When Was the Golden Age of Video Games?

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 Notebook has been uploaded for this project. It is noted that the project is completed in mySQL, you can fork and clone the project and make a PR for notebook in any other language.

List of tasks for the Project:

Task 1:

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.*

Task 2:

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.

Task 3:

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.

Task 4:

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.

Task 5:

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`.

Task 6:

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.

Task 7:

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.

Task 8:

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`.