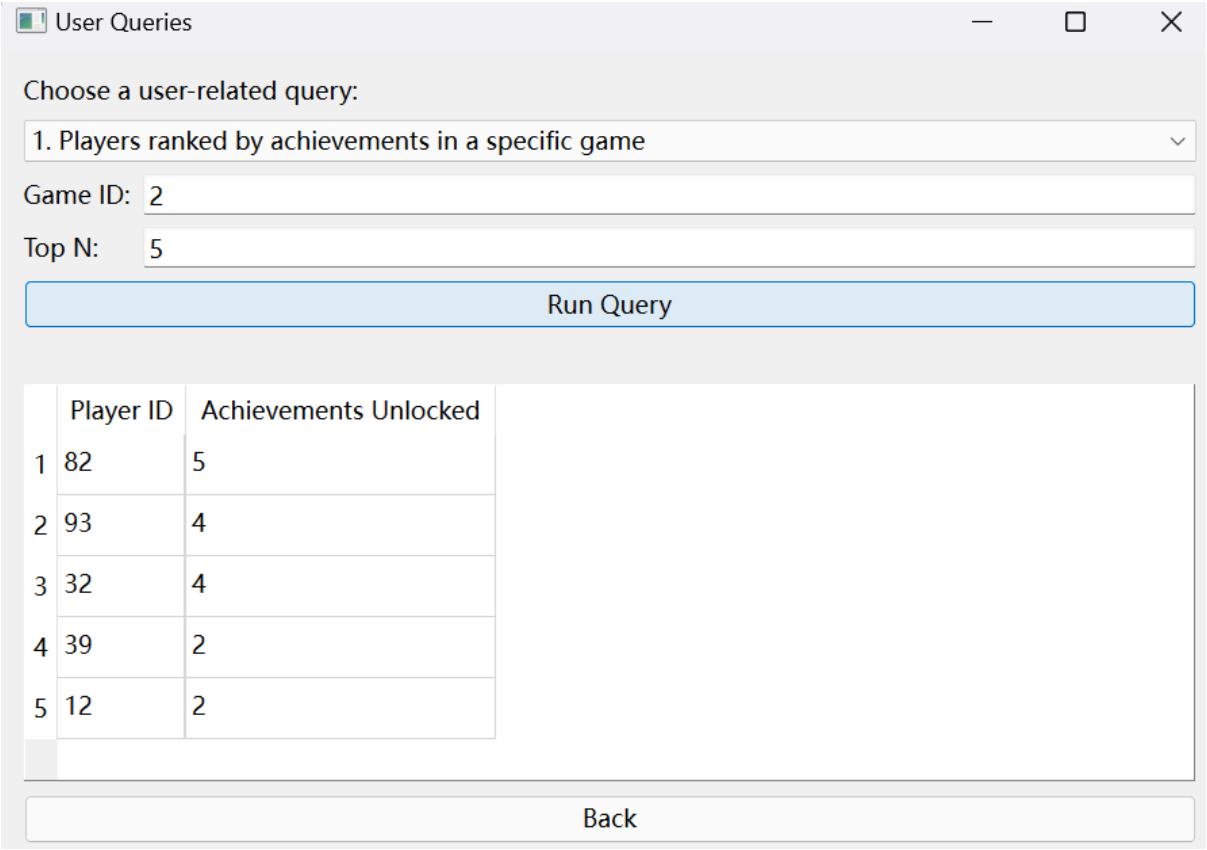
Users

1 For a given game, list all players sorted by the number of achievements they have unlocked in that game, from most to least

query attributes: game_id top n

results: user_id

```
SELECT PlayerID, COUNT(*) AS AchievementsUnlocked
FROM Player_Unlock_Achievement
WHERE GameID = %s
GROUP BY PlayerID
ORDER BY AchievementsUnlocked DESC
LIMIT %s;
```



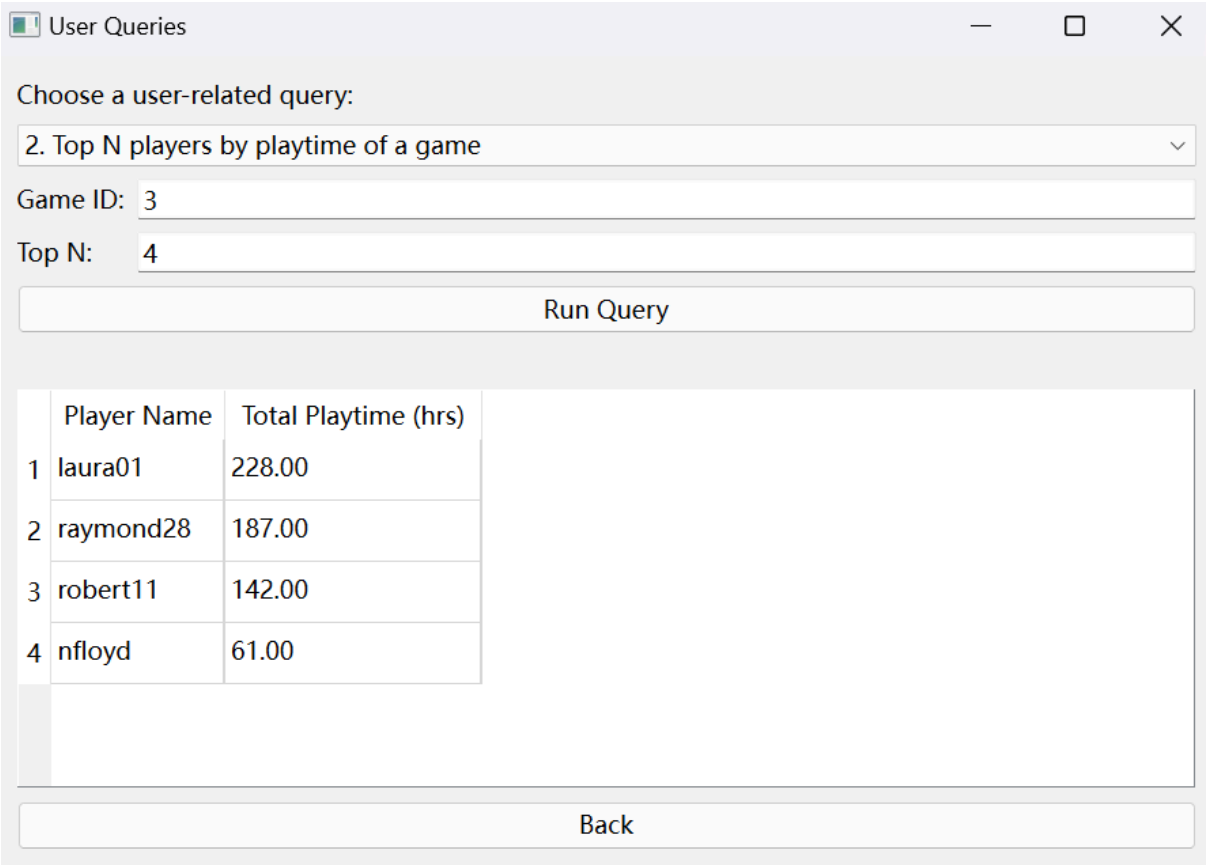
	Player ID	Achievements Unlocked
1	82	5
2	93	4
3	32	4
4	39	2
5	12	2

2 Find the top N players with the highest total playtime of certain specific game across all platforms.

query attributes: game_id top n

results: user_id …

```
SELECT P.UserName, SUM(PPGP.TotalPlayingTime) AS TotalTime
FROM Player_Platform_Games_Play PPGP
JOIN Player P ON PPGP.PlayerID = P.UserID
WHERE PPGP.GameID = %s
GROUP BY P.UserID, P.UserName
ORDER BY TotalTime DESC
LIMIT %s;
```



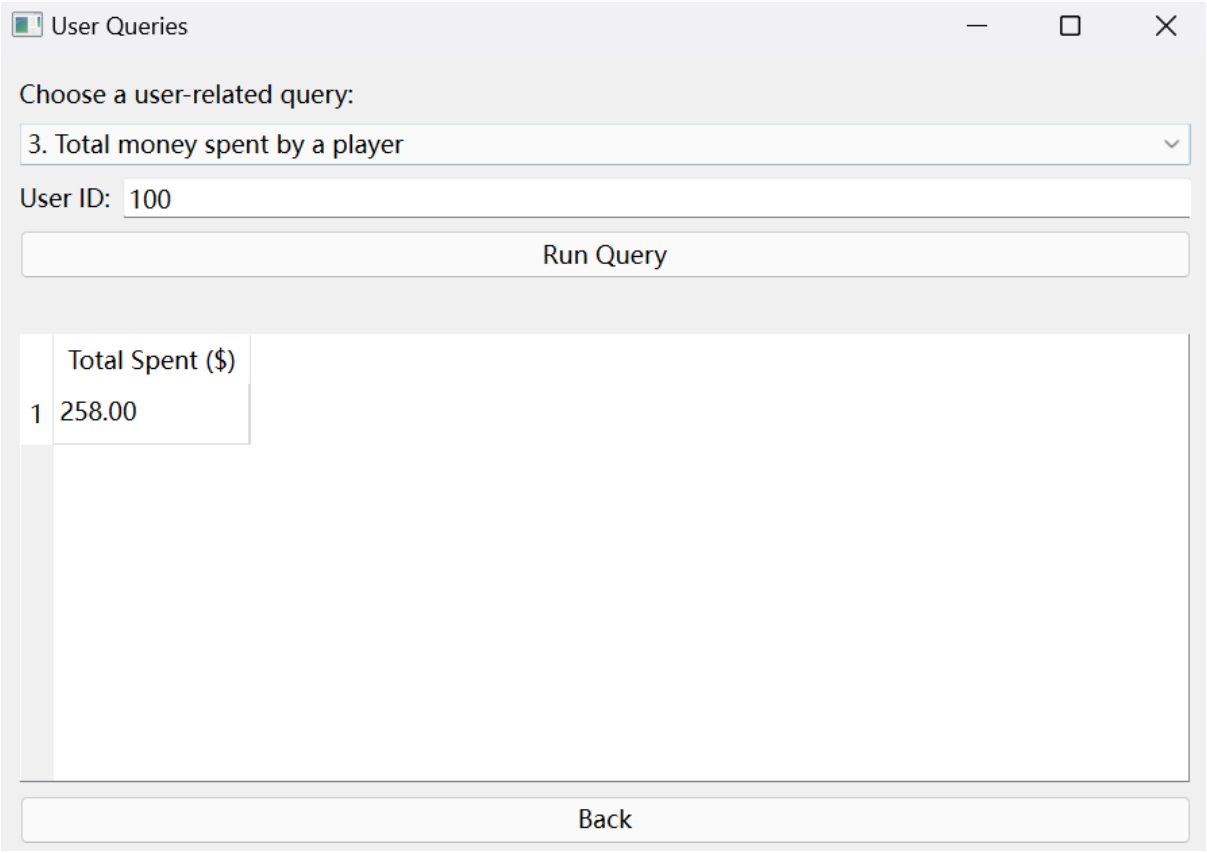
	Player Name	Total Playtime (hrs)
1	laura01	228.00
2	raymond28	187.00
3	robert11	142.00
4	nfloyd	61.00

3 Display the total amount of money a specific player has spent across all platforms

query attributes: user_id top n

results: total amount of money

```
SELECT SUM(PurchasePrice) AS TotalSpent
FROM Player_Platform_Games_Play
WHERE PlayerID = %s;
```

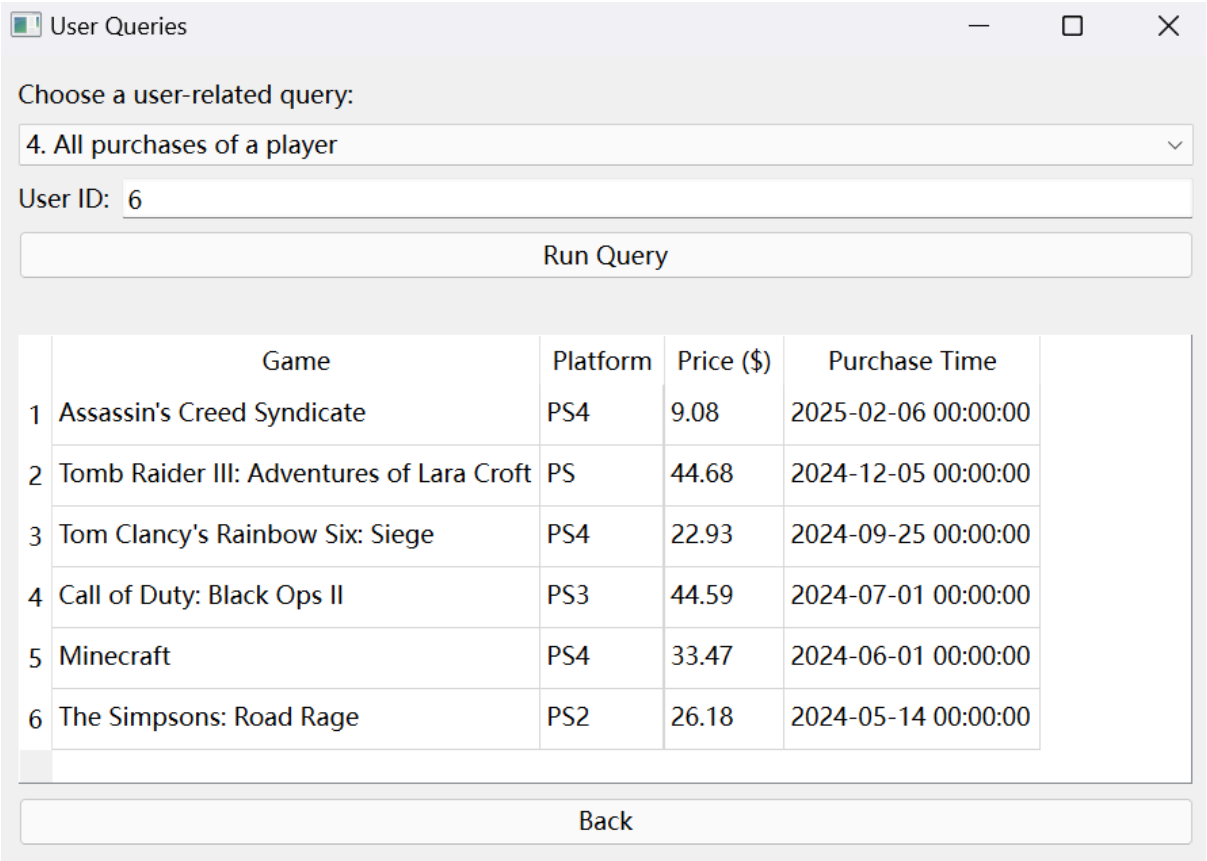


4 List all games a player has purchased, including the platform, price at time of purchase, and transaction time.

query attributes: user_id top n

results: game info

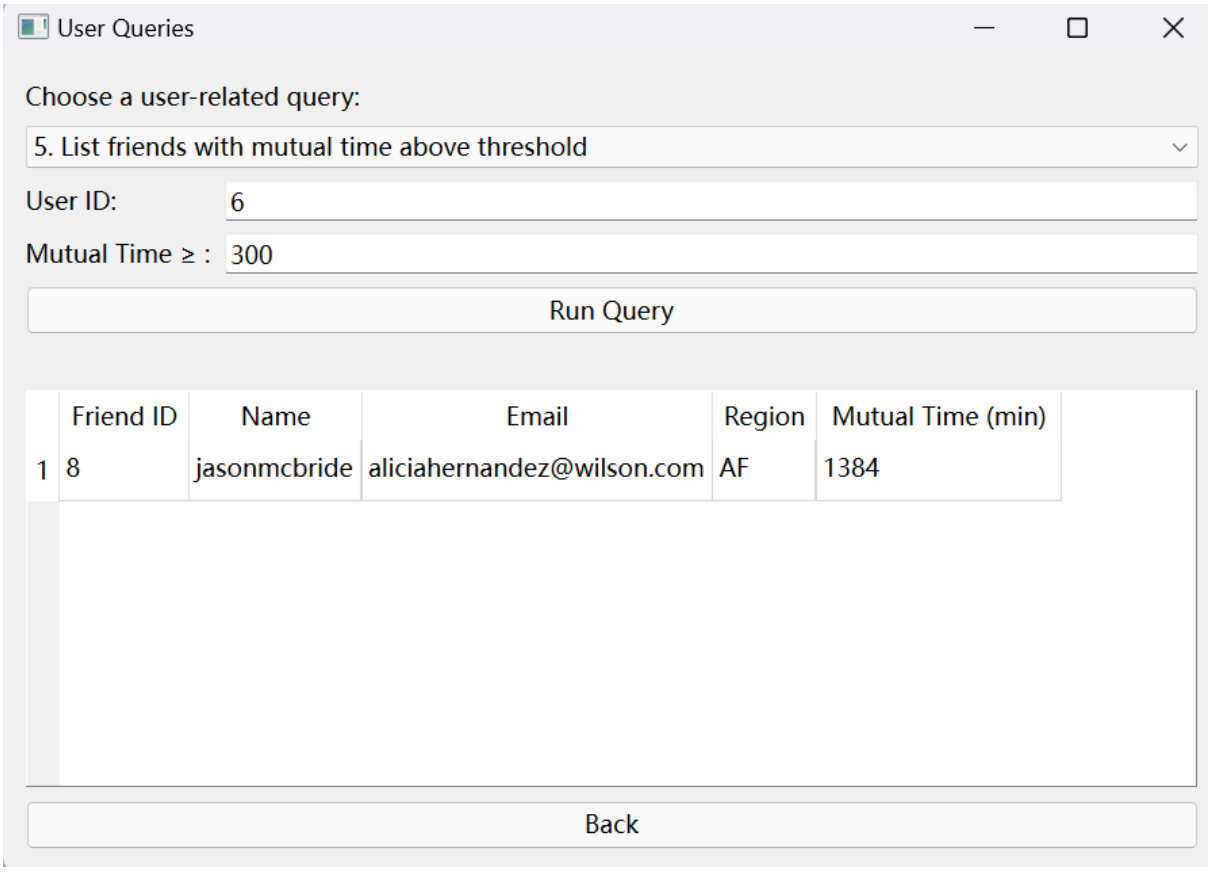
```
SELECT G.Name AS GameName, PF.PlatformName, PPGP.PurchasePrice,
PPGP.PurchaseTime
FROM Player_Platform_Games_Play AS PPGP
JOIN Games G ON G.GameID = PPGP.GameID
JOIN Platform PF ON PF.PlatformID = PPGP.PlatformID
WHERE PPGP.PlayerID = %s
ORDER BY PPGP.PurchaseTime DESC;
```



	Game	Platform	Price ($)	Purchase Time
1	Assassin's Creed Syndicate	PS4	9.08	2025-02-06 00:00:00
2	Tomb Raider III: Adventures of Lara Croft	PS	44.68	2024-12-05 00:00:00
3	Tom Clancy's Rainbow Six: Siege	PS4	22.93	2024-09-25 00:00:00
4	Call of Duty: Black Ops II	PS3	44.59	2024-07-01 00:00:00
5	Minecraft	PS4	33.47	2024-06-01 00:00:00
6	The Simpsons: Road Rage	PS2	26.18	2024-05-14 00:00:00

5.List friends with mutual time above threshold

```
SELECT p2.UserID, p2.UserName, p2.Email, p2.Region, pf.MutualTime
    FROM Player_Friends pf
    JOIN Player p2 ON pf.Player2ID = p2.UserID
    WHERE pf.Player1ID = %s AND pf.MutualTime >= %s;
```



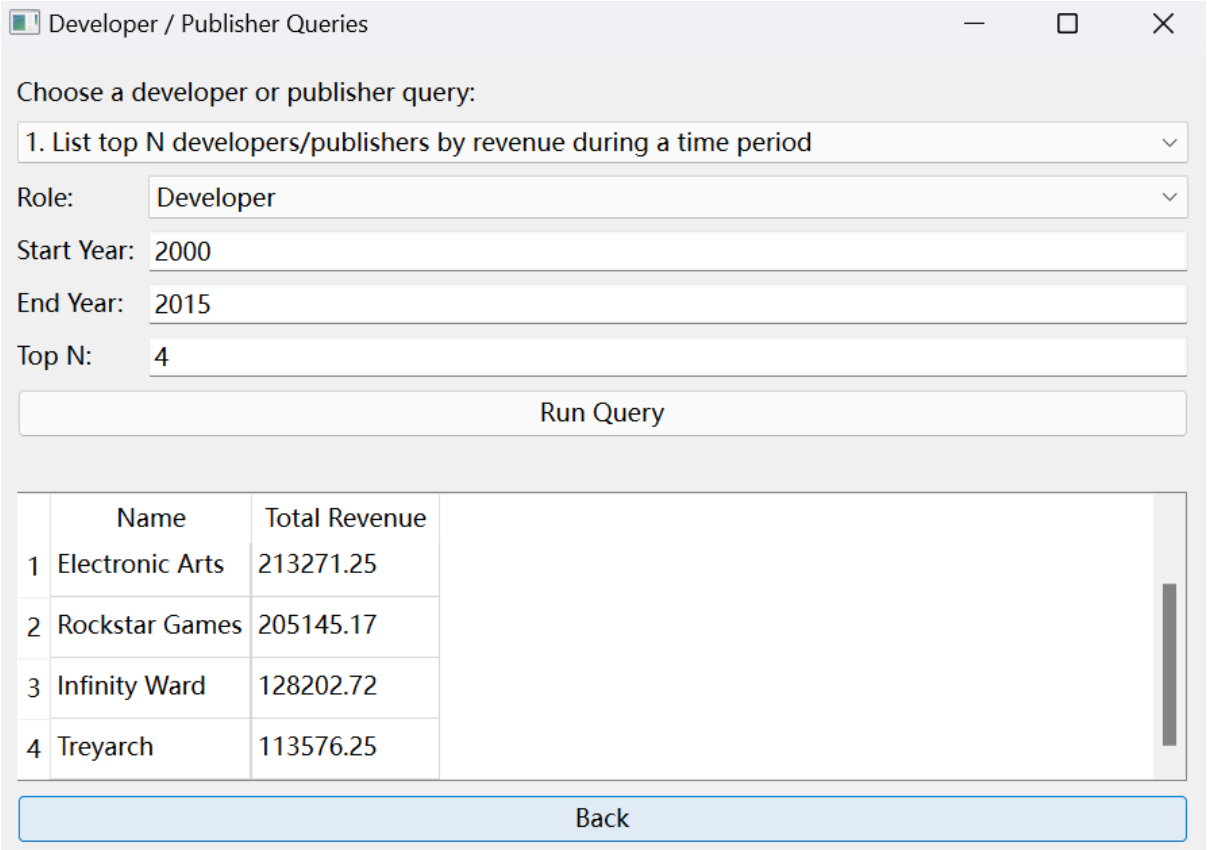
Developer/ Publisher

1 List revenues of all the developers/publishers during a certain time period.

query attributes: years top n

results: revenues

```
def q_dev_pub_revenues(cursorObject, role, start_year, end_year, top_n):
    if role == "Developer":
        query = """
            SELECT D.DeveloperName, SUM(G.UnitsSold * PSG.Price) AS TotalRevenue
            FROM Games G
            INNER JOIN Developer_Games DG ON G.GameID = DG.GameID
            INNER JOIN Developer D ON DG.DeveloperID = D.DeveloperID
            INNER JOIN Platform_Support_Games PSG ON G.GameID = PSG.GameID
            WHERE DG.DevelopeFinishYear BETWEEN %s AND %s
            GROUP BY D.DeveloperID
            ORDER BY TotalRevenue DESC
            LIMIT %s
        """
    elif role == "Publisher":
        query = """
            SELECT P.PublisherName, SUM(G.UnitsSold * PSG.Price) AS TotalRevenue
            FROM Games G
            INNER JOIN Publisher_Games PG ON G.GameID = PG.GameID
            INNER JOIN Publisher P ON PG.PublisherID = P.PublisherID
            INNER JOIN Platform_Support_Games PSG ON G.GameID = PSG.GameID
            WHERE PG.PublishYear BETWEEN %s AND %s
            GROUP BY P.PublisherID
            ORDER BY TotalRevenue DESC
            LIMIT %s
        """
    else:
        return []
```

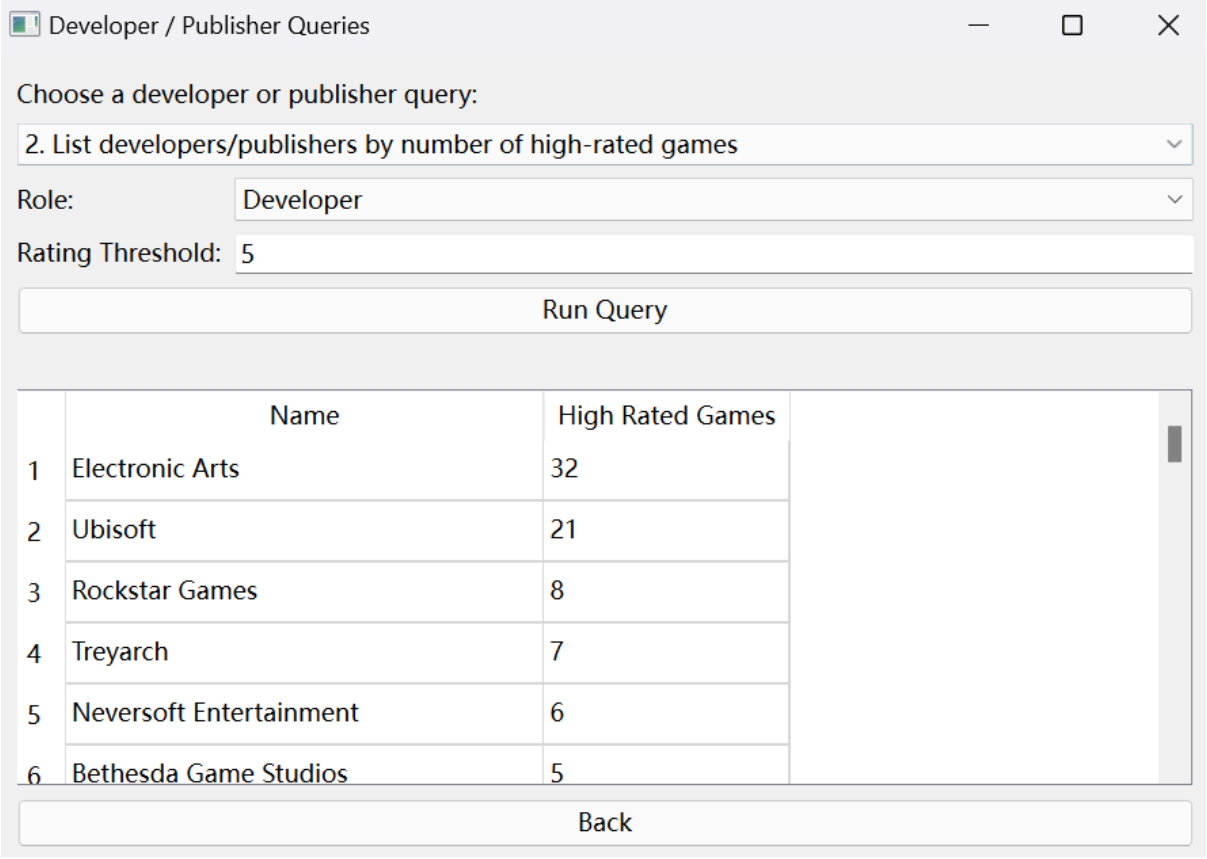

Developer / Publisher Queries
Choose a developer or publisher query:
1. List top N developers/publishers by revenue during a time period
Role:
Developer
Start Year:
2000
End Year:
2015
Top N:
4
Run Query
Name
Total Revenue
1
Electronic Arts
213271.25
2
Rockstar Games
205145.17
3
Infinity Ward
128202.72
4
Treyarch
113576.25
Back

2 the number of games rated over n developed by any publisher/developer .

query attributes: rating threshold

results: the numbers of games

```python
def q_dev_pub_rating(cursorObject, role, rating_threshold):
    if role == "Developer":
        query = """
            SELECT D.DeveloperName, COUNT(DISTINCT G.GameID) AS HighRatedGames
            FROM Games G
            INNER JOIN Developer_Games DG ON G.GameID = DG.GameID
            INNER JOIN Developer D ON DG.DeveloperID = D.DeveloperID
            INNER JOIN Player_Platform_Games_Play PPGP ON G.GameID = PPGP.GameID
            WHERE PPGP.Rating > %s
            GROUP BY D.DeveloperID
            ORDER BY HighRatedGames DESC
        """
    elif role == "Publisher":
        query = """
            SELECT P.PublisherName, COUNT(DISTINCT G.GameID) AS HighRatedGames
            FROM Games G
            INNER JOIN Publisher_Games PG ON G.GameID = PG.GameID
            INNER JOIN Publisher P ON PG.PublisherID = P.PublisherID
            INNER JOIN Player_Platform_Games_Play PPGP ON G.GameID = PPGP.GameID
            WHERE PPGP.Rating > %s
            GROUP BY P.PublisherID
            ORDER BY HighRatedGames DESC
        """
    else:
        return []
```



3 the number of games that are compatible on over n platforms developed by any publisher/developer.

query attributes: platform compatibility threshold

results: the numbers of games

```
def q_dev_pub_compatibility(cursorObject, role, platform_threshold):
    if role == "Developer":
        query = """
            SELECT D.DeveloperName, COUNT(*) AS GameCount
            FROM (
                SELECT G.GameID, COUNT(DISTINCT PSG.PlatformID) AS PlatformCount
                FROM Games G
                INNER JOIN Platform_Support_Games PSG ON G.GameID = PSG.GameID
                GROUP BY G.GameID
                HAVING PlatformCount > %s
            ) AS CompatibleGames
            INNER JOIN Developer_Games DG ON CompatibleGames.GameID = DG.GameID
            INNER JOIN Developer D ON DG.DeveloperID = D.DeveloperID
            GROUP BY D.DeveloperID
            ORDER BY GameCount DESC
        """
    elif role == "Publisher":
        query = """
            SELECT P.PublisherName, COUNT(*) AS GameCount
            FROM (
                SELECT G.GameID, COUNT(DISTINCT PSG.PlatformID) AS PlatformCount
                FROM Games G
                INNER JOIN Platform_Support_Games PSG ON G.GameID = PSG.GameID
                GROUP BY G.GameID
                HAVING PlatformCount > %s
            ) AS CompatibleGames
            INNER JOIN Publisher_Games PG ON CompatibleGames.GameID = PG.GameID
            INNER JOIN Publisher P ON PG.PublisherID = P.PublisherID
            GROUP BY P.PublisherID
            ORDER BY GameCount DESC
        """
    else:
        return []
```

Developer / Publisher Queries

Choose a developer or publisher query:

3. List developers/publishers by number of platform-compatible games

Role: Developer

Min Platform Count: 2

Run Query

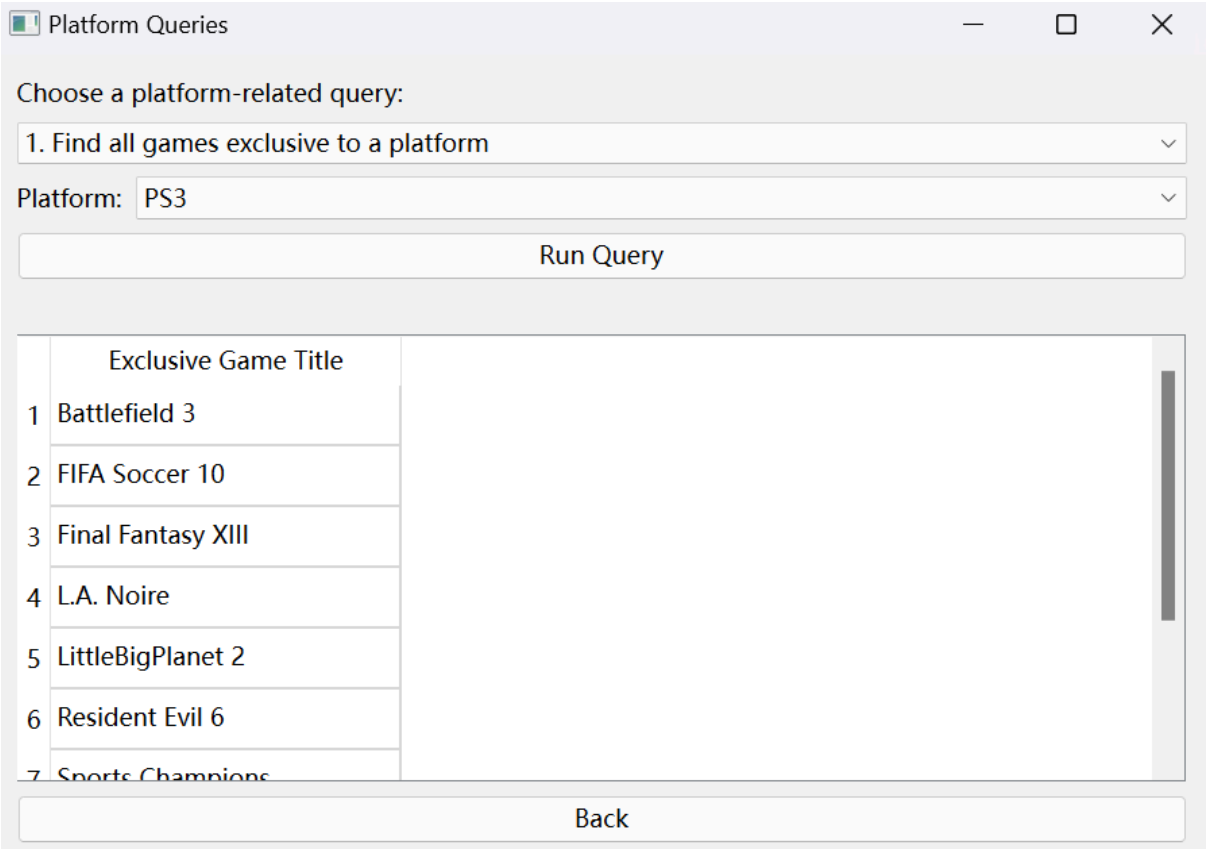
	Name	Platform Count
1	Electronic Arts	2
2	Ubisoft	1
3	High Moon Studios	1
4	Sledgehammer Games	1
5	Infinity Ward	1
6	Rockstar Games	1

Back

Platform

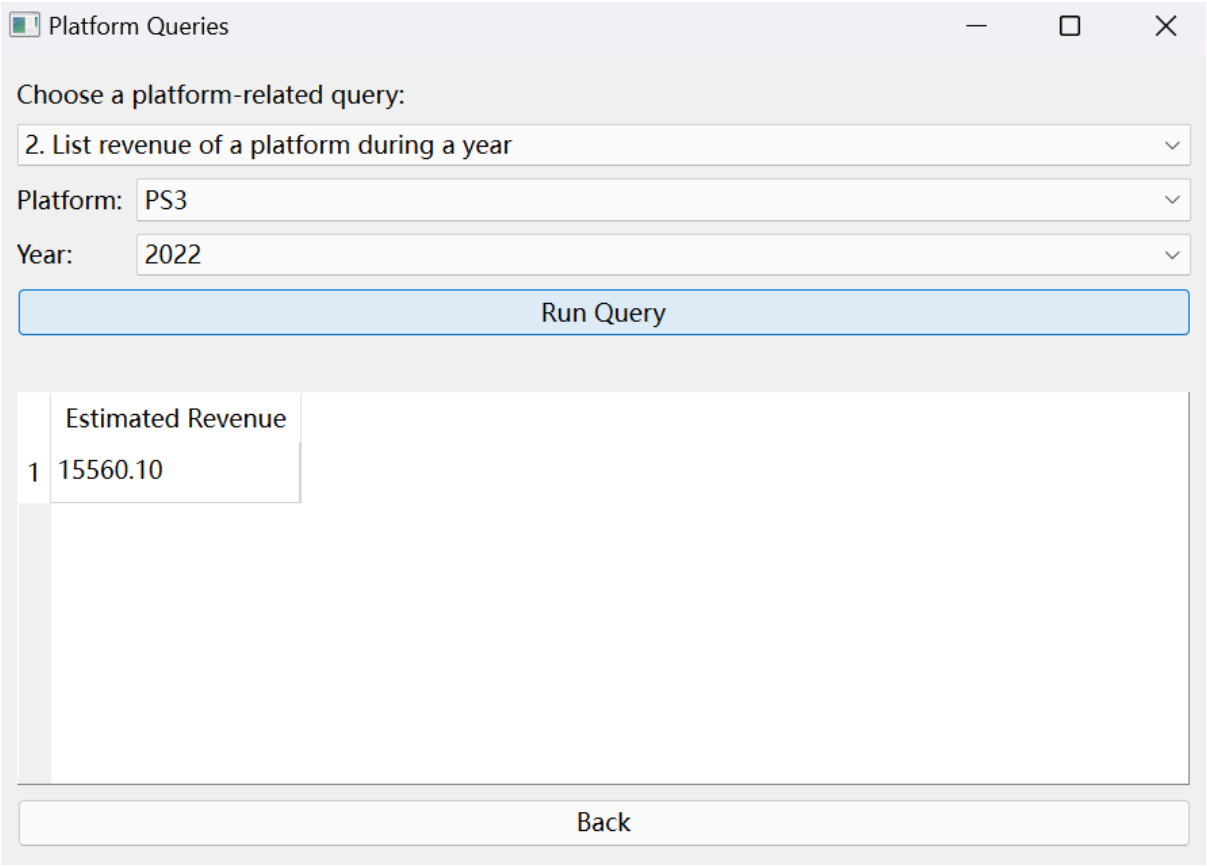
1. Find games that are exclusive to a certain platform.
   Query attributes: platform
   Results: Exclusive game title

```python
def q_platform_exclusive_games(cursorObject, platform_name):
    sql = """
        SELECT g.Name
        FROM Games g
        JOIN Platform_Support_Games p ON g.GameID = p.GameID
        JOIN Platform pf ON pf.PlatformID = p.PlatformID
        WHERE pf.PlatformName = %s
        AND g.GameID NOT IN (
            SELECT p2.GameID
            FROM Platform_Support_Games p2
            JOIN Platform pf2 ON p2.PlatformID = pf2.PlatformID
            WHERE pf2.PlatformName != %s
        )
        LIMIT 20;
    """
```



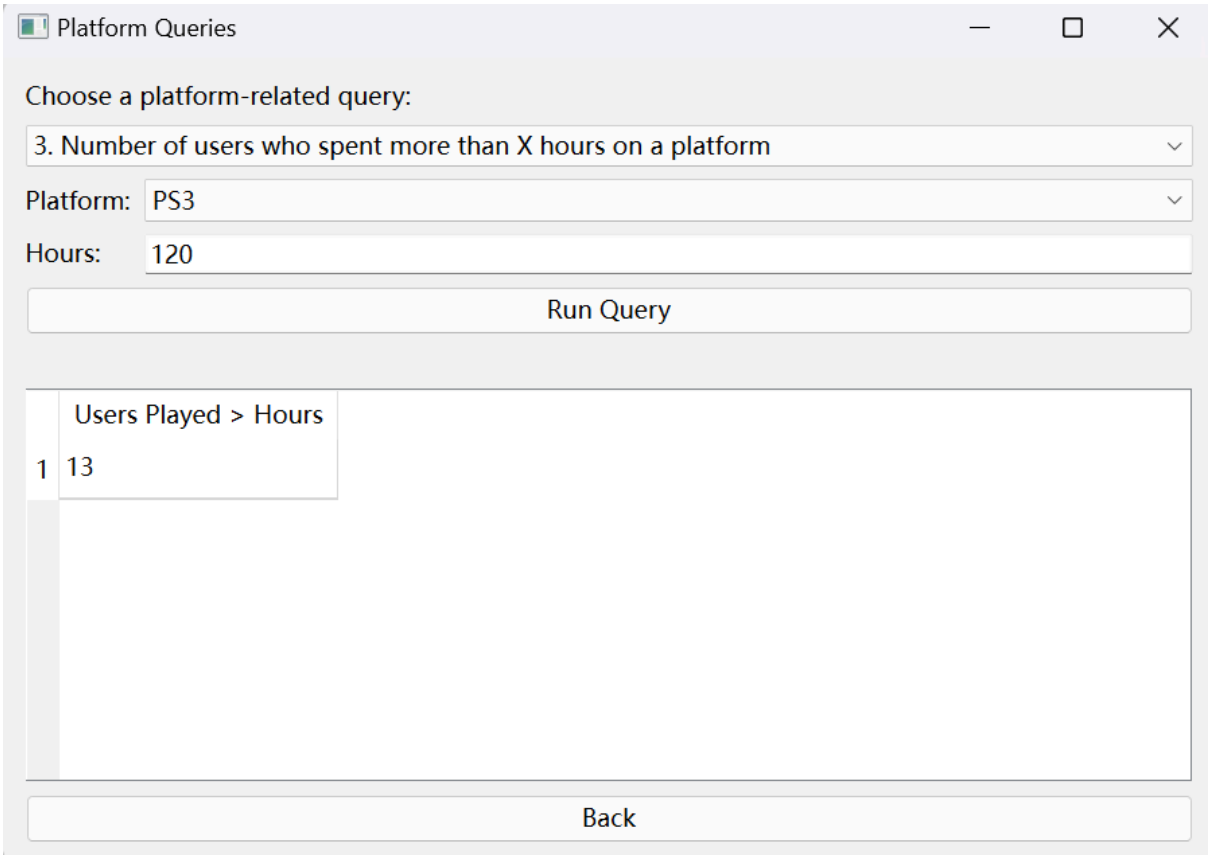
2. List revenues of all certain platforms during a certain time period.
   Query attributes: Platform, Year
   Results: Estimated revenue

```python
def q_platform_revenue(cursorObject, platform_name, year):
    sql = """
        SELECT SUM(g.UnitsSold * psg.Price) AS EstimatedRevenue
        FROM Platform_Support_Games psg
        JOIN Platform pf ON pf.PlatformID = psg.PlatformID
        JOIN Games g ON psg.GameID = g.GameID
        WHERE pf.PlatformName = %s AND YEAR(psg.IssuedTime) = %s;
    """
```



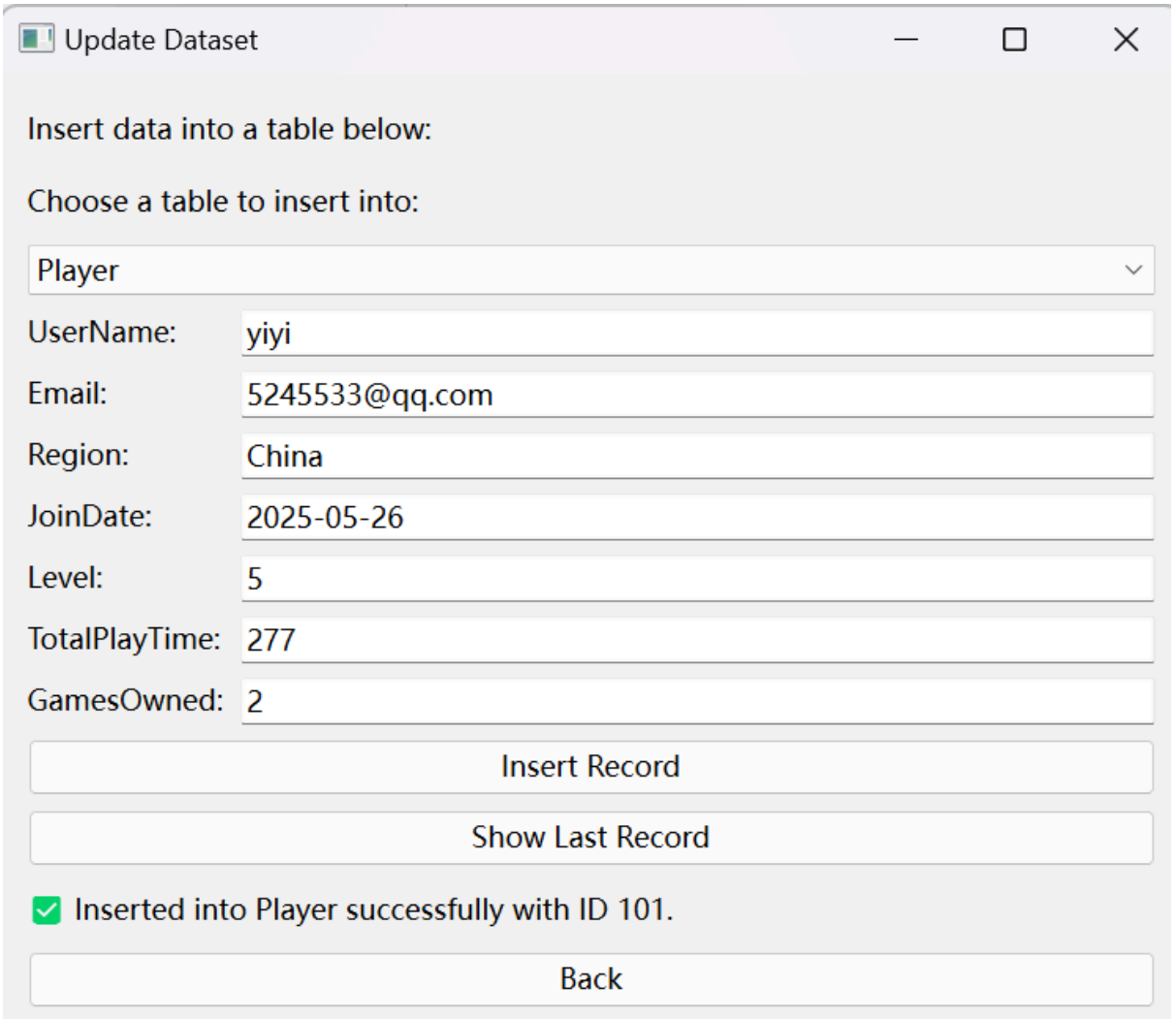
3. The number of users spent more than n hours on a certain platform.
   Query attributes: Platform, Hours threshold
   Results: The numbers of users

```python
def q_platform_user(cursorObject, platform_name, min_hours):
    sql = """
        SELECT COUNT(*)
        FROM Player_Use_Platform pup
        JOIN Platform pf ON pf.PlatformID = pup.PlatformID
        WHERE pf.PlatformName = %s AND pup.TotalTimeSpent >= %s;
    """
```

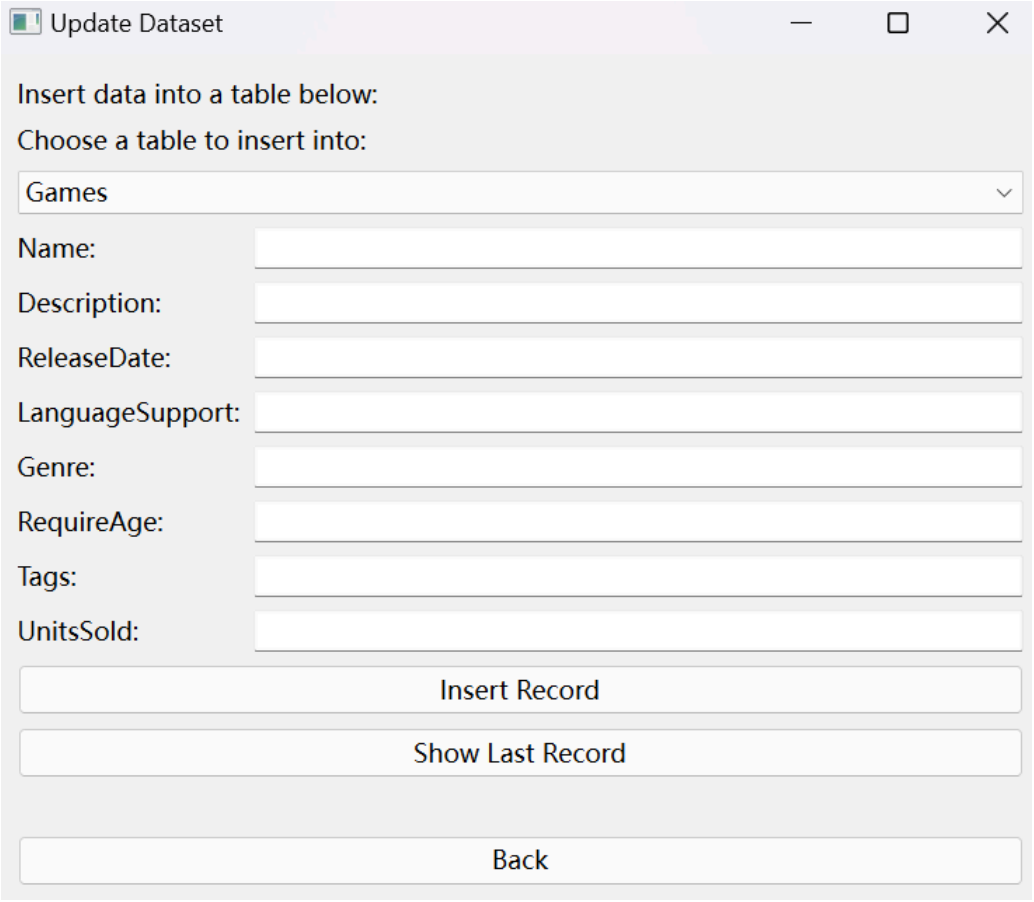


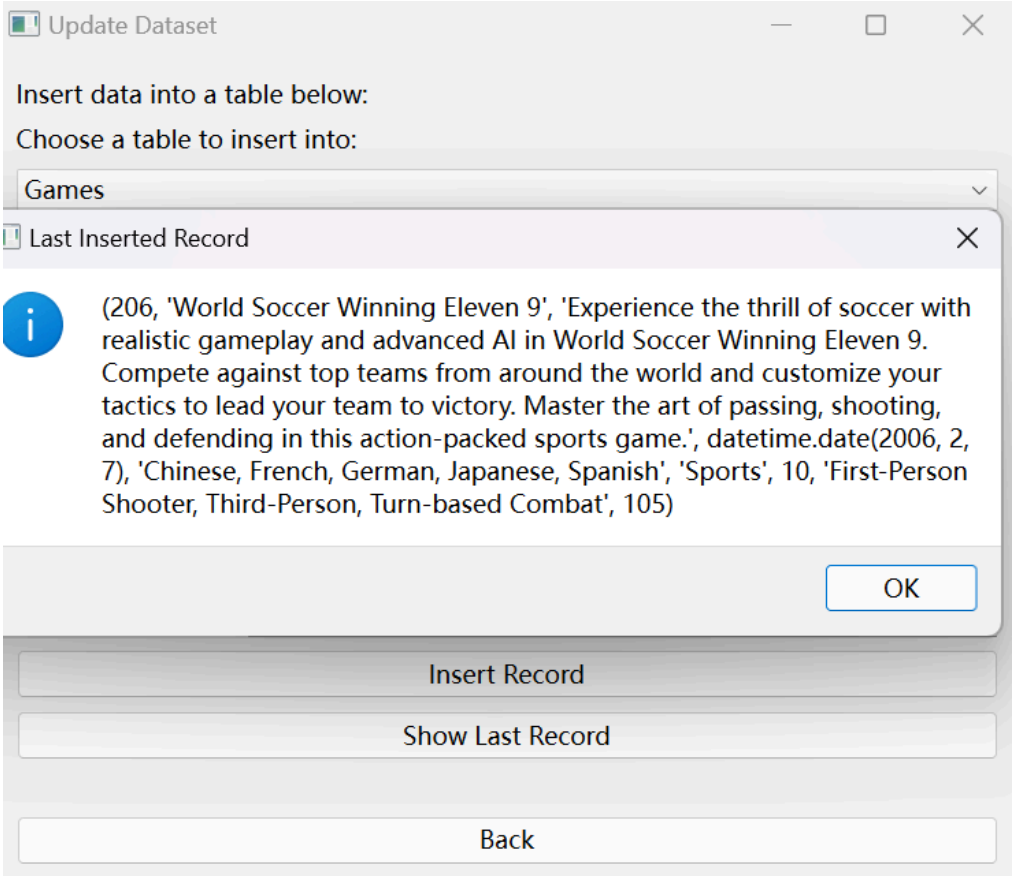
Some Insert Examples

Insert User



Insert Game


Update Dataset
Insert data into a table below:
Choose a table to insert into:
Games
Name:
Description:
ReleaseDate:
LanguageSupport:
Genre:
RequireAge:
Tags:
UnitsSold:
Insert Record
Show Last Record
Back



Insert/Update **Player-Platform-Games-Play**

Update Dataset

Insert data into a table below:

Choose a table to insert into:

Player_Platform_Games_Play

GameID:	2
PlatformID:	5
PlayerID:	5
TotalPlayingTime:	100
LastPlayTime:	2025-05-26
PurchaseTime:	2025-05-01
PurchasePrice:	10
Review:	/
Rating:	8

Insert Record

Show Last Record

Inserted into Player_Platform_Games_Play successfully with ID Composed.

Back