CA2: SQL REFLECTION

Jonathan Love

Reflection on Group Work

Throughout this group project, the five of us collaborated together and encountered issues that reflect those of a proper development team. Prior to starting any coding at all, we created a group chat and tried to find times outside of practical time that we could meet to divvy up the work relatively equally and begin working, which was where we encountered our first issue. Finding significant amounts of time during which we were all free was difficult, so we chose to meet for a short time in the Meston building together after a shared lecture to assign everyone five or six questions and instead do much of the work in our own time, helping each other where necessary via our group chat. I was assigned questions 2, 7, 12, 17, 22, and 27.

We also decided that using a GitHub repository would help us work on our questions individually. We put the .rb and .db files into the repo, and created our own branches to work on. Once we'd verified that our assigned questions ran properly in Ruby, we merged our individual branches with main to create the final, completed file.

There were also instances in which we had to change our queries to fit the expected answer. For example, the expected output for SQLQ12 required a lower case 'COUNT' statement as the column header, whereas I would typically write 'COUNT' instead of 'count'. Additionally, while semicolons aren't necessary in SQL, we were instructed to use them in this assessment as good practice.

As for improvements, many of our queries are extremely vulnerable to SQL injection attacks, which would be my first action for any further work on these queries. Using wildcards in our queries allows anyone to input their own queries, allowing the ability to access to potentially sensitive data or delete or modify existing data. Further improvements could be made to the efficiency of queries, too. Querying entire tables or databases with potential repeat entries is inefficient, and could be improved by using the 'DISTINCT' keyword to eliminate duplicates (especially when working with 'SUM' or 'AVG'). Also, sorting data would improve the experience for anyone using the database, such as sorting the 'customers' or 'employees' tables alphabetically by last name.

Reflection on Individual Questions

(SQLQ2)

My first query required me to display data from the ‘employees’ table while renaming a the columns in the output table. I selected the three columns from the table and used the ‘AS’ command to create aliases for each.

(SQLQ7)

Question 7 asked to display the invoice ID and the full names of both the customer and the support representative for the invoice with an ID of 1 without using the ‘JOIN’ keyword. Rather than using ‘JOIN’, I added commas in the ‘FROM’ clause to select data from multiple tables, using ‘WHERE’ to join them using rows with matching values and search for the invoice with an ID of 1.

(SQLQ12)

Question 12 required me to write a query that outputs a table with the number of customers in the ‘customers’ table who are from either Germany or Canada in ascending order. I selected the ‘Country’ column and a select-all count function where the customer’s nationality is ‘Canada’ or ‘Germany’, ordered by the ‘Country’ column using ‘ASC’ to put them in ascending order. Finally, I had to use a ‘GROUP BY’ statement to make sure the resulting table would differentiate between the two countries and not group them both into a single row.

(SQLQ17)

My next query needs to display all employees who are reported to by other employees. I selected the two name columns and title column from the ‘employees’ table, and used a nested query to select all employees who’s employee ID appears in the ‘ReportsTo’ column.

(SQLQ22)

Question 22 asked for a query that displays the album ID and title of all albums by AC/DC using a nested query. I selected the two required columns from the ‘albums’ table, then matched the ‘ArtistId’ columns from ‘albums’ and ‘artists’ where the artist’s name entry is ‘AC/DC’ using a nested query.

(SQLQ27)

The final question asked for artist's name and ID, and the number of albums produced containing the genre 'Soundtrack' on one of the album's tracks using only 'LEFT JOIN' statements. I selected the artist name and ID from the 'artists' table, a count of albums from the 'albums' table, and the genre name ('Soundtrack') from the 'genres' table. I joined artists and albums on the 'ArtistId' column, albums and tracks on the 'AlbumId' column, and tracks and genres on the 'GenreId' column. With all of the necessary tables joined using 'LEFT JOIN' statements, I searched for albums and their tracks with the genre 'Soundtrack', excluded any results that had multiple or no composers in the 'tracks' table, and separated the resulting data by their album ID.