

GAMING ECOSYSTEM

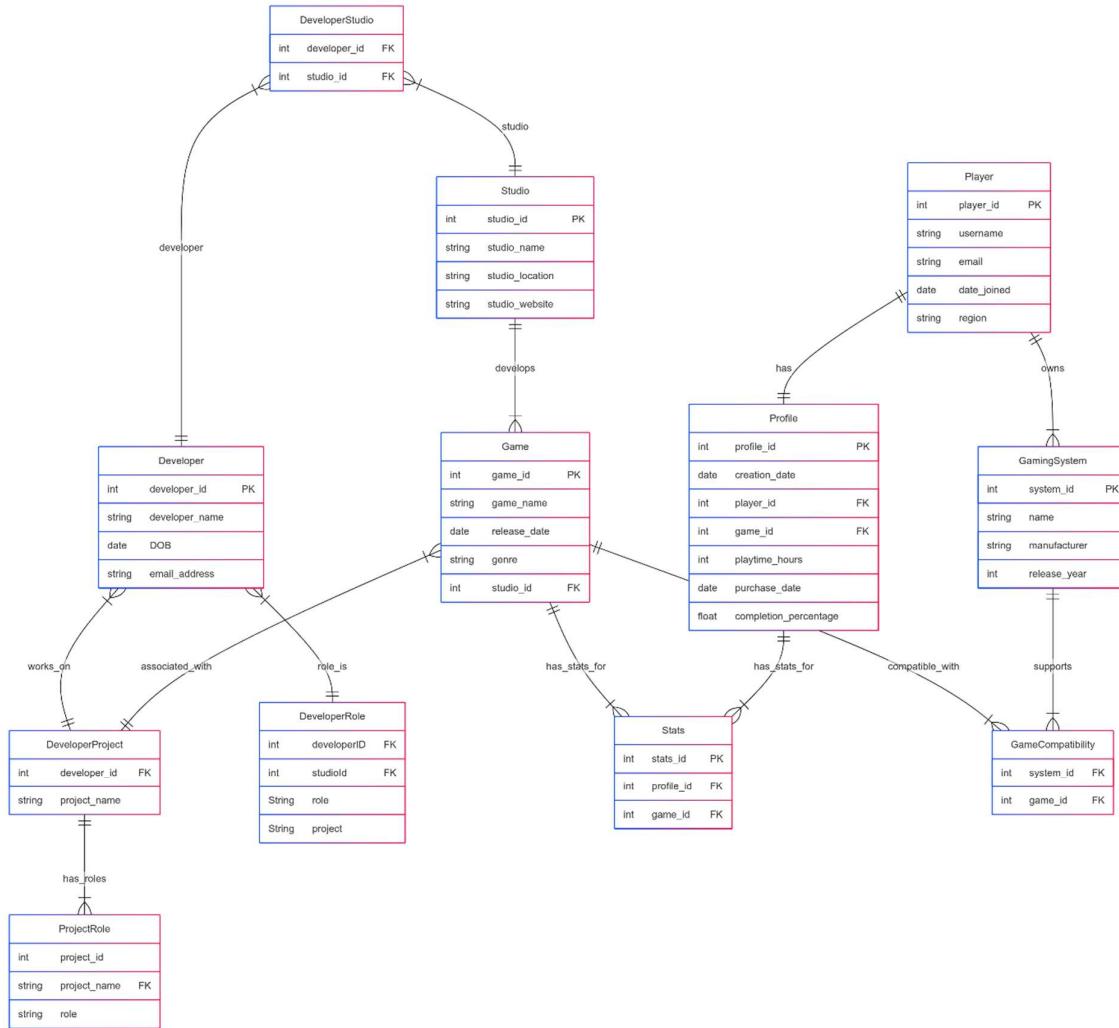
GROUP 3

ROMAN KOROLOVYCH GVIDONAS BUIKYS

STAGE 3 – LAB REPORT

GAMING ECOSYSTEM – GVIDONAS & ROMAN

SECTION 1: FINAL ER DIAGRAM



SECTION 2: 10 QUERIES

QUERY 1: GET PLAYERS WITH LESS THAN 50 HOURS OF PLAYTIME

SQL STATEMENT

SELECT

```
player.ID,  
player.Username,  
profile.playtime_hours
```

FROM

```
player
```

JOIN

```
profile ON player.ID = profile.player_id
```

WHERE

```
profile.playtime_hours < 50;
```

EXPLANATION

This SQL Query retrieves a list of players who have played for less than 50 hours. It does this by:

- Selecting the player's ID and Username from the player table.
- Joining the player table with the profile table using a JOIN on the matching ID from player and player ID from profile.
- Filtering the results with a WHERE clause so that only records where playtime hours is less than 50 are returned.

The result shows which users are less active based on their total hours played.

QUERY 2: RETRIEVES THE TOTAL NUMBER OF CONSOLES FOR EACH MANUFACTURER

SQL STATEMENT

```
SELECT manufacturer, COUNT(name) AS total_consoles  
FROM gaming_system  
GROUP BY manufacturer  
ORDER BY total_consoles DESC;
```

EXPLANATION:

This SQL query shows how many gaming consoles each manufacturer has produced, sorted from the highest to the lowest.

- It selects the manufacturer name from the gaming system table.
- It counts the number of entries in the name column(i.e, different console models) for each manufacturer using COUNT(Name).
- The GROUP by manufacturer groups the results so that the count is calculated per manufacturer.
- The ORDER by total consoles DESC sorts the output in descending order, so the manufacturer with the most console models appears first.

This is useful for identifying which manufacturers are most active or diverse in their gaming system production.

QUERY 3: RETRIEVE THE LATEST GAME FOR EACH GENRE

SQL STATEMENT

```
SELECT game_name, release_date, genre
FROM game g
WHERE release_date =
    (SELECT MAX(release_date)
     FROM game
     WHERE genre = g.genre
    )
ORDER BY genre;
```

EXPLANATION:

This query retrieves the most recently released game in each genre from the game table

- It selects the game name, release date, and genre from the game table (aliased as g)
- The WHERE clause uses a subquery to find the maximum (latest) release date for each genre.
- The subquery compares each row's release date with the maximum date for that specific genre (WHERE genre = g.genre), ensuring only the latest game per genre is returned.
- The results are then ordered by genre for easier readability and grouping.

This is helpful when you want to highlight the newest game in every genre category.

QUERY 4: RETRIEVES THE TOP 5 PLAYERS WITH THE HIGHEST AVERAGE GAME COMPLETION PERCENTAGE

SQL STATEMENT

SELECT

```
player.ID,  
player.Username,  
ROUND(AVG(profile.completion_percentage), 2) AS  
avg_completion  
FROM player  
JOIN profile ON player.ID = profile.player_id  
GROUP BY player.ID, player.Username  
ORDER BY avg_completion DESC  
LIMIT 5;
```

EXPLANATION:

This query displays the top 5 players with the highest average game completion percentage.

- It selects the ID and Username from the player table.
- It JOINS the player table with the profile table using the player's ID
- It calculates the average completion percentage for each player using AVG(profile completion percentage) and rounds it to 2 decimal places with ROUND(...,2)
- This data is grouped by player ID and username to ensure the average is calculated per player
- It orders the results in descending order based on average completion so that the top players appears first.
- The LIMIT 5 ensures that only top 5 players are shown.

This query is useful for leaderboard-style statistics based on how thoroughly players complete their games.

QUERY 5: RANK SONY CONSOLES RELEASED BETWEEN 2015 AND 2023 BY LATEST RELEASE

SQL STATEMENT

```
SELECT  
    name,  
    manufacturer,  
    release_year,  
    RANK() OVER (ORDER BY release_year DESC) AS release_rank  
FROM gaming_system  
WHERE manufacturer = 'Sony'  
AND release_year BETWEEN 2015 AND 2023;
```

EXPLANATION:

This query lists Sony gaming systems released between 2015 and 2023, ranked by their release year from newest to oldest.

- It selects the name, manufacturer, and release year from the gaming system table.
- The WHERE clause filters the results to only include systems made by 'Sony' and released between 2015 and 2023.
- It uses the RANK() window function with OVER(ORDER By release DESC) to assign rank to each console based on how recently it was released – newer consoles get a higher rank (i.e, lower number).
- The result includes a release rank column showing this order

This helpful for seeing Sony's console release timeline and comparing the relative order of their newer systems.

QUERY 6: SHOW PLAYERS WHOSE PLAYTIME IS GREATER THAN THE AVERAGE PLAYTIME

SQL STATEMENT

```
SELECT player.Username, SUM(profile.playtime_hours) AS  
total_playtime  
  
FROM player  
  
JOIN profile ON player.ID = profile.player_id  
  
GROUP BY player.Username  
  
HAVING total_playtime > (  
  
    SELECT AVG(playtime_hours) FROM profile  
  
);
```

EXPLANATION:

This query shows the usernames of players whose total playtime is above the average playtime across all profiles.

- It joins the player and profile tables on the player's ID
- It groups the results by player username and calculates the total playtime for each player using `SUM(profile playtime hours)`
- The HAVING clause filters out players whose total playtime is less than or equal to the average. The average is computed with a subquery: `SELECT AVG(playtime hours) FROM profile`
- Only those players whose total playtime is greater than the overall average are included in the final result.

This is useful for identifying the most active or engaged players based on total hours played.

QUERY 7: CLASSIFY PLAYERS BASED ON THEIR TOTAL PLAYTIME HOURS

SQL STATEMENT

```
SELECT
```

```
    Player.Username,
```

```
    SUM(Profile.playtime_hours) AS total_playtime,
```

```
    CASE
```

```
        WHEN SUM(Profile.playtime_hours) > 100 THEN 'Hardcore'
```

```
        WHEN SUM(Profile.playtime_hours) > 50 THEN 'Regular'
```

```
        WHEN SUM(Profile.playtime_hours) > 10 THEN 'Casual'
```

```
        ELSE 'Newbie'
```

```
    END AS player_category
```

```
FROM Player
```

```
JOIN Profile ON Player.ID = Profile.player_id
```

```
GROUP BY Player.Username
```

```
ORDER BY total_playtime DESC;
```

EXPLANATION:

This query classifies players based on their total playtime into categories like Hardcore, Regular, Casual or Newbie.

- It joins player and profile tables using the player ID
- It calculates the total playtime for each player by summing profile playtime hours using SUM
- It uses CASE statement to assign a player category:
 1. Hardcore for playtime over 100 hours
 2. Regular for over 50 hours
 3. Casual for over 10 hours
 4. Newbie for 10 or fewer hours
- The results are grouped by player username so each player appears once.

- Finally, the list is sorted by total playtime in descending order so the most active players are listed first.

This is great for generating a simple engagement-based player segmentation.

QUERY 8: FIND ALL DEVELOPERS THAT HAVE DIRECTOR IN THEIR ROLE

SQL STATEMENT

```

SELECT DISTINCT

    Developer.developer_name,

    Project_Role.Role

FROM

    Developer

JOIN

    Developer_Project ON Developer.developer_id =
Developer_Project.Developer_ID

JOIN

    Project_Role ON Developer_Project.Project = Project_Role.Project

WHERE

    Project_Role.Role LIKE '%Director%'

ORDER BY

    Developer.developer_name;

```

EXPLANATION:

This query lists all distinct developers who held a role containing the word “Director” on any project.

- It performs a series of joins between the Developer, developer project, and project role tables to link developers to their roles in projects

- The WHERE clause filters for roles that include the term “Director” using LIKE ‘%Director%’(e.g. Game Director, Art Director).
- SELECT DISTINCT ensures that each developer-role combination appears only once on the results.
- The output includes the developer name and their matching role.
- The final result is sorted alphabetically by developer name

This query is useful for identifying key leadership contributors across development projects.

QUERY 9: GET THE BEST PERFORMING DEVELOPERS BASED ON THEIR GAME RATING WHICH IS CALCULATED BASED ON AVERAGE COMPLETION AND PLAYTIME

SQL STATEMENT

```
WITH GameRatings AS (
  SELECT
    Game.game_id,
    game_name,
    ROUND(AVG(completion_percentage),2) AS avg_completion,
    ROUND(AVG(playtime_hours),2) AS avg_playtime,
    ROUND(AVG((playtime_hours+completion_percentage)/2),2)
  AS game_rating
  FROM
    Profile
  JOIN
    Game ON Profile.game_id = Game.game_id
  GROUP BY
    Game.game_id, game_name
```

)

SELECT

 Developer.developer_name,
 Studio.studio_name,
 GameRatings.game_name,
 GameRatings.avg_completion,
 GameRatings.avg_playtime,
 GameRatings.game_rating

FROM

 Developer

JOIN

 Developer_Studio ON Developer.developer_id =
 Developer_Studio.Developer_ID

JOIN

 Studio ON Developer_Studio.studio_id = Studio.studio_id

JOIN

 Game ON Studio.studio_id = Game.studio_id

JOIN

 GameRatings ON Game.game_id = GameRatings.game_id

ORDER BY

 GameRatings.game_rating DESC

EXPLANATION:

This SQL query provides a ranked list of games developed by specific studios, including their average completion, playtime, and a calculated game rating, along with the developers and studios behind them.

A common table expression (CTE) named Game ratings is created to:

1. Calculate the average completion percentage, average playtime, and custom game rating (average of completion and playtime) for each game.
2. The data comes from joining profile and game tables
3. Results are grouped by each genre (game ID, game name) and rounded to 2 decimal places for readability.

Final Select

This part retrieves:

- The developer name, studio name, all fields from Game ratings.
- It joins multiple tables (Developer, Developer studio, studio, game, and game ratings) to associate developers with studios and their games.
- Finally, it orders the results by game rating in descending order so the top-rated games appear first.

This query is powerful for performance analysis, allowing you to see which developers and studios are producing the most engaging games based on aggregated player data.

QUERY 10: PURPOSE: CREATE A BASIC VIEW OF STUDIO GAME STATISTICS,

SQL STATEMENT

```
CREATE VIEW Studio_Game_Stats AS
```

```
SELECT
```

```
    Studio.studio_id,
```

```
    Studio.studio_name,
```

```
COUNT(Game.game_id) AS number_of_games,  
MIN(Game.release_date) AS first_release,  
MAX(Game.release_date) AS latest_release  
FROM  
Studio  
LEFT JOIN  
Game ON Studio.studio_id = Game.studio_id  
GROUP BY  
Studio.studio_id, Studio.studio_name;  
  
SELECT  
studio_name,  
number_of_games,  
first_release,  
latest_release,  
YEAR(CURRENT_DATE()) - YEAR(first_release) AS years_active  
FROM  
Studio_Game_Stats  
WHERE  
number_of_games >= 2  
AND latest_release >= '2022-01-01'  
ORDER BY  
number_of_games DESC;
```

EXPLANATION:

Part 1 – Create VIEW studio game stats:

This view summarizes basic game release statistics per studio

- It joins the studio and Game tables (using a LEFT JOIN to include studios that may have no games)
- For each studio, it calculates:
 1. The total number of games (COUNT(Game.Game ID))
 2. The first release date (MIN(...))
 3. The last release date (MAX(...))
- The results are grouped by the studios ID and name

Part 2 – Final SELECT from Studios Game stats

This query filters and present key data for studios with at least 2 games and a recent release (2022 or later).

- It selects each studios name, number of games, first and latest release dates.
- It also calculates how long the studio has been active by subtracting the year of their first release from the current year: YEAR(CURRENT DATE)- YEAR(FIRST RELEASE)AS years active)
- The results are sorted by the number of games, showing the most prolific studios first.

This is great for spotting active and long-standing studios with recent output.

SCREENSHOTS OF EACH QUERY WORKING IN JAVA

Query 1

PLAYERS WITH <50 HOURS PLAYTIME				
ID	Username	total_playtime	Date_Joined	Region
14	NoobMaster	42	2024-03-20	EU
6	DragonSlayer	18	2023-07-22	AS
7	PixelHunter	47	2023-08-14	NA
9	SpeedRunner99	28	2023-10-22	AS

Query 2

CONSOLES BY MANUFACTURER	
manufacturer	total_consoles
Sony	6
Nintendo	4
Microsoft	4
Valve	1

Query 3

LATEST GAME PER GENRE			
game_id	game_name	release_date	genre
14	Marvel's Spider-Man	2023-10-20	Action-Adventure
6	Starfield	2023-09-06	Action-RPG
5	The Legend of Zelda: Breath of the Wild	2017-03-03	Adventure
11	Psychonauts 2	2021-08-25	Platformer
15	Mass Effect Legendary Edition	2021-05-14	RPG
8	Half-Life: Alyx	2020-03-23	VR Shooter

Query 4

TOP 5 PLAYERS BY COMPLETION %				
ID	Username	avg_completion	Date_Joined	Region
11	MagicMike	100.0	2023-12-15	EU
13	EpicGamer	95.2	2024-02-15	NA
8	NecroQueen	92.1	2023-09-05	EU
12	SniperWolf	87.4	2024-01-10	AS
2	MageFire	86.3	2023-02-20	EU

Query 5

RANKED SONY CONSOLES			
name	manufacturer	Release_year	release_rank
PlayStation VR2	Sony	2023	1
PlayStation 5	Sony	2020	2
PlayStation VR	Sony	2016	3

Query 6

PLAYERS ABOVE AVERAGE PLAYTIME				
ID	Username	total_playtime	Date_Joined	Region
1	Shadow Ninja	113	2023-01-15	NA
11	MagicMike	112	2023-12-15	EU
12	SniperWolf	89	2024-01-10	AS
13	EpicGamer	145	2024-02-15	NA
15	ProGamer123	76	2024-04-25	AS
2	MageFire	175	2023-02-20	EU
3	ArcherQueen	94	2023-03-10	AS
4	TankLord	122	2023-04-05	NA
5	Stealthy	124	2023-05-12	EU

Query 7

PLAYER CLASSIFICATION		
Username	total_playtime	player_category
MageFire	175	Hardcore
EpicGamer	145	Hardcore
Stealthy	124	Hardcore
TankLord	122	Hardcore
Shadow Ninja	113	Hardcore
MagicMike	112	Hardcore
ArcherQueen	94	Regular
SniperWolf	89	Regular
ProGamer123	76	Regular
NecroQueen	62	Regular
LootGoblin	56	Regular
PixelHunter	47	Casual
NoobMaster	42	Casual
SpeedRunner99	28	Casual
DragonSlayer	18	Casual

Query 8

DEVELOPERS WITH DIRECTOR ROLES			
developer_id	developer_name	Role	Project
11	Cliff Bleszinski	Design Director	Gears of War
3	Hideki Kamiya	Game Director	Bayonetta
1	Hideo Kojima	Director	Death Stranding
13	Ken Levine	Creative Director	BioShock
9	Neil Druckmann	Creative Director	The Last of Us
5	Todd Howard	Game Director	The Elder Scrolls V: Skyrim
15	Warren Spector	Director	Deus Ex
4	Yoko Taro	Creative Director	Nier: Automata

Query 9

DEVELOPER PERFORMANCE RATINGS					
developer_name	studio_name	game_name	avg_completion	avg_playtime	game_rating
Amy Hennig	Naughty Dog	The Last of Us Part II	92.1	62.0	77.05
Neil Druckmann	Naughty Dog	The Last of Us Part II	92.1	62.0	77.05
Todd Howard	Bethesda Game Studios	Starfield	43.45	88.5	65.98
Cliff Bleszinski	BioWare	Mass Effect Legendary Edition	53.1	76.0	64.55
Ken Levine	BioWare	Mass Effect Legendary Edition	53.1	76.0	64.55
Warren Spector	BioWare	Mass Effect Legendary Edition	53.1	76.0	64.55
Amy Hennig	Naughty Dog	Uncharted 4: A Thief's End	78.3	47.0	62.65
Neil Druckmann	Naughty Dog	Uncharted 4: A Thief's End	78.3	47.0	62.65
Hideki Kamiya	PlatinumGames	Nier: Automata	68.9	56.0	62.45
Yoko Taro	PlatinumGames	Nier: Automata	68.9	56.0	62.45
Shigeru Miyamoto	Nintendo EPD	The Legend of Zelda: Breath of the Wild	21.0		53.0
Tim Schafer	Double Fine	Psychonauts 2	35.7	28.0	31.85
John Carmack	Valve	Half-Life: Alyx	12.5	18.0	15.25
Gabe Newell	Valve	Half-Life: Alyx	12.5	18.0	15.25

Query 10

ACTIVE STUDIO STATISTICS				
studio_name	number_of_games	first_release	latest_release	years_active
Naughty Dog	2	2016-05-10	2020-06-19	9
CD Projekt Red	2	2015-05-19	2020-12-10	10

SECTION 3: RESPONSIBILITIES

Task / Component	Gvidonas Buikys (D00280972)	Roman Korolovych (D0080847)
SQL Schema	✓	
Queries 1–5	✓	
Queries 6–10		✓
Java – DBConnect		✓
Java – DBCommand		✓
Java – OutputFormatter		✓
Java – Main		✓
ER Diagram (Mermaid)	✓	
PDF Section 1 & 2	✓	
PDF Compilation & Formatting	✓	
Testing – SQLite	✓	

Task / Component	Gvidonas Buikys (D00280972)	Roman Korolovych (D0080847)
Testing – Java Code		