Dhirubhai Ambani
Institute of Information and Communication Technology

DBMS - IT214

Team Id- 3.13 G4 S3

Case Study :
Digital Video Game Distribution Database

https://github.com/DivCode4664/Video-Game-Distribution-Project.git

Sr No.	Name	Student ID
1.	Dhruv Prajapati	202001219
2.	Divyesh Baravaliya	202001229

INDEX

Section 1: Final version of SRS

1. Introduction
   a. Purpose
   b. Intended audience and reading suggestion
   c. Product scope
   d. Description
2. Document the Requirements Collection/ Fact Finding Phase
   a. Description of reading and References
   b. Interviews
   c. Questionnaires
   d. Observations
3. Fact Finding Chart
4. List requirements
5. User classes and Privileges
6. Assumptions
7. Business Constraints

1. Introduction

a. Purpose

- The purpose of this study is to obtain a database design for a distribution simulation game that will be used as a learning medium in industrial engineering.

- Purpose of this SRS document is to provide all the requirements to the developers to manage all the activities on the distribution platform.

- It will provide details about video games that users have bought, their achievements, connectivity of multiple users, complaint requests from the users, transaction history of in-game purchases, keep track of user's progress, gaming subscriptions etc.

- The purpose of this system is to eliminate traditional hardcopy game sale systems, combine game developers and players under one platform, making games easier to distribute. At the same time, it offers a market for independent developers, where they can easily show themselves.

b. Intended Audience and Reading Suggestions

- The SRS document is intended for ,
  Gamers : Gamers can get all the information about new games and can purchase a new game.
  Game publishers : They can publish their new games,
  Developers : They can get data about users reviews so that they can improve their work,
  Distributors and Retailers : They can get all the data about distribution of the games,
  Marketing staff : They can get data about the popularity of the game.

- It is also geared toward younger users who enjoy playing video games.

- The SRS document's remainder is devoted to describing the product's scope, which outlines the kinds of data that distributors, publishers and the other audience may obtain, among other things.

- An overview of the database that will connect all the relevant data.

c. Product Scope

This software is intended to provide a relevant platform to users for buying new games and provide relevant data to the developers.

- Goal of the product is to provide relevant data of the game distribution to the distribution staff of the company.

- The basic concept is to create a database where various users may look for and buy games while also rating and reviewing them. Furthermore, a messaging platform will be made available for users of the system to communicate with one another. They will be able to talk with one another either privately or in groups as a result.

- Users can download any game if and only if that game is supported for the user's PC.

- This system is built with keeping normal problems like assigning games to the customers, checking balance in the debit card before buying the game, and checking the required age of the customer for any particular game.

- The buyers can see the prices of the game and discount by using this database.

- Retailers can see the sales and revenue of the different digital copies of games.

- If a user likes a particular genre then the database would recommend the game of that genre.

- Users can complain and give reviews about games but they are restricted to change or update the reviews of the other users. They can only reply and can like or dislike the comment.

- Using this database, retailers, distributors, and wholesalers can get information about the games' final stock, order quantity, and backlog.

- Developers can analyze game outcomes and performance by using the main entities but they are restricted to change or update any reviews or other information about the game.

- The users are restricted to download the game if the software requirements of their PC is not sufficient.

d. Description

- This document serves as a guide for developers for adding new features and improving performance. By using this database developers get information about users reviews about the game so that they can update all the requirements of the game.

- Any game has a subscription feature so we have to collect all the data about the user's subscriptions using that data. This System can send a message to users whose subscriptions have expired by using information about user subscriptions data.

- Information about users and games will be stored in a database. By using queries, the user of this database will be able to look up data on both people and games. The webpage interface, for instance, allows the visitor to choose to view a list of games that fit into a specific category. As a result, a query will be used to project only the games that fall within the category, in alphabetical order. The user may optionally want to order the list of games based on other characteristics that the "Game" entity must have, such as popularity, which is determined by how many people own the game, or game difficulty.

- System can restrict game use based on the user's age data.

- Users can give the review about the game and also give their ratings of the game, so based on the popularity of games, this database will be helpful to the marketing team by statistics for marketing.

- This system will collect all the data about the users and their friends so by using this system users can get to know about new friends and share their rewards.

- This information can be used to forecast the growth of esports and suggest appropriate market investments for generating income.

- It would allow users to interact with the database securely and it would maintain confidentiality of the user's information.

2. Fact-Finding Phase

a. Description of reading and References :

- **Reference of Digital video game distribution from** https://cs-agents.com/blog/steam/
  From this blog we understood about the gaming distribution platform - **Steam**. We learnt about the services provided by the Steam platform.The Valve Corporation developed Steam, a digital platform for the distribution of video games. Players and game producers can purchase and sell video games online in this gaming community.

- We learnt about, **Database** related to gaming Distribution from : https://steamdb.info/
  By this website we understood about entities and relations which will be used to create gaming distribution platforms like, Steam.

- HOW CAN GAME DEVELOPERS LEVERAGE DATA FROM ONLINE DISTRIBUTION PLATFORMS? **A CASE STUDY OF THE STEAM PLATFORM**
  https://sail.cs.queensu.ca/data/pdfs/PhD_Thesis_Defense_Dayi.pdf
  This document is focused on mining online distribution platforms like Steam. This document is based on the following four aspects of online game distribution: user reviews, user-recorded gameplay videos on the Steam platform, urgent updates, and the early access model (which lets game developers sell unfinished versions of their games). So we learnt how these factors affect the online distribution platform.We learnt how important reviews are to developers and how they take them into account. It suggests we can add a feature for gamers to share and discuss their Youtube gameplay video, In which gamers have covered several topics like game tutorial, setup, game-bugs, game play-thoughts.

- **The Business Model of Steam Because steam is much more than play online**
  https://www.dsi.unive.it/~marek/files/seminari/Andrea%20Pretotto/Steam%20Relazione.pdf
  This document contains the information about what are the success points of the largest gaming distribution platform Steam. It shows ecommerce strategy , consumer behavior and satisfaction, payments, security and how these factors are important for a reliable distribution platform. It shows that for advertising

- **List the combined Requirements gathered from Background Reading/s.**

- A well functioning Database management system is required to maintain and update all the information about, all about games from publication to use, customer support, user's connectivity and their achievements.

- A user interface is required so that the customers can easily access the data required without having to know much about the actual implementation of the system.

- All the basic functionality that is expected to be in a gaming distribution system.
  E.g. Keeping a record of game purchases, player's statistics, popularity of the game, reviews etc…

- Security is a very important factor, System requires an authorization feature by this feature. If anyone attempts to login the user's account from an unrecognized computer then it must provide additional authorization.
  The access to user's games is based on a customer account, not tied to a computer. We can do this by custom executable generation.

- To maintain contact between the user and friend system requires a special feature like Vocal chat.

- System structure and functions should be designed in such a way that it ensures the fast performance of the system.

- To increase user interaction and interest in the game, the system must notify users as soon as a new update is available.

- Reviews are crucial for developers, so this system ought to take into account every user comment.

- The interface should be clean and without any redundant data.

b. Interview/s:

1. Interview Plan :

System : Flash (Video Game distribution platform)

Interviewee : Aniket Roy(Role play)
Designation : Developer at GamersGate (Video Game distribution platform)

Interviewer : 1) Dhruv Prajapati
Designation : Business Development Executive
2) Divyesh Baravaliya
Designation : Developer

Date : 25 / 09 / 2022 **Time** : 4:30 PM **Place :** Interview Office

Duration : 60 minutes

Purpose of Interview :

Preliminary meeting to identify problems and requirements regarding developing new features of customer support at the GamersGate.

Agenda :

- Introduction and current status

- Initial ideas and follow-up action regarding the developing new features for customer support based on customer reviews.
- Discussion on the setup a good interference of the customer support section, so that they don't have to do more searches to fulfill their need and queries.

Documents to brought at interview :

- Rough plan of changing customer support setup.
- Any document report relating to the current customer reviews and current needs of the customers.

- ## Interview summary :

System : Flash (Video Game distribution platform)

Interviewee :

Aniket Roy(Role play)
Designation : Developer at GamersGate (Video Game distribution platform)

Interviewer :

1) Dhruv Prajapati
Designation : Business Development Executive

2) Divyesh Baravaliya
Designation : Developer

Date : 25 / 09 / 2022 **Time** : 4:30 PM **Place :** Interview Office

Duration : 60 minutes

Purpose of Interview :

Preliminary meeting to identify problems and requirements regarding developing new features of customer support at the GamersGate.

Results :

- Customers can add their queries and complaints with the snapshot of the game.
- Allowing players to report bugs that they encounter while playing is a common practice to ensure the user-perceived quality of a game.
  By this Bug-reporting feature sellers can track most frequent crashes in real time and this is before customer support contact.
- Separate section for statistics of customer's solved queries, so that it can build trust for the customers.
- Regular feedback from customers will be taken and improvements will be done to the system accordingly.
- Customers can like and dislike the comments and reviews given by other customers and replies given by a developer.
- Due to several demands, a new privacy function was created, allowing users to conceal their statistics.

2. Interview Plan :

System : Flash (Video Game distribution platform)

Interviewee : Shikha Singh (Role play)
Designation : Developer at GamersGate (Video Game distribution platform)

Interviewer : 1) Dhruv Prajapati
Designation : Business Development Executive
2) Divyesh Baravaliya
Designation : Developer

Date : 27 / 09 / 2022 **Time** : 9 AM **Place :** Interview Office

Duration : 120 minutes

Purpose of Interview :

Preliminary meeting to discuss adding a new feature related animation of game play and displaying new updates.

Agenda :

- Discussion on current status
- Initial ideas and follow-up action regarding the developing new features for adding animated videos and separate a section for display about new updates of the game.

Documents to brought at interview :

- Digital video copy of any game play as an example.
- Document related new features of animation which are new in the market.

• Interview summary :

System : Flash (Video Game distribution platform)

Interviewee :

Sikha Singh(Role play)
Designation : Developer at GamersGate (Video Game distribution platform)

Interviewer :

1) Dhruv Prajapati
Designation : Business Development Executive

2) Divyesh Baravaliya
Designation : Developer

Date : 27 / 09 / 2022 **Time** : 9 AM **Place :** Interview Office

Duration : 120 minutes

Purpose of Interview :
Preliminary meeting to discuss adding a new feature related animation of game play and displaying new updates.

Results :
- Discussion on adding a new function that would automatically play an animated video of the game that is visible on the display when a customer accesses a website. Customers will be drawn to the game by this automatic function.
- Adding a new feature What's new which will display the new update of the game.
- Customers will be able to see the new characters and new offers of the game.
- We will increase the performance of the system by efficient implementation and a well structured system.

3. Interview Plan :

System : Flash (Video Game distribution platform)

Interviewee : Fenil Mansara (Role play)
Designation : Expert Gamer on Youtube platform (User)

Interviewer : 1) Dhruv Prajapati
Designation : Business Development Executive
2) Divyesh Baravaliya
Designation : Developer

Date : 29 / 09 / 2022 **Time** : 9 AM **Place :** Live Youtube session

Duration : 90 minutes

Purpose of Interview :
Preliminary meeting to discuss on the experience, reviews and suggestions for the distribution platform (**Flash**)

Agenda :

- Discussion on current status
- Initial ideas and follow-up action regarding the developing new features based on user (Expert in Gaming) experience.
- Consider the reviews of the other users from the live youtube session.

● Interview summary :

System : Flash (Video Game distribution platform)

Interviewee :

Fenil Manasara (Role play)
Designation : Expert Gamer on Youtube platform (User)

Interviewer :

1) Dhruv Prajapati
Designation : Business Development Executive

2) Divyesh Baravaliya
Designation : Developer

Date : 29 / 09 / 2022 **Time** : 9 AM **Place :** Live Youtube session

Duration : 90 minutes

Purpose of Interview :

Preliminary meeting to discuss on the experience, reviews and suggestions for the distribution platform (**Flash**)

Results :

- Gamers are able to share and discuss their youtube game play videos, In which gamers have covered several topics like game tutorial, setup, game-bugs, game play-thoughts. It will also help to increase advertisement and sales of the games.

- **Combined Requirements gathered from all Interview/s.**

- We need to add a feature that can complete an analysis of a specific game and generate a separate report for the game's developer in order to improve customer support.

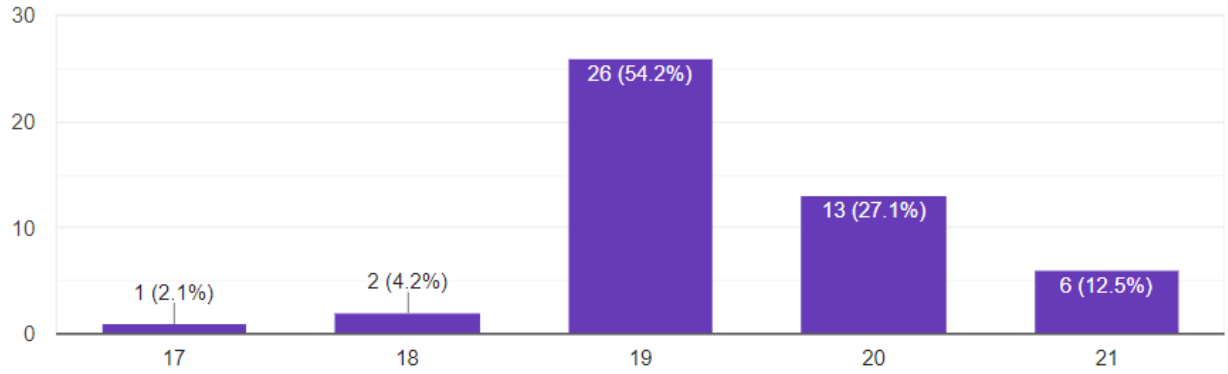
- System requires a separate section based on Update available, which will show the list of installed games on the user's PC in which the update is available so that the user does not have to look for each game individually.

- Youtube will become a good platform for advertising and promotion for any game so we will try to add a new feature related to youtube so that youtubers and influencers can promote the game which can improve rating and sales of the game.

c. Questionnaire/s -

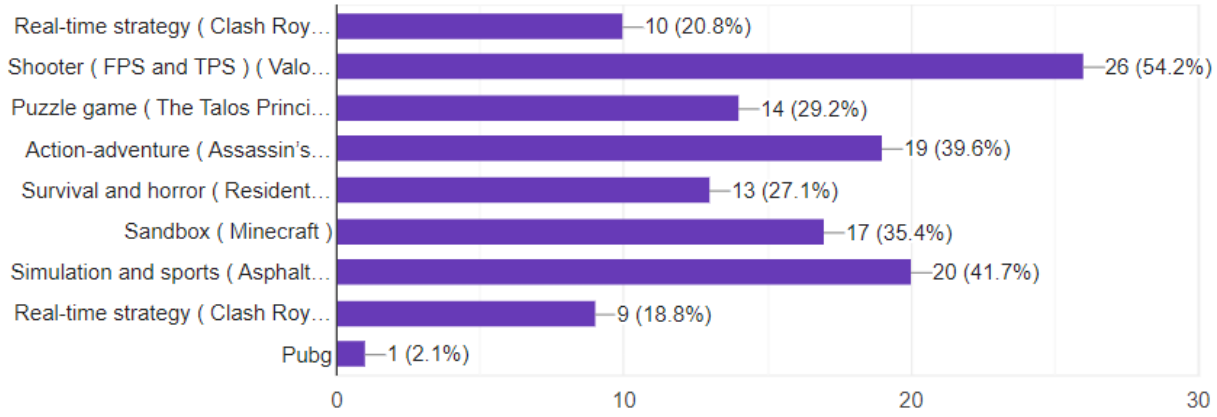
- Here is the Q/A screenshot of google form

Age?



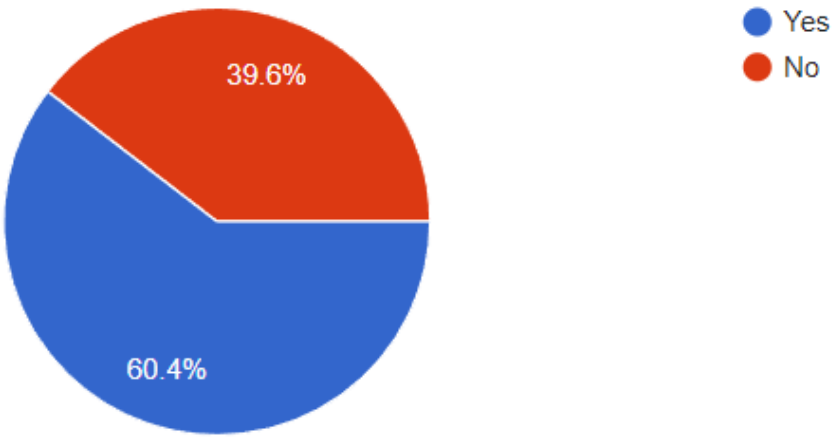
Intent - To get to know what the age group of the customers/players is and set the games and the software User Interface accordingly.
Observation - Video games are primarily played by young people.

What kind of video game genre do you like?



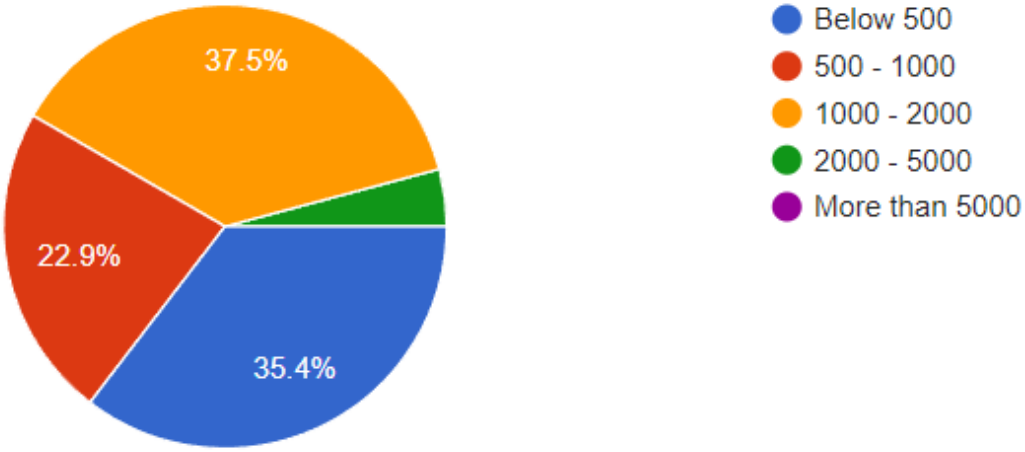
Intent - To get details about the popularity of the genre of the game by which we can add the most popular genre in the suggested games section.
Observation - Most gamers like to play shooter games, Simulation and sports and Action-adventure.

Have you ever bought a video game from the online game store?



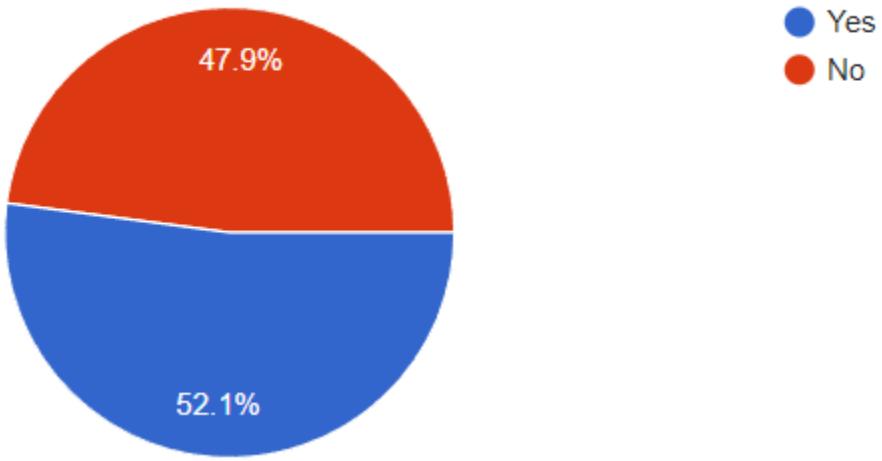
Intent - To get information about the interest of users to buy a new video game.
Observation - Users are more interested in buying games online than offline.

How much money do you typically spend on new video games?



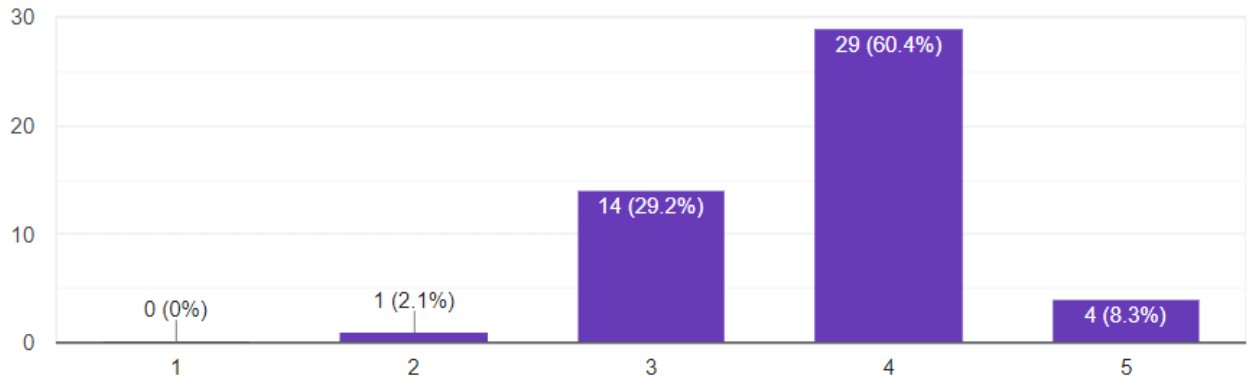
Intent - Are users interested in spending money(more/less) in game which gives more benefits to users
Observation - Most of them spend less money , only a fewer prefers spending more money

Do you ever experience dissatisfaction after buying a new game based on its presentation and user reviews on the distribution platform?



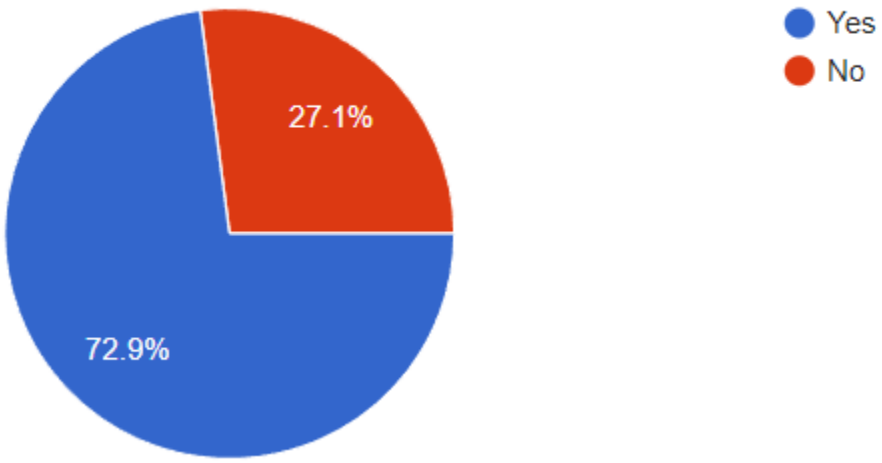
Intent - To get to know that publishers and developers are providing relevant content to the users based on the presentation of the game.
Observation - Users are not fully satisfied with the performance of the games on current distribution platforms.

What is your opinion of the video game distribution platform from which you purchased it?



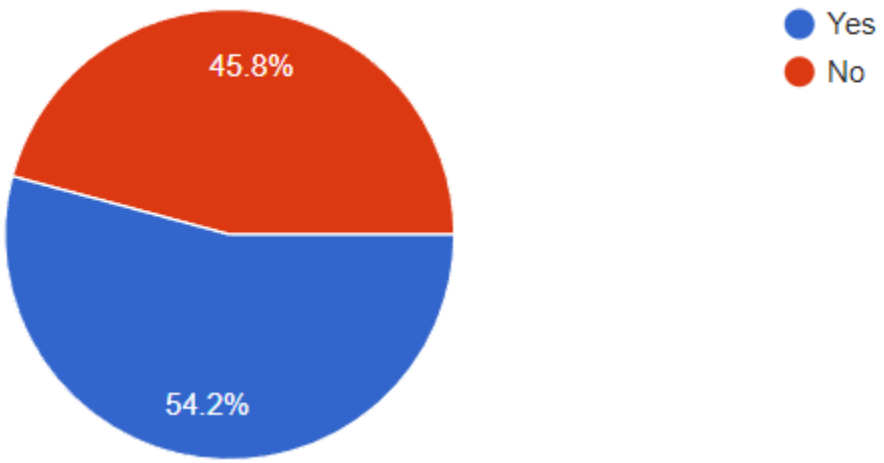
Intent - To know about the quality of the services offered
Observation - opinions are favored to be above average. Users are happy with the service of the platform.

Does the customer service on your game distribution platform meet your standards?



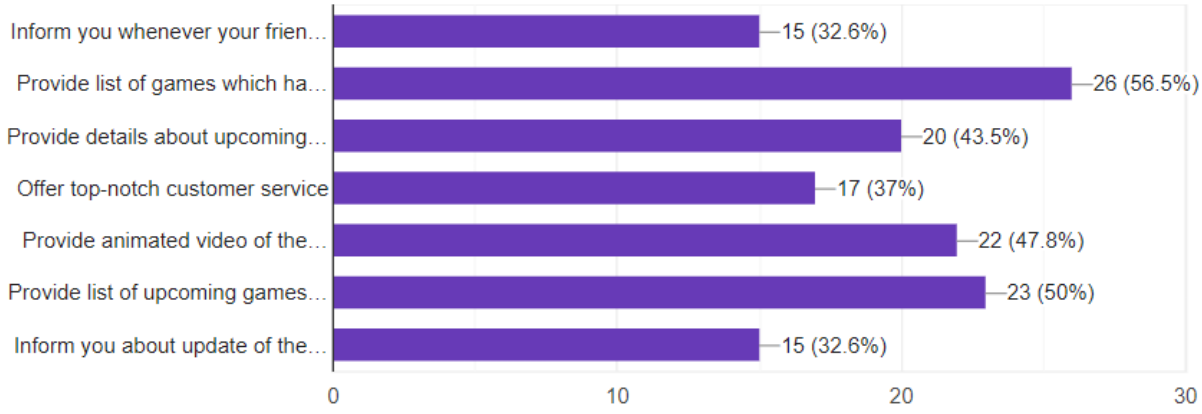
Intent - To get to know about satisfaction of the users towards the distribution platform
Observation - Most users are satisfied with the service provided by the distribution platform.

Have you ever given a review about any video game on the distribution platform?



Intent - To get to know how many users are interested in giving reviews about the games.
Observation - Most Users are interested in giving reviews about the games.

What do you anticipate from a reliable distribution platform?



Intent - To get to know about the needs of the users towards the distribution platform.
Observation - Users are more interested in Limited-time Offer and users are demanding for good customer service, animated video of game play.

- **Combined Requirements gathered from Response/s.**

- Most users are willing to play the Shooter, Action-adventure genre of the game so by this information we can add the most popular genre in the suggested games section.

- The "Editors' Choice" section, which provides information on games ranked by their effectiveness and quality as recommended by experts, could be added.

- Users are willing to include a Section in exchange for a limited-time offer on quality games, so the system requires that section and we can add a pop-up window which will showcase best offers on trending games.

- Some Users are not interested in spending more money on video games so for that we can add a filter feature based on the price of the game.

- Users are also interested in pre-registering for new games, so we can add a feature for that based on their expectations for the platform.

d. Observation(s) :

IT solutions : Observation

System : GamersGate

Project reference :

Observations by : Dhruv Prajapati

Date : 30 / 09 / 2022

Duration : 45 minutes **Place** : GamersGate

Observations :

- Users should receive daily updates regarding installed and paid games.
- The system's workflow should be efficient to prevent users from becoming trapped in any procedure.
- Since privacy and security of the data were given priority, not all data was accessible to everyone.
- User’s positive and negative comments are important for developers.
- The game's presentation plays a key role in drawing users.
- Communities that are distinct for each game might improve user connectivity.
- Users are interested in discounts for first purchase and on the special events.

● **Requirements from observations** :

- The system should optimize the workload of the server so that customers can receive their service fast.
- To guarantee that consumers have a positive experience, the registration process is simple and the user interface is effective.
- To link new communities for certain games on that platform, a system should be connected to another system that is only based on user connectivity.
- For the games, there should be a pop-up window offering exclusive discounts on initial purchases and during special events.
- The system ought to accommodate as many versions as possible.
- Rules and regulation for any game should be provided in the proper way.
- Different user classes should have varying levels of access and system update permission.

3. Fact-Finding chart :

Objective	Technique	Subject	Time commitment
To get background knowledge on the distribution website	Background readings	Blog and similar project on distribution system	0.5 day
Meeting to identify problems and requirements regarding customer support at GamersGate	Interview	Developer at GamerGate	60 minutes
To identify requirements regarding new updates and video of the game play	Interview	Developer at GamerGate	120 minutes
To identify requirements of the expert gamers of Youtube platform	Online live session	Expert Gamer on Youtube	90 minutes
To get knowledge of the real world Gaming distribution platform	Observations	Customers	90 minutes
To remove bugs identified by users	Announcement for give away to identify bugs	Users	1 week
To understand the user's perspective	Questionnaire	Student developers	1 day

4. List of Requirements :

- A well functioning Database management system is required to maintain and update all the information about, all about games from publication to use, customer support, user's connectivity and their achievements.
  The interface should be clean and without any redundant data.
  (**Background Readings**)

- A user interface is required so that the customers can easily access the data required without having to know much about the actual implementation of the system.To guarantee that consumers have a positive experience, the registration process is simple and the user interface is effective. (**Observations , Background readings**)

- System structure and functions should be designed in such a way that it ensures the fast performance of the system and interface should be more interesting for users of age 18 to 22. (**Questionnaire and Background readings**)

- Users are more interested in Limited-time offers, To get information about upcoming games and their pre registration, and users like to play real-time strategy and shooter games than any other genre, so the UI should be created to meet their needs. (**Questionnaire**)

- Security is a very important factor, System requires an authorization feature. System should be based on custom executable generation, which will make it difficult for any user to share the game with any other user.
  (**Background readings**)

- System requires a separate section based on Update available, which will show the list of installed games on the user's PC in which the update is available so that the user does not have to look for each game individually. To increase user interaction and interest in the game, the system must notify users as soon as a new update is available. (**Interview and Background readings**)

- Reviews are crucial for developers, so this system ought to take into account every user comment.User's positive and negative comments are important for developers. (**Observations, Background readings**)

- We need to add a feature that can complete an analysis of a specific game and generate a separate report for the game's developer in order to improve customer support. (**Interview**)

- For advertising effectiveness the model should be based on the AIDA(S) model.
  A - Attention : Attract the attention of the customers (great discounts, games already owned by friends)
  I - Interest : System should be full of advantages and features (In-game chat, In-game friends) for the customers and producers (a lot of tools support the development of their games)
  D - Desire : customer is convinced that products satisfy his needs.
  A - Action : The customer hasn’t a lot of time to decide and the customer's problem should be solved as soon as possible.
  (**Background readings**)

5. User categories and Privileges :

Gamers (Users) :
Permission for update the personal data
Search for the games with filters and search bar
Play multiplayer games with friends
Send gifts to the friends
Chat with the friends
Interact with developer
Permission for purchase the game
Permission for only view :

- Payment history
- Information about the game
- Profile of other users

Developer :
Developer for system :
Permission for update the all features of the system

Developer of the games :
Develops a new games
Inform to system developer about the new updates in the game

Publisher :
Responsible for publish a new game on the system
Help with financial and job support for the developers
Help with marketing and quality assurance of the game

Employee :
They are the employees working for a distribution platform like : Product manager, Business intelligence analyst, Technical writer, designer, database manager erc…

Database administrator :
A database administrator manages and maintains software databases such as client records, statistical surveys and user accounts.

Technical writer :
A technical writer creates documentation and educational materials for digital platform users.
They produce and edit installation instructions, user guides or in-platform instructions to increase user satisfaction rates.

Marketing staff :
For the system :They are responsible for the advertisement of the game on the distribution platform.
For the game : There is a particular staff for In-game advertisements.

System Owner :
They are the owner of the system. They have access to all the databases.
The database of employees can be updated by their permission.

6. Assumption :

- System will be more functional with transactions during purchasing the game and all the payments will be successful, there is not any dissatisfaction from the customer side for this part.
- System will be supported for all the VPN.
- When doing system maintenance updates, the system should shut down as little as feasible.
- Every digital copy will function identically for every user..
- The level of security will be high enough that no user will be able to update any data other than their personal information, primarily financial information.

7. Business constraints :

- If the game is not getting more popular, then the business analyst of the system would suggest to developers to change the essential constraints of the game and the system would add discount offers on it.
- System should ensure that the user is satisfied with the quality of the game and the system should analyze changes in customer behavior and expectations.
- We should ensure that the maintenance time of the system is as little as possible.
- We should analyze the market for the top trending games which are not available on the system and try to give good offers to them.
- We should analyze daily targets and constraints of the distribution.

Section 2: Noun Analysis

1. Noun Analysis
2. Entity-Attribute
3. Rejected Nouns and Verbs

1. Noun Analysis

Sr.No.	Noun	Verbs
1.	data	update
2.	developer	develop,update
3.	User or customer	Add,purchase,play,view, Complaint, log in
4.	publisher	publish,analysis
5.	employee	handle,change
6.	owner	remove,access,authorize
7.	game	Like,dislike,play,restrict, download
8.	reviews	update,like,dislike
9.	Intended audience	view
10.	genre	like, dislike
11.	price	Decrease, increase
12.	community	suggest
13.	groups	chat
14.	discount	get
15.	Marketing staff	advertisement
16.	Database manager	track,access
17.	Technical writer	write
18.	analyst	analyze
19.	Influencers	influence
20.	coupon	redeem
21.	bugs	remove

22.	Age	permit
23.	subscription	buy, manage
24.	performance	increase
25.	database	handle
26.	quality	satisfy
27.	Offer	Valid, invalid
28.	ratings	Increase, decrease
29.	Support system	support
30.	date	insert
31.	editors	edit
32.	followers	follow
34.	language	support
35.	achievements	Maintain, record
36.	Play time	record
37.	friends	Help, share
38.	characters	play
39.	bundle	register
40.	Gaming events	organize
41.	security	Secure, restrict
42.	Reading suggestions	suggest
43.	Payment method	pay
44.	Transactions	distribute
45.	Transaction history	view
46.	maintenance	maintain
47.	connectivity	connect

48.	Registration	register
49.	requirements	support
50.	statistics	analyze
51.	account	Handle, view
52.	permission	grant
53.	feature	available
54.	Terms and condition	apply
55.	online	check
56.	popularity	increase
57.	quality	Assure, increase
58.	Limited-time offer	available
59.	filter	search
60.	comments	Like, dislike
61.	communicate	communication
62.	interface	interact
63.	players	play
64.	Platform	handle, maintain
65.	Information	fetch
66.	market	analysis
67.	section	divide
68.	category	categorize
69.	rewards	won
70.	Shut down	maintain
71.	stock	available/not available
72.	videos	stream

73.	document	discuss
74.	animation	add, update
75.	efficiency	increase
76.	idea	implement
77.	meeting	discuss, resolve
78.	tutorial	upload
79.	sales	increase

2. Entity-Attribute

Sr.No.	Candidate Entity set	Candidate attribute set	Candidate relationship set
1.	User	user_id, user_name, age, email_id, gender	View, update, purchase, share
2.	Game	game_id, game_name, genre, release_date, last_update, total_players, dev_id, pub_id	Play, purchase
3.	Developer	dev_id, developer_name, ratings	develop, update
4.	Publisher	pub_id, pub_name, ratings	publish, support
5.	Reviews	rev_no, user_id, game_id, rev_date, content	review, comment
6.	Game_purchase	pur_no, user_id, game_id, pur_date, price, payment_method	purchase
7.	Requirements	game_id, supported_system	support
8.	Employee	emp_id, emp_name, role	Insert, delete, update
9.	Inventory	inv_no, user_id, total_purchases, total_expence, total_ingame	add
10.	Friendlist	user1_id, user2_id, game_id	add, request send

3. Rejected Nouns and Verbs

No.	Noun	Reject reason
1.	data	attributes
2.	Intended audience	irrelevant
3.	genre	attributes
4.	price	attributes
5.	community	irrelevant
6.	groups	general
7.	discount	general
8.	Marketing staff	general
9.	Database manager	general
10.	Technical writer	general
11.	analyst	general
12.	Influencers	Irrelevant
13.	coupon	attributes
14.	bugs	general
15.	Age	attributes
16.	subscription	general
17.	performance	irrelevant
18.	database	irrelevant
19.	quality	irrelevant
20.	Offer	duplicate
21.	ratings	attributes
23.	Support system	attributes

24.	date	attributes
25.	editors	attributes
26.	followers	attributes
27.	language	attributes

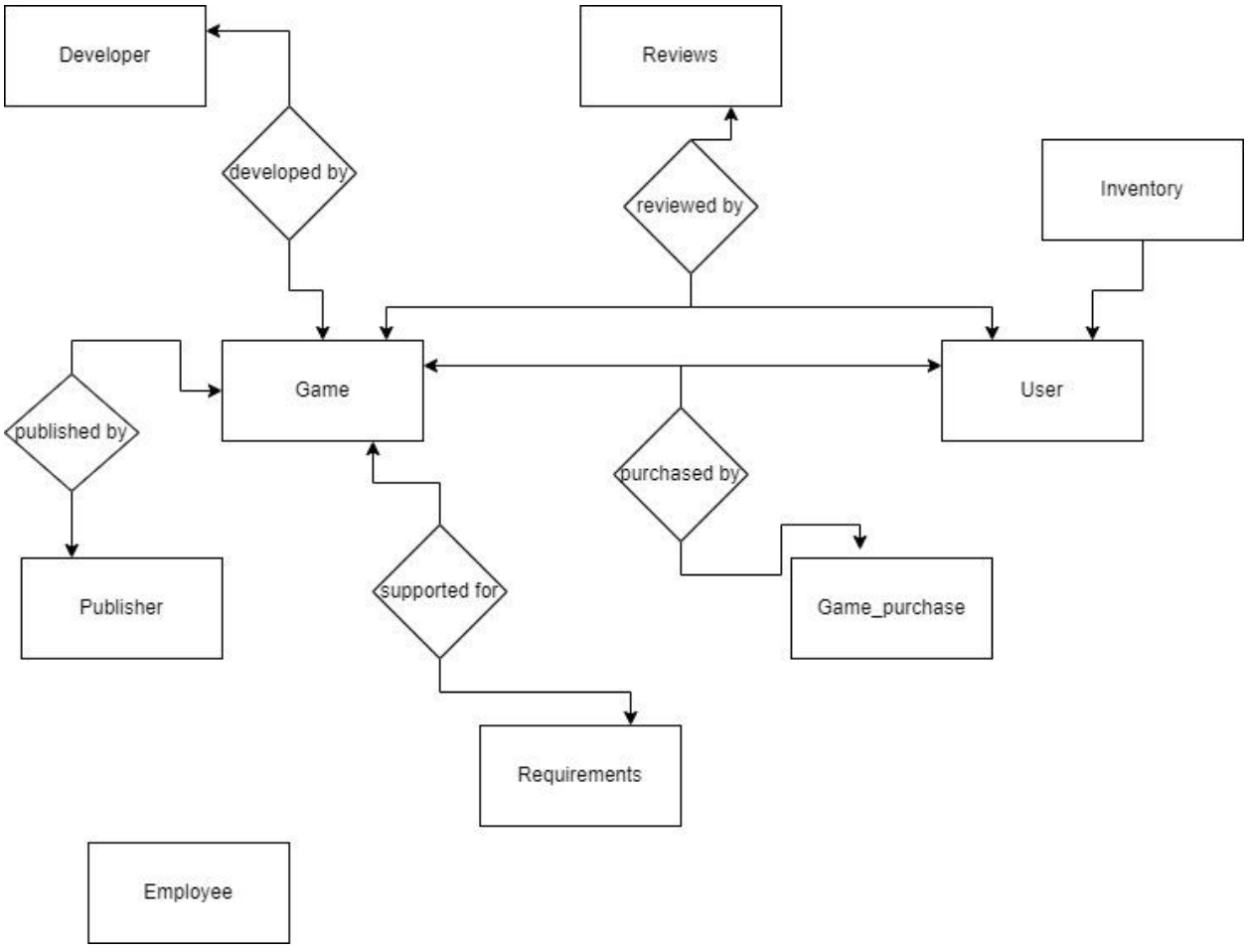
28.	achievements	attributes
29.	Play time	attributes
30.	friends	duplicate
31.	characters	general
32.	bundle	general
33.	Gaming events	general
34.	security	general
35.	Reading suggestions	general
36.	Payment method	attributes
37.	Transactions	attributes
38.	Transaction history	attributes
39.	maintenance	general
40.	connectivity	associations
41.	Registration	general
43.	statistics	general
44.	account	general
45.	permission	associations
46.	feature	geereal
47.	Terms and condition	apply
48.	online	attributes
49.	popularity	attributes

50.	quality	general
51.	Limited-time offer	general
52.	filter	general
53.	comments	duplicate
54.	communicate	associations
55.	interface	irrelevant
56.	players	duplicate
57.	Platform	irrelevant
58.	Information	general
59.	market	irrelevant
60.	section	general
61.	category	associations
62.	rewards	associations
63.	Shut down	irrelevant
64.	Stock	irrelevant
65.	videos	irrelevant
66.	document	general
67.	animation	irrelevant
68.	efficiency	irrelevant
69.	idea	irrelevant

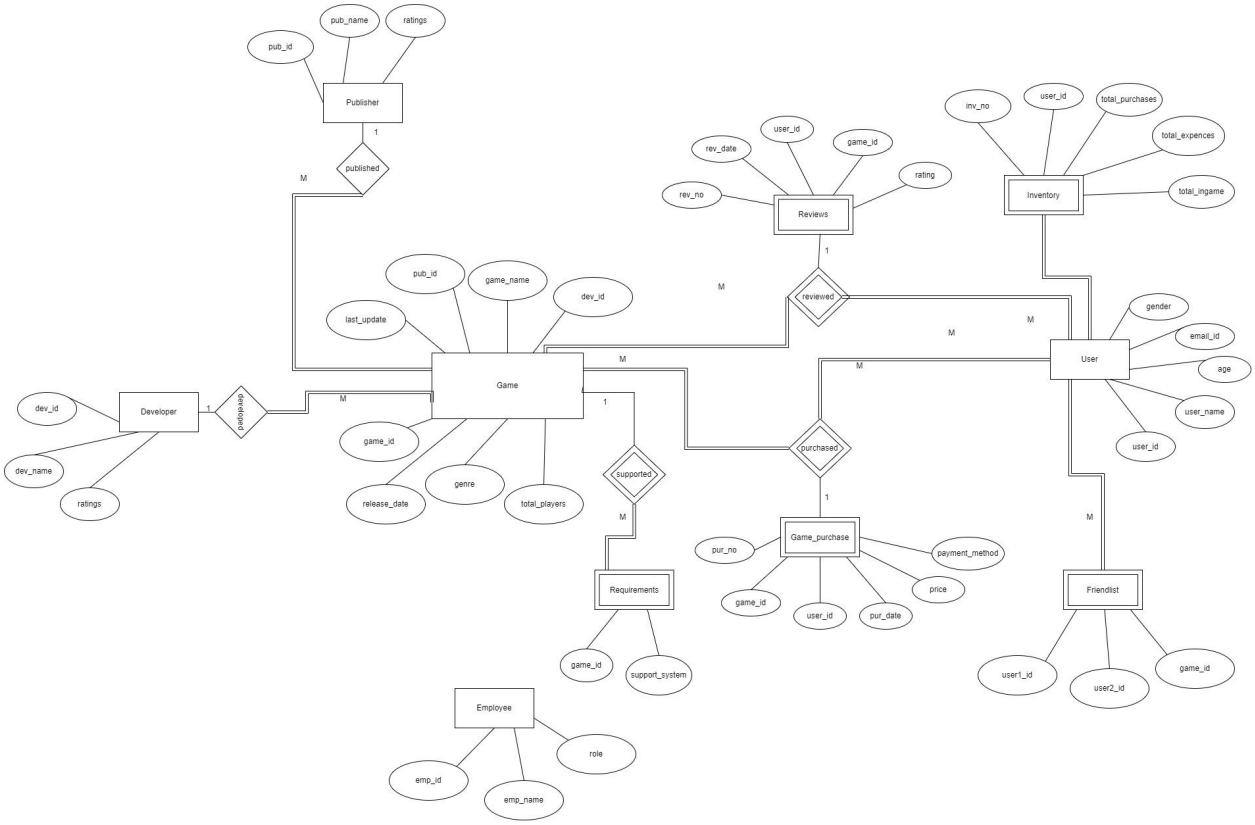
Section 3: ER-Diagrams all versions.

1. ER Diagram Version 1
2. ER Diagram Version 2
3. ER Diagram Version 2 (Final)
4. Relationship Types

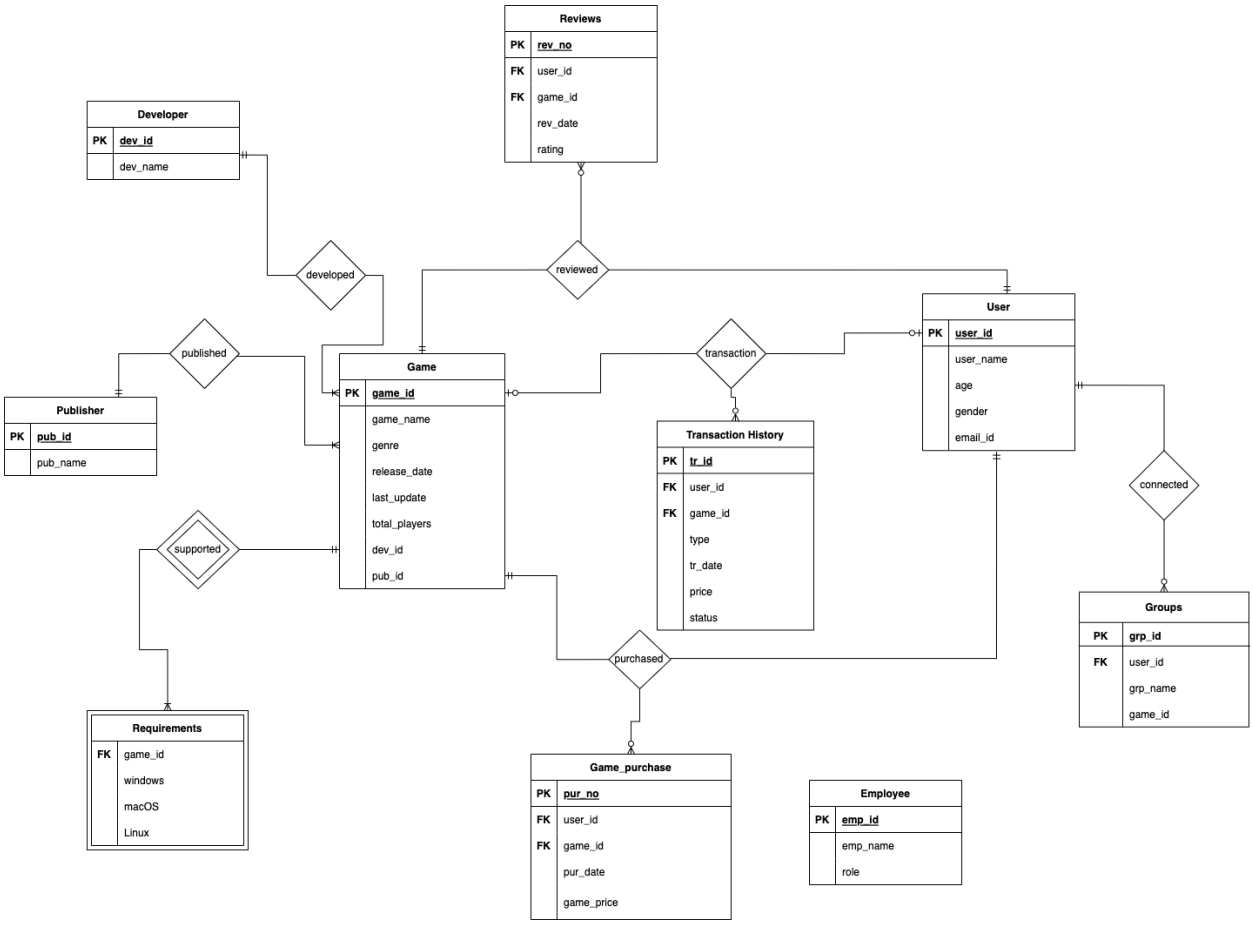
1. ER Diagram Version 1



2. ER Diagram Version 2



3. ER Diagram Version 2 (Final)



3. Relationship Types

Binary relationship :

- Publish
- Develop
- Support
- connect
- Maintains

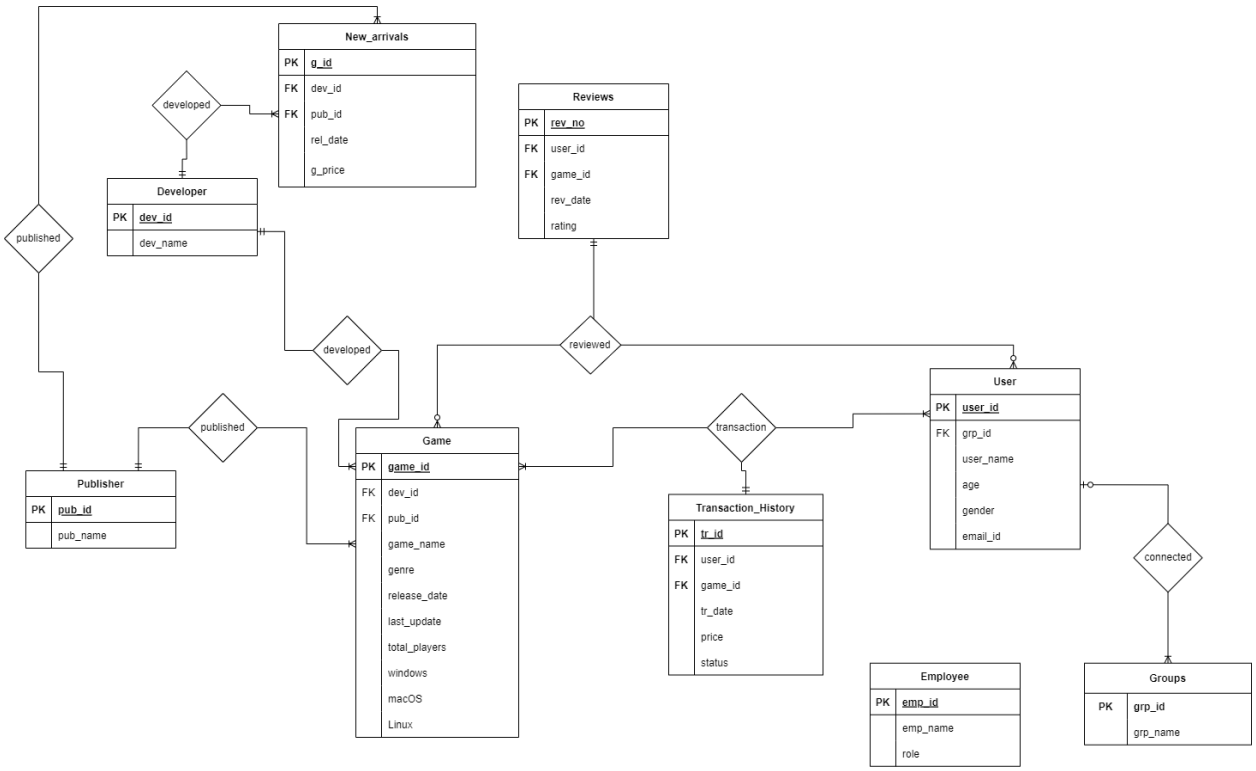
Ternary relationship:

- Reviewed
- Purchase

Section 4 : Conversion of Final ER-Diagram to Relational Model

1. Relational Model Diagram Version 1
2. All the Relations

1. Relational Model Diagram Version 1



2. All the Relations

1. Game(<u>game_id,</u> game_name,genre,relase_date,last_update, total_players, dev_id, pub_id, windows, macOS, Linux**)**

- **game_id** is **primary key** for which datatype is integer and NOT NULL
- **dev_id** is a **foreign key** because development of a game is dependent on particular developer
    - more than one game can be developed by one developer
- **pub_id** is a **foreign key** because publishment of a game is dependent on particular publisher
    - more than one game can be published by one publisher

2. Publisher(<u>pub_id</u>, pub_name**)**

- **pub_id** is **primary key** for which datatype is integer and NOT NULL

3. Developer(dev_id, dev_name**)**

- **dev_id** is **primary key** for which datatype is integer and NOT NULL

4. New_arrivals(<u>g_id</u>, dev_id, pub_id, rel_date, g_price**)**

- **g_id** is **primary key** for which datatype is integer and NOT NULL
- **dev_id** is a **foreign key** because development of a game is dependent on particular develope
    - more than one game can be developed by one developer
- **pub_id** is a **foreign key** because publishment of a game is dependent on particular publisher
    - more than one game can be published by one publisher

5. Reviews(<u>rev_no</u>, user_id, game_id, rev_date, rating**)**

- **rev_no** is **primary key** for which datatype is integer and NOT NULL
- **user_id** is a **foreign key** because reviews for a game are dependent on users
    - one user can review one game
- **game_id** is a **foreign key** because reviews given by users are dependent on games
    - one game can be reviewed many users

6. User(<u>user_id</u>, user_name, age, gender, email_id, grp_id**)**

- **user_id** is **primary key** for which datatype is integer and NOT NULL
- **grp_id** is a **foreign key** because groups are dependent on users

    - more than one user can exist in one group

7. Groups(<u>grp_id</u>, grp_name**)**

- **grp_id** is **primary key** for which datatype is integer and NOT NULL

8. Transaction_History(<u>tr_id</u>, user_id, game_id, tr_date, price, status**)**

- **tr_id** is **primary key** for which datatype is integer and NOT NULL
- **user_id** is a **foreign key** because transactions for games are dependent on users
    - more than one transaction can be done by one user
- **game_id** is a **foreign key** because transactions done by users are dependent on games
    - more than one game can be purchased by one use

9. Employee(<u>emp_id</u>, emp_name, role**)**

- **emp_id** is **primary key** for which datatype is integer and NOT NULL

Section 5 : Normalization and Schema Refinement

1. Functional Dependencies
2. Redundancy and Analysis
3. Normalization upto 3NF/BCNF
4. Final Relational Schema
    a. List all final Relations & Schemas with all details

1. Functional Dependencies

1. **Game(game_id**→game_name,genre,relase_date,last_update,total_players, dev_id, pub_id, windows, macOS, Linux**)**
2. **Publisher(pub_id** → pub_name**)**
3. **Developer(dev_id**→ dev_name**)**
4. **New_arrivals(g_id** → g_name, pub_id, rel_date, g_price**)**
5. **Reviews(rev_no** → user_id, game_id, rev_date, rating**)**
6. **User(user_id** → user_name, age, gender, email_id, grp_id**)**
7. **Groups(grp_id** → grp_name**)**
8. **Transaction_History(tr_id** →user_id, game_id, tr_date, price, status**)**
9. **Employee(emp_id** → emp_name, role**)**

2. Redundancy and Analysis

- **Game(<u>game_id,</u>** game_name,genre,relase_date,last_update, total_players, dev_id, pub_id, windows, macOS, Linux**)**
  - **There can be multiple developers for a single game so we have to create multiple rows for a single game so there is a redundancy in this table.**
  - There is no partial dependency
  - There is no transitive dependency
  - Foreign Key pub_id refers to publisher

- **Publisher(<u>pub_id</u>,** pub_name**)**
  - There is no redundancy in this table
  - There no anomalies in this table

- **Developer(<u>dev_id</u>,** dev_name**)**
  - There is no redundancy in this table
  - There no anomalies in this table

- **New_arrivals(<u>g_id</u>**, g_name, pub_id, rel_date, g_price**)**
  - There is no redundancy in this table
  - There no anomalies in this table

- **Reviews(<u>rev_no</u>,** user_id, game_id, rev_date, rating**)**
  - There is no data redundancy in this table
  - There no anomalies in this table
  - Foreign Key user_id refers to user
  - Foreign Key game_id refers to game

- **User(<u>user_id</u>,** user_name, age, gender, email_id, grp_id**)**
  - There is no data redundancy in this table
  - Foreign Key grp_id refers to groups.
  - There no anomalies in this table

- **Groups(<u>grp_id</u>,** grp_name**)**
  - There is no data redundancy in this table
  - There no anomalies in this table

- **Transaction_History(<u>tr_id</u>,** user_id, game_id, tr_date, price, status**)**
  - There is no data redundancy in this table
  - There no anomalies in this table
  - Foreign Key user_id refers to user
  - Foreign Key game_id refers to game

- **Employee(<u>emp_id</u>,** emp_name, role**)**
  - There is no data redundancy in this table
  - There no anomalies in this table

3. Normalization upto 3NF/BCNF

- **Game(<u>game_id,</u>** game_name,genre,relase_date,last_update, total_players, pub_id, windows, macOS, Linux**)**
    - There can be multiple developers for a single game,so it is not in 1NF, so we can create a new table in which dev_id and game_id will be the composite primary key.
    - There is no partial dependency so the table is in 2NF form.
    - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.
    - **New table : Game_dev(<u>dev_id</u>, <u>game_id</u>)**

- **Publisher(<u>pub_id</u>,** pub_name**)**
    - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
    - There is no partial dependency in this table so it is in 2NF.
    - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

- **Developer(<u>dev_id</u>,** dev_name**)**
    - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
    - There is no partial dependency in this table so it is in 2NF.
    - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

- **New_arrivals(<u>g_id</u>**, g_name, pub_id, rel_date, g_price**)**
    - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
    - There is no partial dependency in this table so it is in 2NF.
    - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

- **Reviews(<u>rev_no</u>,** user_id, game_id, rev_date, rating**)**
    - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
    - There is no partial dependency in this table so it is in 2NF.

- All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

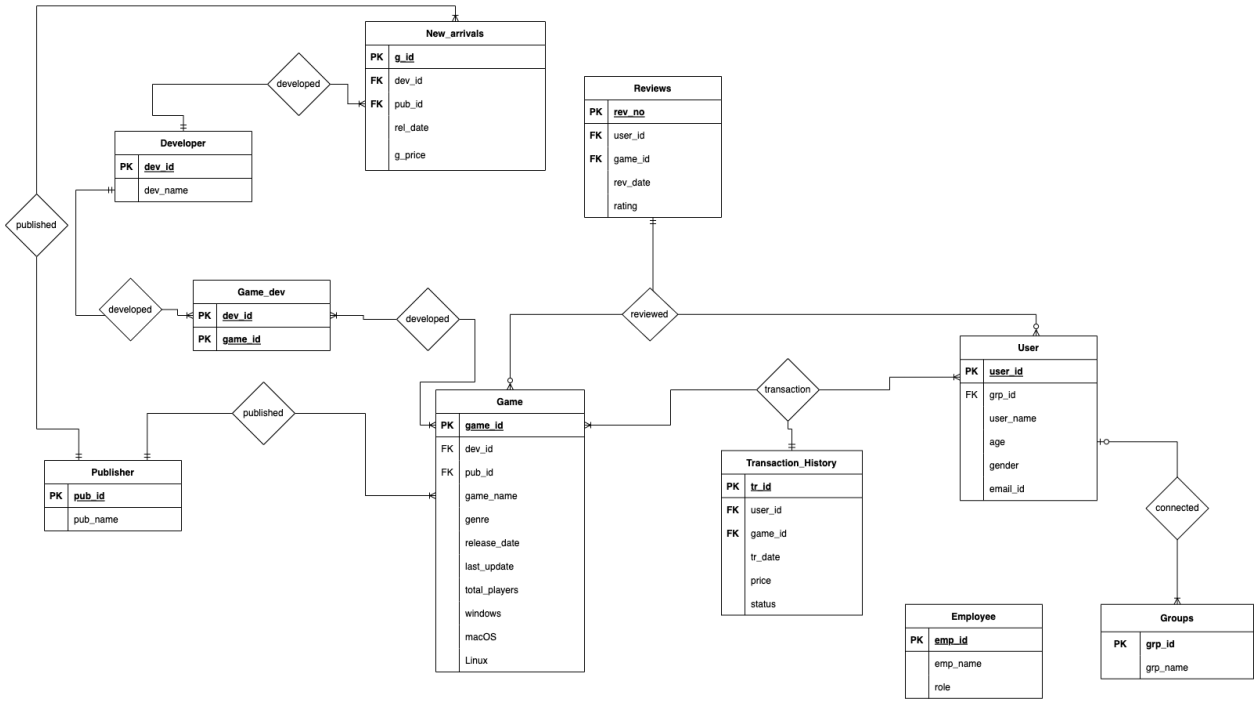
- **User(<u>user_id</u>,** user_name, age, gender, email_id, grp_id**)**
  - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
  - There is no partial dependency in this table so it is in 2NF.
  - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

- **Groups(<u>grp_id</u>,** grp_name**)**
  - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
  - There is no partial dependency in this table so it is in 2NF.
  - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

- **Transaction_History(<u>tr_id</u>,** user_id, game_id, tr_date, price, status**)**
  - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
  - There is no partial dependency in this table so it is in 2NF.
  - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

- **Employee(<u>emp_id</u>,** emp_name, role**)**
  - This schema does not have any composite or multivalued attribute. It satisfies atomicity. Hence, it is already in 1NF form.
  - There is no partial dependency in this table so it is in 2NF.
  - All the functional dependencies have a candidate key on the left side. So, it is in BCNF form which implies it is 3NF form.

4. Final Relational Schema



List all final Relations & Schemas with all details -

- **Game(<u>game_id,</u>** game_name,genre,relase_date,last_update, total_players, pub_id, windows, macOS, Linux**)**
- **Publisher(<u>pub_id</u>,** pub_name**)**
- **Developer(<u>dev_id</u>,** dev_name**)**
- **Game_dev(<u>dev_id, game_id</u>)**
- **New_arrivals(<u>g_id</u>**, g_name, pub_id, rel_date, g_price**)**
- **Reviews(<u>rev_no</u>,** user_id, game_id, rev_date, rating**)**
- **User(<u>user_id</u>,** user_name, age, gender, email_id, grp_id**)**
- **Groups(<u>grp_id</u>,** grp_name**)**
- **Transaction_History(<u>tr_id</u>,** user_id, game_id, tr_date, price, status**)**
- **Employee(<u>emp_id</u>,** emp_name, role**)**

Section 6 : SQL: Final DDL Scripts, Insert statements, 20 SQL Queries with Snapshots of output of each query.

1. Final DDL Scripts
2. SQL Queries

1. Final DDL Scripts

1. Game

```
-- Table: dbms_3_13.Game

-- DROP TABLE IF EXISTS dbms_3_13."Game";

CREATE TABLE IF NOT EXISTS dbms_3_13."Game"
(
    game_id integer NOT NULL,
    pub_id integer NOT NULL,
    game_name character varying COLLATE
pg_catalog."default" NOT NULL,
    genre character varying COLLATE pg_catalog."default"
NOT NULL,
    release_date date NOT NULL,
    last_update date NOT NULL,
    total_players integer NOT NULL,
    windows character varying COLLATE pg_catalog."default"
NOT NULL,
    "macOS" character varying COLLATE pg_catalog."default"
NOT NULL,
    "Linux" character varying COLLATE pg_catalog."default"
NOT NULL,
    CONSTRAINT "Game_pkey" PRIMARY KEY (game_id),
    CONSTRAINT "FK1" FOREIGN KEY (pub_id)
        REFERENCES dbms_3_13."Publisher" (pub_id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Game"
    OWNER to postgres;

-- Trigger: Trigger_insert
```

```
-- DROP TRIGGER IF EXISTS "Trigger_insert" ON
dbms_3_13."Game";

CREATE TRIGGER "Trigger_insert"
    BEFORE INSERT
    ON dbms_3_13."Game"
    FOR EACH ROW
    EXECUTE FUNCTION dbms_3_13."check"();
```

2. User

```
-- Table: dbms_3_13.User

-- DROP TABLE IF EXISTS dbms_3_13."User";

CREATE TABLE IF NOT EXISTS dbms_3_13."User"
(
    user_id integer NOT NULL,
    user_name character varying COLLATE
pg_catalog."default" NOT NULL,
    age integer NOT NULL,
    gender character varying COLLATE pg_catalog."default"
NOT NULL,
    email_id character varying COLLATE pg_catalog."default"
NOT NULL,
    grp_id integer NOT NULL,
    CONSTRAINT "User_pkey" PRIMARY KEY (user_id),
    CONSTRAINT "FK1" FOREIGN KEY (grp_id)
        REFERENCES dbms_3_13."Groups" (grp_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."User"
    OWNER to postgres;
```

3. Developer

```
-- Table: dbms_3_13.Developer

-- DROP TABLE IF EXISTS dbms_3_13."Developer";

CREATE TABLE IF NOT EXISTS dbms_3_13."Developer"
(
    dev_id integer NOT NULL,
    dev_name character varying COLLATE pg_catalog."default"
NOT NULL,
    CONSTRAINT "Developer_pkey" PRIMARY KEY (dev_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Developer"
    OWNER to postgres;
```

4. Publisher

```
-- Table: dbms_3_13.Publisher

-- DROP TABLE IF EXISTS dbms_3_13."Publisher";

CREATE TABLE IF NOT EXISTS dbms_3_13."Publisher"
(
    pub_id integer NOT NULL,
    pub_name character varying COLLATE pg_catalog."default"
NOT NULL,
    CONSTRAINT "Publisher_pkey" PRIMARY KEY (pub_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Publisher"
    OWNER to postgres;
```

5. Employee

```
-- Table: dbms_3_13.Employee

-- DROP TABLE IF EXISTS dbms_3_13."Employee";

CREATE TABLE IF NOT EXISTS dbms_3_13."Employee"
(
    emp_id integer NOT NULL,
    emp_name character varying COLLATE pg_catalog."default"
NOT NULL,
    role character varying COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT "Employee_pkey" PRIMARY KEY (emp_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Employee"
    OWNER to postgres;
```

6. Reviews

```
-- Table: dbms_3_13.Game_dev

-- DROP TABLE IF EXISTS dbms_3_13."Game_dev";

CREATE TABLE IF NOT EXISTS dbms_3_13."Game_dev"
(
    dev_id integer NOT NULL,
    game_id integer NOT NULL,
    CONSTRAINT "Game_dev_pkey" PRIMARY KEY (dev_id,
game_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Game_dev"
```

```
    OWNER to postgres;
```

7. Transaction_History

```
-- Table: dbms_3_13.Transaction_History

-- DROP TABLE IF EXISTS "dbms_3_13"."Transaction_History";

CREATE TABLE IF NOT EXISTS
"dbms_3_13"."Transaction_History"
(
    tr_id integer NOT NULL,
    user_id integer NOT NULL,
    game_id integer NOT NULL,
    tr_date date NOT NULL,
    price integer NOT NULL,
    status boolean NOT NULL,
    CONSTRAINT "Transaction_History_pkey" PRIMARY KEY
(tr_id),
    CONSTRAINT "FK1" FOREIGN KEY (user_id)
        REFERENCES "dbms_3_13"."User" (user_id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "FK2" FOREIGN KEY (game_id)
        REFERENCES "dbms_3_13"."Game" (game_id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS "dbms_3_13"."Transaction_History"
    OWNER to postgres;
```

8. New_arrivals

```
-- Table: dbms_3_13.New_arrivals

-- DROP TABLE IF EXISTS dbms_3_13."New_arrivals";

CREATE TABLE IF NOT EXISTS dbms_3_13."New_arrivals"
(
    g_id integer NOT NULL,
    pub_id integer NOT NULL,
    g_name character varying COLLATE pg_catalog."default"
NOT NULL,
    rel_date date NOT NULL,
    g_price integer NOT NULL,
    CONSTRAINT "New_arrivals_pkey" PRIMARY KEY (g_id),
    CONSTRAINT "FK2" FOREIGN KEY (pub_id)
        REFERENCES dbms_3_13."Publisher" (pub_id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."New_arrivals"
    OWNER to postgres;
```

9. Groups

```
-- Table: dbms_3_13.Groups

-- DROP TABLE IF EXISTS dbms_3_13."Groups";

CREATE TABLE IF NOT EXISTS dbms_3_13."Groups"
(
    grp_id integer NOT NULL,
    grp_name character varying COLLATE pg_catalog."default"
NOT NULL,
    CONSTRAINT "Groups_pkey" PRIMARY KEY (grp_id)
```

```
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Groups"
    OWNER to postgres;
```

10. Game_dev

```
-- Table: dbms_3_13.Game_dev

-- DROP TABLE IF EXISTS dbms_3_13."Game_dev";

CREATE TABLE IF NOT EXISTS dbms_3_13."Game_dev"
(
    dev_id integer NOT NULL,
    game_id integer NOT NULL,
    CONSTRAINT "Game_dev_pkey" PRIMARY KEY (dev_id,
game_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS dbms_3_13."Game_dev"
    OWNER to postgres;
```

2. SQL Queries

1. **Count the number of games for which genre is Shooter.**

```
1  -- query 1
2  SELECT COUNT(*)
3  FROM dbms_3_13."Game"
4  WHERE dbms_3_13."Game"."genre" = 'Shooter'
5
6  Loading...
7  SELECT dbms_3_13."Game"."game_name"
8  FROM dbms_3_13."Game"
9  WHERE dbms_3_13."Game"."windows" = 'TRUE'
10
```

Data output | Messages | Notifications

	count bigint
1	15

Total rows: 1 of 1 | Query complete 00:00:00.128

Tuple count = 1

2. **Print all names of the game which are supported for windows.**

```
5
6   -- query 2
7   SELECT dbms_3_13."Game"."game_name"
8   FROM dbms_3_13."Game"
9   WHERE dbms_3_13."Game"."windows" = 'TRUE'
10
```

Loading...

Data output | Messages | Notifications

	game_name character varying
1	Gigashots
2	Cogibox
3	Jabbertype
4	Zooxo
5	Eadel
6	Buzzster

Total rows: 50 of 50 | Query complete 00:00:00.073 | Successfully run. Total query runtime: 73 msec.

Tuple count = 50

3. **Print all the user names which are below age 18.**

```
10
11  -- query 3
12  SELECT dbms_3_13."User"."user_name"
13  FROM dbms_3_13."User"
14  WHERE dbms_3_13."User"."age" < 18
15
```

Loading...

Data ... Notifications

	user_name character varying
1	jkondratovich0
2	athringc
3	bcarasf
4	cnugentj
5	emitchardo
6	gjocict

Total rows: 22 of 22 | Query complete 00:00:00.063

Tuple count = 22

4. **List Top 10 Games which has the most total no. of players with its genre.**

```
16  -- query 4
17  SELECT dbms_3_13."Game"."game_name",dbms_3_13."Game"."genre",dbms_3_13."Game"."total
18  FROM dbms_3_13."Game"
19  order by dbms_3_13."Game"."total_players" DESC
20  LIMIT 10
21
```

Data output | Messages | Notifications

	game_name character varying	genre character varying	total_players integer
1	Wikido	FPS	995
2	Skynoodle	Shooter	982
3	Einti	Action	972
4	Gabcube	RTS	963
5	Camido	Survival	957
6	Bubbletube	Survival	945

Total rows: 10 of 10 | Query complete 00:00:00.109

Tuple count = 10

5. **List the game names for which transaction was done successfully and price of the game is greater than average price.**

```
22  -- query 5
23  SELECT dbms_3_13."Game"."game_name"
24  FROM dbms_3_13."Game"
25  JOIN dbms_3_13."Transaction_History" ON dbms_3_13."Game"."game_id" = dbms_3_13."Tra
26  WHERE dbms_3_13."Transaction_History"."price" > (select avg(dbms_3_13."Transaction_H
27
28
```

Loading...

Data output | Messages | Notifications

	game_name character varying
1	Gigashots
2	Zoozzy
3	Jabbertype
4	Browsetype
5	Eadel
6	Tagpad

Total rows: 25 of 25 | Query complete 00:00:00.187

Tuple count = 25

6. **List all the female players.**

```
29  -- query 6
30
31  SELECT dbms_3_13."User"."user_name"
32  FROM dbms_3_13."User"
33  WHERE dbms_3_13."User" Loading... 'Female'
34
```

Data output | Messages | Notifications

	user_name character varying
1	brisdale3
2	kbricket4
3	gkydde8
4	astollen9
5	wbaselioa
6	athringc

Total rows: 47 of 47 | Query complete 00:00:00.158 | Successfully run. Total query runtime: 1

Tuple count = 47

7. **Create a view of the maximum amount of game price.**

```
34
35  -- query 7
36  CREATE OR REPLACE VIEW lst_update as
37  SELECT dbms_3_13."Game"."last_update" FROM dbms_3_13."Game"
38  SELECT * FROM lst_update
```

lst_update

Data output | Messages | Notifications

	last_update date
1	2022-02-10
2	2022-02-11
3	2022-02-12
4	2022-02-13
5	2022-02-14
6	2022-02-15

Total rows: 100 of 100 | Query complete 00:00:00.308

Tuple count = 100

8. **Print the count of different types of transaction status.**

```
40  -- query 8
41  SELECT dbms_3_13."Transaction_History"."status",count(dbms_3_13."Transaction_Histor
42  FROM dbms_3_13."Transaction_Hist  Loading...  dbms_3_13."Transaction_History"."statu
43
```

Data output | Messages | Notifications

	status character varying	count bigint
1	TRUE	53
2	FALSE	47

Total rows: 2 of 2 | Query complete 00:00:00.073 | Successfully run. Total query runtime

Tuple count = 2

9. **Print the game names whose ratings are greater than 3.**

```
44  -- query 9
45      SELECT dbms_3_13."Game"."game_name"
46      FROM dbms_3_13."Game"
47      JOIN dbms_3_13."Reviews" ON dbms_3_13."Game"."game_id" = dbms_3_13."Reviews"."ga
48      WHERE dbms_3_13."Reviews"."rating" > 3
49
```

Data output | Messages | Notifications

	game_name character varying
1	Gigashots
2	Cogibox
3	InnoZ
4	Zooxo
5	Browsetype
6	Tagpad

Total rows: 36 of 36 | Query complete 00:00:00.083

Tuple count = 36

10. List all the users whose group name is ‘Schmidt Group’.

```
50  -- query 10
51  SELECT dbms_3_13."User"."user_id",dbms_3_13."User"."user_name"
52  FROM dbms_3_13."User"
53  JOIN dbms_3_13."Groups" ON dbms_3_13."User"."grp_id" = dbms_3_13."Groups"."grp_id"
54  WHERE dbms_3_13."Groups"."grp_name" = 'Schmidt Group'
55
```

Data output | Messages | Notifications

	user_id [PK] integer	user_name character varying
1	24	karnoudn
2	25	emitchardo
3	26	btumilityp

Total rows: 3 of 3 | Query complete 00:00:00.246

Tuple count = 3

11. List the no. of Testers, Designers, Artist, Programmer, Database Handler.

```
56  -- query 11
57  SELECT dbms_3_13."Employee"."role",count(dbms_3_13."Employee"."role")
58  FROM dbms_3_13."Employee" GROUP BY dbms_3_13."Employee"."role"
```

Data output | Messages | Notifications

	role character varying	count bigint
1	Designer	7
2	Database handler	7
3	Tester	5
4	Programmer	7
5	Artist	4

Total rows: 5 of 5 | Query complete 00:00:00.355

Successfully run. Total query runtime: 3

Tuple count = 5

12. Count the no. of Transactions done between 07/12/2022 and 25/12/2022.

```
60  -- query 12
61  SELECT count(*)
62  FROM dbms_3_13."Transaction_History"
63  WHERE dbms_3_13."Transaction_History"."tr_date" BETWEEN '2022-12-07' AND '2022-12-25'
64
```

Data output | Messages | Notifications

	count bigint
1	19

Total rows: 1 of 1 | Query complete 00:00:00.097

Tuple count = 1

13. Which games have been published by Valve.

```
65  -- query 13
66  SELECT dbms_3_13."Game"."game_id",dbms_3_13."Game"."game_name"
67  FROM dbms_3_13."Game"
68  JOIN dbms_3_13."Publisher" ON dbms_3_13."Game"."pub_id" = dbms_3_13."Publisher"."pub
69  WHERE dbms_3_13."Publisher"."pub_name" = 'Valve'
70
```

Data output | Messages | Notifications

	game_id [PK] integer	game_name character varying
1	1	Ainyx
2	2	Voomm

Total rows: 2 of 2 | Query complete 00:00:00.099

Tuple count = 2

14. **Display all the details of users whose transaction status was FALSE for purchase and create a view for the same.**

```
71  -- query 14
72  SELECT dbms_3_13."User"."user_id",dbms_3_13."User"."user_name",dbms_3_13."User"."ger
73  FROM dbms_3_13."User"
74  JOIN dbms_3_13."Transaction_History" ON dbms_3_13."User"."user_id" = dbms_3_13."Trar
75  WHERE dbms_3_13."Transaction_History"."status" = 'FALSE'
76
77
```

Data output | Messages | Notifications

	user_id [PK] integer	user_name character varying	gender character varying	age integer	email_id character varying
1	2	sfronek1	Male	18	sclaus1@cloudfl...
2	4	brisdale3	Female	20	tkabos3@google....
3	5	kbricket4	Female	21	lstirman4@vkont...
4	7	pbalsom6	Male	23	alightollers6@fd...
5	8	ahaitlie7	Male	24	ggyer7@cloudfla...
6	12	rpearsonb	Male	19	bpyrahb@google....

Total rows: 47 of 47 | Query complete 00:00:00.073

Tuple count = 47

15. **List 10 new_release games which have lower prices.**

```
76
77  -- query 15
78  SELECT dbms_3_13."New_arrivals"."g_id",dbms_3_13."New_arrivals"."g_name"
79  FROM dbms_3_13."New_arrivals"
80  order by dbms_3_13."New_arrivals"."g_price" asc
81  LIMIT 10
82
```

Data output | Messages | Notifications

	g_id [PK] integer	g_name character varying
1	19	Senger PLC
2	27	Erdman-Mohr
3	41	Connelly-Brekke
4	6	Zulauf, Buckridge...
5	23	Konopelski, Mohr...
6	39	Schamberger PLC

Total rows: 10 of 10 | Query complete 00:00:00.162

Tuple count = 10

16. List the publisher of the game which is supported in all systems.

```
83  -- query 16
84  SELECT dbms_3_13."Publisher"."pub_name"
85  FROM dbms_3_13."Publisher"
86  JOIN dbms_3_13."Game" ON dbms_3_13."Game"."pub_id" = dbms_3_13."Publisher"."pub_id"
87  WHERE dbms_3_13."Game"."windows" = 'TRUE' AND dbms_3_13."Game"."macOS" = 'TRUE' AND
88
```

Data output | Messages | Notifications

	pub_name character varying
1	Amazon Games
2	Electronic Arts
3	Rockstar Games
4	Netease Games
5	Astyr
6	Fireshine Games

Total rows: 21 of 21 | Query complete 00:00:00.088 | Successfully run. Total query runtime:

Tuple count = 21

17. What is the rating given by user_id 4.

```
89  -- query 17
90  SELECT dbms_3_13."Reviews"."rating"
91  FROM dbms_3_13."Reviews"
92  JOIN dbms_3_13."User" ON dbms_3_13."Reviews"."user_id" = dbms_3_13."User"."user_id"
93  WHERE dbms_3_13."User"."user_id" = 4
94
```

Data output | Messages | Notifications

	rating integer
1	1

Total rows: 1 of 1 | Query complete 00:00:00.081

Tuple count = 1

18. **List total no. of players that play Skynoodle.**

```
95  -- query 18
96  SELECT dbms_3_13."Game"."total_players"
97  FROM dbms_3_13."Game"
98  WHERE dbms_3_13."Game"."game_name" = 'Skynoodle'
99
```

Data output | Messages | Notifications

	total_players integer
1	982

Total rows: 1 of 1 | Query complete 00:00:00.734

Tuple count = 1

19. **Print details about the game for which total players are greater than 500. Return a temp table with the game_id, pub_id, game_name, genre, total_players in the result table.**

```
100 -- query 19
101
102 CREATE OR REPLACE FUNCTION dbms_3_13.fun( )
103 RETURNS TABLE(game_id integer,pub_id integer,game_name character varying,genre char
104 LANGUAGE 'plpgsql'
105 AS $BODY$
106 DECLARE
107 temp_t record;
108 BEGIN
109 CREATE TEMP TABLE test1 (game_id integer,pub_id integer,game_name character varyin
```

Data output | Messages | Notifications

	fun record
1	(1,101,Ain...
2	(2,101,Vo...
3	(3,102,Gi...
4	(6,102,Tw...
5	(7,102,Ski...
6	(8,102,Ju...

Total rows: 48 of 48 | Query complete 00:00:00.098

Tuple count = 48

20. **Create a trigger on the Table of your choice to check if the Primary key ID already exists or not before inserting a new record & Send a custom reply instead of an error message.**

```
121  -- query 20
122  CREATE OR REPLACE FUNCTION dbms_3_13.check()
123  returns TRIGGER
124  LANGUAGE 'plpgsql'
125  AS
126  $BODY$
127  DECLARE
128  n_id integer;
129
130  BEGIN
131  SELECT "game_id" INTO n_id FROM dbms_3_13."Game" WHERE dbms_3_13."Game"."game_id"
```

Data output | Messages | Notifications

```
NOTICE:  game_id: 10 already exists

ERROR:  control reached end of trigger procedure without RETURN
CONTEXT:  PL/pgSQL function dbms_3_13."check"()
SQL state: 2F005
```

Total rows: 48 of 48 | Query complete 00:00:00.293

Tuple count = 48

Section 7 : Project Code with output screenshots

https://github.com/DivCode4664/Video-Game-Distribution-Project.git

1. Backend Code
2. Screenshots of the Website that connects the Database.

1. Backend Code

Connects frontend with backend (Using Django):

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'dummy',
        'USER': 'postgres',
        'PASSWORD': 'Divyesh202001229_@',
        'HOST' : 'localhost' ,
        'PORT': '5432',
    }
}
```

Code for setting up the URL:

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.style),
    path('homepage',views.homepage,name="homepage"),
    path('',views.homepage,name="homepage"),
    path('showPublisher',views.showPublisher, name = "showPublisher"),
    path('showGame',views.showGame,name="showGame"),
    path('InsertGame',views.InsertGame,name="InsertGame"),
    path('InsertPublisher',views.InsertPublisher,name="InsertPublisher"),
    path('updateGame/<int:id>',views.updateGame,name="updateGame"),
    path('EditGame/<int:id>',views.EditGame,name="EditGame"),
    path('EditPublisher/<int:id>',views.EditPublisher,name="EditPublisher"),
    path('DelGame/<int:id>',views.DelGame,name="DelGame"),
    path('deletedGame/<int:id>',views.deletedGame,name="deletedGame"),
    path('DelPublisher/<int:id>',views.DelPublisher,name="DelPublisher"),
    path('deletedPublisher/<int:id>',views.deletedPublisher,name="deletedPublisher"),
    path('updatePublisher/<int:id>',views.updatePublisher,name="updatePublisher"),
```

```
    path('SortGame',views.SortGame,name="SortGame"),
    path('SortPublisher',views.SortPublisher,name="SortPublisher"),
    path('QueryforGame',views.QueryforGame,name="QueryforGame"),
    path('QueryforPublisher',views.QueryforPublisher,name="QueryforPublisher"),
]
```

Code for setting up the modules used for frontend:

```
from django.db import models

class Game(models.Model):
    game_id = models.IntegerField(primary_key=True)
    pub_id = models.IntegerField()
    game_name = models.CharField(max_length=100)
    genre = models.CharField(max_length=100)
    release_date = models.DateField()
    last_update = models.DateField()
    total_players = models.IntegerField()
    windows = models.CharField(max_length=5)
    macos = models.CharField(db_column='macOS', max_length=5)  # Field name made
lowercase.
    linux = models.CharField(db_column='Linux', max_length=5)  # Field name made
lowercase.

    class Meta:
        managed = False
        db_table = 'Game'


class Publisher(models.Model):
    pub_id = models.IntegerField(primary_key=True)
    pub_name = models.CharField(max_length=100)

    class Meta:
        managed = False
        db_table = 'Publisher'


class AuthGroup(models.Model):
```

```
    name = models.CharField(unique=True, max_length=150)

    class Meta:
        managed = False
        db_table = 'auth_group'


class AuthGroupPermissions(models.Model):
    id = models.BigAutoField(primary_key=True)
    group = models.ForeignKey(AuthGroup, models.DO_NOTHING)
    permission = models.ForeignKey('AuthPermission', models.DO_NOTHING)

    class Meta:
        managed = False
        db_table = 'auth_group_permissions'
        unique_together = (('group', 'permission'),)


class AuthPermission(models.Model):
    name = models.CharField(max_length=255)
    content_type = models.ForeignKey('DjangoContentType', models.DO_NOTHING)
    codename = models.CharField(max_length=100)

    class Meta:
        managed = False
        db_table = 'auth_permission'
        unique_together = (('content_type', 'codename'),)


class AuthUser(models.Model):
    password = models.CharField(max_length=128)
    last_login = models.DateTimeField(blank=True, null=True)
    is_superuser = models.BooleanField()
    username = models.CharField(unique=True, max_length=150)
    first_name = models.CharField(max_length=150)
    last_name = models.CharField(max_length=150)
    email = models.CharField(max_length=254)
    is_staff = models.BooleanField()
    is_active = models.BooleanField()
    date_joined = models.DateTimeField()
```

```
    class Meta:
        managed = False
        db_table = 'auth_user'


class AuthUserGroups(models.Model):
    id = models.BigAutoField(primary_key=True)
    user = models.ForeignKey(AuthUser, models.DO_NOTHING)
    group = models.ForeignKey(AuthGroup, models.DO_NOTHING)

    class Meta:
        managed = False
        db_table = 'auth_user_groups'
        unique_together = (('user', 'group'),)


class AuthUserUserPermissions(models.Model):
    id = models.BigAutoField(primary_key=True)
    user = models.ForeignKey(AuthUser, models.DO_NOTHING)
    permission = models.ForeignKey(AuthPermission, models.DO_NOTHING)

    class Meta:
        managed = False
        db_table = 'auth_user_user_permissions'
        unique_together = (('user', 'permission'),)


class DjangoAdminLog(models.Model):
    action_time = models.DateTimeField()
    object_id = models.TextField(blank=True, null=True)
    object_repr = models.CharField(max_length=200)
    action_flag = models.SmallIntegerField()
    change_message = models.TextField()
    content_type = models.ForeignKey('DjangoContentType', models.DO_NOTHING,
blank=True, null=True)
    user = models.ForeignKey(AuthUser, models.DO_NOTHING)

    class Meta:
        managed = False
```

```
        db_table = 'django_admin_log'


class DjangoContentType(models.Model):
    app_label = models.CharField(max_length=100)
    model = models.CharField(max_length=100)

    class Meta:
        managed = False
        db_table = 'django_content_type'
        unique_together = (('app_label', 'model'),)


class DjangoMigrations(models.Model):
    id = models.BigAutoField(primary_key=True)
    app = models.CharField(max_length=255)
    name = models.CharField(max_length=255)
    applied = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'django_migrations'


class DjangoSession(models.Model):
    session_key = models.CharField(primary_key=True, max_length=40)
    session_data = models.TextField()
    expire_date = models.DateTimeField()

    class Meta:
        managed = False
        db_table = 'django_session'
```

Code for fetching, editing, deleting and sorting in the database:

```
from django.shortcuts import render
from gamingdistribution.models import Game
from gamingdistribution.models import Publisher
from django.contrib import messages
from django.http import HttpResponse
from gamingdistribution.forms import GameForms,PublisherForms
from django.db import connection

def homepage(request):
    return render(request,'homepage.html')

def showGame(request):
    showall = Game.objects.all()
    return render(request,'showGame.html',{"data":showall})

def showPublisher(request):
    showall = Publisher.objects.all()
    return render(request,'showPublisher.html',{"data":showall})

def InsertPublisher(request):
    if request.method == 'POST':
        if request.POST.get('pub_id') and request.POST.get('pub_name'):
            saverecord = Publisher()
            saverecord.pub_id = request.POST.get('pub_id')
            saverecord.pub_name = request.POST.get('pub_name')

            allval = Publisher.objects.all()

            for i in allval:
                if int(i.pub_id)==int(request.POST.get('pub_id')):
                    messages.warning(request,'Publisher already exists with this ID...!');
                    return render(request,'InsertPublisher.html')

            saverecord.save()
            messages.success(request, 'Publisher ' + saverecord.pub_name + ' Is Saved
Successfully')
            return render(request ,'InsertPublisher.html')
    else:
```

```
        return render(request, 'InsertPublisher.html')

def InsertGame(request):
    if request.method == 'POST':
        request.POST.get('game_id') and request.POST.get('pub_id') and
request.POST.get('game_name') and request.POST.get('genre') and
request.POST.get('release_date') and request.POST.get('last_update') and
request.POST.get('total_players') and request.POST.get('windows') and
request.POST.get('macos') and request.POST.get('linux')
        saverecord = Game()
        saverecord.game_id = request.POST.get('game_id')
        saverecord.pub_id = request.POST.get('pub_id')
        saverecord.game_name = request.POST.get('game_name')
        saverecord.genre = request.POST.get('genre')
        saverecord.release_date = request.POST.get('release_date')
        saverecord.last_update = request.POST.get('last_update')
        saverecord.total_players = request.POST.get('total_players')
        saverecord.windows = request.POST.get('windows')
        saverecord.macos = request.POST.get('macos')
        saverecord.linux = request.POST.get('linux')

        allval = Game.objects.all()
        for i in allval:
            if int(i.game_id)==int(request.POST.get('game_id')):
                messages.warning(request,'Game already exists with this ID....!');
                return render(request,'InsertGame.html')

        allvl = Publisher.objects.all()
        x = False
        for i in allvl:
            if int(i.pub_id)==int(request.POST.get('pub_id')):
                x = True
                break

        if x == False:
            messages.warning(request,'Publisher with Publisher ID ' + saverecord.pub_id + '
does not exits...');
            return render(request,'InsertGame.html')

        saverecord.save()
```

```
        messages.success(request, 'Game ' + saverecord.game_name + ' Is Saved
Successfully')
        return render(request ,'InsertGame.html')

    else:
        return render(request, 'InsertGame.html')

def EditGame(request,id):
    editGameObj=Game.objects.get(game_id=id)
    context={
        "Game":editGameObj
    }
    return render(request,'EditGame.html',context)

def updateGame(request,id):
    game_id=id
    pub_id=request.POST.get('pub_id')
    game_name=request.POST.get('game_name')
    genre=request.POST.get('genre')
    release_date=request.POST.get('release_date')
    last_update=request.POST.get('last_update')
    total_players=request.POST.get('total_players')
    windows=request.POST.get('windows')
    macos=request.POST.get('macOS')
    linux=request.POST.get('linux')
    NewGame = Game(game_id = game_id, pub_id = pub_id, game_name =
game_name,genre = genre, release_date = release_date, last_update = last_update,
total_players =total_players, windows = windows, macos = macos, linux = linux)
    game=Game.objects.get(game_id=id)
    game=NewGame
    game.save()
    context={
        "Game":game
    }
    messages.success(request, 'Game ' + game_name + ' Is Updated Successfully')
    return render(request,'EditGame.html',context)

def DelGame(request,id):
    delGameObj=Game.objects.get(game_id=id)
    context={
```

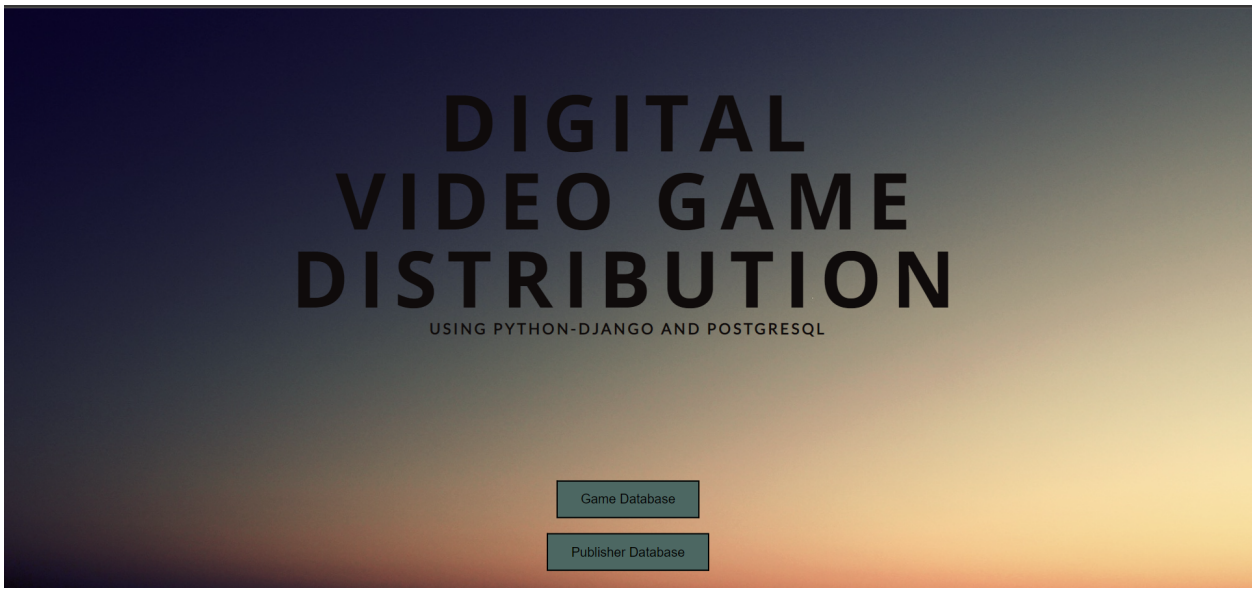
```
        "Game":delGameObj
    }
    return render(request,'DelGame.html',context)

def deletedGame(request,id):
    delGameObj=Game.objects.get(game_id=id)
    delGameObj.delete()
    showall=Game.objects.all()
    messages.success(request,'Game deleted successfully!!')
    return render(request,'DelGame.html',{"Game": delGameObj})

def DelPublisher(request,id):
    delPubObj=Publisher.objects.get(pub_id=id)
    context={
        "Publisher":delPubObj
    }
    return render(request,'DelPublisher.html',context)

def deletedPublisher(request,id):
    delPubObj=Publisher.objects.get(pub_id=id)
    delPubObj.delete()
    showall=Publisher.objects.all()
    messages.success(request,'Record deleted successfully!!')
    return render(request,'DelPublisher.html',{"Publisher": delPubObj})

def updatePublisher(request,id):
    pub_id = id
    pub_name=request.POST.get('pub_name')
    NewPublisher = Publisher(pub_id = pub_id,pub_name = pub_name)
    publisher=Publisher.objects.get(pub_id=id)
    publisher=NewPublisher
    publisher.save()
    context={
        "Publisher":publisher
    }
    messages.success(request, 'Publisher ' + pub_name + ' Is Saved Successfully')
    return render(request,'EditPublisher.html',context)

def EditPublisher(request,id):
    editPublisherObj=Publisher.objects.get(pub_id=id)
```

```
    context={
        "Publisher":editPublisherObj
    }
    return render(request,'EditPublisher.html',context)

def SortGame(request):
    if request.method=="POST":
        if request.POST.get('Sort'):
            type=request.POST.get('Sort')
            sorted=Game.objects.all().order_by(type)
            context = {
                'data': sorted
            }
            return render(request,'SortGame.html',context)
    else:
        return render(request,'SortGame.html')

def SortPublisher(request):
    if request.method=="POST":
        if request.POST.get('Sort'):
            type=request.POST.get('Sort')
            sorted=Publisher.objects.all().order_by(type)
            context = {
                'data': sorted
            }
            return render(request,'SortPublisher.html',context)
    else:
        return render(request,'SortPublisher.html')

def QueryforGame(request):
    query = "select * from \"Game\" where \"Game\".\"total_players\" > (select
avg(\"Game\".\"total_players\") from \"Game\");"

    cursor = connection.cursor()
    cursor.execute(query)
    alldata=cursor.fetchall()

    return render(request,'QueryforGame.html',{'data':alldata})

def QueryforPublisher(request):
```

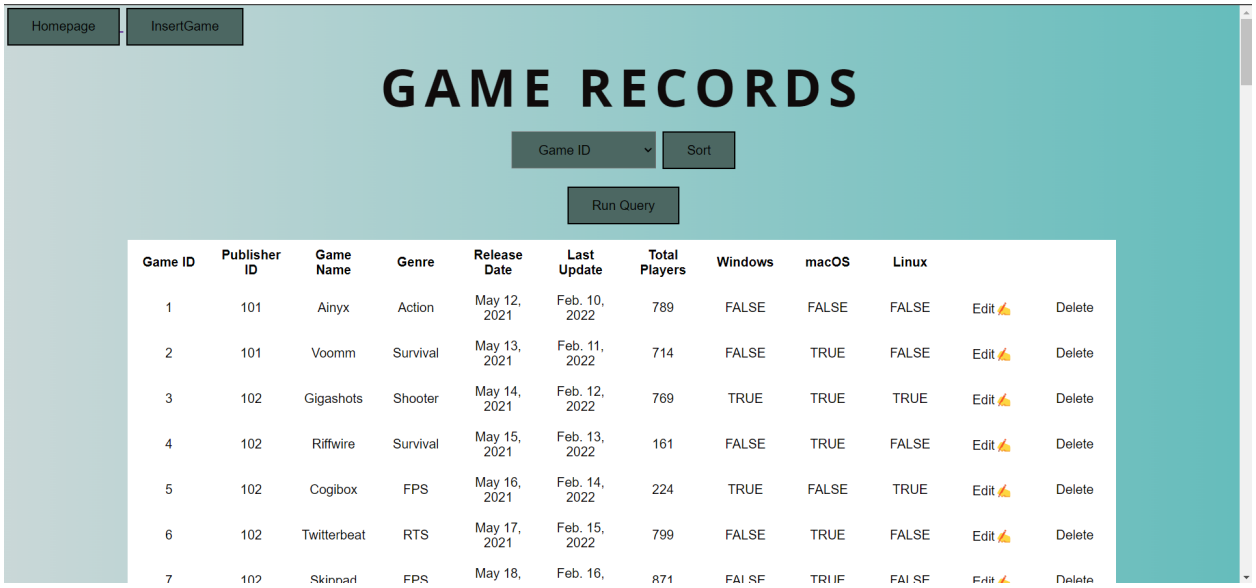
```
    query = "select
\"Game\".\"game_id\",\"Game\".\"game_name\",\"Game\".\"genre\",\"Game\".\"release_d
ate\",\"Game\".\"last_update\",\"Game\".\"total_players\",\"Game\".\"windows\",\"Game\".\
"macOS\",\"Game\".\"Linux\" from \"Game\" JOIN \"Publisher\" on \"Publisher\".\"pub_id\"
= \"Game\".\"pub_id\" where \"Publisher\".\"pub_name\" = 'Valve';"

    cursor = connection.cursor()
    cursor.execute(query)
    alldata=cursor.fetchall()

    return render(request,'QueryforPublisher.html',{'data':alldata})

def style(request):
    return render(request,'homepage.html')
```

2. Screenshots of the Website that connects the Database.

1. Homepage

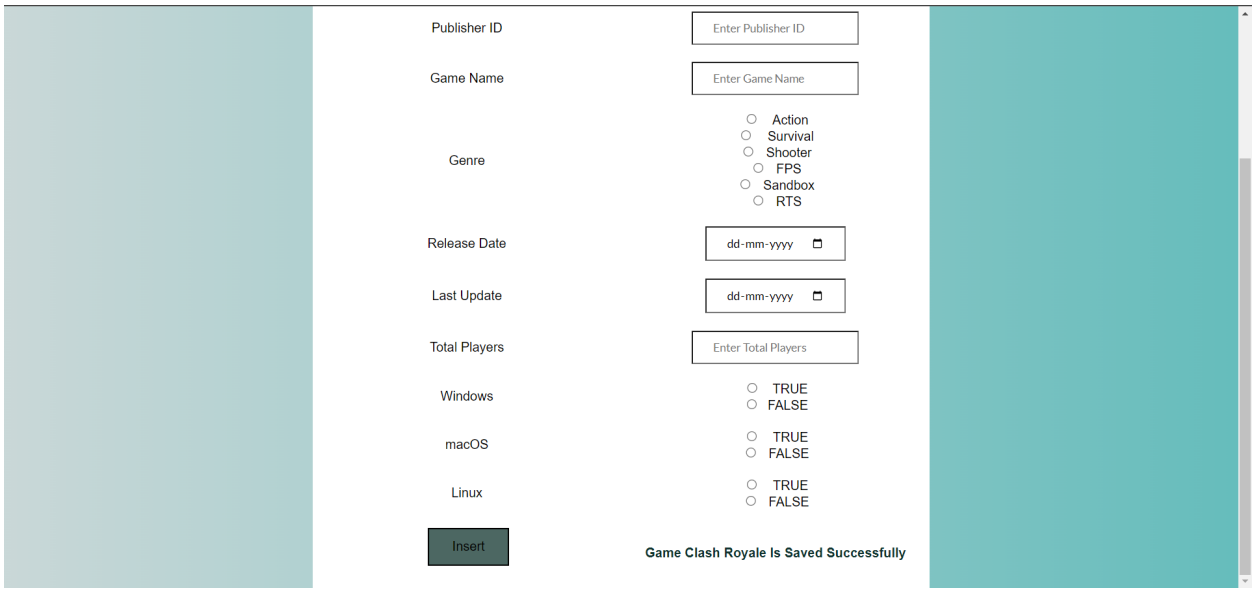


2. Game Database



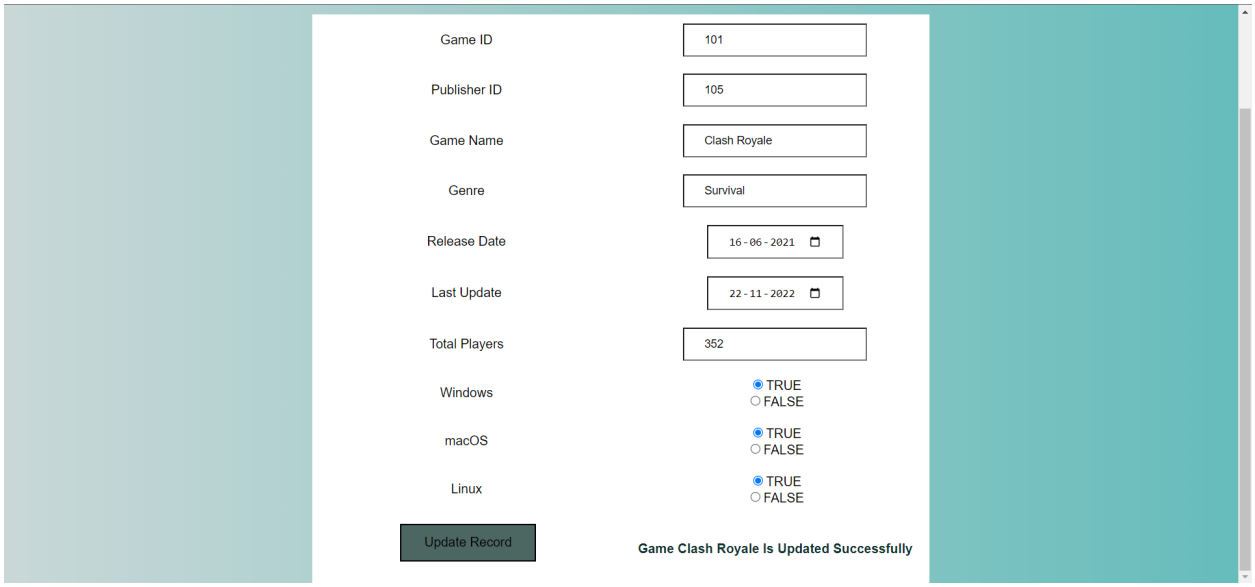
Game ID	Publisher ID	Game Name	Genre	Release Date	Last Update	Total Players	Windows	macOS	Linux		
1	101	Ainyx	Action	May 12, 2021	Feb. 10, 2022	789	FALSE	FALSE	FALSE	Edit ✍	Delete
2	101	Voomm	Survival	May 13, 2021	Feb. 11, 2022	714	FALSE	TRUE	FALSE	Edit ✍	Delete
3	102	Gigashots	Shooter	May 14, 2021	Feb. 12, 2022	769	TRUE	TRUE	TRUE	Edit ✍	Delete
4	102	Riffwire	Survival	May 15, 2021	Feb. 13, 2022	161	FALSE	TRUE	FALSE	Edit ✍	Delete
5	102	Cogibox	FPS	May 16, 2021	Feb. 14, 2022	224	TRUE	FALSE	TRUE	Edit ✍	Delete
6	102	Twitterbeat	RTS	May 17, 2021	Feb. 15, 2022	799	FALSE	TRUE	FALSE	Edit ✍	Delete
7	102	Skipnad	FPS	May 18,	Feb. 16,	871	FALSE	TRUE	FALSE	Edit ✍	Delete

3. Insert a Game Record



				2021	2022						
95	123	Tagopia	FPS	Aug. 14, 2021	May 15, 2022	52	TRUE	FALSE	TRUE	Edit ✍	Delete
96	124	Lajo	RTS	Aug. 15, 2021	May 16, 2022	683	TRUE	FALSE	FALSE	Edit ✍	Delete
97	124	Jetpulse	FPS	Aug. 16, 2021	May 17, 2022	438	FALSE	TRUE	TRUE	Edit ✍	Delete
98	124	Wikizz	Action	Aug. 17, 2021	May 18, 2022	395	TRUE	TRUE	TRUE	Edit ✍	Delete
99	125	Skipstorm	RTS	Aug. 18, 2021	May 19, 2022	850	FALSE	TRUE	TRUE	Edit ✍	Delete
100	125	Flashpoint	Survival	Aug. 19, 2021	May 20, 2022	143	TRUE	FALSE	TRUE	Edit ✍	Delete
101	105	Clash Royale	Survival	June 16, 2021	Nov. 16, 2022	276	TRUE	TRUE	FALSE	Edit ✍	Delete

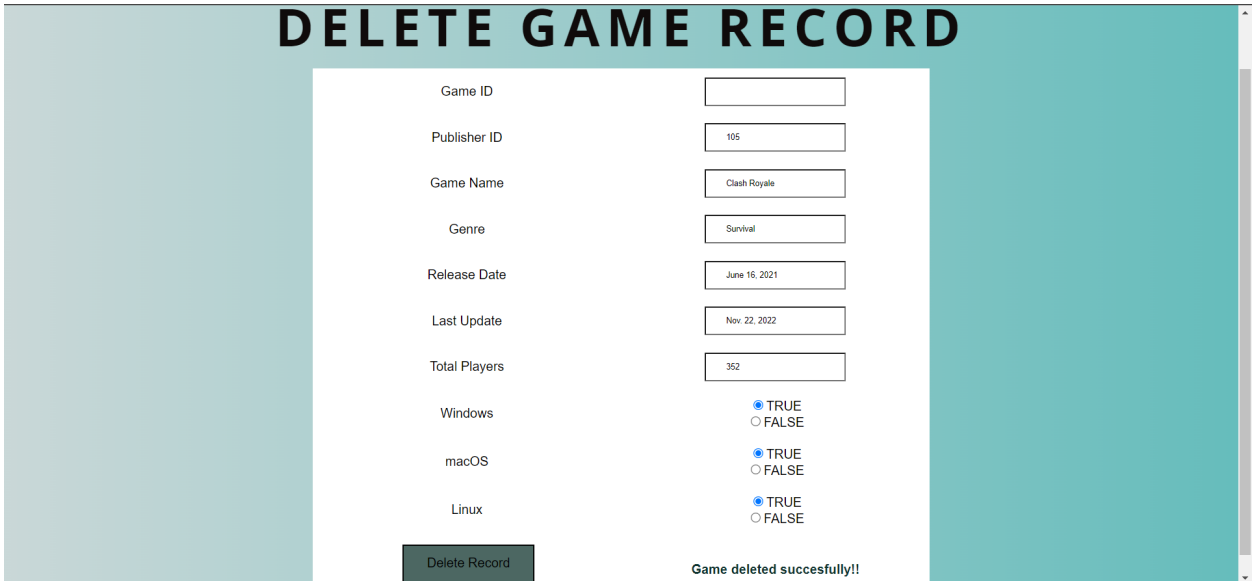
4. Edit a Game Record



96	124	Lajo	RTS	Aug. 15, 2021	May 16, 2022	683	TRUE	FALSE	FALSE	Edit ✍	Delete
97	124	Jetpulse	FPS	Aug. 16, 2021	May 17, 2022	438	FALSE	TRUE	TRUE	Edit ✍	Delete
98	124	Wikizz	Action	Aug. 17, 2021	May 18, 2022	395	TRUE	TRUE	TRUE	Edit ✍	Delete
99	125	Skipstorm	RTS	Aug. 18, 2021	May 19, 2022	850	FALSE	TRUE	TRUE	Edit ✍	Delete
100	125	Flashpoint	Survival	Aug. 19, 2021	May 20, 2022	143	TRUE	FALSE	TRUE	Edit ✍	Delete
101	105	Clash Royale	Survival	June 16, 2021	Nov. 22, 2022	352	TRUE	TRUE	TRUE	Edit ✍	Delete

	game_id [PK] integer	pub_id integer	game_name character varying (100)	genre character varying (100)	release_date date	last_update date	total_players integer	windows character varying (5)	macOS character varying (5)	Linux character varying (5)
93	93	123	Avamba	Shooter	2021-08-12	2022-05-13	159	FALSE	TRUE	TRUE
94	94	123	Twitterlist	Survival	2021-08-13	2022-05-14	601	TRUE	TRUE	TRUE
95	95	123	Tagopia	FPS	2021-08-14	2022-05-15	52	TRUE	FALSE	TRUE
96	96	124	Lajo	RTS	2021-08-15	2022-05-16	683	TRUE	FALSE	FALSE
97	97	124	Jetpulse	FPS	2021-08-16	2022-05-17	438	FALSE	TRUE	TRUE
98	98	124	Wikizz	Action	2021-08-17	2022-05-18	395	TRUE	TRUE	TRUE
99	99	125	Skipstorm	RTS	2021-08-18	2022-05-19	850	FALSE	TRUE	TRUE
100	100	125	Flashpoint	Survival	2021-08-19	2022-05-20	143	TRUE	FALSE	TRUE
101	101	105	Clash Royale	Survival	2021-06-16	2022-11-22	352	TRUE	TRUE	TRUE

5. Delete Game record



95	123	Tagopia	FPS	Aug. 14, 2021	May 15, 2022	52	TRUE	FALSE	TRUE	Edit ✍	Delete
96	124	Lajo	RTS	Aug. 15, 2021	May 16, 2022	683	TRUE	FALSE	FALSE	Edit ✍	Delete
97	124	Jetpulse	FPS	Aug. 16, 2021	May 17, 2022	438	FALSE	TRUE	TRUE	Edit ✍	Delete
98	124	Wikizz	Action	Aug. 17, 2021	May 18, 2022	395	TRUE	TRUE	TRUE	Edit ✍	Delete
99	125	Skipstorm	RTS	Aug. 18, 2021	May 19, 2022	850	FALSE	TRUE	TRUE	Edit ✍	Delete
100	125	Flashpoint	Survival	Aug. 19, 2021	May 20, 2022	143	TRUE	FALSE	TRUE	Edit ✍	Delete

93	93	123	Avamba	Shooter	2021-08-12	2022-05-13	159	FALSE	TRUE	TRUE
94	94	123	Twitterlist	Survival	2021-08-13	2022-05-14	601	TRUE	TRUE	TRUE
95	95	123	Tagopia	FPS	2021-08-14	2022-05-15	52	TRUE	FALSE	TRUE
96	96	124	Lajo	RTS	2021-08-15	2022-05-16	683	TRUE	FALSE	FALSE
97	97	124	Jetpulse	FPS	2021-08-16	2022-05-17	438	FALSE	TRUE	TRUE
98	98	124	Wikizz	Action	2021-08-17	2022-05-18	395	TRUE	TRUE	TRUE
99	99	125	Skipstorm	RTS	2021-08-18	2022-05-19	850	FALSE	TRUE	TRUE
100	100	125	Flashpoint	Survival	2021-08-19	2022-05-20	143	TRUE	FALSE	TRUE

Total rows: 100 of 100 Query complete 00:00:00.144 Ln 1, Col 1

6. **Sort Game record (By total players) (We can sort it by all the columns)**

Game ID	Publisher ID	Game Name	Genre	Release Date	Last Update	Total Players	Windows	macOS	Linux		
18	104	Buzzster	Survival	May 29, 2021	Feb. 27, 2022	13	TRUE	TRUE	FALSE	Edit ✍	Delete
58	113	Latz	RTS	July 8, 2021	April 8, 2022	30	TRUE	FALSE	TRUE	Edit ✍	Delete
95	123	Tagopia	FPS	Aug. 14, 2021	May 15, 2022	52	TRUE	FALSE	TRUE	Edit ✍	Delete
51	112	Wordpedia	FPS	July 1, 2021	April 1, 2022	59	FALSE	FALSE	FALSE	Edit ✍	Delete
45	111	Yoveo	Sandbox	June 25, 2021	March 26, 2022	62	FALSE	TRUE	TRUE	Edit ✍	Delete
64	114	Tambee	Shooter	July 14, 2021	April 14, 2022	73	FALSE	TRUE	FALSE	Edit ✍	Delete
25	105	Skyndu	Survival	June 5, 2021	March 6, 2022	96	TRUE	FALSE	FALSE	Edit ✍	Delete
15	104	Browsetype	FPS	May 26, 2021	Feb. 24, 2022	106	FALSE	FALSE	FALSE	Edit ✍	Delete

7. **Query on Game record:**

Query : Print all the Games which have total players greater than average of total players of all Games

Game ID	Publisher ID	Game Name	Genre	Release Date	Last Update	Total Players	Windows	macOS	Linux
1	101	Ainyx	Action	May 12, 2021	Feb. 10, 2022	789	FALSE	FALSE	FALSE
2	101	Voomm	Survival	May 13, 2021	Feb. 11, 2022	714	FALSE	TRUE	FALSE
3	102	Gigashots	Shooter	May 14, 2021	Feb. 12, 2022	769	TRUE	TRUE	TRUE
6	102	Twitterbeat	RTS	May 17, 2021	Feb. 15, 2022	799	FALSE	TRUE	FALSE
7	102	Skippad	FPS	May 18, 2021	Feb. 16, 2022	871	FALSE	TRUE	FALSE
8	102	JumpXS	Action	May 19, 2021	Feb. 17, 2022	774	FALSE	TRUE	TRUE
9	102	InnoZ	RTS	May 20, 2021	Feb. 18, 2022	886	FALSE	FALSE	FALSE
11	103	Jabbertype	Action	May 22, 2021	Feb. 20, 2022	549	TRUE	TRUE	TRUE
13	103	Kaymbo	FPS	May 24, 2021	Feb. 22, 2022	702	FALSE	TRUE	FALSE
14	103	Zooxo	Shooter	May 25, 2021	Feb. 23, 2022	866	TRUE	FALSE	TRUE
20	105	Skynoodle	Shooter	May 31, 2021	March 1, 2022	982	TRUE	FALSE	FALSE
22	105	Tagchat	Action	June 2, 2021	March 3, 2022	843	FALSE	TRUE	FALSE
26	106	Mybuzz	FPS	June 6, 2021	March 7, 2022	756	TRUE	TRUE	FALSE
28	106	Einti	Action	June 8, 2021	March 9, 2022	972	TRUE	FALSE	FALSE

```
1  select *
2  from public."Game"
3  where public."Game"."total_players" > (select avg(public."Game"."total_players") from public."Game")
4
```

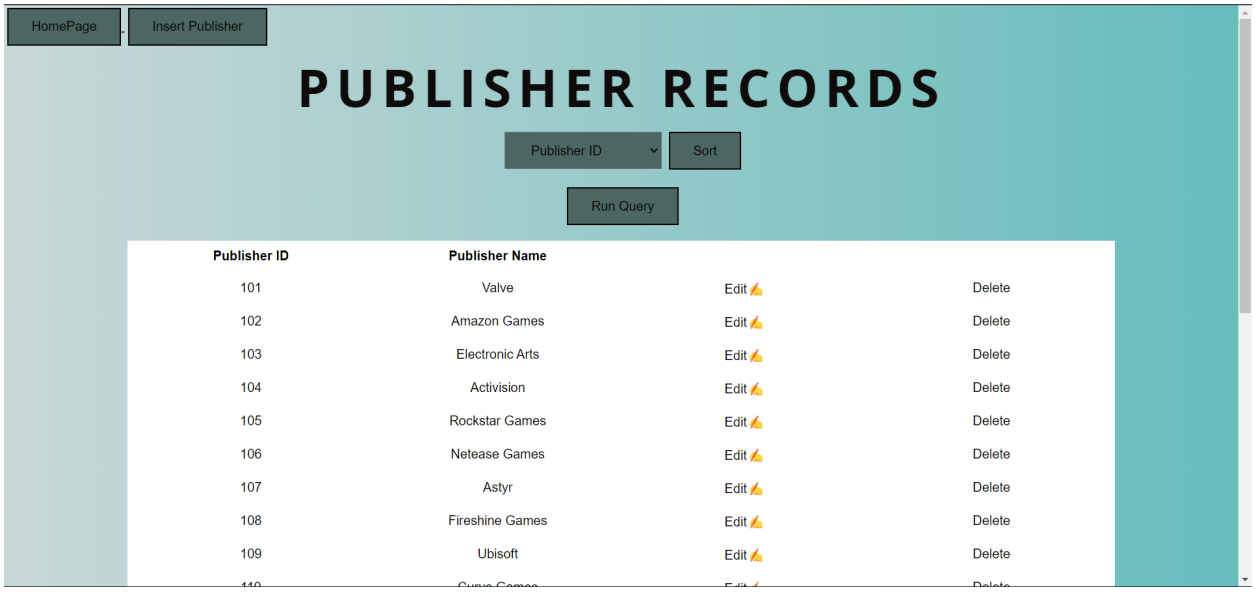
Loading...

Data output | Messages | Notifications

	game_id [PK] integer	pub_id integer	game_name character varying (100)	genre character varying (100)	release_date date	last_update date	total_players integer	windows character varying (5)	macOS character varying (5)	Linux character varying (5)
1	1	101	Ainyx	Action	2021-05-12	2022-02-10	789	FALSE	FALSE	FALSE
2	2	101	Voomm	Survival	2021-05-13	2022-02-11	714	FALSE	TRUE	FALSE
3	3	102	Gigashots	Shooter	2021-05-14	2022-02-12	769	TRUE	TRUE	TRUE
4	6	102	Twitterbeat	RTS	2021-05-17	2022-02-15	799	FALSE	TRUE	FALSE
5	7	102	Skippad	FPS	2021-05-18	2022-02-16	871	FALSE	TRUE	FALSE
6	8	102	JumpXS	Action	2021-05-19	2022-02-17	774	FALSE	TRUE	TRUE
7	9	102	InnoZ	RTS	2021-05-20	2022-02-18	886	FALSE	FALSE	FALSE
8	11	103	Jabbertype	Action	2021-05-22	2022-02-20	549	TRUE	TRUE	TRUE

Total rows: 47 of 47 | Query complete 00:00:00.162 | Ln 4, Col 1

8. Publisher record

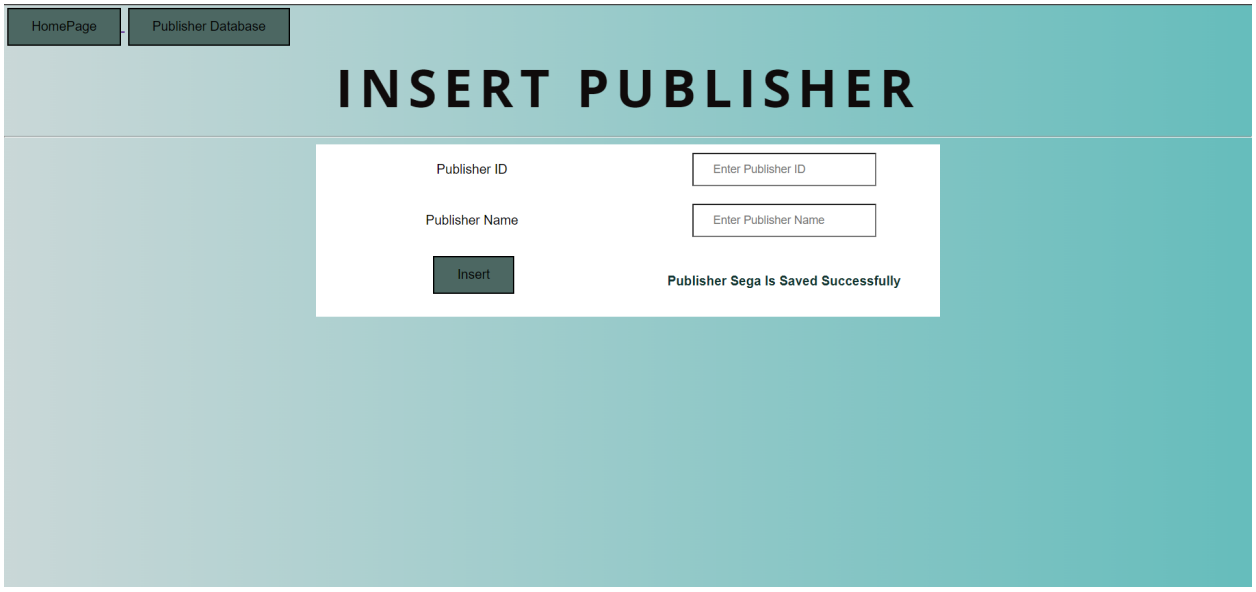


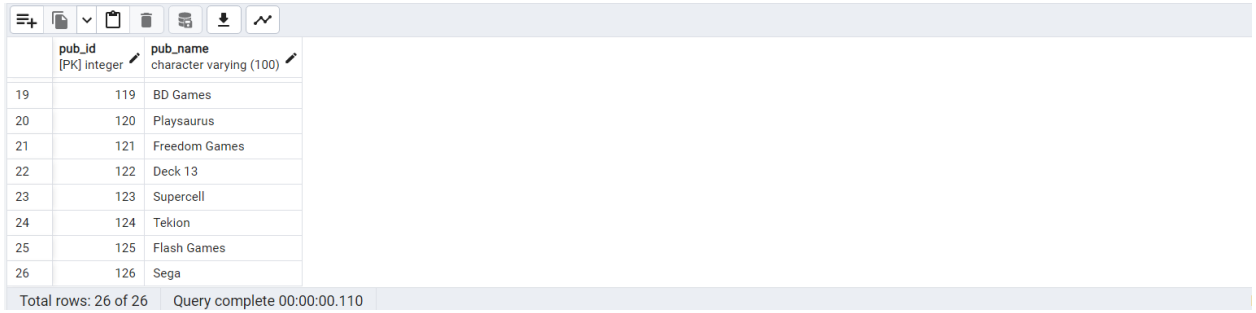
Data output | Messages | Notifications

	pub_id [PK] integer	pub_name character varying (100)
1	101	Valve
2	102	Amazon Games
3	103	Electronic Arts
4	104	Activision
5	105	Rockstar Games
6	106	Netease Games
7	107	Astyr
8	108	Fireshine Games

Total rows: 25 of 25 | Query complete 00:00:00.280

9. Insert Publisher :

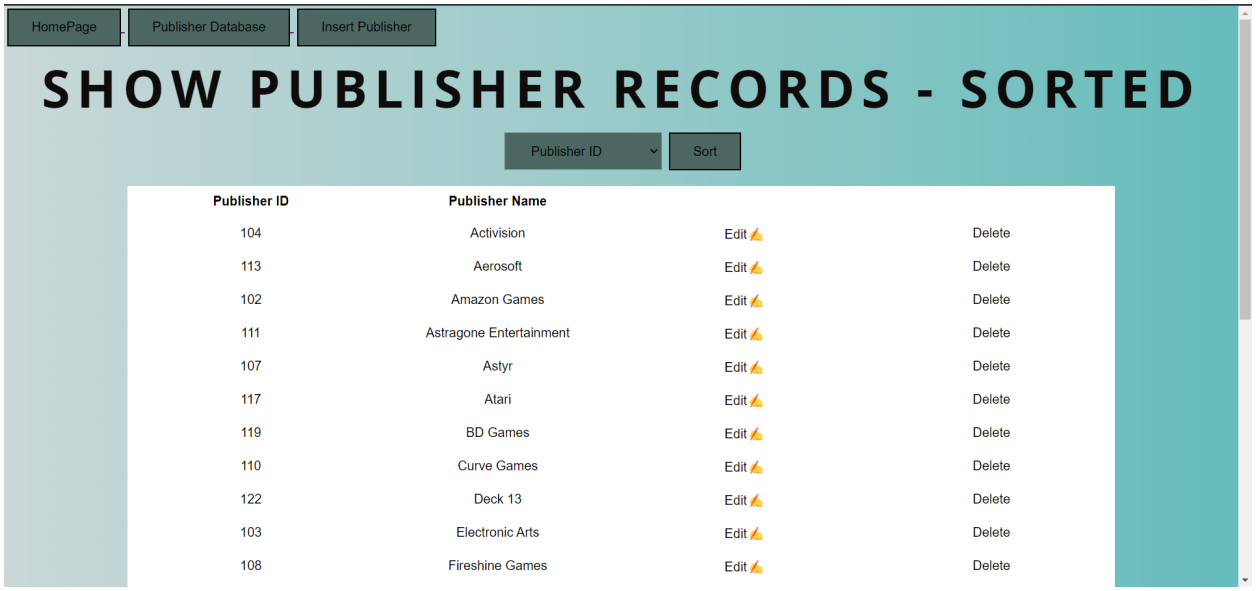




	pub_id [PK] integer	pub_name character varying (100)
19	119	BD Games
20	120	Playsaurus
21	121	Freedom Games
22	122	Deck 13
23	123	Supercell
24	124	Tekion
25	125	Flash Games
26	126	Sega

Total rows: 26 of 26 Query complete 00:00:00.110

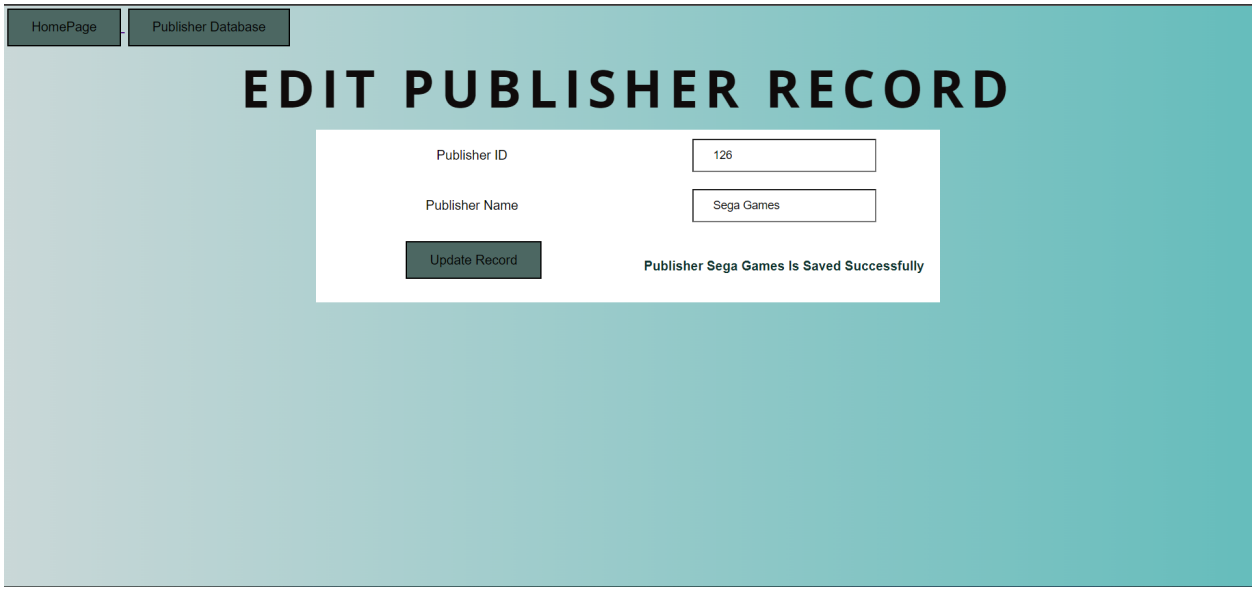
10. **Sort Publisher record (By Publisher name)(We can sort it by all the columns)**

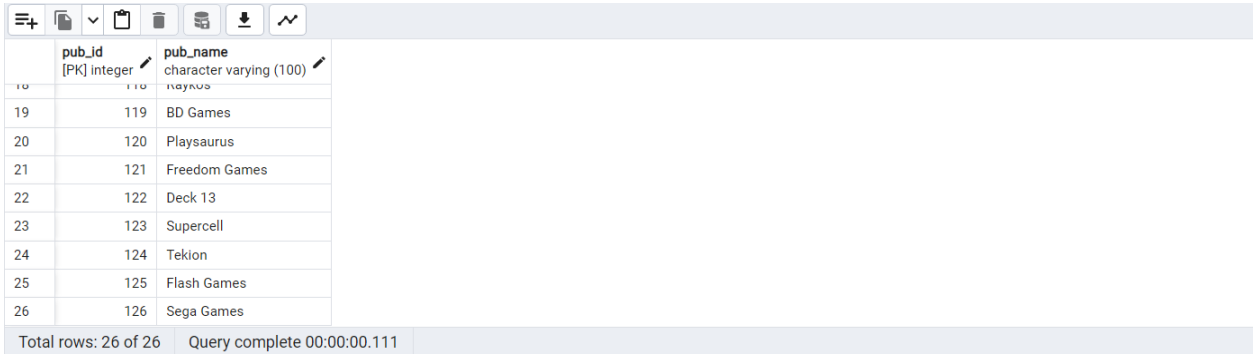


11. **Query for Publisher record :**

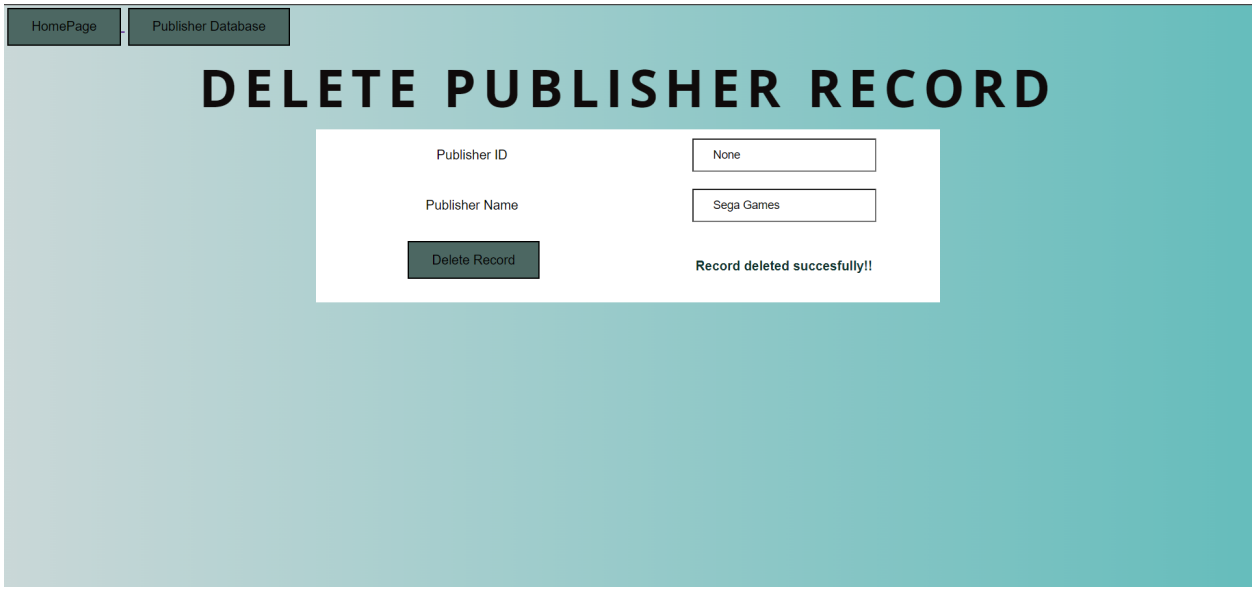
Query : Print all the Game names which are published by Valve Publisher

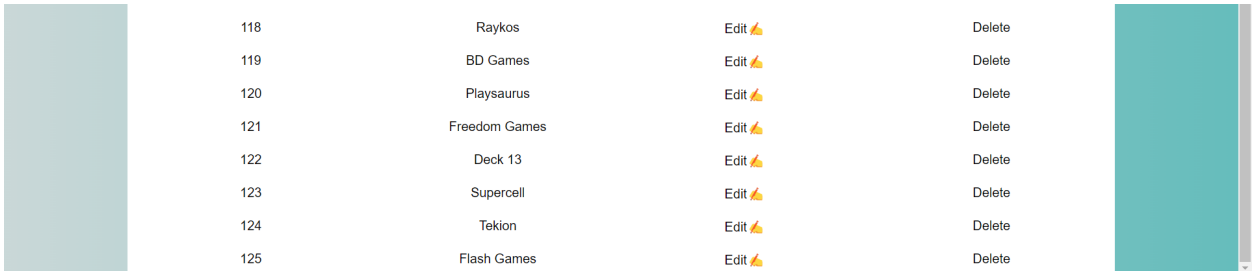
12. Edit Publisher Record :



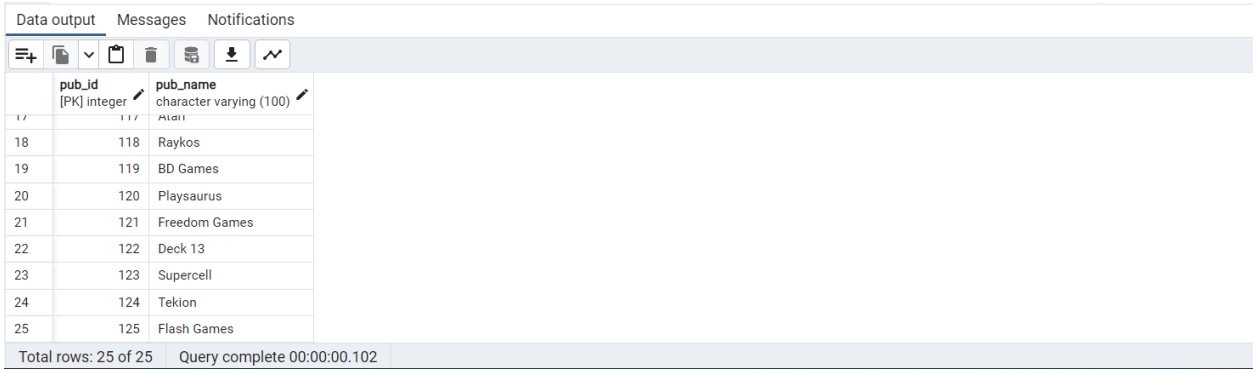


13. Delete Publisher Record :





118	Raykos	Edit ✍	Delete
119	BD Games	Edit ✍	Delete
120	Playsaurus	Edit ✍	Delete
121	Freedom Games	Edit ✍	Delete
122	Deck 13	Edit ✍	Delete
123	Supercell	Edit ✍	Delete
124	Tekion	Edit ✍	Delete
125	Flash Games	Edit ✍	Delete



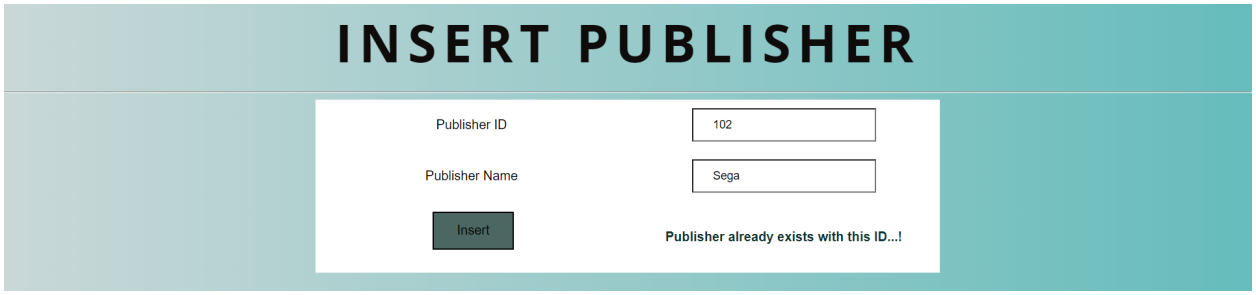
Data output | Messages | Notifications

	pub_id [PK] integer	pub_name character varying (100)
18	118	Raykos
19	119	BD Games
20	120	Playsaurus
21	121	Freedom Games
22	122	Deck 13
23	123	Supercell
24	124	Tekion
25	125	Flash Games

Total rows: 25 of 25 | Query complete 00:00:00.102

14. If any Game ID already exists then it will print that game already exists.

15. If any Publisher ID already exists then it will print this message.



16. **If any Publisher does not exist with the ID which we have entered then it will print this message.**