National University of Computer and Emerging Sciences

Database Project Report

Video Game Sales Analysis Dashboard

Name	Roll Number	Sub-section
Momina Humayun	24L-7349	1A

CL219 Database Systems Lab

Fall 2024

Department of Management Sciences

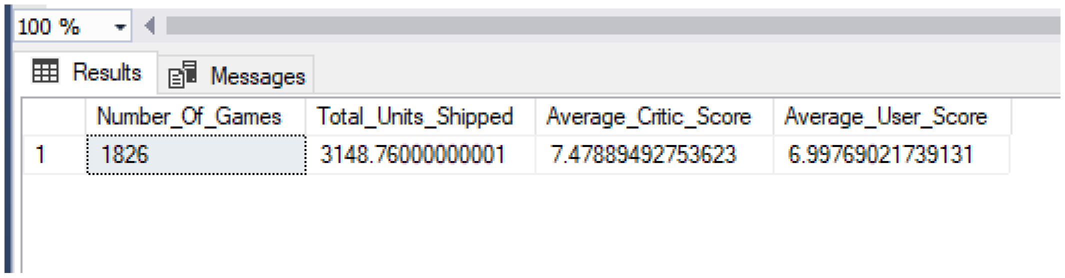
FAST-NU, Lahore, Pakistan

Video Game Sales Analysis Queries

1. Sales Overview Dashboard

Display total units shipped, average critic score, and user score for the games, providing a snapshot of overall industry performance.

```
SELECT COUNT(games.Name) AS Number_Of_Games,
SUM(scores.Total_Shipped) AS Total_Units_Shipped,
AVG(scores.Critic_Score) AS Average_Critic_Score,
AVG(scores.User_Score) AS Average_User_Score
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
```



	Number_Of_Games	Total_Units_Shipped	Average_Critic_Score	Average_User_Score
1	1826	3148.76000000001	7.47889492753623	6.99769021739131

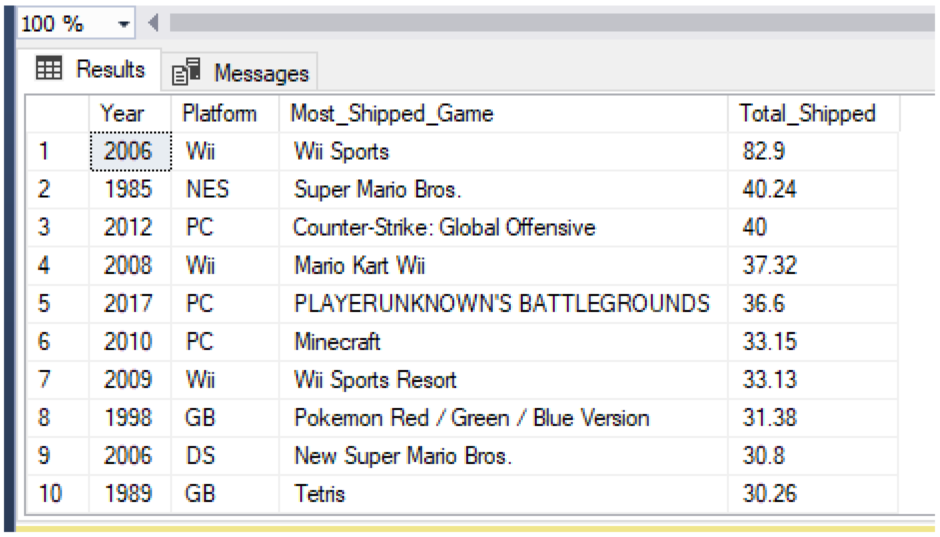
The SQL query calculates four key metrics to provide an overview of video game industry performance: the total number of games, total units shipped, average critic score, and average user score. It combines data from the games table, which includes game details, and the scores table, which tracks shipment numbers and review scores.

The query uses COUNT(games.Name) to determine the total number of games, while SUM(scores.Total_Shipped) calculates the overall units shipped as a measure of sales performance. To evaluate game quality, it computes the average critic score (AVG(scores.Critic_Score)) and user score (AVG(scores.User_Score)). A WHERE clause filters out records with missing critic or user scores (IS NOT NULL), ensuring the calculations are accurate and based on complete data.

2. Best-Selling Games

Identify the top-selling games based on total shipped across different years and platforms, offering insights into the most commercially successful titles.

```
SELECT games.Year, platforms.Platform, games.Name AS
Most_Shipped_Game, scores.Total_Shipped
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
JOIN platforms ON games.Platform_ID = platforms.Platform_ID
WHERE scores.Total_Shipped IN (SELECT MAX(s1.Total_Shipped) FROM
games g1
JOIN scores s1 ON g1.Game_ID = s1.Game_ID
WHERE g1.Year = games.Year OR g1.Platform_ID = games.Platform_ID
GROUP BY g1.Year, g1.Platform_ID)
ORDER BY scores.Total_Shipped DESC
```



	Year	Platform	Most_Shipped_Game	Total_Shipped
1	2006	Wii	Wii Sports	82.9
2	1985	NES	Super Mario Bros.	40.24
3	2012	PC	Counter-Strike: Global Offensive	40
4	2008	Wii	Mario Kart Wii	37.32
5	2017	PC	PLAYERUNKNOWN'S BATTLEGROUNDS	36.6
6	2010	PC	Minecraft	33.15
7	2009	Wii	Wii Sports Resort	33.13
8	1998	GB	Pokemon Red / Green / Blue Version	31.38
9	2006	DS	New Super Mario Bros.	30.8
10	1989	GB	Tetris	30.26

The SQL query identifies the top-selling video games based on total units shipped, analyzing data across different years and platforms. It combines information from three tables: games, which contains game details, scores, which tracks shipment numbers, and platforms, which specifies the platform for each game.

The query retrieves the year, platform, game name, and total units shipped for the games. It uses a subquery to find the game with the highest units shipped for each year and platform combination. The subquery calculates the maximum total shipped (MAX(s1.Total_Shipped)) for each specific year and platform, ensuring that only the top-selling games for each year and platform are included.

The WHERE clause ensures the games returned in the main query match the top-selling titles from the subquery. The results are ordered by the total units shipped in descending order, highlighting the most commercially successful games first.

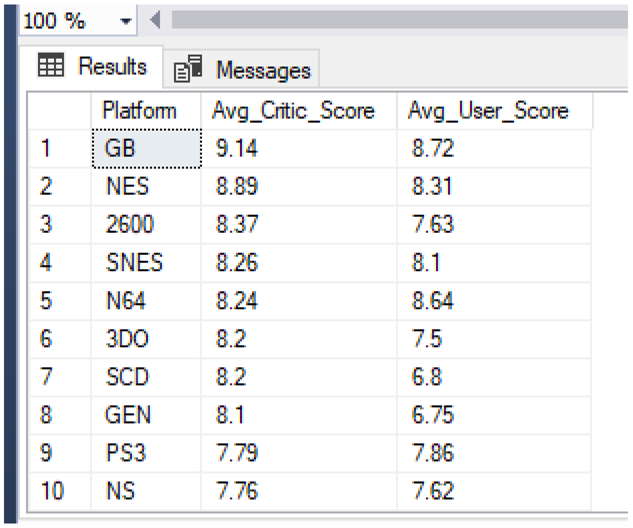
The names of the tables are different (i.e s1, g1) in the nested queries since this is a correlated subquery. To avoid confusion, the subquery needs its own aliases so that it can refer to the same tables without conflict. For example, the outer query uses games.Year, but within the subquery, we need to refer to g1.Year to avoid ambiguity.

There are 104 rows of data in the output but for relevance purposes, there are 10 rows of output shown. It shows that the top selling game was in 2006 on Wii as a platform, and the game was Wii Sports according to the total amount shipped of games.

3. Game Performance by Platform

Compare average critic and user scores across various platforms (e.g., PC, NES, Wii), revealing which platforms host higher-rated games.

```
SELECT platforms.Platform,
       ROUND(AVG(scores.Critic_Score), 2) AS Avg_Critic_Score,
       ROUND(AVG(scores.User_Score), 2) AS Avg_User_Score
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
JOIN platforms ON games.Platform_ID = platforms.Platform_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
GROUP BY platforms.Platform
ORDER BY Avg_Critic_Score DESC
```



100 %

Results | Messages

	Platform	Avg_Critic_Score	Avg_User_Score
1	GB	9.14	8.72
2	NES	8.89	8.31
3	2600	8.37	7.63
4	SNES	8.26	8.1
5	N64	8.24	8.64
6	3DO	8.2	7.5
7	SCD	8.2	6.8
8	GEN	8.1	6.75
9	PS3	7.79	7.86
10	NS	7.76	7.62

The SQL query analyzes the performance of video games across different platforms by comparing the average critic and user scores. It focuses on determining which platforms (e.g., PC, NES, Wii) host higher-rated games. This is achieved by calculating the average critic and user scores for games on each platform, based on data from three tables: games, scores, and platforms.

The query first joins the games table with the scores table to access the review scores (critic and user scores) for each game, and then joins the platforms table to associate each game with its respective platform. It calculates the average critic score (AVG(scores.Critic_Score)) and the average user score (AVG(scores.User_Score)) for each platform. The ROUND() function is used to limit the averages to two decimal places.

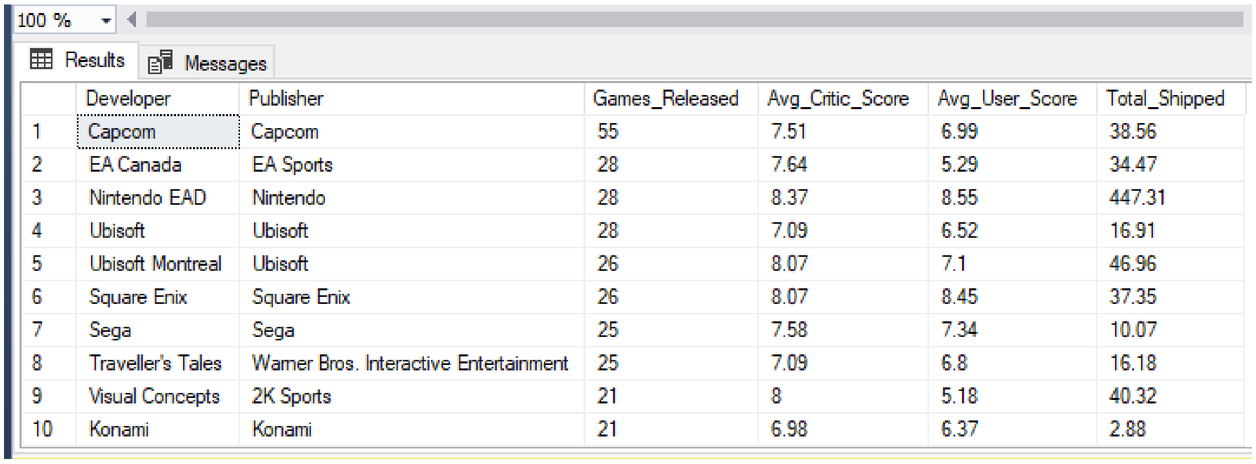
To ensure that only games with available review data are included, the query filters out records with missing critic or user scores by using the WHERE clause (IS NOT NULL). The GROUP BY clause groups the results by platform, so the averages are calculated separately for each platform. Finally, the ORDER BY clause sorts the results by the average critic score in descending order, highlighting platforms with the highest-rated games based on critic scores.

There were 28 platforms shown in the output but for relevance purposes, only the first 10 rows are shown. The GB platform was the highest-rated out of all them. The lowest average critic score was 6.99 and the lowest average user score was 6.4, which suggests that there was a higher threshold for the average critic score.

4. Publisher and Developer Analysis

Examine the top publishers and developers based on the number of games released and their average scores, highlighting those with the highest-rated and most successful games.

```
SELECT developers.Developer, publishers.Publisher,
COUNT(games.Game_ID) AS Games_Released,
ROUND(AVG(ISNULL(scores.Critic_Score, 0)), 2)
AS Avg_Critic_Score, ROUND(AVG(ISNULL(scores.User_Score, 0)), 2) AS
Avg_User_Score, ROUND(SUM(ISNULL(scores.Total_Shipped, 0)), 2) AS
Total_Shipped
FROM games JOIN developers ON games.Developer_ID =
developers.Developer_ID JOIN publishers ON games.Publisher_ID =
publishers.Publisher_ID
JOIN scores ON scores.Game_ID = games.Game_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
GROUP BY developers.Developer, publishers.Publisher
ORDER BY Games_Released DESC
```



	Developer	Publisher	Games_Released	Avg_Critic_Score	Avg_User_Score	Total_Shipped
1	Capcom	Capcom	55	7.51	6.99	38.56
2	EA Canada	EA Sports	28	7.64	5.29	34.47
3	Nintendo EAD	Nintendo	28	8.37	8.55	447.31
4	Ubisoft	Ubisoft	28	7.09	6.52	16.91
5	Ubisoft Montreal	Ubisoft	26	8.07	7.1	46.96
6	Square Enix	Square Enix	26	8.07	8.45	37.35
7	Sega	Sega	25	7.58	7.34	10.07
8	Traveller's Tales	Warner Bros. Interactive Entertainment	25	7.09	6.8	16.18
9	Visual Concepts	2K Sports	21	8	5.18	40.32
10	Konami	Konami	21	6.98	6.37	2.88

The SQL query identifies the top publishers and developers based on the number of games released and their average scores. It also calculates the total units shipped for each publisher and developer. The query joins the games, developers, publishers, and scores tables.

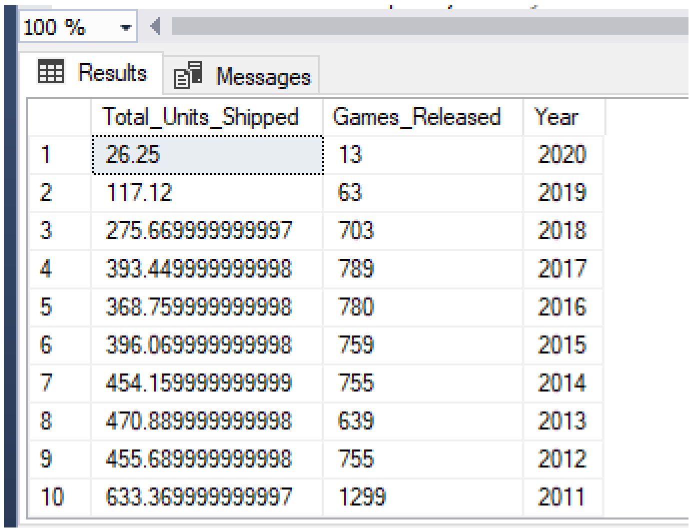
It counts the games released (COUNT(games.Game_ID)), calculates average critic (AVG(ISNULL(scores.Critic_Score, 0))) and user scores (AVG(ISNULL(scores.User_Score, 0))), and sums total units shipped (SUM(ISNULL(scores.Total_Shipped, 0))). The ISNULL function handles missing values as zero.

The query filters out games without critic or user scores, groups by developer and publisher, and orders by the number of games released in descending order. This highlights the most successful publishers and developers. There are 1132 rows of data in the output, the first 10 are shown for relevance.

5. Yearly Sales and Release Trends

Analyse total units shipped and the number of games released per year to identify peak periods in the video game industry.

```sql
SELECT SUM(scores.Total_Shipped) AS Total_Units_Shipped,
COUNT(games.Game_ID) AS Games_Released, games.Year
FROM games JOIN scores ON scores.Game_ID = games.Game_ID
GROUP BY games.Year
ORDER BY games.Year DESC
```



100 %

Results | Messages

	Total_Units_Shipped	Games_Released	Year
1	26.25	13	2020
2	117.12	63	2019
3	275.669999999997	703	2018
4	393.449999999998	789	2017
5	368.759999999998	780	2016
6	396.069999999998	759	2015
7	454.159999999999	755	2014
8	470.889999999998	639	2013
9	455.689999999998	755	2012
10	633.369999999997	1299	2011

The SQL query analyzes the total units shipped and the number of games released per year, helping to identify peak periods in the video game industry. It combines data from the games and scores tables.

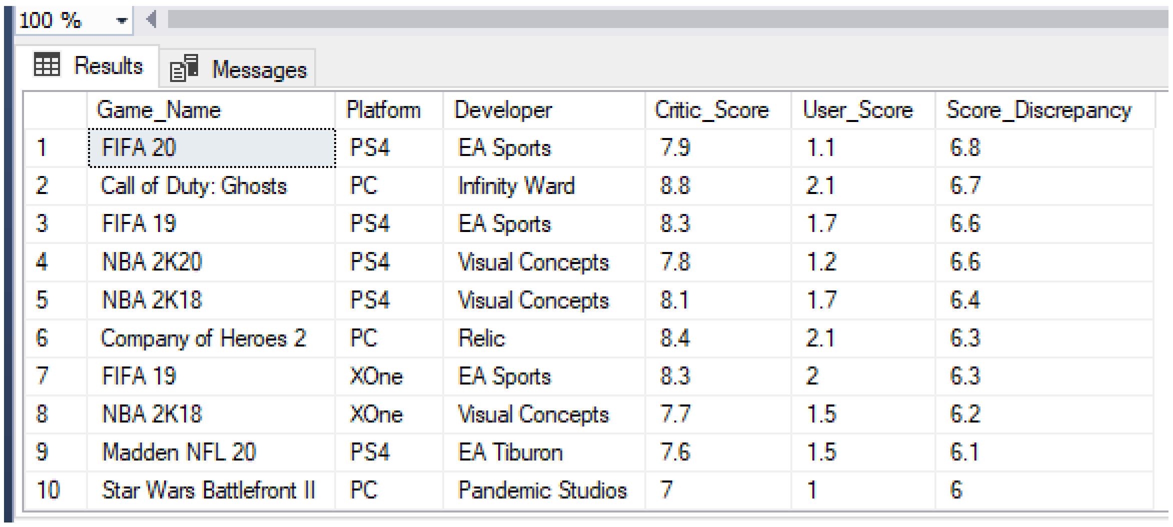
The query calculates the total units shipped (SUM(scores.Total_Shipped)) and counts the number of games released (COUNT(games.Game_ID)) for each year. It groups the data by the games.Year to generate the yearly summary and orders the results by year in descending order. This allows us to observe trends over time, highlighting the years with the most games released and the highest total units shipped.

The result set shows the total units shipped and games released for each year from 1992 to 2020. For example, in 2020, 26.25 million units were shipped across 13 games, while 2018 saw 275.67 million units shipped across 703 games.

6. Critic vs. User Score Comparison

Visualize and compare critic scores with user scores for each game to spot trends and discrepancies in perception.

```
SELECT games.Name AS Game_Name, platforms.Platform,
developers.Developer, scores.Critic_Score, scores.User_Score,
(scores.Critic_Score - scores.User_Score) AS Score_Discrepancy
FROM games JOIN scores ON games.Game_ID = scores.Game_ID JOIN
platforms ON games.Platform_ID = platforms.Platform_ID
JOIN developers ON developers.Developer_ID = games.Developer_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
ORDER BY Score_Discrepancy DESC, Game_Name
```



	Game_Name	Platform	Developer	Critic_Score	User_Score	Score_Discrepancy
1	FIFA 20	PS4	EA Sports	7.9	1.1	6.8
2	Call of Duty: Ghosts	PC	Infinity Ward	8.8	2.1	6.7
3	FIFA 19	PS4	EA Sports	8.3	1.7	6.6
4	NBA 2K20	PS4	Visual Concepts	7.8	1.2	6.6
5	NBA 2K18	PS4	Visual Concepts	8.1	1.7	6.4
6	Company of Heroes 2	PC	Relic	8.4	2.1	6.3
7	FIFA 19	XOne	EA Sports	8.3	2	6.3
8	NBA 2K18	XOne	Visual Concepts	7.7	1.5	6.2
9	Madden NFL 20	PS4	EA Tiburon	7.6	1.5	6.1
10	Star Wars Battlefront II	PC	Pandemic Studios	7	1	6

This SQL query compares critic scores and user scores for video games, highlighting discrepancies in how the games are perceived by critics versus players. The query selects the game name, platform, developer, critic score, user score, and calculates the difference between the critic score and the user score (referred to as Score_Discrepancy). It joins multiple tables—games, scores, platforms, and developers—to retrieve relevant information, ensuring that only records with both a critic and user score are included in the results. The query then orders the results by Score_Discrepancy in descending order, followed by the game name.

Critics, who tend to evaluate games based on specific criteria such as technical performance, storytelling, graphics, and innovation, may give high scores to games that they see as artistically or technically impressive, even if they are not as enjoyable or accessible to the average player. On the other hand, users might rate the game lower if it does not meet their expectations in terms of gameplay, difficulty, or fun factor. There are 2,208 rows of data in the output and only top 10 are showed for relevance.

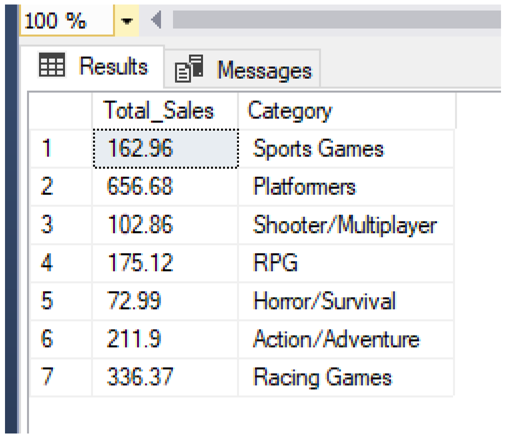
7. Sales by Game Category (Proxy by Name or Known Categories):

If applicable, group games into broad categories based on their titles or known genres and analyse sales distribution among these groups.

```sql
CREATE TABLE GameCategories (
    Total_Sales FLOAT,
    Category NVARCHAR(150)
)

-- Inserting data for 'Sports Games'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'Sports Games'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE games.Name LIKE '%Sports%';

-- Inserting data for 'Platformers'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'Platformers'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
```

```sql
WHERE games.Name LIKE '%Mario%';

-- Inserting data for 'Shooter/Multiplayer'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'Shooter/Multiplayer'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE games.Name LIKE '%Counter-Strike%' OR games.Name LIKE
'%Battlegrounds%'

-- Inserting data for 'RPG'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'RPG'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE games.Name LIKE '%Final Fantasy%' OR games.Name LIKE '%Dragon
Age%' OR games.Name LIKE '%The Witcher%'

-- Inserting data for 'Horror/Survival'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'Horror/Survival'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE games.Name LIKE '%Resident Evil%' OR games.Name LIKE '%Silent
Hill%' OR games.Name LIKE '%Outlast%'

-- Inserting data for 'Action/Adventure'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'Action/Adventure'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE games.Name LIKE '%Zelda%' OR games.Name LIKE '%Uncharted%' OR
games.Name LIKE '%Tomb Raider%'

-- Inserting data for 'Racing Games'
INSERT INTO GameCategories (Total_Sales, Category)
SELECT SUM(scores.Total_Shipped), 'Racing Games'
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
```

```sql
WHERE games.Name LIKE '%Need for Speed%' OR games.Name LIKE '%Gran
Turismo%' OR games.Name LIKE '%Mario Kart%'

SELECT * FROM GameCategories
```



100 %

Results | Messages

	Total_Sales	Category
1	162.96	Sports Games
2	656.68	Platformers
3	102.86	Shooter/Multiplayer
4	175.12	RPG
5	72.99	Horror/Survival
6	211.9	Action/Adventure
7	336.37	Racing Games

The SQL query analyzes video game sales by grouping games into broad categories based on keywords in their titles or known genres. The query first creates a new table, GameCategories, which will store the total sales and the corresponding category for each genre of games. Then, for each game category—such as "Sports Games," "Platformers," "Shooter/Multiplayer," "RPG," "Horror/Survival," "Action/Adventure," and "Racing Games"—it inserts data into the GameCategories table by summing up the total units shipped (Total_Shipped) from the scores table for games that match specific keywords in their titles.

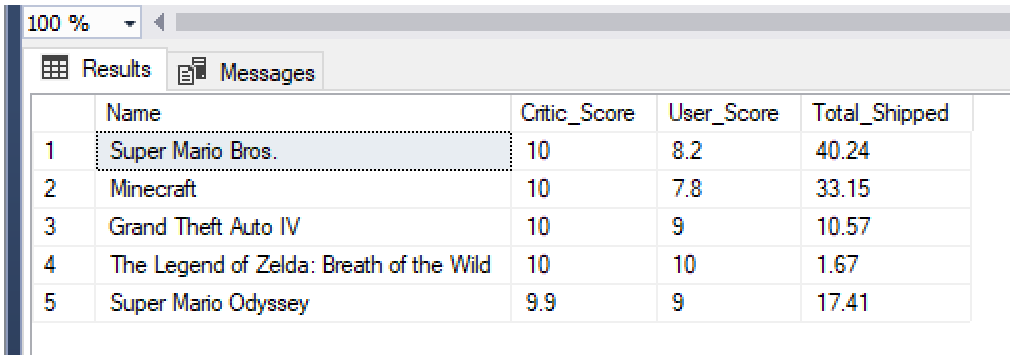
The resulting data provides an overview of total sales by game category. For instance, "Platformers" is the highest-grossing category with 656.68 million units shipped, followed by "Racing Games" at 336.37 million units, and "Action/Adventure" at 211.9 million units. Categories like "Horror/Survival" and "Shooter/Multiplayer" have lower total sales, at 72.99 million and 102.86 million units, respectively. This analysis offers valuable insights into the relative popularity and financial success of different game genres over time.

8. High and Low Scoring Games:

Identify games with the highest and lowest scores, providing insights into what might have contributed to their critical and commercial success or failure.

```
-- Highest Rated Games
SELECT TOP 5 games.Name, scores.Critic_Score, scores.User_Score,
scores.Total_Shipped
FROM games INNER JOIN scores ON games.Game_ID = scores.Game_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
ORDER BY scores.Critic_Score DESC

-- Lowest Rated Games
SELECT TOP 5 games.Name, scores.Critic_Score, scores.User_Score,
scores.Total_Shipped
FROM games INNER JOIN scores ON games.Game_ID = scores.Game_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
ORDER BY scores.Critic_Score ASC
```

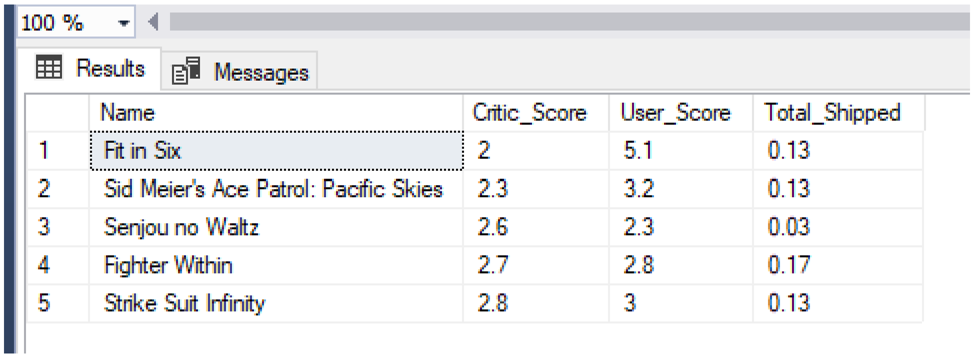
Highest Rated:



	Name	Critic_Score	User_Score	Total_Shipped
1	Super Mario Bros.	10	8.2	40.24
2	Minecraft	10	7.8	33.15
3	Grand Theft Auto IV	10	9	10.57
4	The Legend of Zelda: Breath of the Wild	10	10	1.67
5	Super Mario Odyssey	9.9	9	17.41

For the highest-rated games, the query orders the results in descending order of critic score (ORDER BY scores.Critic_Score DESC) to show the top-rated games at the top. This list provides insight into the games that received the most favorable critical reception. These games are often seen as industry milestones or genre-defining titles, likely due to factors such as exceptional gameplay mechanics, innovative design, storytelling, or technical execution. The data underscores a strong correlation between high critic scores and positive user reception, highlighting the importance of game quality in driving engagement and legacy. Titles like *Breath of the Wild* show that excellence can outweigh commercial factors, while *Super Mario Bros.* and *Minecraft* exemplify games that blend critical acclaim with massive popularity.

Lowest Rated:



100 %

Results Messages

	Name	Critic_Score	User_Score	Total_Shipped
1	Fit in Six	2	5.1	0.13
2	Sid Meier's Ace Patrol: Pacific Skies	2.3	3.2	0.13
3	Senjou no Waltz	2.6	2.3	0.03
4	Fighter Within	2.7	2.8	0.17
5	Strike Suit Infinity	2.8	3	0.13

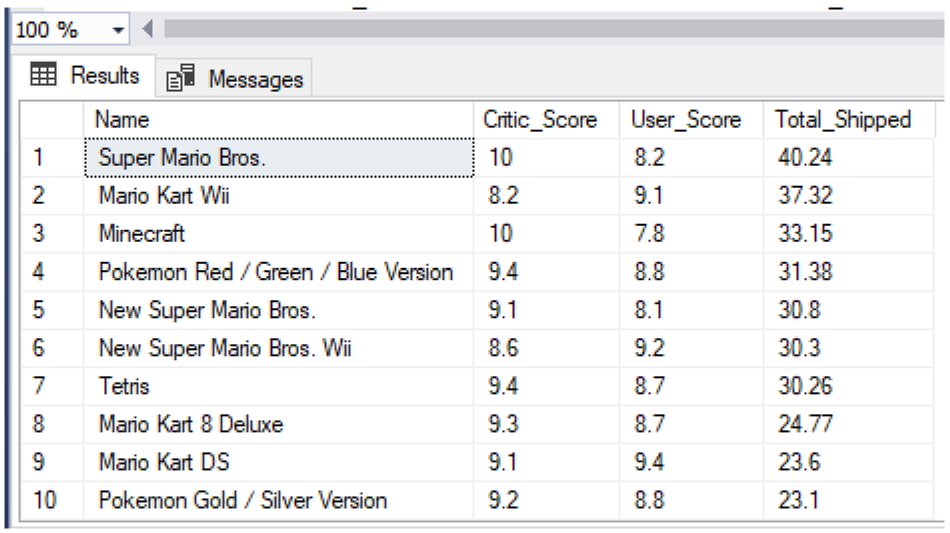
These games exhibit a trend of poor critical reception paired with low user scores and minimal sales. The discrepancies between critic and user scores for some titles, such as *Fit in Six*, suggest that while critics found significant flaws, certain aspects may have appealed to specific user demographics. The poor commercial performance across the board highlights the direct impact of low ratings on a game's market success. Developers might learn from these cases by addressing gameplay quality, technical issues, or audience alignment to avoid similar pitfalls.

The SQL query identifies the highest and lowest-rated games based on critic scores, and analyzes the potential factors contributing to their critical and commercial success or failure. It achieves this by selecting the top 5 games with the highest and lowest critic scores from the database. The query first joins the games and scores tables, ensuring that only games with valid critic and user scores are included by filtering out rows where either of these values is missing (IS NOT NULL).

9. Sales and Score Correlation Analysis

Explore whether there is a relationship between high critic/user scores and higher sales, helping to understand the impact of quality on commercial success.

```
SELECT games.Name, scores.Critic_Score, scores.User_Score,
scores.Total_Shipped
FROM games JOIN scores ON games.Game_ID = scores.Game_ID
WHERE (scores.Critic_Score >= 9.0 OR scores.User_Score >= 9.0)
  AND scores.Critic_Score IS NOT NULL
  AND scores.User_Score IS NOT NULL
ORDER BY scores.Total_Shipped DESC
```



	Name	Critic_Score	User_Score	Total_Shipped
1	Super Mario Bros.	10	8.2	40.24
2	Mario Kart Wii	8.2	9.1	37.32
3	Minecraft	10	7.8	33.15
4	Pokemon Red / Green / Blue Version	9.4	8.8	31.38
5	New Super Mario Bros.	9.1	8.1	30.8
6	New Super Mario Bros. Wii	8.6	9.2	30.3
7	Tetris	9.4	8.7	30.26
8	Mario Kart 8 Deluxe	9.3	8.7	24.77
9	Mario Kart DS	9.1	9.4	23.6
10	Pokemon Gold / Silver Version	9.2	8.8	23.1

The SQL query aims to explore the potential correlation between high critic/user scores and higher sales by identifying games that received high ratings and analyzing their commercial performance. The query specifically selects games with either a critic score or a user score of 9.0 or higher, reflecting titles that have been highly regarded by both critics and players. The query filters games that have a critic score of at least 9.0 or a user score of at least 9.0 (WHERE (scores.Critic_Score >= 9.0 OR scores.User_Score >= 9.0)), ensuring that only games with strong critical and user reception are included. The results are ordered by scores.Total_Shipped DESC, which sorts the games by total units shipped, in descending order.

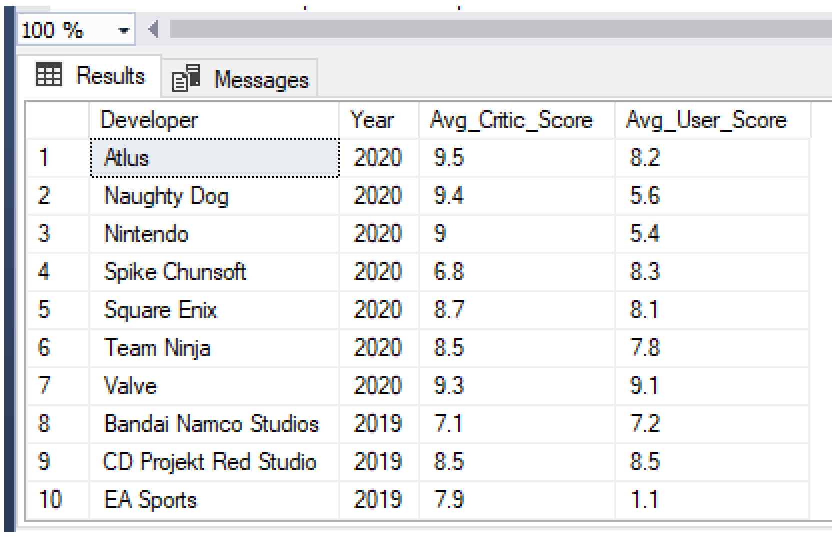
If high-rated games show a strong sales performance, it would suggest that critical acclaim and player satisfaction can drive commercial success. Conversely, if high ratings do not correlate with high sales, other factors (such as marketing, franchise reputation, or platform exclusivity) might be influencing sales outcomes. There are 263 rows of output but for relevance, only top 10 are shown here.

10. Developer Consistency Analysis:

Track the performance of individual developers over time by examining their average critic and user scores for each year.

```
SELECT developers.Developer, games.Year,
ROUND(AVG(scores.Critic_Score), 2) AS Avg_Critic_Score,
ROUND(AVG(scores.User_Score), 2) AS Avg_User_Score
```

```
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
JOIN developers ON games.Developer_ID = developers.Developer_ID
WHERE scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT
NULL
GROUP BY developers.Developer, games.Year
ORDER BY games.Year DESC
```



	Developer	Year	Avg_Critic_Score	Avg_User_Score
1	Atlus	2020	9.5	8.2
2	Naughty Dog	2020	9.4	5.6
3	Nintendo	2020	9	5.4
4	Spike Chunsoft	2020	6.8	8.3
5	Square Enix	2020	8.7	8.1
6	Team Ninja	2020	8.5	7.8
7	Valve	2020	9.3	9.1
8	Bandai Namco Studios	2019	7.1	7.2
9	CD Projekt Red Studio	2019	8.5	8.5
10	EA Sports	2019	7.9	1.1

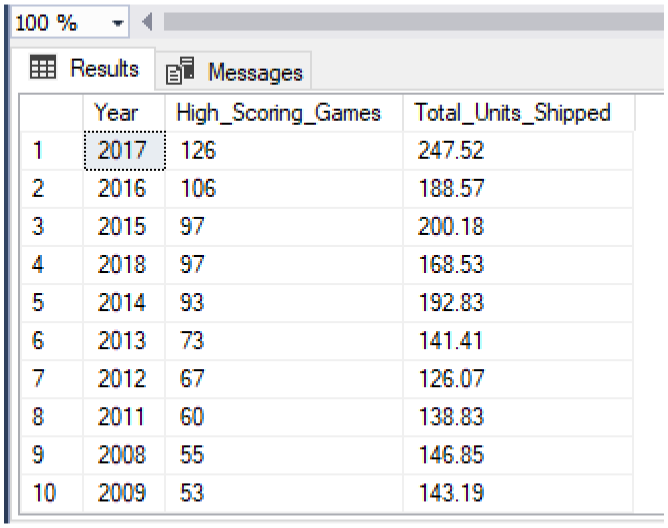
The query groups the data by both the developer's name (developers.Developer) and the release year of the game (games.Year). The query calculates the average critic score and user score for each developer in each year. It does so by using the AVG() function on the scores.Critic_Score and scores.User_Score fields, and the results are rounded to two decimal places using ROUND(). The WHERE clause ensures that only games with both a valid critic score and a user score (not null) are considered in the analysis (scores.Critic_Score IS NOT NULL AND scores.User_Score IS NOT NULL), which improves the accuracy of the analysis. The ORDER BY games.Year DESC clause sorts the results by the year in descending order, meaning the most recent years will appear first.

A developer with consistently high average scores over the years may be seen as reliable and trusted to deliver quality games. On the other hand, a developer with fluctuating or declining average scores could indicate challenges or a decline in the quality of their releases. This kind of analysis can also highlight shifts in the gaming industry, such as when a developer experiences a breakthrough year or a downturn. There are 1570 rows of output but for relevance, only first ten are shown here.

11. Release Year Insights:

Highlight standout years for the gaming industry by reviewing the number of high-scoring games and total units shipped.

```
SELECT games.Year, COUNT(*) AS High_Scoring_Games,
ROUND(SUM(scores.Total_Shipped), 2) AS Total_Units_Shipped
FROM games
JOIN scores ON games.Game_ID = scores.Game_ID
WHERE (scores.Critic_Score >= 8.0 OR scores.User_Score >= 8.0)
    AND scores.Critic_Score IS NOT NULL
    AND scores.User_Score IS NOT NULL
GROUP BY games.Year
ORDER BY High_Scoring_Games DESC, Total_Units_Shipped DESC
```



	Year	High_Scoring_Games	Total_Units_Shipped
1	2017	126	247.52
2	2016	106	188.57
3	2015	97	200.18
4	2018	97	168.53
5	2014	93	192.83
6	2013	73	141.41
7	2012	67	126.07
8	2011	60	138.83
9	2008	55	146.85
10	2009	53	143.19

The WHERE clause filters games that have either a critic score (scores.Critic_Score) or a user score (scores.User_Score) of 8.0 or higher. This threshold is used to identify games that were generally well-received by either critics or players, ensuring that only high-quality games are considered. The query calculates the total units shipped (SUM(scores.Total_Shipped)) for the high-scoring games. The query orders the results first by the number of high-scoring games (High_Scoring_Games) in descending order, and then by the total units shipped (Total_Units_Shipped) in descending order. This ensures that years with the most successful games, in terms of both quality and sales, appear at the top of the list. There are 38 rows of data but for relevance, only top 10 are shown here.

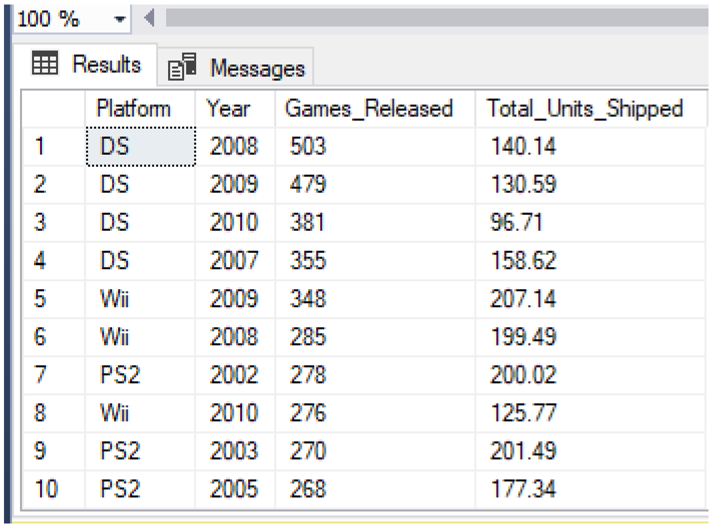
If a year is shown to have both a high number of high-scoring games and a large total number of units shipped, it may indicate that the industry had a particularly successful year, both in terms of quality and consumer demand. This might correlate with the release of landmark games or major technological advancements in gaming, such as the launch of new consoles or breakthrough titles.

Conversely, if a year shows a high number of high-scoring games but lower total units shipped, it might suggest that the quality of games was high, but there were other factors (like competition, economic conditions, or market saturation) that prevented those games from achieving commercial success.

12. Platform Popularity Over Time:

Analyse how the popularity of different platforms has evolved by looking at the number of games released and their respective sales over the years.

```
SELECT platforms.Platform, games.Year, COUNT(games.Game_ID) AS
Games_Released, ROUND(SUM(scores.Total_Shipped), 2) AS
Total_Units_Shipped
FROM games JOIN scores ON games.Game_ID = scores.Game_ID JOIN
platforms ON games.Platform_ID = platforms.Platform_ID
GROUP BY platforms.Platform, games.Year
ORDER BY Games_Released DESC
```



	Platform	Year	Games_Released	Total_Units_Shipped
1	DS	2008	503	140.14
2	DS	2009	479	130.59
3	DS	2010	381	96.71
4	DS	2007	355	158.62
5	Wii	2009	348	207.14
6	Wii	2008	285	199.49
7	PS2	2002	278	200.02
8	Wii	2010	276	125.77
9	PS2	2003	270	201.49
10	PS2	2005	268	177.34

The query groups data by two key dimensions: the gaming platform (platforms.Platform) and the year of release (games.Year). The query counts the number of games released on each platform for each year using the COUNT(games.Game_ID) function. This metric helps identify which platforms saw the highest number of game releases in a given year. A higher count of games released on a platform typically suggests that the platform was more popular and supported by a larger number of developers. The query calculates the total units shipped for all games released on each platform and in each year using the SUM(scores.Total_Shipped) function. The result is rounded to two decimal places using the ROUND() function for precision. The query orders the results by the number of games released (Games_Released) in descending order.

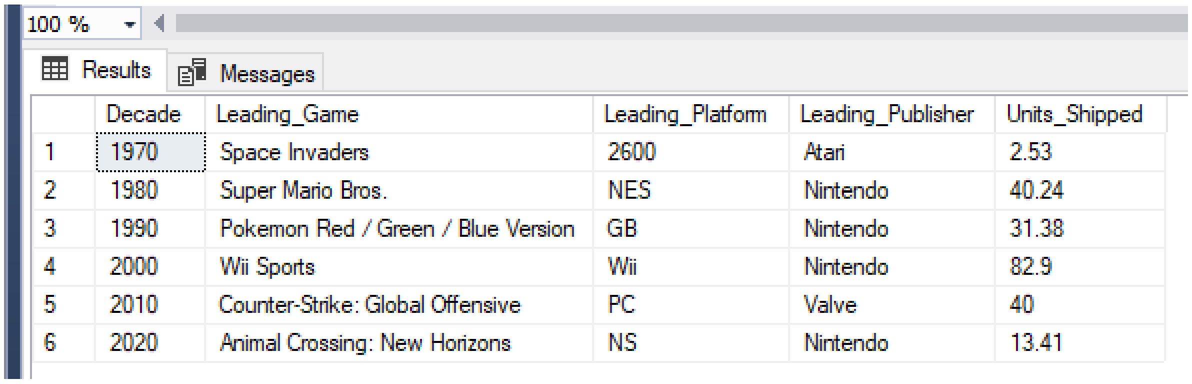
There are 294 rows of data but for relevance's sake, only top 10 are shown.

A higher number of games released on a platform generally indicates greater developer support and higher adoption of that platform. A higher number of games released on a platform generally indicates greater developer support and higher adoption of that platform. By comparing the data year-over-year, the query allows for tracking the rise and fall of different platforms. For example, a spike in the number of games released and total units shipped for a platform could indicate a major launch (like a new console or a big game franchise). Conversely, a drop in either metric could indicate declining popularity or market saturation.

13. Sales Leaders by Era:

Segment the data by decade to see which games and platforms were leaders in their respective periods.

```
SELECT (games.Year/10) * 10 AS Decade, games.Name AS Leading_Game,
platforms.Platform AS Leading_Platform, publishers.Publisher AS
Leading_Publisher, ROUND(scores.Total_Shipped, 2) AS Units_Shipped
FROM games JOIN scores ON games.Game_ID = scores.Game_ID
JOIN platforms ON games.Platform_ID = platforms.Platform_ID
JOIN publishers ON games.Publisher_ID = publishers.Publisher_ID
WHERE scores.Total_Shipped = (SELECT MAX(s1.Total_Shipped) FROM games
g1
JOIN scores s1 ON g1.Game_Id = s1.Game_ID
WHERE (g1.Year/10) * 10 = (games.Year / 10) * 10)
ORDER BY Decade
```



	Decade	Leading_Game	Leading_Platform	Leading_Publisher	Units_Shipped
1	1970	Space Invaders	2600	Atari	2.53
2	1980	Super Mario Bros.	NES	Nintendo	40.24
3	1990	Pokemon Red / Green / Blue Version	GB	Nintendo	31.38
4	2000	Wii Sports	Wii	Nintendo	82.9
5	2010	Counter-Strike: Global Offensive	PC	Valve	40
6	2020	Animal Crossing: New Horizons	NS	Nintendo	13.41

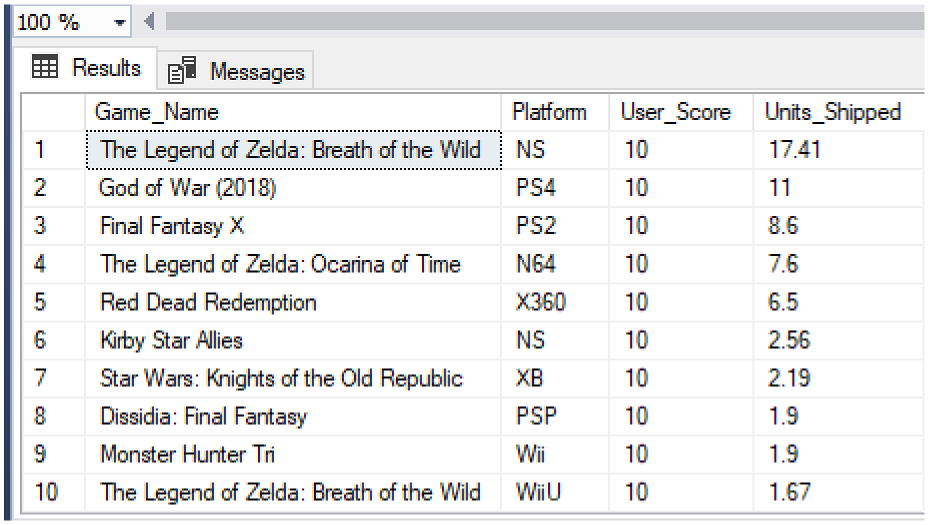
The SQL query is designed to identify the leading games, platforms, and publishers in the video game industry for each decade, based on the total units shipped. It segments the data by decade, calculating which games had the highest sales in their respective periods. The query first calculates the decade of each game by dividing the release year by 10 and multiplying by 10. It then selects the top-performing game, platform, publisher, and units shipped for each decade. To determine the leading game of the decade, the query uses a subquery to find the game with the maximum units shipped for each decade. The data is then grouped by decade, and the results are ordered by the decade, showcasing the commercial success of games, platforms, and publishers over time.

The results provide valuable insights into the evolution of the gaming industry, highlighting the games, platforms, and publishers that dominated each era. By analyzing the total units shipped, the query reveals which games had the most commercial success in each decade, providing a clearer understanding of market trends and how the industry has evolved. This analysis reveals valuable insights into customer preferences by pinpointing games that players rated highly. The findings include titles from various decades, showcasing trends and popular genres. For instance, classic games like *Super Mario Bros.* on the NES (1980) and *Space Invaders* on the Atari 2600 (1970) are noted for their significant impact, while modern hits such as *Counter-Strike: Global Offensive* on PC (2010) and *Animal Crossing: New Horizons* on the Nintendo Switch (2020) demonstrate how user engagement continues to be a critical success factor. Notably, *Wii Sports* (2000) achieved an unmatched combination of high user satisfaction and massive sales, shipping 82.9 million units.

14. Top Games by User Engagement

Focus on games with high user scores to understand which titles resonate most with players, offering potential insights into customer preferences.

```
SELECT TOP 10 games.Name AS Game_Name, platforms.Platform AS
Platform, ROUND(scores.User_Score, 2) AS User_Score,
ROUND(scores.Total_Shipped, 2) AS Units_Shipped
FROM games JOIN scores ON games.Game_ID = scores.Game_ID JOIN
platforms ON platforms.platform_ID = games.Platform_ID
WHERE scores.User_Score IS NOT NULL
ORDER BY scores.User_Score DESC
```



	Game_Name	Platform	User_Score	Units_Shipped
1	The Legend of Zelda: Breath of the Wild	NS	10	17.41
2	God of War (2018)	PS4	10	11
3	Final Fantasy X	PS2	10	8.6
4	The Legend of Zelda: Ocarina of Time	N64	10	7.6
5	Red Dead Redemption	X360	10	6.5
6	Kirby Star Allies	NS	10	2.56
7	Star Wars: Knights of the Old Republic	XB	10	2.19
8	Dissidia: Final Fantasy	PSP	10	1.9
9	Monster Hunter Tri	Wii	10	1.9
10	The Legend of Zelda: Breath of the Wild	WiiU	10	1.67

The SQL query is designed to identify the top 10 games based on user engagement, specifically focusing on games with the highest user scores. The query selects the name of each game, the platform on which it was released, the user score, and the total units shipped for each game. By filtering out games with missing user scores, the query ensures that only those with user feedback are considered. The data is then ordered by the user score in descending order to highlight the games that resonated the most with players.

This analysis helps to identify which games have garnered the highest levels of user satisfaction, offering valuable insights into customer preferences and what makes a game successful from a player's perspective.

The analysis of user engagement reveals that *The Legend of Zelda: Breath of the Wild* stands as the highest-rated game, achieving a perfect user score of 10 on both the Nintendo Switch and Wii U. With an impressive 17.41 million units shipped on the Nintendo Switch, it is evident that the game has not only captivated players with its quality but also achieved significant commercial success. Other top-rated games include *God of War (2018)* for the PlayStation 4, *Final Fantasy X* for the PlayStation 2, and *The Legend of Zelda: Ocarina of Time* for the Nintendo 64—all with user scores of 10. These titles, each recognized for their groundbreaking gameplay, narratives, and design, also exhibit varying degrees of sales performance, reflecting their popularity and legacy in gaming history.