SET OPERATIONS:

I used UNION

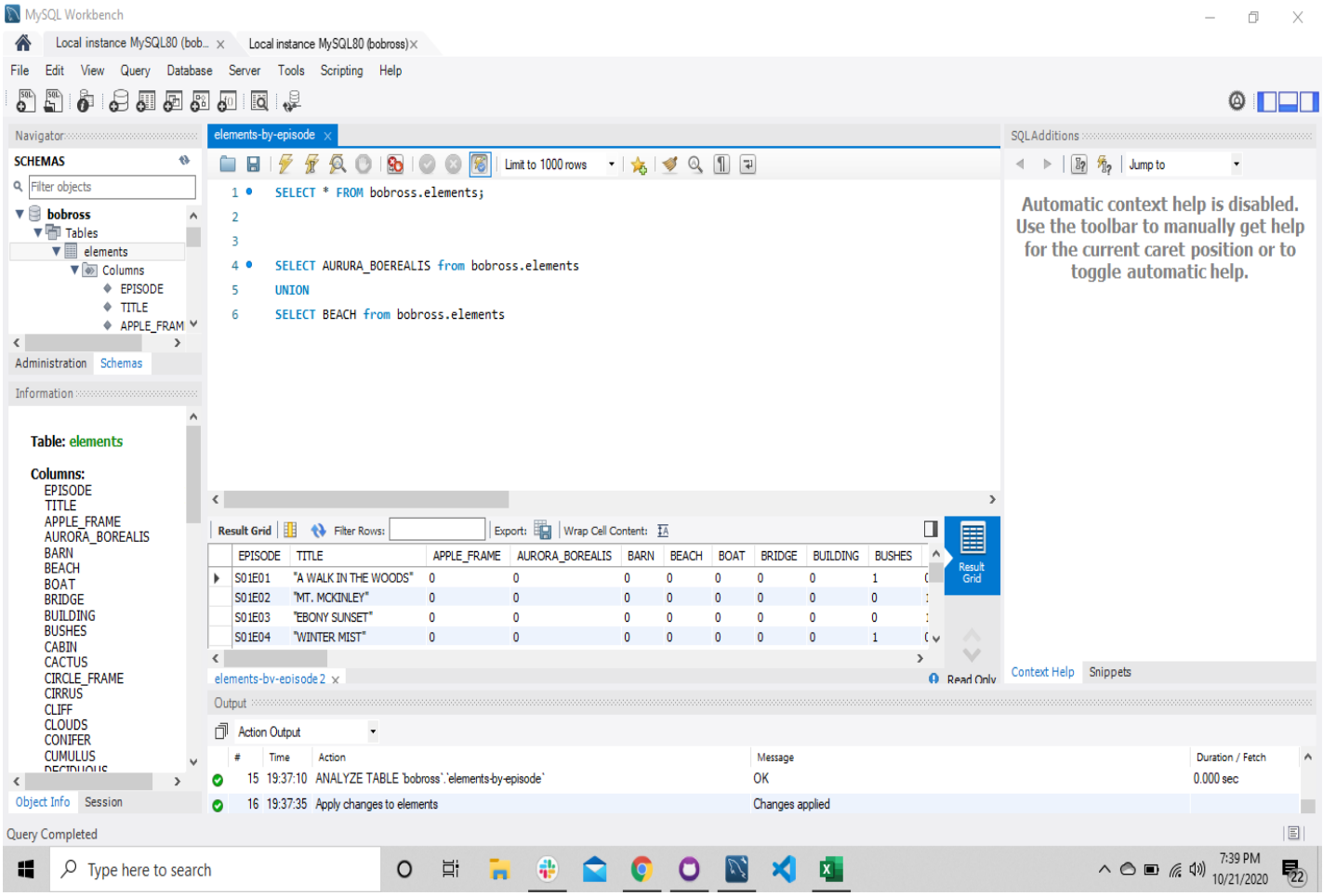
UNION operator

In SQL the UNION clause combines the results of two SQL queries into a single table of all matching rows. The two queries must result in the same number of columns and compatible data types in order to unite. Any duplicate records are automatically removed unless UNION ALL is used.

SELECT AURORA_BOREALIS from bobross.elements

UNION

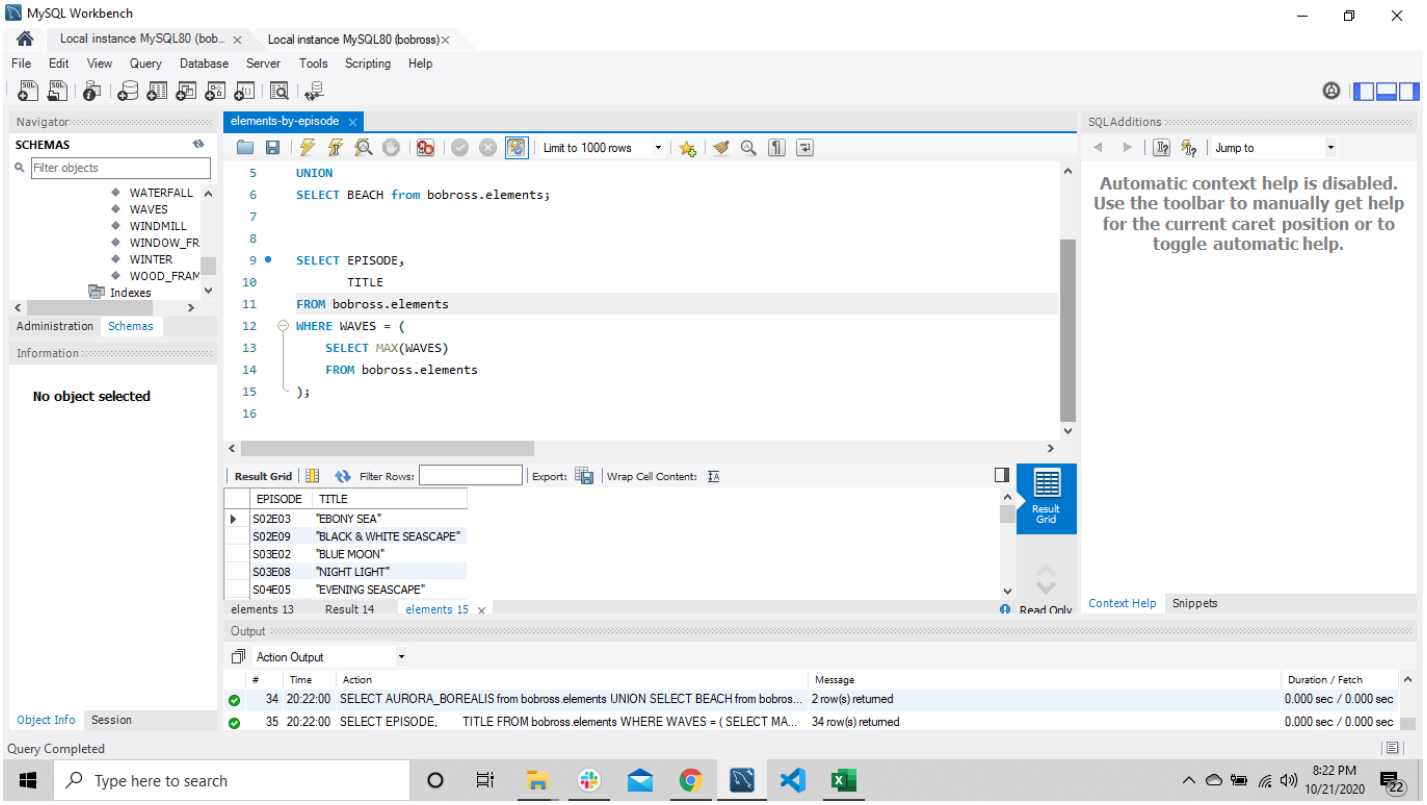
SELECT BEACH from bobross.elements;



SUBQUERIES:

Subqueries have may be called other names as well such as an Inner query. A Nested query is a query within another SQL query and embedded within the WHERE clause. Which is what I believe I used. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN,

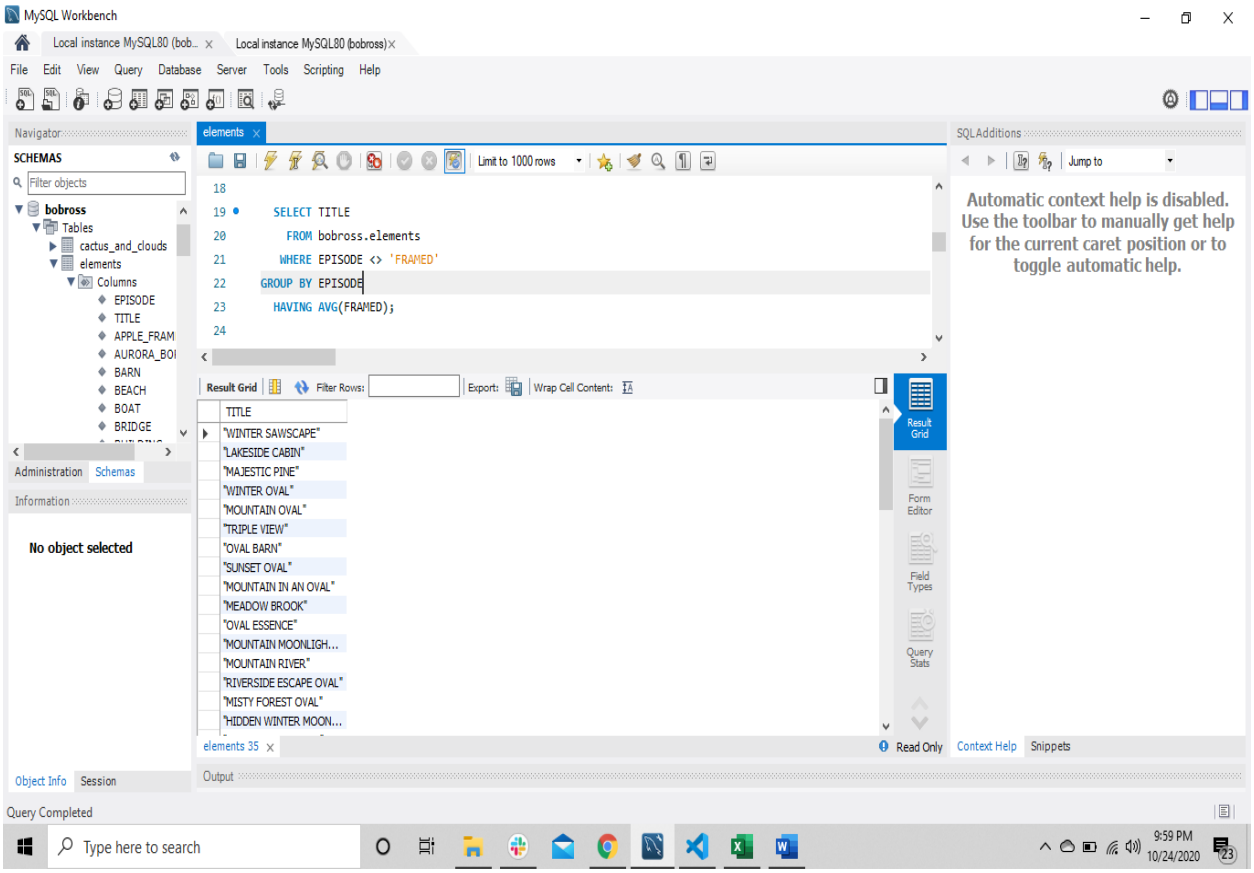
```
SELECT EPISODE,
    TITLE
FROM bobross.elements
WHERE WAVES = (
    SELECT MAX(WAVES)
    FROM bobross.elements
);
```



ORDER OF OPERATIONS:

The SQL order of execution defines the order in which the clauses of a query are evaluated. Six orders of operations are SELECT, FROM, WHERE, GROUP, HAVING and ORDER BY. I have demonstrated my example below.

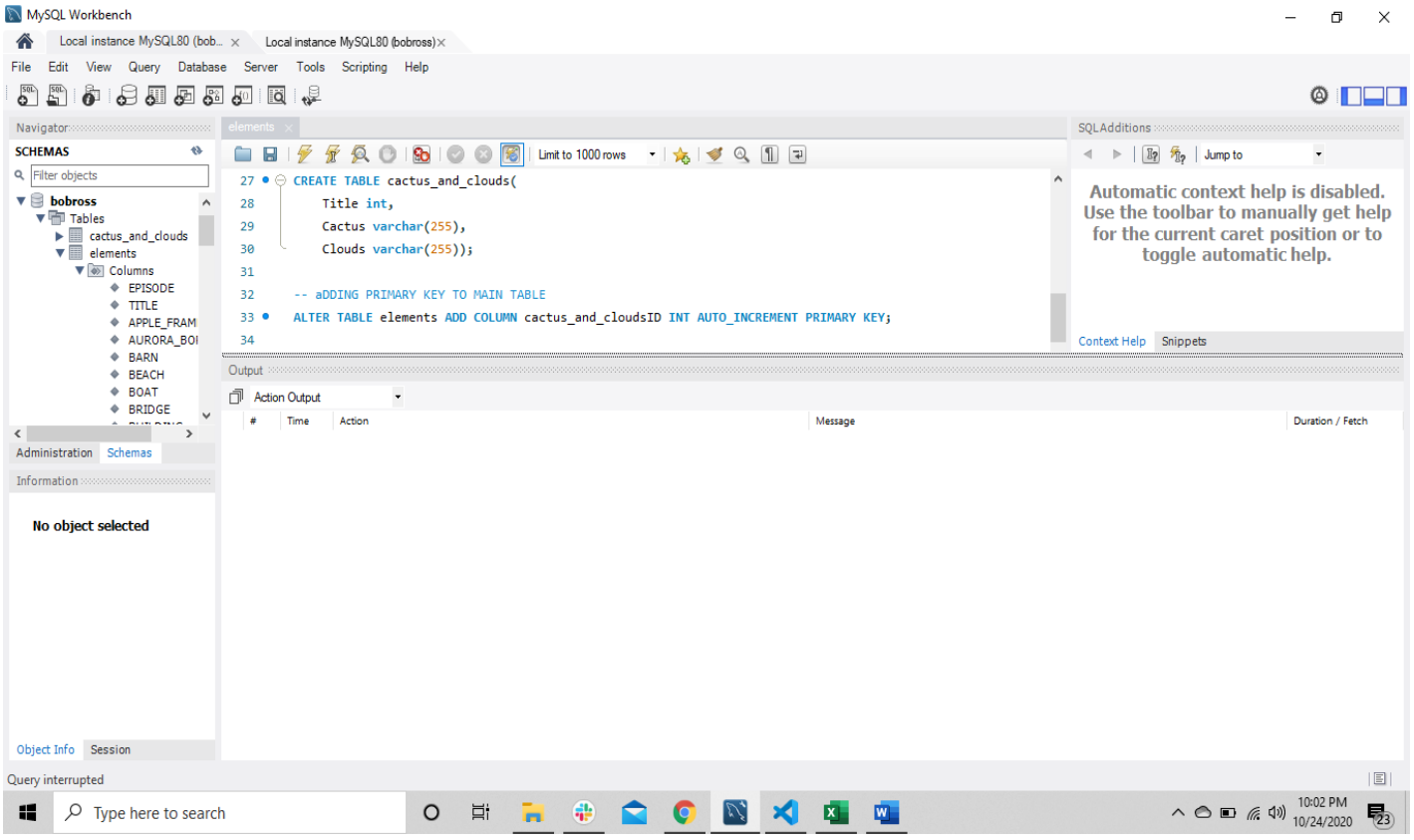
```
SELECT TITLE
  FROM bobross.elements
  WHERE EPISODE <> 'FRAMED'
GROUP BY EPISODE
  HAVING AVG(FRAMED)
```



CREATING and ALTERING TABLES:

You can define the columns and assign field attributes when you create the table. After a table has been set up, it can be modified using the ALTER table command. To create a table, use the following command with the new table's name, column names, and data type for each column. To drop a table simply use DROP TABLE.

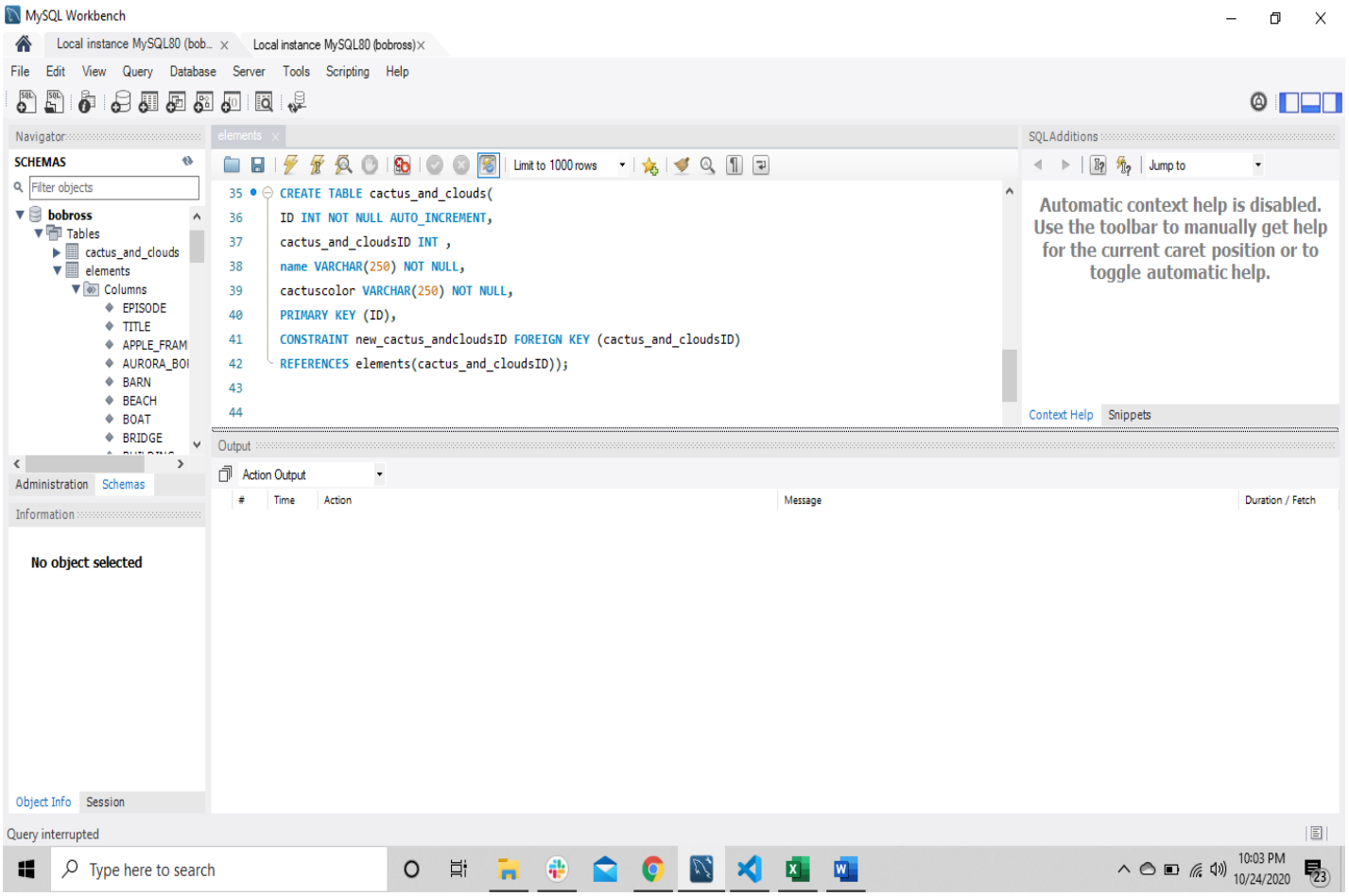
```
CREATE TABLE cactus_and_clouds(
    Title int,
    Cactus varchar(255),
    Clouds varchar(255));

ALTER TABLE elements ADD COLUMN cactus_and_cloudsID INT AUTO_INCREMENT PRIMARY KEY;
```



ASSOCIATIONS:

An association are tables that are used for many-to-many relationships between two objects. They consist of at least two foreign keys, each of which references one of the two objects. My example is below.

```
CREATE TABLE cactus_and_clouds(

ID INT NOT NULL AUTO_INCREMENT,

cactus_and_cloudsID INT ,

name VARCHAR(250) NOT NULL,

cactuscolor VARCHAR(250) NOT NULL,

PRIMARY KEY (ID),

CONSTRAINT new_cactus_andcloudsID FOREIGN KEY (cactus_and_cloudsID)

REFERENCES elements(cactus_and_cloudsID));

 INSERT INTO cactus_and_clouds(cactus_and_cloudsID, cactuscolor, name)
values (5,'blue','prickly'),(10, 'green', 'soft'),(15, 'white','fluffy');
```


MySQL Workbench
Local instance MySQL80 (bob...
Local instance MySQL80 (bobross)
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bobross
Tables
cactus_and_clouds
elements
Columns
EPISODE
TITLE
APPLE_FRAM
AURORA_BOI
BARN
BEACH
BOAT
BRIDGE
Administration Schemas
Information
No object selected
Object Info Session
elements
Limit to 1000 rows
35 CREATE TABLE cactus_and_clouds(
36 ID INT NOT NULL AUTO_INCREMENT,
37 cactus_and_cloudsID INT ,
38 name VARCHAR(250) NOT NULL,
39 cactuscolor VARCHAR(250) NOT NULL,
40 PRIMARY KEY (ID),
41 CONSTRAINT new_cactus_andcloudsID FOREIGN KEY (cactus_and_cloudsID)
42 REFERENCES elements(cactus_and_cloudsID));
43
44
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
Query interrupted
Type here to search
10:03 PM
10/24/2020

JOIN:

A JOIN creates a set that can be saved as a table or used as it is. A JOIN is a means for combining columns from one or more tables by using values common to each.

```
SELECT distinct a.title, a.episode, b.name, b.cactuscolor, b.cactus_and_cloudsID
FROM elements a
JOIN cactus_and_clouds b
ON b.cactus_and_cloudsID = b.cactus_and_cloudsID
ORDER by a.title;
```