FINAL PROJECT

DBAS 3035

By: Matthew Daley, Mitchell Smith

Table of Contents

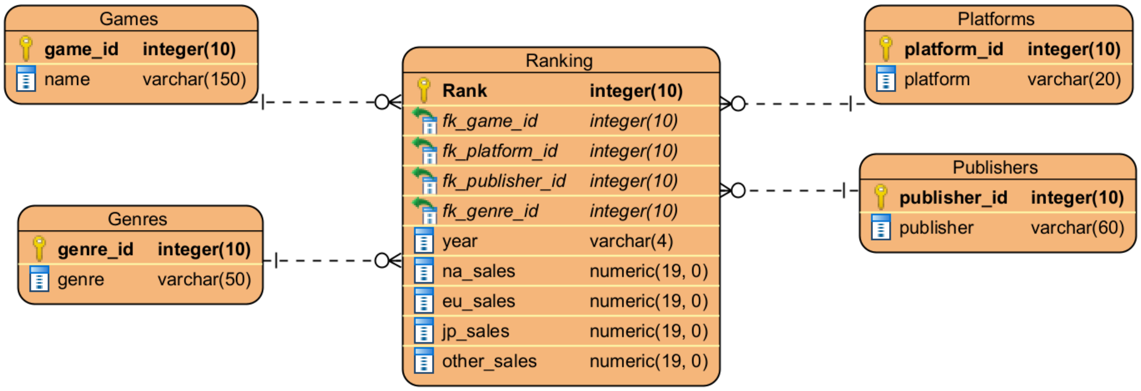
Introduction

In this document, we will be demonstrating the completion of our final project for DBAS3035. We will achieve this by providing an Entity Relationship Diagram (ERD) for the "Video Game Sales" dataset that has been created in Third-Normal Form (3NF), providing a data dictionary for the dataset, creating a database and tables for the dataset that consist of the dataset's imported data, creating SQL queries that obtain meaningful results, and analyzing our queries by utilizing various methods.

Part 1: Entity Relationship Diagram & Data Dictionary

Entity Relationship Diagram

The following screenshot displays our Entity Relationship Diagram, which showcases the relational structure of our database.



Games	
game_id	**integer(10)**
name	varchar(150)

Genres	
genre_id	**integer(10)**
genre	varchar(50)

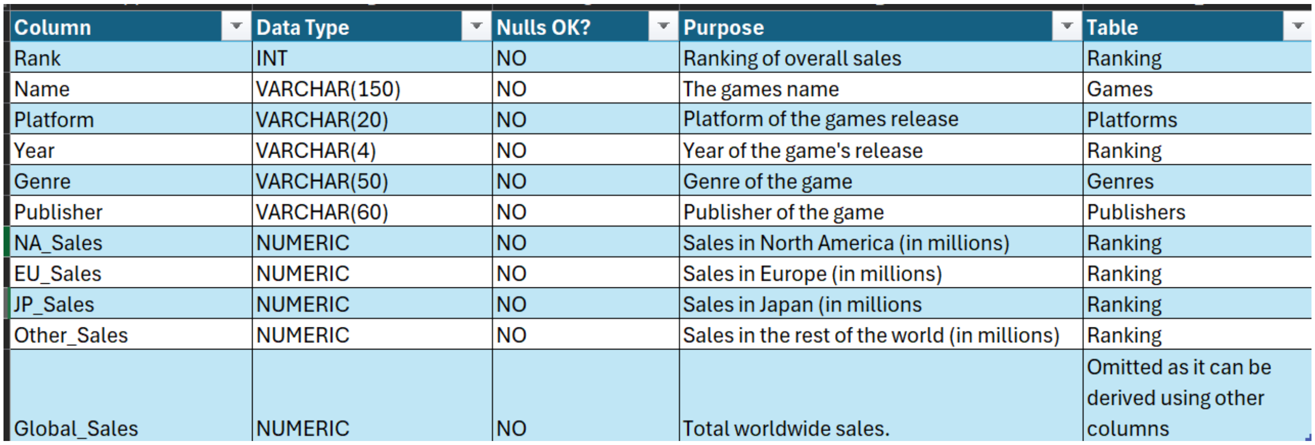
Ranking	
Rank	**integer(10)**
fk_game_id	*integer(10)*
fk_platform_id	*integer(10)*
fk_publisher_id	*integer(10)*
fk_genre_id	*integer(10)*
year	varchar(4)
na_sales	numeric(19, 0)
eu_sales	numeric(19, 0)
jp_sales	numeric(19, 0)
other_sales	numeric(19, 0)

Platforms	
platform_id	**integer(10)**
platform	varchar(20)

Publishers	
publisher_id	**integer(10)**
publisher	varchar(60)

Data Dictionary

The following screenshot displays the data dictionary that we created for the dataset, which was used to populate our database.



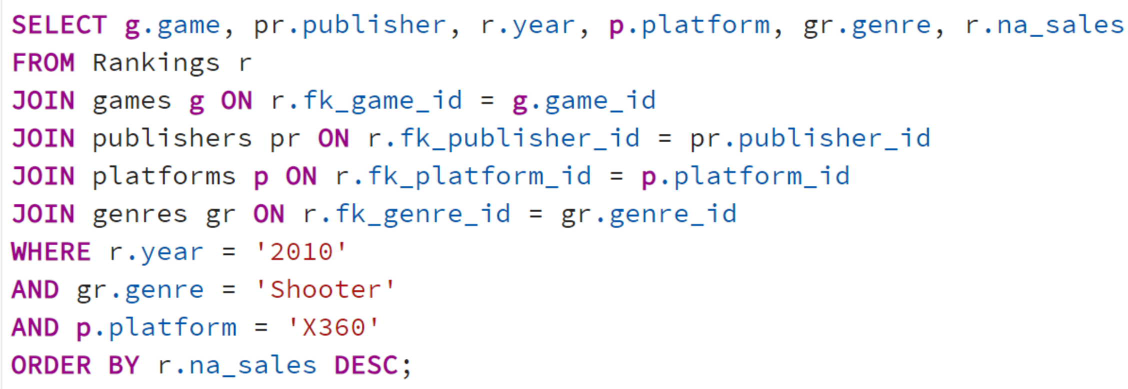
Column	Data Type	Nulls OK?	Purpose	Table
Rank	INT	NO	Ranking of overall sales	Ranking
Name	VARCHAR(150)	NO	The games name	Games
Platform	VARCHAR(20)	NO	Platform of the games release	Platforms
Year	VARCHAR(4)	NO	Year of the game's release	Ranking
Genre	VARCHAR(50)	NO	Genre of the game	Genres
Publisher	VARCHAR(60)	NO	Publisher of the game	Publishers
NA_Sales	NUMERIC	NO	Sales in North America (in millions)	Ranking
EU_Sales	NUMERIC	NO	Sales in Europe (in millions)	Ranking
JP_Sales	NUMERIC	NO	Sales in Japan (in millions	Ranking
Other_Sales	NUMERIC	NO	Sales in the rest of the world (in millions)	Ranking
Global_Sales	NUMERIC	NO	Total worldwide sales.	Omitted as it can be derived using other columns

Part 2: SQL Queries

Basic Queries

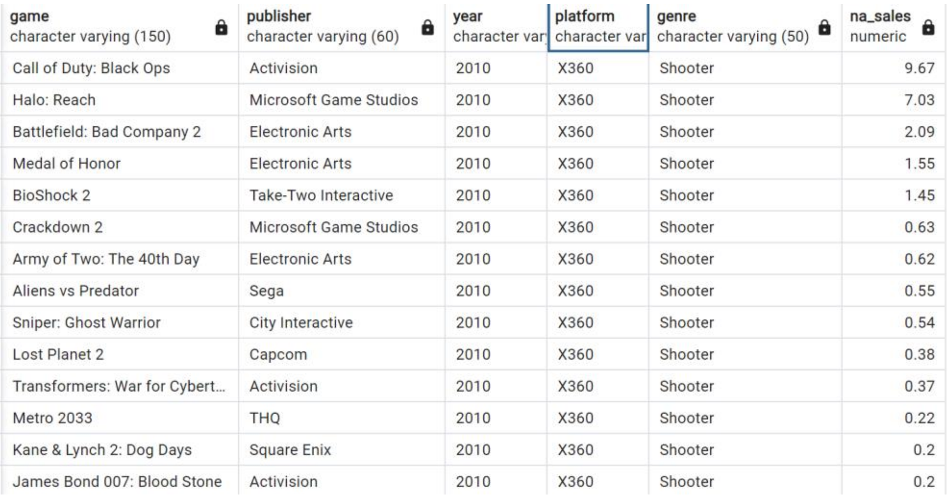
Query #1

Screenshot of Query:



```
SELECT g.game, pr.publisher, r.year, p.platform, gr.genre, r.na_sales
FROM Rankings r
JOIN games g ON r.fk_game_id = g.game_id
JOIN publishers pr ON r.fk_publisher_id = pr.publisher_id
JOIN platforms p ON r.fk_platform_id = p.platform_id
JOIN genres gr ON r.fk_genre_id = gr.genre_id
WHERE r.year = '2010'
AND gr.genre = 'Shooter'
AND p.platform = 'X360'
ORDER BY r.na_sales DESC;
```

Screenshot of Result Set:



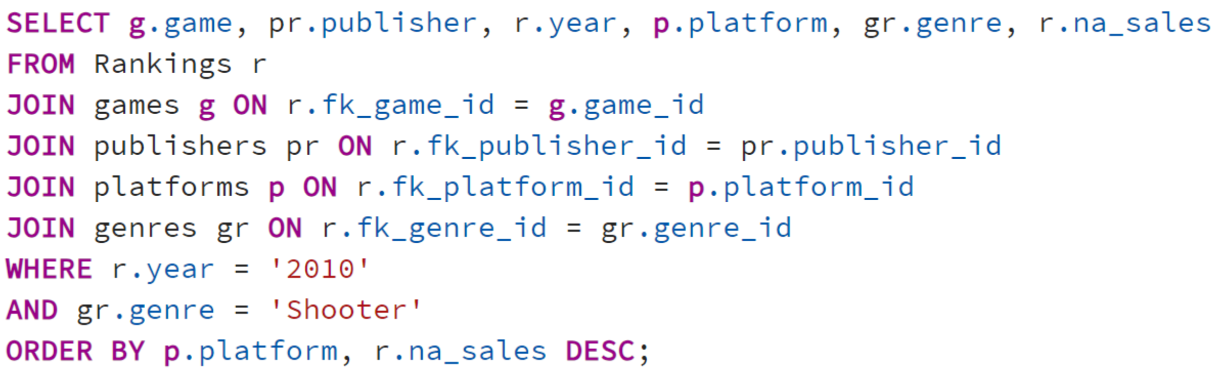
game character varying (150)	publisher character varying (60)	year character var	platform character var	genre character varying (50)	na_sales numeric
Call of Duty: Black Ops	Activision	2010	X360	Shooter	9.67
Halo: Reach	Microsoft Game Studios	2010	X360	Shooter	7.03
Battlefield: Bad Company 2	Electronic Arts	2010	X360	Shooter	2.09
Medal of Honor	Electronic Arts	2010	X360	Shooter	1.55
BioShock 2	Take-Two Interactive	2010	X360	Shooter	1.45
Crackdown 2	Microsoft Game Studios	2010	X360	Shooter	0.63
Army of Two: The 40th Day	Electronic Arts	2010	X360	Shooter	0.62
Aliens vs Predator	Sega	2010	X360	Shooter	0.55
Sniper: Ghost Warrior	City Interactive	2010	X360	Shooter	0.54
Lost Planet 2	Capcom	2010	X360	Shooter	0.38
Transformers: War for Cybert...	Activision	2010	X360	Shooter	0.37
Metro 2033	THQ	2010	X360	Shooter	0.22
Kane & Lynch 2: Dog Days	Square Enix	2010	X360	Shooter	0.2
James Bond 007: Blood Stone	Activision	2010	X360	Shooter	0.2

Query Purpose:

In this query, we derived all information pertaining to the best-selling shooter games for the XBOX 360 in the year 2010. We performed this by selecting the relevant columns for the information we wanted to derive, joining the additional (child) tables (Games, Publishers, Platforms, Genres) that contained the relevant columns we wanted to use in the query to our parent table (Rankings), adding conditions to the query to output the specific information we wanted to obtain, and ordering the sales column in descending order to list the information from the best-selling XBOX 360 games of 2010 to the worst selling XBOX 360 games of 2010.

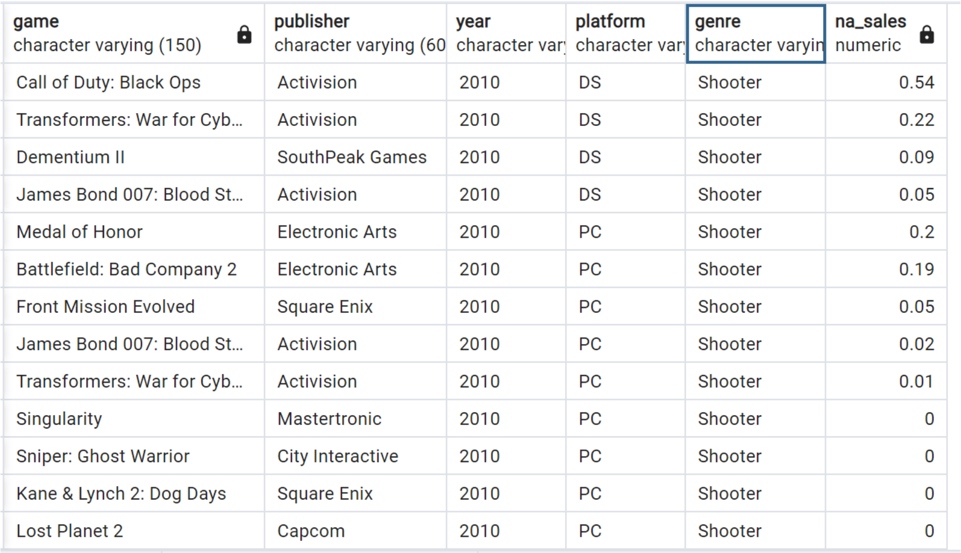
Query #2

Screenshot of Query:



```sql
SELECT g.game, pr.publisher, r.year, p.platform, gr.genre, r.na_sales
FROM Rankings r
JOIN games g ON r.fk_game_id = g.game_id
JOIN publishers pr ON r.fk_publisher_id = pr.publisher_id
JOIN platforms p ON r.fk_platform_id = p.platform_id
JOIN genres gr ON r.fk_genre_id = gr.genre_id
WHERE r.year = '2010'
AND gr.genre = 'Shooter'
ORDER BY p.platform, r.na_sales DESC;
```

Screenshot of Result Set:



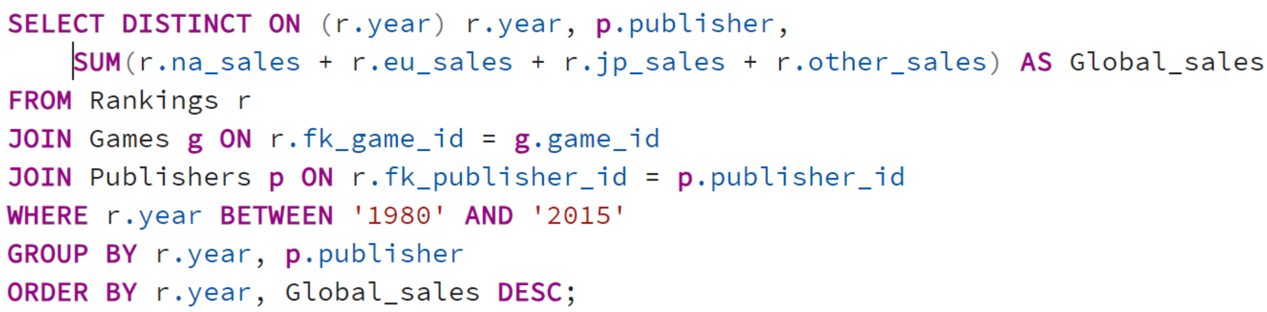
game character varying (150)	publisher character varying (60	year character var	platform character var	genre character varyin	na_sales numeric
Call of Duty: Black Ops	Activision	2010	DS	Shooter	0.54
Transformers: War for Cyb...	Activision	2010	DS	Shooter	0.22
Dementium II	SouthPeak Games	2010	DS	Shooter	0.09
James Bond 007: Blood St...	Activision	2010	DS	Shooter	0.05
Medal of Honor	Electronic Arts	2010	PC	Shooter	0.2
Battlefield: Bad Company 2	Electronic Arts	2010	PC	Shooter	0.19
Front Mission Evolved	Square Enix	2010	PC	Shooter	0.05
James Bond 007: Blood St...	Activision	2010	PC	Shooter	0.02
Transformers: War for Cyb...	Activision	2010	PC	Shooter	0.01
Singularity	Mastertronic	2010	PC	Shooter	0
Sniper: Ghost Warrior	City Interactive	2010	PC	Shooter	0
Kane & Lynch 2: Dog Days	Square Enix	2010	PC	Shooter	0
Lost Planet 2	Capcom	2010	PC	Shooter	0

Query Purpose:

In this query, we derived all information pertaining to the best-selling shooter games for each platform in the year 2010. We performed this by selecting the relevant columns for the information we wanted to derive, joining the additional (child) tables (Games, Publishers, Platforms, Genres) that contained the relevant columns we wanted to use in the query to our parent table (Rankings), adding conditions to the query to output the specific information we wanted to obtain, and ordering the sales column and platform column in descending order to display the information beginning with the best-selling shooter games of 2010 for each platform at the top of the list and ending with the worst-selling shooter games of 2010 for each platform.

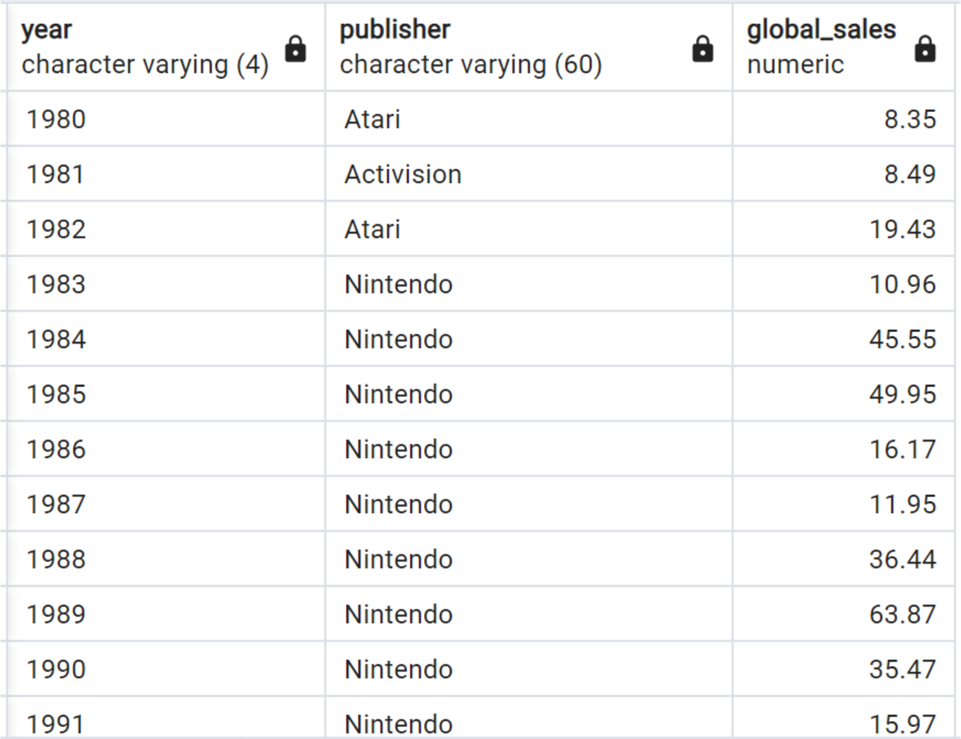
Query #3

Screenshot of Query:



```
SELECT DISTINCT ON (r.year) r.year, p.publisher,
    SUM(r.na_sales + r.eu_sales + r.jp_sales + r.other_sales) AS Global_sales
FROM Rankings r
JOIN Games g ON r.fk_game_id = g.game_id
JOIN Publishers p ON r.fk_publisher_id = p.publisher_id
WHERE r.year BETWEEN '1980' AND '2015'
GROUP BY r.year, p.publisher
ORDER BY r.year, Global_sales DESC;
```

Screenshot of Result Set:



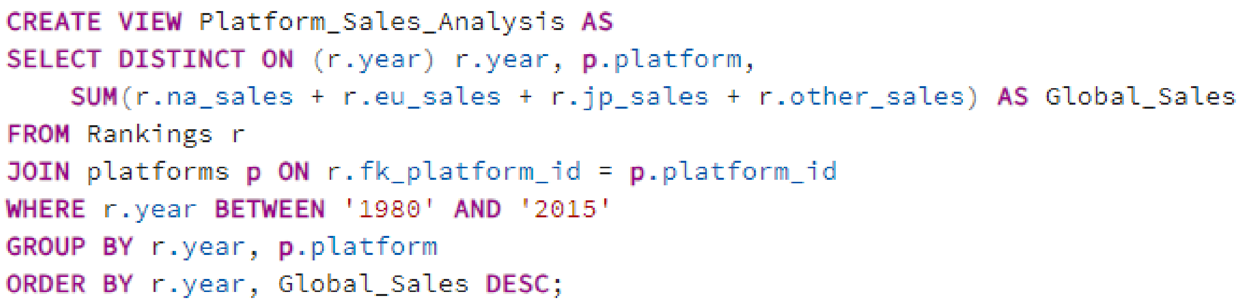
year character varying (4)	publisher character varying (60)	global_sales numeric
1980	Atari	8.35
1981	Activision	8.49
1982	Atari	19.43
1983	Nintendo	10.96
1984	Nintendo	45.55
1985	Nintendo	49.95
1986	Nintendo	16.17
1987	Nintendo	11.95
1988	Nintendo	36.44
1989	Nintendo	63.87
1990	Nintendo	35.47
1991	Nintendo	15.97

Query Purpose:

In this query, we derived all relevant information pertaining to the top-selling publishers for each year spanning from the years 1980 to 2015. We designed the query to output this information in a standard view and return only the top selling publisher for each individual year. We performed this by selecting the relevant columns (year and publisher) for the information we wanted to derive, using an aggregate function (SUM) to calculate the global sales for each game, joining the additional (child) tables (Games table and Publishers table) that contained the relevant columns that we wanted to use in the query to our parent table (Rankings), grouping all of the non-aggregate columns together, adding a condition / filter to the query to output the specific information we wanted to obtain, and ordering the years column and global sales column in descending order to list the information from the best-selling platform for each year.

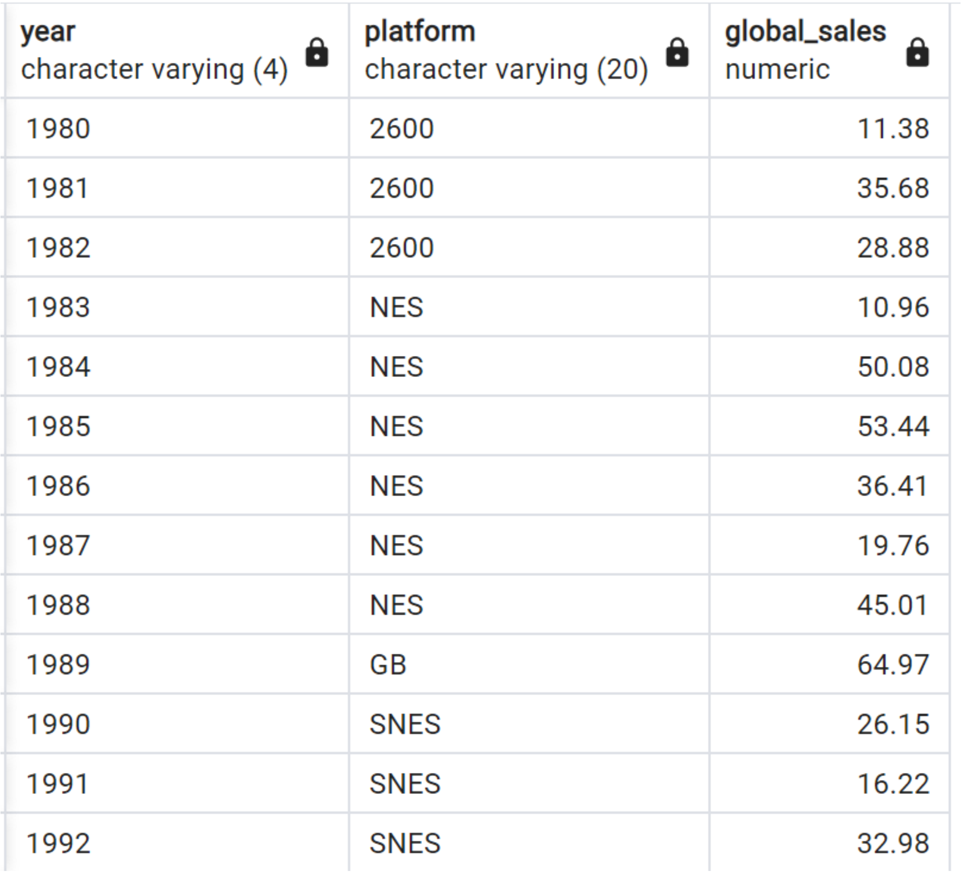
View Query

Screenshot of Query:



```
CREATE VIEW Platform_Sales_Analysis AS
SELECT DISTINCT ON (r.year) r.year, p.platform,
    SUM(r.na_sales + r.eu_sales + r.jp_sales + r.other_sales) AS Global_Sales
FROM Rankings r
JOIN platforms p ON r.fk_platform_id = p.platform_id
WHERE r.year BETWEEN '1980' AND '2015'
GROUP BY r.year, p.platform
ORDER BY r.year, Global_Sales DESC;
```

Screenshot of Result Set:



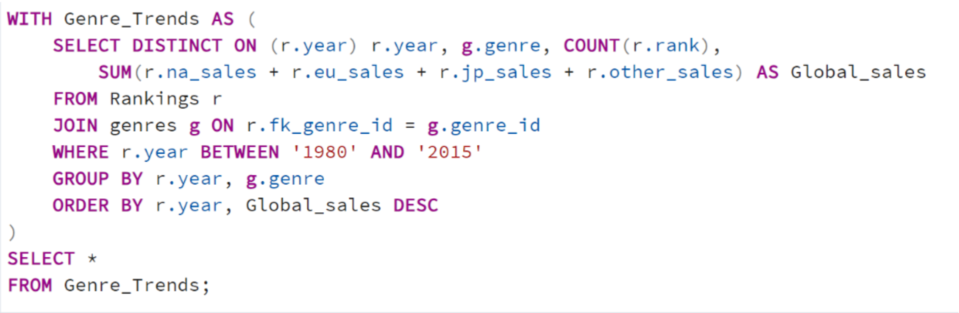
year character varying (4)	platform character varying (20)	global_sales numeric
1980	2600	11.38
1981	2600	35.68
1982	2600	28.88
1983	NES	10.96
1984	NES	50.08
1985	NES	53.44
1986	NES	36.41
1987	NES	19.76
1988	NES	45.01
1989	GB	64.97
1990	SNES	26.15
1991	SNES	16.22
1992	SNES	32.98

Query Purpose:

In this query, we created a view that derived all relevant information pertaining to the top-selling platforms for each year spanning from the years 1980 to 2015. We designed the query to output this information in a standard view and return only the top selling platform for each individual year. We performed this by selecting the relevant columns for the information we wanted to derive, using an aggregate function (SUM) to calculate the global sales for each game, joining the additional (child) table (Platforms) that contained the relevant column that we wanted to use in the query to our parent table (Rankings), grouping all of the non-aggregate columns together, adding a condition / filter to the query to output the specific information we wanted to obtain, and ordering the years column and sales column in descending order to list the information from the best-selling platform for each year.

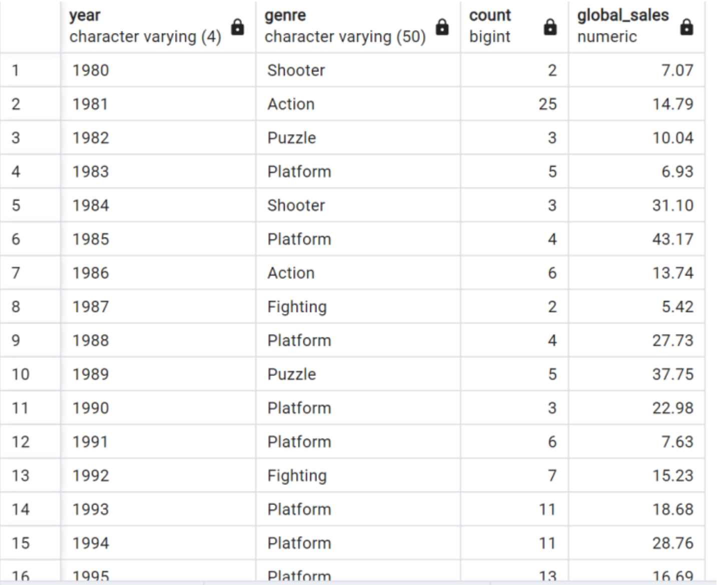
CTE Query

Screenshot of Query:



```
WITH Genre_Trends AS (
    SELECT DISTINCT ON (r.year) r.year, g.genre, COUNT(r.rank),
        SUM(r.na_sales + r.eu_sales + r.jp_sales + r.other_sales) AS Global_sales
    FROM Rankings r
    JOIN genres g ON r.fk_genre_id = g.genre_id
    WHERE r.year BETWEEN '1980' AND '2015'
    GROUP BY r.year, g.genre
    ORDER BY r.year, Global_sales DESC
)
SELECT *
FROM Genre_Trends;
```

Screenshot of Result Set



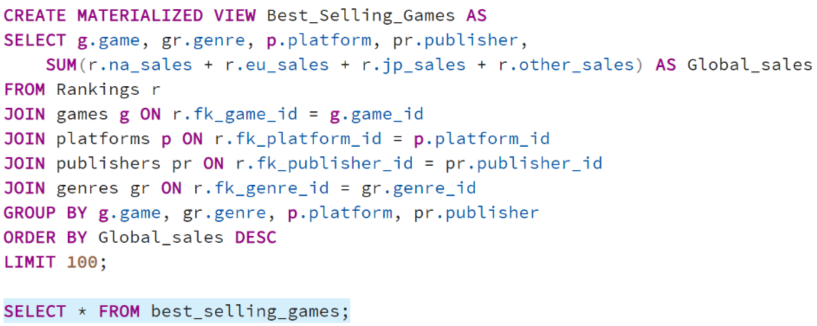
	year character varying (4)	genre character varying (50)	count bigint	global_sales numeric
1	1980	Shooter	2	7.07
2	1981	Action	25	14.79
3	1982	Puzzle	3	10.04
4	1983	Platform	5	6.93
5	1984	Shooter	3	31.10
6	1985	Platform	4	43.17
7	1986	Action	6	13.74
8	1987	Fighting	2	5.42
9	1988	Platform	4	27.73
10	1989	Puzzle	5	37.75
11	1990	Platform	3	22.98
12	1991	Platform	6	7.63
13	1992	Fighting	7	15.23
14	1993	Platform	11	18.68
15	1994	Platform	11	28.76
16	1995	Platform	13	16.69

Query Purpose:

In this query, we created a Common Table Expression (CTE) that derived all relevant information pertaining to the top-selling genres of games for each year spanning from the years 1980 to 2015. We created this query for the purpose of identifying possible trends over the past 25 years regarding the popularity of game genres. We designed the query to output this information in a standard view and return only the top selling genre for each individual year. We performed this by selecting the relevant columns for the information we wanted to derive, using aggregate functions (SUM and COUNT) to calculate the global sales for each game, joining the additional (child) table (Genres) that contained the relevant column that we wanted to use in the query to our parent table (Rankings), grouping all of the non-aggregate columns together, adding a condition / filter to the query to output the specific information we wanted to obtain, and ordering the years column and global sales column in descending order to list the best-selling platform for each year.

Creation and Use of Materialized View

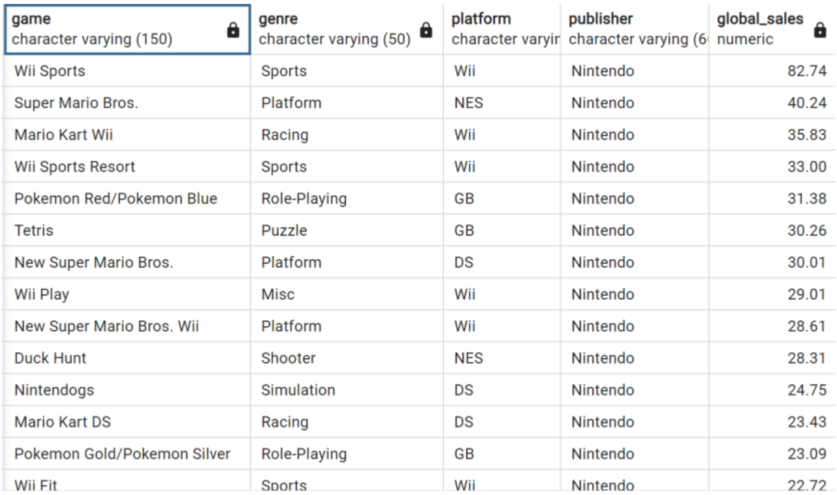
Screenshot of Query:



```
CREATE MATERIALIZED VIEW Best_Selling_Games AS
SELECT g.game, gr.genre, p.platform, pr.publisher,
    SUM(r.na_sales + r.eu_sales + r.jp_sales + r.other_sales) AS Global_sales
FROM Rankings r
JOIN games g ON r.fk_game_id = g.game_id
JOIN platforms p ON r.fk_platform_id = p.platform_id
JOIN publishers pr ON r.fk_publisher_id = pr.publisher_id
JOIN genres gr ON r.fk_genre_id = gr.genre_id
GROUP BY g.game, gr.genre, p.platform, pr.publisher
ORDER BY Global_sales DESC
LIMIT 100;

SELECT * FROM best_selling_games;
```

Screenshot of Result Set:



game character varying (150)	genre character varying (50)	platform character varyir	publisher character varying (6	global_sales numeric
Wii Sports	Sports	Wii	Nintendo	82.74
Super Mario Bros.	Platform	NES	Nintendo	40.24
Mario Kart Wii	Racing	Wii	Nintendo	35.83
Wii Sports Resort	Sports	Wii	Nintendo	33.00
Pokemon Red/Pokemon Blue	Role-Playing	GB	Nintendo	31.38
Tetris	Puzzle	GB	Nintendo	30.26
New Super Mario Bros.	Platform	DS	Nintendo	30.01
Wii Play	Misc	Wii	Nintendo	29.01
New Super Mario Bros. Wii	Platform	Wii	Nintendo	28.61
Duck Hunt	Shooter	NES	Nintendo	28.31
Nintendogs	Simulation	DS	Nintendo	24.75
Mario Kart DS	Racing	DS	Nintendo	23.43
Pokemon Gold/Pokemon Silver	Role-Playing	GB	Nintendo	23.09
Wii Fit	Sports	Wii	Nintendo	22.72

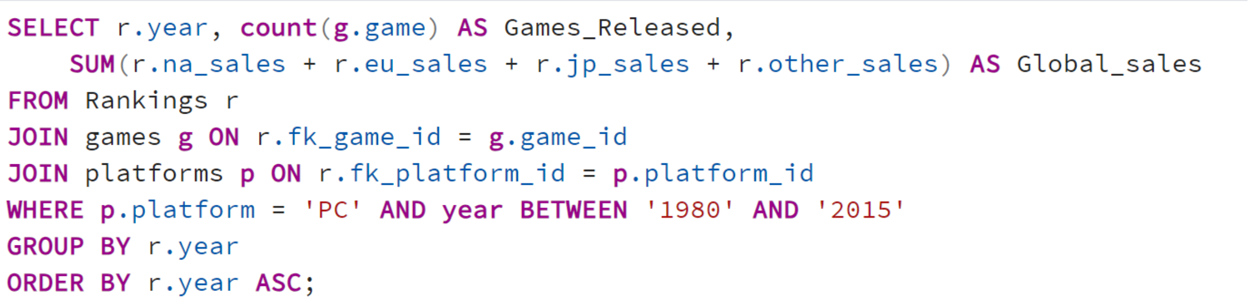
Query Purpose:

In this query, we created a materialized view that derives information regarding the top 100 best-selling video games of all time across all platforms and genres. We performed this by selecting the relevant columns for the information we wanted to derive, using an aggregate function (SUM) to calculate the global sales for each game, joining the additional (child) tables that contained the relevant columns that we wanted to use in the query to our parent table (Rankings), grouping all of the non-aggregate columns together, ordering the global sales column in descending order to list the information from the number one best-selling video game of all time to the one hundredth bestselling video game of all time, and limiting the amount of rows that are returned in the output to one hundred.

Part 3: Query Analysis

Graphical Analysis of a Query

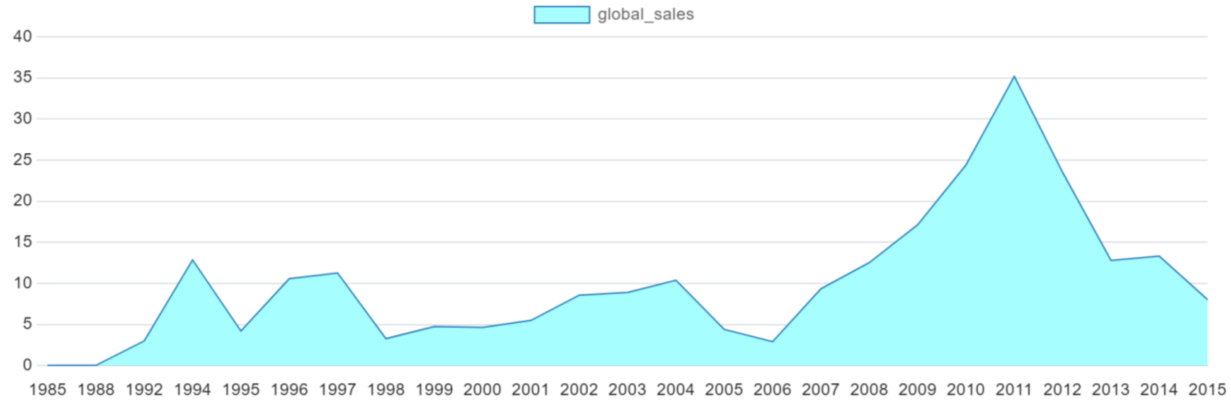
Screenshot of Query:



```
SELECT r.year, count(g.game) AS Games_Released,
    SUM(r.na_sales + r.eu_sales + r.jp_sales + r.other_sales) AS Global_sales
FROM Rankings r
JOIN games g ON r.fk_game_id = g.game_id
JOIN platforms p ON r.fk_platform_id = p.platform_id
WHERE p.platform = 'PC' AND year BETWEEN '1980' AND '2015'
GROUP BY r.year
ORDER BY r.year ASC;
```

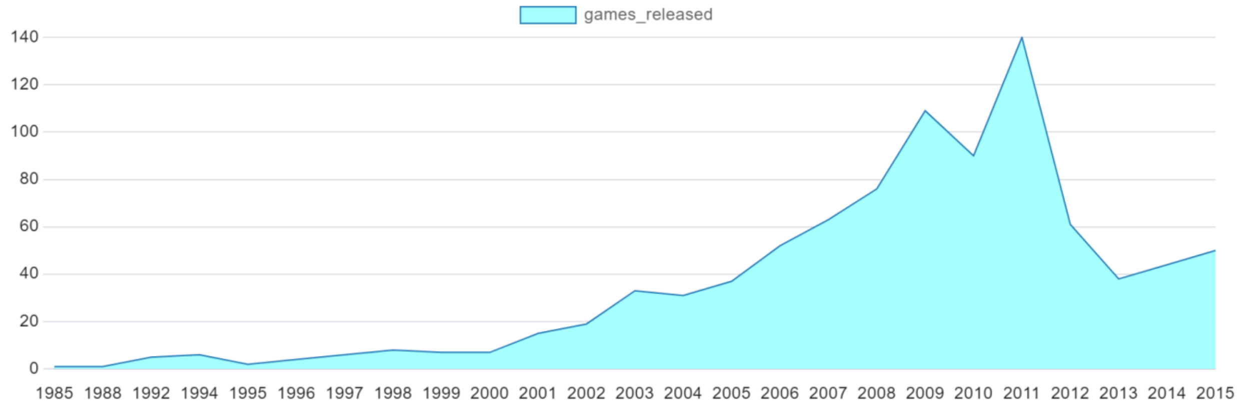
Query Purpose: This query returns a count of games released and the total global sales of these games by year for the PC platform. The purpose of finding this information is to create visuals which can show us the growth of PCs as a gaming platform through 2 different metrics.

Global Sales Over Time:



This graph displays steady rising and falling from the early 90's all the way to the early 2000's where we see an extreme rise peaking in 2011 before falling dramatically. We are unsure of the cause of this incredible fall.

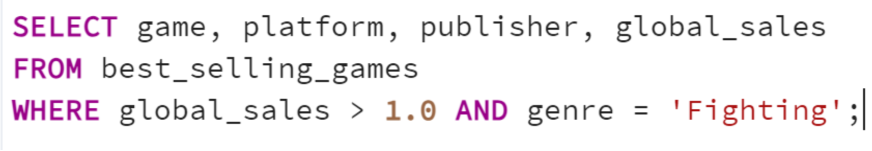
Games Released Over Time:



We see a very similar pattern in the games released, a steadier climb into the 2000's towards 2 peaks instead of the 1 we saw in sales. Also of note is the climbing pattern after the dramatic fall from 2011's peak. This climb is noticeably absent in the global sales graph.

Optimal Indexing of a Query

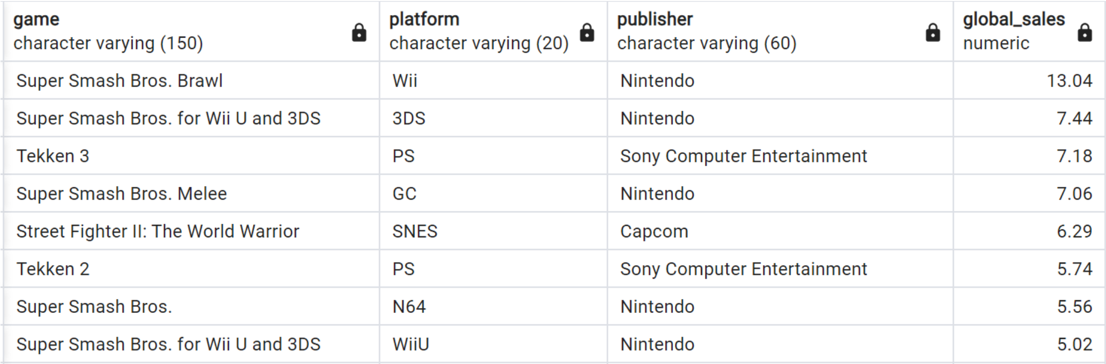
Screenshot of Query:



```
SELECT game, platform, publisher, global_sales
FROM best_selling_games
WHERE global_sales > 1.0 AND genre = 'Fighting';
```

This query above returns all fighting games that have more than a million total global sales.

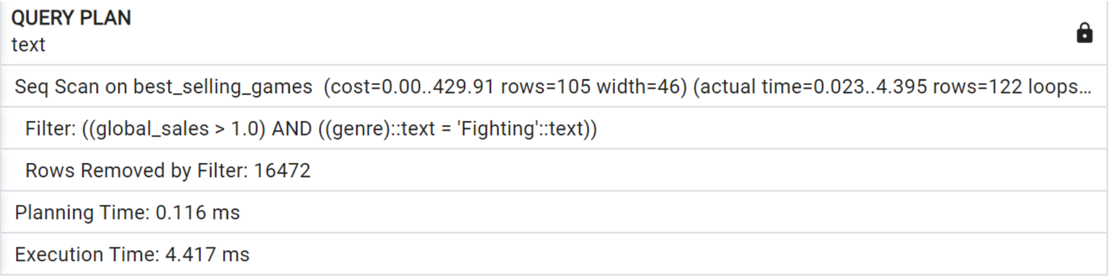
Screenshot of Result Set:



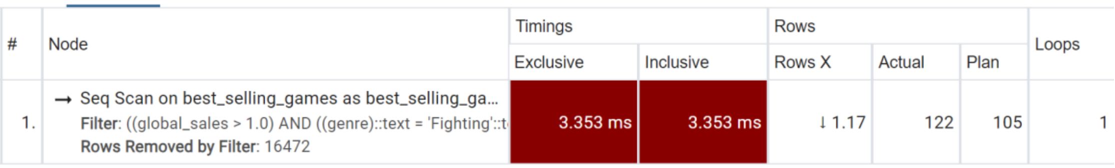
game character varying (150)	platform character varying (20)	publisher character varying (60)	global_sales numeric
Super Smash Bros. Brawl	Wii	Nintendo	13.04
Super Smash Bros. for Wii U and 3DS	3DS	Nintendo	7.44
Tekken 3	PS	Sony Computer Entertainment	7.18
Super Smash Bros. Melee	GC	Nintendo	7.06
Street Fighter II: The World Warrior	SNES	Capcom	6.29
Tekken 2	PS	Sony Computer Entertainment	5.74
Super Smash Bros.	N64	Nintendo	5.56
Super Smash Bros. for Wii U and 3DS	WiiU	Nintendo	5.02

And this above is an example of its results.

Query Plan Before Indexing:



QUERY PLAN text
Seq Scan on best_selling_games (cost=0.00..429.91 rows=105 width=46) (actual time=0.023..4.395 rows=122 loops...
Filter: ((global_sales > 1.0) AND ((genre)::text = 'Fighting'::text))
Rows Removed by Filter: 16472
Planning Time: 0.116 ms
Execution Time: 4.417 ms



#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Seq Scan on best_selling_games as best_selling_ga... Filter: ((global_sales > 1.0) AND ((genre)::text = 'Fighting'::t Rows Removed by Filter: 16472	3.353 ms	3.353 ms	↓ 1.17	122	105	1

best_selling_games

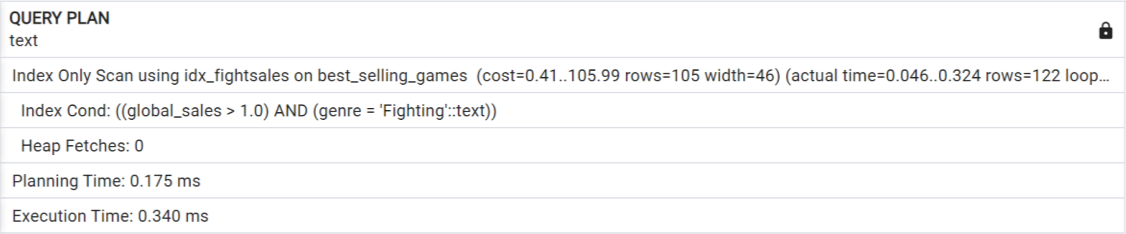
Above is the text-based query plan as well as graphical and analysis tools provided in PGAdmin. The query they describe is very simple and requires only a sequential scan on the materialized view containing the data.

Index Applied to the Materialized View

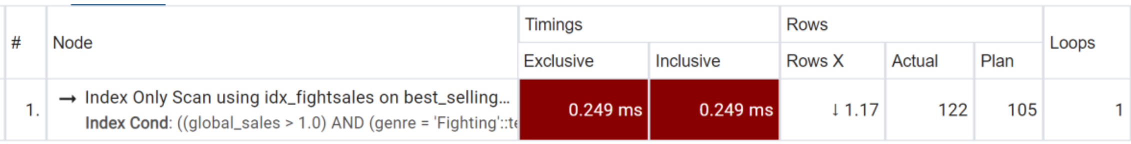
```
CREATE INDEX idx_fightSales
ON best_selling_games(global_sales, genre, game, platform, publisher);
```

This index is being applied to every column which is relevant to the query which we want to improve. The columns “global_sales and genre” must be positioned in the index in that order or else the index will not be used. This is because of their order in the where clause of the query.

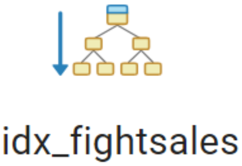
Query Plan After Indexing:



```
QUERY PLAN
text
Index Only Scan using idx_fightsales on best_selling_games  (cost=0.41..105.99 rows=105 width=46) (actual time=0.046..0.324 rows=122 loop…
  Index Cond: ((global_sales > 1.0) AND (genre = 'Fighting'::text))
  Heap Fetches: 0
Planning Time: 0.175 ms
Execution Time: 0.340 ms
```



#	Node	Timings		Rows			Loops
		Exclusive	Inclusive	Rows X	Actual	Plan	
1.	→ Index Only Scan using idx_fightsales on best_selling… **Index Cond**: ((global_sales > 1.0) AND (genre = 'Fighting'::te	0.249 ms	0.249 ms	↓ 1.17	122	105	1



Above is the text-based query plan as well as graphical and analysis tools provided in PGAdmin. After applying an index to all the columns involved in the query, the query executor recognizes it would be faster to use the index and performs an index-only scan because all the data it needs can be found within. This is the fastest way a database can retrieve data using an index and is possible because of the extreme simplicity of the query itself. But as is evident by comparing the two execution plans, the index has vastly improved the speed and lowered the cost considerably. As such it is clear that this single index is the optimal number of indexes for maximum performance.

Conclusion

In this document, we have successfully demonstrated the completion of our final project for DBAS3035. We have achieved this by providing an Entity Relationship Diagram (ERD) for the "Video Game Sales" dataset that has been created in Third-Normal Form (3NF), providing a data dictionary for the dataset, creating a database and tables for the dataset that consist of the datasets imported data, creating SQL queries that obtain meaningful results, and analyzing our queries by utilizing various methods.

References

Video game sales. (2016, October 26). Kaggle. https://www.kaggle.com/datasets/gregorut/videogamesales