Make a series of queries that meet the following requirements. Write a paragraph about each one including a general overview of the process (i.e. talk about functions in SQL), the expected result, the actual query, and a screenshot of the result.

* Creating, altering, and dropping tables

Creating a table is just as simple as creating a new table of data within the database to be able to access and manipulate as needed, altering is the manipulation of that data. You would use altering if you needed to rename a column, add columns, change constraints on a column or table as needed. Lastly dropping a table is used to remove the table completely, you can also drop a column if needed. I demonstrated the alter table statement to add a “watched” column to my original table to show if that episode has been watched before. The values start out as null until I update the column later with a random number generated to show which episodes have been watched.

* SET operations

Set operations are very convenient if you want to combine date together or separate data between two tables. I used the union all as an example, I combined two tables, the original table of the bob ross episodes, and the new table I downloaded off git hub where the list is of the paintings rather than the episodes and the tables do cross over with the episode name/painting name, and the season and episode. I actually completed this query after the subquery below since the table I added didn’t line up season and episode wise, I needed to update the table in order to match them up later in a join. The union all query I ran, combined both tables showing the episode column and the title column.

* Subqueries

Subqueries are great for getting more done in one query and simplifying what you need manipulated in the table. For this query, I needed to update my second table to match the episode column in the first table. The second table had a season and episode column and I used a case when function combined with concat to combine both columns into a new column where the syntax matched the first table. The case when was needed so I wouldn’t add any extra 0’s or not have enough when combining both columns.

* Order of operation of queries

Just like math, the process of a query is ran in a certain order, I decided to demonstrate that using a query with extra constraints to show what is ran first to complete the query. I used a select statement, but inside the select statement, first the inner join is ran to combine the tables, following that the where clause to filter the list down to those parameters, then the select query pulls what’s needed, after that the order by clause shows how to order the data when completing the query, and finally the limit clause to only show 15 rows that were pulled from the tables.

* Associations (foreign keys)
* Joins and/or multiple table joins

Join statements are a good way to get pieces of data from more than one table, which is a huge part of using a relational database, because you have that convenience to be able to combine tables by columns that line up and you’re able to line up what data you need like if you needed to see orders by order date, by who completed them, who ordered them, and you get everything you need all in one query. I used an inner join just to combine both tables and show the episodes that were watched alongside with the second table that doesn’t have the watched column to be able to see that. I did a very simple query, but there’s a lot of power in using join statements to get information