SQL Project On Gaming Platform Data

Creating the Database : -

```sql
-- Creating DataBase
CREATE DATABASE IF NOT EXISTS gaming_platform;

USE gaming_platform;

-- Creating Tables

-- User Table
CREATE TABLE users (
user_id INT auto_increment PRIMARY KEY,
user_name VARCHAR(50) NOT NULL,
email VARCHAR(50) NOT NULL UNIQUE,
join_date DATE NOT NULL,
age INT
);

-- Games Talbe
CREATE TABLE games (
game_id INT auto_increment PRIMARY KEY,
title VARCHAR(100) NOT NULL,
gener VARCHAR(50) ,
release_date DATE,
price DECIMAL(10,2) DEFAULT(0.0)
);

-- Reviews Table
CREATE TABLE reviews (
review_id INT auto_increment PRIMARY KEY,
user_id INT NOT NULL,
game_id INT NOT NULL,
rating INT CHECK(rating BETWEEN 1 AND 10),
review_text TEXT,
review_date DATE,
FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
FOREIGN KEY (game_id) REFERENCES games(game_id) ON DELETE CASCADE
/*Cascade ensure that when record are deleted in user, games table the corresponding
records in reviews table would be deleted. ( Cascade help in maintaining Data Consistency )
eg. When certain game is deleted, the reveiws and users in the game shall also be deleted other wise
there would be redundant data. */
);

-- Purchase Table
CREATE TABLE purchases (
purchase_id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT NOT NULL,
game_id INT NOT NULL,
purchase_date DATE NOT NULL,
amount_paid DECIMAL(10,2) NOT NULL,
FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE,
FOREIGN KEY (game_id) REFERENCES games(game_id) ON DELETE CASCADE
);

-- Inserting Data Into Tables
INSERT INTO Users (user_name, email, join_date, age) VALUES
('gamer_one', 'gamer1@example.com', '2023-05-10', 25),
('player_two', 'player2@example.com', '2024-01-15', 30),
('pro_gamer', 'progamer@example.com', '2022-11-20', 22),
('casual_player', 'casual@example.com', '2023-07-25', 28),
('elite_shooter', 'elite@example.com', '2024-03-30', 27);
-- Here we aren't inserting user_id as it is auto increment and would be assigned to each row automatically.

INSERT INTO Games (title, gener, release_date, price) VALUES
('Battle Quest', 'Action', '2023-09-15', 59.99),
('Mystic Lands', 'RPG', '2022-06-20', 49.99),
('Speed Racer', 'Racing', '2024-02-10', 39.99),
('Puzzle Mania', 'Puzzle', '2023-12-05', 19.99),
('Space Odyssey', 'Sci-Fi', '2024-07-22', 69.99),
('Fantasy World', 'Adventure', '2023-03-18', 29.99),
('Horror Nights', 'Horror', '2024-04-01', 24.99),
('Strategy Master', 'Strategy', '2022-10-30', 34.99),
('Arcade Legends', 'Arcade', '2023-08-08', 14.99),
('Sports Pro', 'Sports', '2024-05-16', 44.99);

INSERT INTO Reviews (user_id, game_id, rating, review_text, review_date) VALUES
(1, 1, 9, 'Amazing graphics and gameplay!', '2023-09-20'),
(2, 2, 8, 'Engaging story but a bit repetitive.', '2024-01-20'),
(3, 3, 10, 'Best racing game I have ever played!', '2024-02-15'),
```

```
77    (3, 3, 10, 'Best racing game I have ever played!', '2024-02-15'),
78    (4, 4, 7, 'Good puzzles but lacks variety.', '2023-12-10'),
79    (5, 5, 9, 'Stunning visuals and immersive experience.', '2024-07-25'),
80    (1, 6, 8, 'Fun adventure with great characters.', '2023-03-25'),
81    (2, 7, 6, 'Too scary for my taste.', '2024-04-05'),
82    (3, 8, 7, 'Solid strategy mechanics.', '2022-11-05'),
83    (4, 9, 8, 'Nostalgic and entertaining.', '2023-08-15'),
84    (5, 10, 9, 'Realistic sports simulation!', '2024-05-20');
85
86    INSERT INTO Purchases (user_id, game_id, purchase_date, amount_paid) VALUES
87    (1, 1, '2023-09-16', 59.99),
88    (1, 6, '2023-03-19', 29.99),
89    (2, 2, '2024-01-18', 49.99),
90    (2, 7, '2024-04-02', 24.99),
91    (3, 3, '2024-02-12', 39.99),
92    (3, 8, '2022-11-22', 34.99),
93    (4, 4, '2023-12-07', 19.99),
94    (4, 9, '2023-08-10', 14.99),
95    (5, 5, '2024-07-25', 69.99),
96    (5, 10, '2024-05-17', 44.99);
97
98    Select * From Reviews;
99
100
```

Basic Select Statements :

1) Selecting all columns from table

Query :

SELECT ***** FROM users;

Results :

user_id	user_name	email	join_date	age
1	gamer_one	gamer1@example.com	2023-05-10	25
2	player_two	player2@example.com	2024-01-15	30
3	pro_gamer	progamer@example.com	2022-11-20	22
4	casual_player	casual@example.com	2023-07-25	28
5	elite_shooter	elite@example.com	2024-03-30	27
NULL	NULL	NULL	NULL	NULL

2) Selecting Specific Columns from the table

Quert :

SELECT **user_id,user_name** FROM users;

Results :

user_id	user_name
1	gamer_one
2	player_two
3	pro_gamer
4	casual_player
5	elite_shooter
NULL	NULL

3) Selecting Distinct Columns From the table

Ouery :

SELECT **DISTINCT** gener from games;

Result :

gener
Action
RPG
Racing
Puzzle
Sci-Fi
Adventure
Horror
Strategy
Arcade
Sports

Filtering Data with WHERE Clause :

1) Select data on basic where clause

Query :

SELECT * FROM GAMES **WHERE GENER = 'Action'**;

Result :

game_id	title	gener	release_date	price
1	Battle Quest	Action	2023-09-15	59.99
NULL	NULL	NULL	NULL	NULL

2) Using Comparison Operator

Query :

Select * from users where age **>=** 25;

Result :

user_id	user_name	email	join_date	age
1	gamer_one	gamer1@example.com	2023-05-10	25
2	player_two	player2@example.com	2024-01-15	30
4	casual_player	casual@example.com	2023-07-25	28
5	elite_shooter	elite@example.com	2024-03-30	27
NULL	NULL	NULL	NULL	NULL

3) Using Logical Operator

Query :

SELECT * FROM users WHERE age>=25 **and** join_date > '2024-01-01';

Result :

user_id	user_name	email	join_date	age
2	player_two	player2@example.com	2024-01-15	30
5	elite_shooter	elite@example.com	2024-03-30	27
NULL	NULL	NULL	NULL	NULL

4) Using Like Pattern matching

Ouery :

Select * from users where email **like '%**@example.com';

Explanation : Here the % is wildcard that replace anything that is before it

Result :

user_id	user_name	email	join_date	age
1	gamer_one	gamer1@example.com	2023-05-10	25
2	player_two	player2@example.com	2024-01-15	30
3	pro_gamer	progamer@example.com	2022-11-20	22
4	casual_player	casual@example.com	2023-07-25	28
5	elite_shooter	elite@example.com	2024-03-30	27
NULL	NULL	NULL	NULL	NULL

5) Using In For Multiple Values

Ouery :

SELECT * FROM Games WHERE gener **IN ('Action', 'RPG', 'Strategy')**;

Explanation : In selects all rows where value is action ,rpg or strategy.

Result :

game_id	title	gener	release_date	price
1	Battle Quest	Action	2023-09-15	59.99
2	Mystic Lands	RPG	2022-06-20	49.99
8	Strategy Master	Strategy	2022-10-30	34.99
NULL	NULL	NULL	NULL	NULL

6) Using Between

Query :

SELECT * FROM Users WHERE join_date **BETWEEN** '2023-01-01' AND '2023-12-31';

Result :

user_id	user_name	email	join_date	age
1	gamer_one	gamer1@example.com	2023-05-10	25
4	casual_player	casual@example.com	2023-07-25	28
NULL	NULL	NULL	NULL	NULL

Sorting results with order by

1) Sort Data By Ascending order

Query :

SELECT * FROM Games **ORDER BY** release_date;

Explanation : It is by default in ascending order

Results :

game_id	title	gener	release_date	price
2	Mystic Lands	RPG	2022-06-20	49.99
8	Strategy Master	Strategy	2022-10-30	34.99
6	Fantasy World	Adventure	2023-03-18	29.99
9	Arcade Legends	Arcade	2023-08-08	14.99
1	Battle Quest	Action	2023-09-15	59.99
4	Puzzle Mania	Puzzle	2023-12-05	19.99
3	Speed Racer	Racing	2024-02-10	39.99
7	Horror Nights	Horror	2024-04-01	24.99
10	Sports Pro	Sports	2024-05-16	44.99
5	Space Odyssey	Sci-Fi	2024-07-22	69.99
NULL	NULL	NULL	NULL	NULL

2) Sorting in Descending order

Query :

SELECT * FROM Users **ORDER BY** age **DESC**;

Results :

user_id	user_name	email	join_date	age
2	player_two	player2@example.com	2024-01-15	30
4	casual_player	casual@example.com	2023-07-25	28
5	elite_shooter	elite@example.com	2024-03-30	27
1	gamer_one	gamer1@example.com	2023-05-10	25
3	pro_gamer	progamer@example.com	2022-11-20	22
NULL	NULL	NULL	NULL	NULL

3) Sorting by multiple columns

Query :

SELECT * FROM Reviews **ORDER BY game_id ASC, rating DESC**;

Explanation : It will first sort by game id in ascending order and then by rating in descending order

Results :

review_id	user_id	game_id	rating	review_text	review_date
1	1	1	9	Amazing graphics and gameplay!	2023-09-20
2	2	2	8	Engaging story but a bit repetitive.	2024-01-20
3	3	3	10	Best racing game I have ever played!	2024-02-15
4	4	4	7	Good puzzles but lacks variety.	2023-12-10
5	5	5	9	Stunning visuals and immersive experience.	2024-07-25
6	1	6	8	Fun adventure with great characters.	2023-03-25
7	2	7	6	Too scary for my taste.	2024-04-05
8	3	8	7	Solid strategy mechanics.	2022-11-05
9	4	9	8	Nostalgic and entertaining.	2023-08-15
10	5	10	9	Realistic sports simulation!	2024-05-20
NULL	NULL	NULL	NULL	NULL	NULL

Limiting Results with LIMIT :

1) Limiting Result

Query :

Select * From Users **limit 2**;

Results :

user_id	user_name	email	join_date	age
1	gamer_one	gamer1@example.com	2023-05-10	25
2	player_two	player2@example.com	2024-01-15	30
NULL	NULL	NULL	NULL	NULL

2) Limit WIth Offset

Query :

Select * From Users **limit 2 offset 2**;

Explanation : The **offset parameter will skip the first 2 rows** and next 2 rows would be considered.

Results :

user_id	user_name	email	join_date	age
3	pro_gamer	progamer@example.com	2022-11-20	22
4	casual_player	casual@example.com	2023-07-25	28
NULL	NULL	NULL	NULL	NULL

Aggregate Functions :

1) Count

Query :

Select **Count(*)** as Total_users from users

Results :

Total_users
5

2) Sum

Query :

Select **sum(amount_paid)** as total_revenue from purchases;

Results :

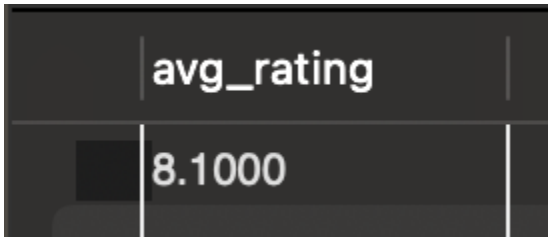
total_revenue
389.90

3) Avg

Query :

Select **avg(rating)** as avg_rating from reviews;

Results :



avg_rating
8.1000

4) Min Max

Query :

Select **Min(price)** as cheapest_game, **Max(price)** as expensive_game from games;

Results :

cheapest_game	expensive_game
14.99	69.99

Group by :

1) Basic Group By

Query :

```
SELECT gener, COUNT(*) AS number_of_games
FROM Games GROUP BY gener;
```

Results :

gener	number_of_gam...
Action	1
RPG	1
Racing	1
Puzzle	1
Sci-Fi	1
Adventure	1
Horror	1
Strategy	1
Arcade	1
Sports	1

2) Group By with multiple columns :

Query :

```
SELECT user_id, game_id, COUNT(*) AS purchase_count
FROM Purchases GROUP BY user_id, game_id;
```

Results :

user_id	game_id	purchase_cou...
1	1	1
1	6	1
2	2	1
2	7	1
3	3	1
3	8	1
4	4	1
4	9	1
5	5	1
5	10	1

Having Clause

1) Filtering Using Having Clause

Query :

```
SELECT gener, COUNT(*) AS number_of_games
FROM Games GROUP BY gener HAVING COUNT(*) > 2;
```

2) Filtering Using Aggregate function

Query :

```
SELECT user_id, AVG(rating) AS average_rating
FROM Reviews GROUP BY user_id HAVING AVG(rating) >= 8;
```

Results :

user_id	average_rating
1	8.5000
3	8.5000
5	9.0000

Joins

1) Inner Joins :

Query :

```
Select users.user_name,games.title,reviews.rating from Reviews
Inner Join Users on reviews.user_id = users.user_id
Inner join games on reviews.game_id = games.game_id;
```

Results :

user_name	title	rating
gamer_one	Battle Quest	9
gamer_one	Fantasy World	8
player_two	Mystic Lands	8
player_two	Horror Nights	6
pro_gamer	Speed Racer	10
pro_gamer	Strategy Master	7
casual_player	Puzzle Mania	7
casual_player	Arcade Legends	8
elite_shooter	Space Odyssey	9
elite_shooter	Sports Pro	9

2) Left Join :

Query :

```
SELECT Users.user_name, Purchases.amount_paid
FROM Users LEFT JOIN Purchases ON Users.user_id =Purchases.user_id;
```

Results :

user_name	amount_paid
gamer_one	59.99
gamer_one	29.99
player_two	49.99
player_two	24.99
pro_gamer	39.99
pro_gamer	34.99
casual_player	19.99
casual_player	14.99
elite_shooter	69.99
elite_shooter	44.99
karan	NULL

3) Right Join :

Query :

```
SELECT Games.title, Purchases.amount_paid
FROM Purchases RIGHT JOIN Games ON Purchases.game_id =
Games.game_id;
```

Results :

title	amount_paid
Battle Quest	59.99
Mystic Lands	49.99
Speed Racer	39.99
Puzzle Mania	19.99
Space Odyssey	69.99
Fantasy World	29.99
Horror Nights	24.99
Strategy Master	34.99
Arcade Legends	14.99
Sports Pro	44.99
karan	NULL

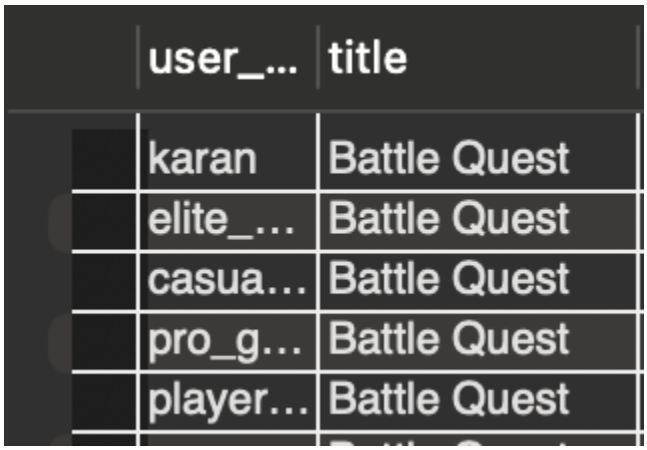
4) Cross Join

Query :

```
SELECT Users.user_name,games.title
FROM Users CROSS JOIN Games;
```

Explanation : Creates cross values for each user

Results :



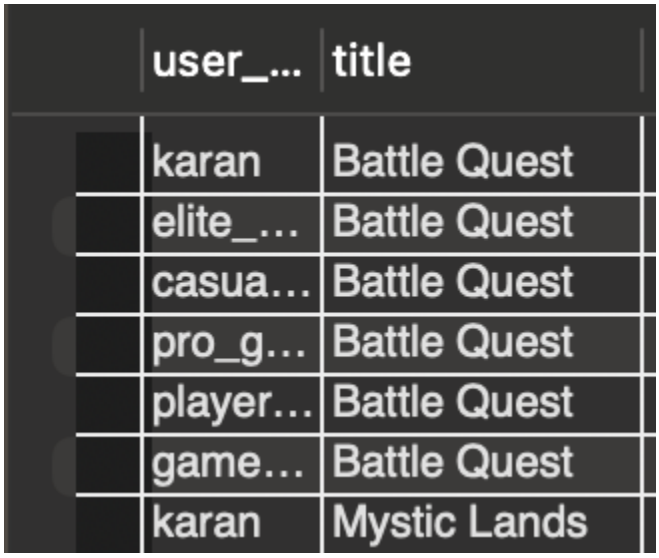
	user_...	title
	karan	Battle Quest
	elite_...	Battle Quest
	casua...	Battle Quest
	pro_g...	Battle Quest
	player...	Battle Quest

5) Full outer join

Query :

```
SELECT Games.title, Purchases.amount_paid FROM Purchases
RIGHT JOIN Games ON Purchases.game_id = Games.game_id
union all
SELECT Games.title, Purchases.amount_paid FROM Purchases
Left JOIN Games ON Purchases.game_id = Games.game_id;
```

Results :



	user_...	title
	karan	Battle Quest
	elite_...	Battle Quest
	casua...	Battle Quest
	pro_g...	Battle Quest
	player...	Battle Quest
	game...	Battle Quest
	karan	Mystic Lands

Combining Clauses for Complex Queries

1) Where with joins

Query :

```
SELECT Users.username, Users.email FROM Users
INNER JOIN Purchases ON Users.user_id = Purchases.user_id
WHERE Purchases.game_id = 3;
```

Results :

user_name	email
pro_gamer	progamer@example.com

2) To find null within joins

Query :

```
SELECT Games.title FROM Games
LEFT JOIN Reviews ON Games.game_id = Reviews.game_id
WHERE Reviews.review_id IS NULL;
```

Results :

title
karan

3) Finding Monthly revenue

Query :

```
SELECT DATE_FORMAT(purchase_date, '%Y-%m') AS month,
SUM(amount_paid) AS monthly_revenue FROM Purchases
GROUP BY month ORDER BY month;
```

Results :

month	monthly_revenue
2022-11	34.99
2023-03	29.99
2023-08	14.99
2023-09	59.99
2023-12	19.99
2024-01	49.99
2024-02	39.99
2024-04	24.99
2024-05	44.99
2024-07	69.99

Best Practices

1) Uses aliases

Query :

SELECT u.username, g.title FROM **Users AS u**
INNER JOIN **Purchases AS p** ON u.user_id = p.user_id
INNER JOIN **Games AS g** ON p.game_id = g.game_id;

More Practice Queries

1) To find most recent joined user

Query :

SELECT * FROM Users
ORDER BY join_date **DESC LIMIT 1**;

user_id	user_name	email	join_date	age
5	elite_shooter	elite@example.com	2024-03-30	27
NULL	NULL	NULL	NULL	NULL

2) To find games released after specific date

Query :

SELECT * FROM Games WHERE **release_date > '2023-01-01**';

game_id	title	gener	release_date	price
1	Battle Quest	Action	2023-09-15	59.99
3	Speed Racer	Racing	2024-02-10	39.99
4	Puzzle Mania	Puzzle	2023-12-05	19.99
5	Space Odys…	Sci-Fi	2024-07-22	69.99
6	Fantasy World	Adventure	2023-03-18	29.99
7	Horror Nights	Horror	2024-04-01	24.99
9	Arcade Leg…	Arcade	2023-08-08	14.99
10	Sports Pro	Sports	2024-05-16	44.99
11	karan	Action	2023-09-15	59.99
NULL	NULL	NULL	NULL	NULL

3) To Find user with most purchases

Query :

SELECT u.username, COUNT(p.purchase_id) AS purchase_count
FROM Users u **INNER JOIN** Purchases p **ON** u.user_id = p.user_id
GROUP BY u.username ORDER BY purchase_count DESC LIMIT 1;

user_name	purchase_cou...
gamer_one	2