1. **SET OPERATIONS:**

**QUERY:**

***SELECT episode, title FROM `elements-by-episode` WHERE title LIKE '%BOAT%'***

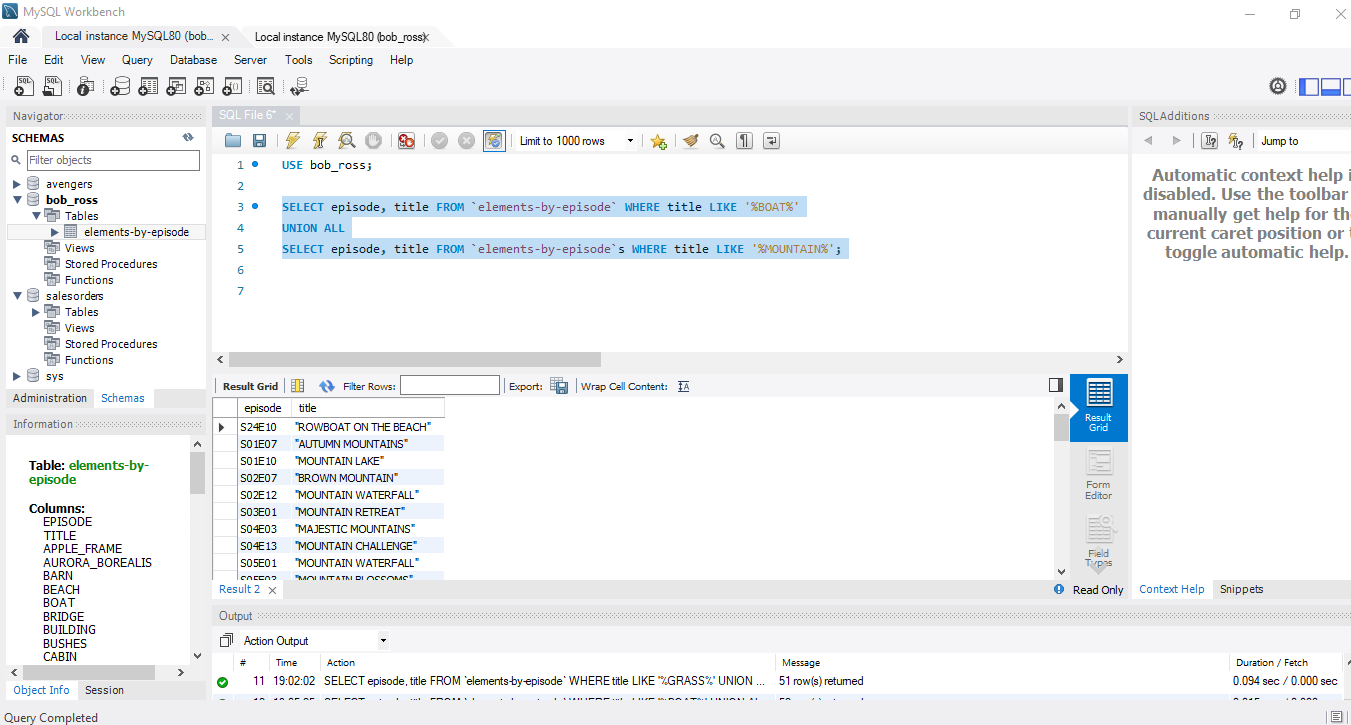
***UNION ALL***

***SELECT episode, title FROM `elements-by-episode`s WHERE title LIKE '%MOUNTAIN%'***

***ORDER BY title;***

**Overview of the process:**

The UNION ALL set operation command was used to combine the results of 2 SELECT statements. UNION ALL the titles that have the words “BOAT” and “MOUNTAIN” and using the while card %%.



1. **Subqueries**

**QUERY:**

**SELECT BARN, BRIDGE, BUILDING, TITLE, EPISODE**

**FROM `elements-by-episode`**

**WHERE BARN IN (SELECT BARN**

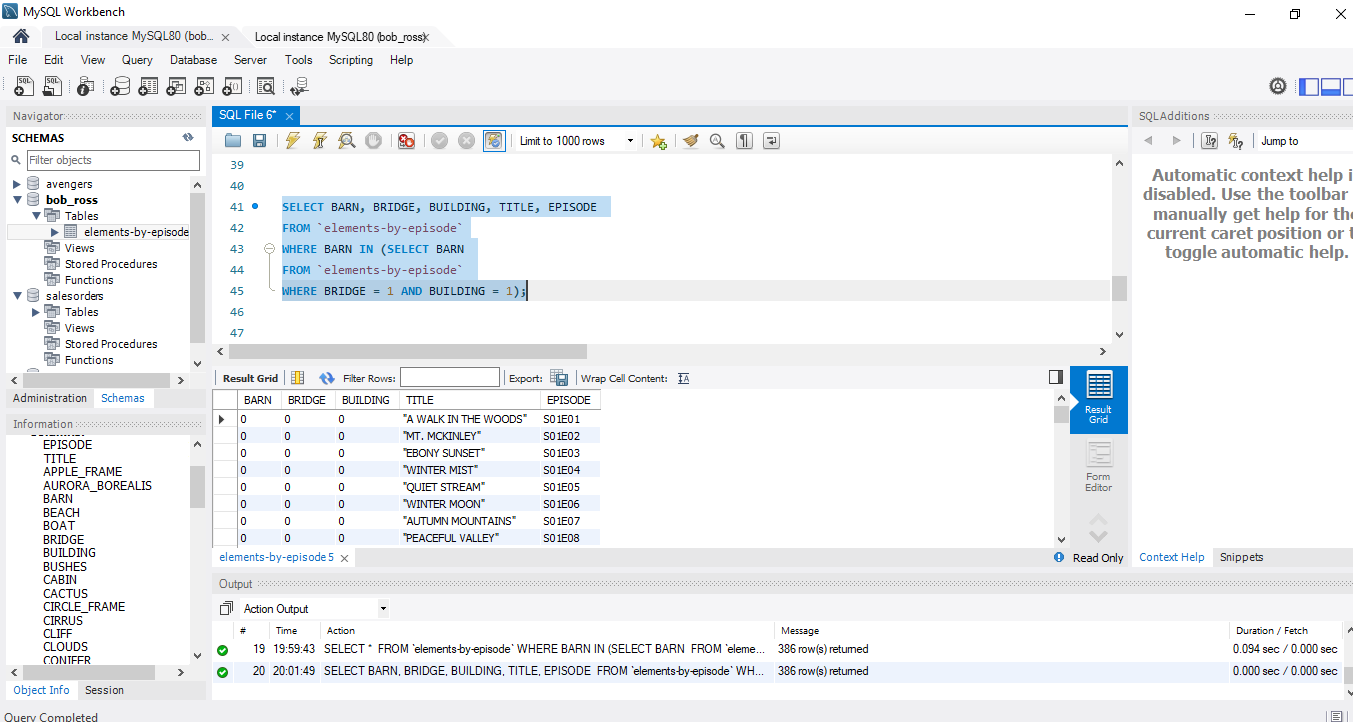
**FROM `elements-by-episode`**

**WHERE BRIDGE = 1 AND BUILDING = 1);**

**Overview of the process:**

**In subqueries I used the Select command to isolate the columns barn, bridge, building, title,**

**Episode. Then using the From command to pick the elements-by-episode table. Finally, I used the Where command to pick which data I wanted to isolate from the titles in the title column.**

****

1. **Order of operation of queries**

**QUERY:**

**SELECT TITLE, BEACH, BOAT**

**FROM `elements-by-episode`**

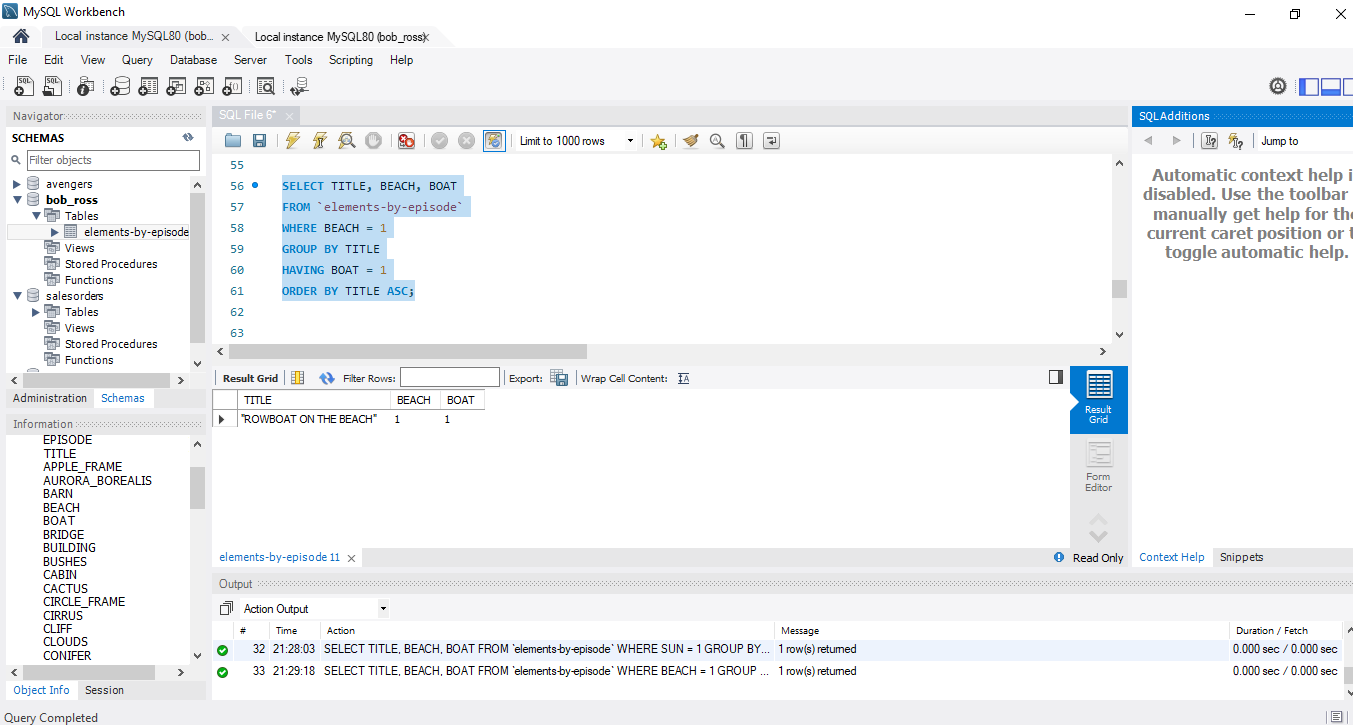
**WHERE BEACH = 1**

**GROUP BY TITLE**

**HAVING BOAT = 1**

**ORDER BY TITLE ASC**

**For the order of operations of queries I used, SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY. The first operation to be executed is SELECT this is used to pick the columns I wanted, then using the FROM command to pick the specific table (episode-by-element). Next, I used the WHERE clause to find SUN, and next using the From command to pick requested columns from the episode-by-element table. The next command order is the GROUP BY which aggregates the data in the column TITLE. After I grouped the data, the next ordered command is HAVING which only searches for BOAT. Finally, I used the ORDER BY command which will take the TITLE column and data values and will associate it to BEACH and BOAT which there was only row that matched. Also, I used the number 1 in the HAVING and WHERE commands for BEACH AND BOAT to see if there was 1 match for each with TITLE column.**



1. **CREATING, ALTERING, AND DROPPING TABLES:**

**QUERY:**

**CREATE TABLE EARTH\_ELEMENTS LIKE `elements-by-episode`;**

**SELECT \* FROM EARTH\_ELEMENTS;**

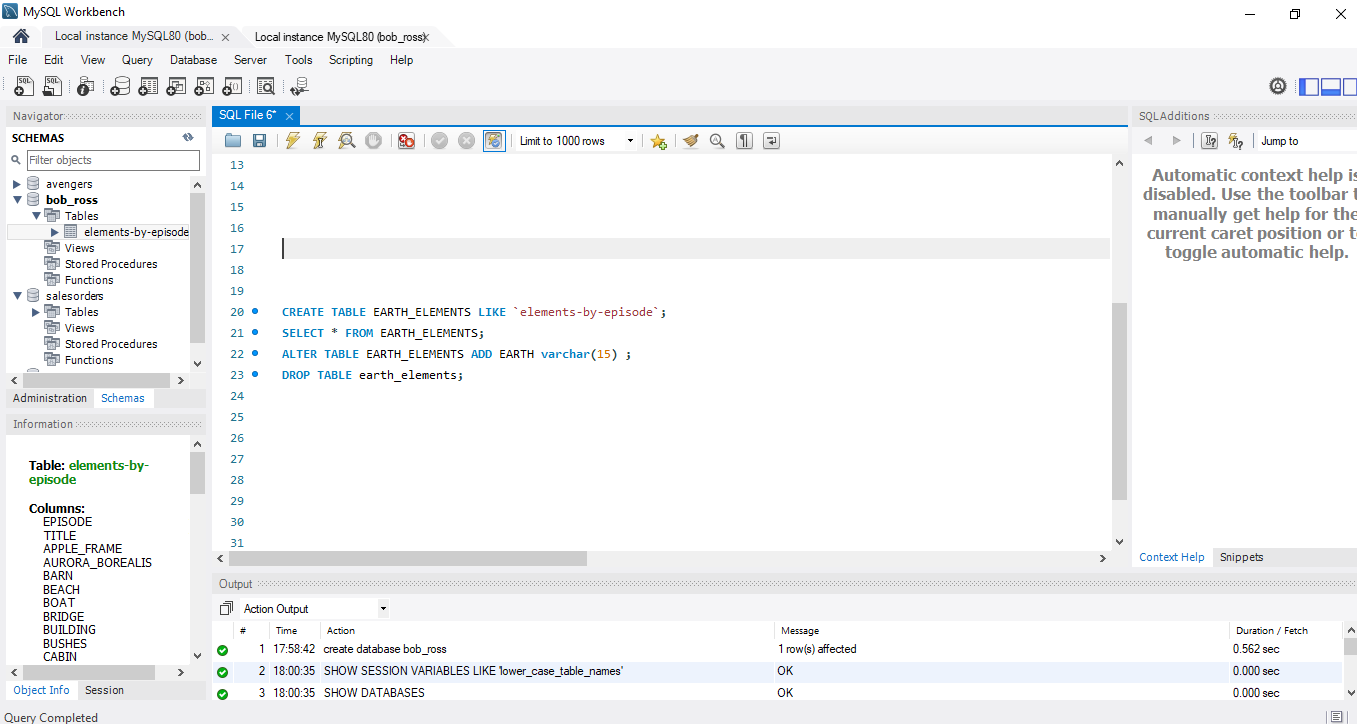
**ALTER TABLE EARTH\_ELEMENTS ADD EARTH varchar(15) ;**

**DROP TABLE artist\_elements**;

**EXPLANATION:**

**Created a table name EARTH\_ELEMENTS like the `elements-by-episode table.**

**Then I selected a new table, the table was then altered by adding a new column and named EARTH. Altering the table was a success then I deleted the table with the command Drop Table which was a success.**



1. **Associations**

**QUERY:**

**CREATE TABLE ADDTL\_EPISODE\_INFO (**

**EPISODE varchar(6) NOT NULL,**

**EPISODE\_NETWORK varchar(3) NOT NULL,**

**EPISODE\_DURATION int,**

**NIELSEN\_RATING int,**

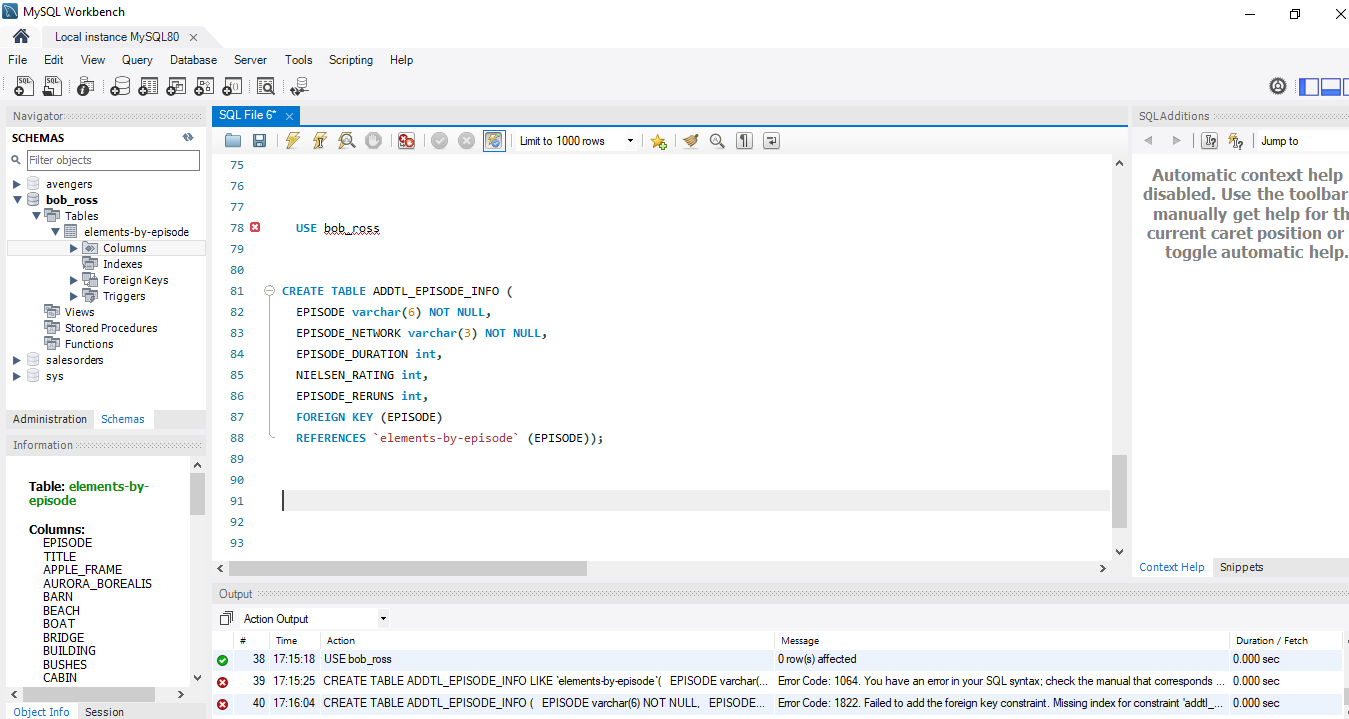
**EPISODE\_RERUNS int,**

**FOREIGN KEY (EPISODE)**

**REFERENCES `elements-by-episode` (EPISODE));**

**Overview:**

**An association defines a relationship between two objects based on common attributes. The relationship can be one-to-one or one-to-many; you can use two one-to-many associations to implement a many-to-many relationship. I was unable to run my query. I am not sure why I am getting the error messages about the foreign key restraints. I have tried to find the resolution on the internet but still can not figure out what is wrong with my commands or the foreign key. From what I understand, association is getting data from multiple tables while using a primary key and foreign key which is where lies the relationship between the tables.**



1. **Joins and multiple table joins**

**Query:**

**SELECT `elements-by-episode`.EPISODE, `elements-by-episode`.TITLE, earth.TITLE, earth.MOUNTAINS, earth.HILLS, earth.TREES, earth.BEACH**

**FROM `elements-by-episode`**

**LEFT JOIN earth**

**ON `elements-by-episode`.title = earth.title**

**ORDER BY earth.title ASC;**

**Overview:**

**The LEFT JOIN keyword returns all records from the left table (elements-by-episode table), and the matching records from the right table (earth table).**

**Graphical user interface, text, application, email

Description automatically generated**