PROJECT in SQL from DATACAMP

When Was the Golden Age of Video Games?

This is a real-world project in SQL covering Intermediate knowledge in SQL.

This project uses techniques learned in Joining Data with SQL, including left and inner joins, set theory concepts such as union and intercept, and subqueries. You'll also be expected to know concepts from Introduction to SQL, such as how to select columns from a table, filter rows where they meet a criterion, use aggregation functions, perform calculations on groups of rows, and filter grouped data.

The project consists of 8 tasks to be completed in order to complete the project.

game_sales

column	type	meaning
`game`	varchar	Name of the video game
`platform`	varchar	Gaming platform
`publisher`	varchar	Game publisher
`developer`	varchar	Game developer
`games_sold`	float	Number of copies sold (millions)
`year`	int	Release year

`reviews`

column	type	meaning
`game`	varchar	Name of the video game
`critic_score`	float	Critic score according to Metacritic
`user_score`	float	User score according to Metacritic

Task 1: Instructions

Video games are big business: the global gaming market is projected to be worth more than $300 billion by 2027 according to Mordor Intelligence. With so much money at stake, the major game publishers are hugely incentivized to create the next big hit. But are games getting better, or has the golden age of video games already passed?

In this project, we'll explore the top 400 best-selling video games created between 1977 and 2020. We'll compare a dataset on game sales with critic and user reviews to determine whether or not video games have improved as the gaming market has grown.

Let's find the ten best-selling video games in `game_sales`.

- Select all columns for the top ten best-selling video games (based on `games_sold`) in `game_sales`.
- Order the results from the best-selling game down to the tenth best-selling game.
- *The line `postgresql:///games` is used to connect to the database; don't remove it.*

```
%%sql
postgresql:///games
-- Select all information for the top ten best-selling games
SELECT *
FROM game_sales
-- Order the results from best-selling game down to tenth best-selling
ORDER BY games_sold DESC
LIMIT 1
```

2. Missing review scores

Wow, the best-selling video games were released between 1985 to 2017! That's quite a range; we'll have to use data from the reviews table to gain more insight on the best years for video games.

First, it's important to explore the limitations of our database. One big shortcoming is that there is not any reviews data for some of the games on the game_sales table.

Let's determine how many games in the `game_sales` table are missing both a `user_score` and a `critic_score`.

- Join the `game_sales` and `reviews` tables together so that all games from the `game_sales` table are listed in the results, whether or not they have associated reviews.
- Select the count of games where both the associated `critic_score` and the associated `user_score` are null.

```
%%sql
-- Join games_sales and reviews
SELECT COUNT(g.game)
FROM game_sales AS g
LEFT JOIN reviews AS r
ON g.game = r.game
WHERE critic_score IS Null AND user_score IS Null
```

3. Years that video game critics loved

It looks like a little less than ten percent of the games on the game_sales table don't have any reviews data. That's a small enough percentage that we can continue our exploration, but the missing reviews data is a good thing to keep in mind as we move on to evaluating results from more sophisticated queries.

There are lots of ways to measure the best years for video games! Let's start with what the critics think.

Find the years with the highest average critic_score.

- Select release year and average critic score for each year; average critic score for each year will be rounded to two decimal places and aliased as avg_critic_score.
- Join the game_sales and reviews tables so that only games which appear on *both* tables are represented.
- Group the data by release year.
- Order the data from highest to lowest avg_critic_score and limit the results to the top ten years.

```
%%sql
-- Select release year and average critic score for each year, rounded and aliased
SELECT g.year, ROUND(AVG(r.critic_score),2) AS avg_critic_Score
-- Join the game_sales and reviews tables
FROM game_sales AS g
LEFT JOIN reviews AS r
USING(game)
-- Group by release year
GROUP BY g.year
-- Order the data from highest to lowest avg_critic_score and limit to 10 results
ORDER BY avg_critic_score DESC
LIMIT 10
```

4. Was 1982 really that great?

The range of great years according to critic reviews goes from 1982 until 2020: we are no closer to finding the golden age of video games!

Hang on, though. Some of those avg_critic_score values look like suspiciously round numbers for averages. The value for 1982 looks especially fishy. Maybe there weren't a lot of video games in our dataset that were released in certain years.

Let's update our query and find out whether 1982 really was such a great year for video games.

Find game critics' ten favorite years, this time with the stipulation that a year must have more than four games released in order to be considered.

- Starting with your query from the previous task, update it so that the selected data additionally includes a count of games released in a given year, and alias this count as num_games.
- Filter your query so that only years with more than four games released are returned.
-

```
%%sql
-- Paste your query from the previous task; update it to add a count of games released in each year called num_games
-- Select release year and average critic score for each year, rounded and aliased
SELECT g.year, COUNT(g.game) AS num_games, ROUND(AVG(r.critic_score),2) AS avg_critic_score
FROM game_sales g
INNER JOIN reviews r
ON g.game = r.game
GROUP BY g.year
HAVING COUNT(g.game) > 4
ORDER BY avg_critic_score DESC
LIMIT 10;
```

5. Years that dropped off the critics' favorites list

That looks better! The num_games column convinces us that our new list of the critics' top games reflects years that had quite a few well-reviewed games rather than just one or two hits. But which years dropped off the list due to having four or fewer reviewed games? Let's identify them so that someday we can track down more game reviews for those years and determine whether they might rightfully be considered as excellent years for video game releases!

It's time to brush off your set theory skills. To get started, we've created tables with the results of our previous two queries:

`top_critic_years`

column	type	meaning
`year`	int	Year of video game release
`avg_critic_score`	float	Average of all critic scores for games released in that year

`top_critic_years_more_than_four_games`

column	type	meaning
`year`	int	Year of video game release
`num_games`	int	Count of the number of video games released in that year
`avg_critic_score`	float	Average of all critic scores for games released in that year

Use set theory to find the years that were on our first critics' favorite list but not the second.

- Select the `year` and `avg_critic_score` for those years that were on our first critics' favorite list but not the second due to having four or fewer reviewed games.
- Order the results from highest to lowest `avg_critic_score`.

```
%%sql

SELECT year, avg_critic_score
FROM top_critic_years
EXCEPT
SELECT year, avg_critic_score
FROM top_critic_years_more_than_four_games
ORDER BY avg_critic_score DESC;
```

6. Years video game players loved

Based on our work in the task above, it looks like the early 1990s might merit consideration as the golden age of video games based on critic_score alone, but we'd need to gather more games and reviews data to do further analysis.

Let's move on to looking at the opinions of another important group of people: players! To begin, let's create a query very similar to the one we used in Task Four, except this one will look at user_score averages by year rather than critic_score averages.

Update your query from Task Four so that it returns years with ten highest `avg_user_score` values.

- You'll still select year and an average of `user_score` for each year, rounded to two decimal places and aliased as `avg_user_score`; also include a count of games released in a given year, aliased as `num_games`.
- Include only years with more than four reviewed games; group the data by year.
- Order data from highest to lowest `avg_user_score`, and limit the results to the top ten years.

```
%%sql

SELECT g.year, COUNT(g.game) AS num_games, ROUND(AVG(r.user_score),2) AS avg_user_score
FROM game_sales g
INNER JOIN reviews r
ON g.game = r.game
GROUP BY g.year
HAVING COUNT(g.game) > 4
ORDER BY avg_user_score DESC
LIMIT 10;
```

7. Years that both players and critics loved

Alright, we've got a list of the top ten years according to both critic reviews and user reviews. Are there any years that showed up on both tables? If so, those years would certainly be excellent ones!

Recall that we have access to the top_critic_years_more_than_four_games table, which stores the results of our top critic years query from Task 4:

`top_critic_years_more_than_four_games`

column	type	meaning
`year`	int	Year of video game release
`num_games`	int	Count of the number of video games released in that year
`avg_critic_score`	float	Average of all critic scores for games released in that year

We've also saved the results of our top user years query from the previous task into a table:

`top_user_years_more_than_four_games`

column	type	meaning
`year`	int	Year of video game release
`num_games`	int	Count of the number of video games released in that year
`avg_user_score`	float	Average of all user scores for games released in that year

Create a list of years that appear on both the `top_critic_years_more_than_four_games` table and the `top_user_years_more_than_four_games` table.

- Using set theory, select only the `year` results that appear on both tables.

```
%%sql

SELECT year
FROM top_user_years_more_than_four_games
INTERSECT
SELECT year
FROM top_critic_years_more_than_four_games;
```

8. Sales in the best video game years

Looks like we've got three years that both users and critics agreed were in the top ten! There are many other ways of measuring what the best years for video games are, but let's stick with these years for now. We know that critics and players liked these years, but what about video game makers? Were sales good? Let's find out.

This time, we haven't saved the results from the previous task in a table for you. Instead, we'll use the query from the previous task as a subquery in this one! This is a great skill to have, as we don't always have write permissions on the database we are querying.

Add a column showing total games_sold in each year to the table you created in the previous task.

- Select year and the sum of games_sold, aliased as total_games_sold; order your results by total_games_sold descending.
- Filter the game_sales table based on whether the year is in the list of years you returned in the previous task, using your code from the previous task as a subquery.
- Group the results by year.

```
%%sql

SELECT g.year, SUM(g.games_sold) AS total_games_sold
FROM game_sales g
WHERE g.year IN (SELECT year
FROM top_user_years_more_than_four_games
INTERSECT
SELECT year
FROM top_critic_years_more_than_four_games)
GROUP BY g.year
ORDER BY total_games_sold DESC;
```