

## Creating, altering, and dropping tables

**CREATE** "wild\_animals" in bob\_ross Schema. **ALTER** the table by adding animals. Snapshot "A" shows the table & Snapshot "B", **DROPPED** the table.

### Snapshot "A"

The screenshot shows the SQL Server Enterprise Manager interface. In the left-hand 'SCHEMAS' pane, the 'bob\_ross' schema is expanded, and the 'wild\_animals' table is selected. The 'Columns' pane shows the table structure with columns: SNOWY\_MOUNTAIN, SPLIT\_FRAME, STEVE\_ROSS, STRUCTURE, SUN, TOMB\_FRAME, TREE, TREES, TRIPLE\_FRAME, WATERFALL, WAVES, WINDMILL, WINDOW\_FRAME, WINTER, WOOD\_FRAMED, and ANIMALS. The 'ALTER TABLE' statement is highlighted in the SQL script pane.

```
1 CREATE TABLE wild_animals LIKE `bob_ross`;
2 SELECT * FROM wild_animals;
3 ALTER TABLE wild_animals ADD ANIMALS varchar(10);
```

The 'Output' pane at the bottom shows the execution results of the SQL script:

#	Time	Action	Message	Duration / Fetch
4	18:49:58	ALTER TABLE wild_animals ADD ANIMALS varchar(10)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
5	18:52:17	CREATE TABLE wild_animals LIKE `bob_ross`	Error Code: 1050. Table 'wild_animals' already exists	0.016 sec
6	18:52:30	SELECT * FROM wild_animals	0 row(s) returned	0.000 sec / 0.000 sec

### Snapshot "B"

The screenshot shows the SQL Server Enterprise Manager interface. In the left-hand 'SCHEMAS' pane, the 'bob\_ross' schema is expanded, and the 'wild\_animals' table is selected. The 'DROP TABLE' statement is highlighted in the SQL script pane.

```
1 CREATE TABLE wild_animals LIKE `bob_ross`;
2 SELECT * FROM wild_animals;
3 ALTER TABLE wild_animals ADD ANIMALS varchar(10);
4 DROP TABLE wild_animals;
```

The 'Output' pane at the bottom shows the execution results of the SQL script:

#	Time	Action	Message
6	18:52:30	SELECT * FROM wild_animals	0 row(s) returned
7	18:54:04	CREATE TABLE wild_animals LIKE `bob_ross`	Error Code: 1050. Tabl
8	18:54:17	DROP TABLE wild_animals	0 row(s) affected

## SET OPERATIONS – Using the **UNION ALL** combines 2 STATEMENTS using both words “**TREE**” & “**TREES**”

The screenshot displays a SQL IDE interface with the following components:

- Navigator:** Shows the database structure with 'avengers' and 'bob\_ross' schemas. The 'bob\_ross' schema is expanded, showing a table 'bob\_ross' with columns: Already\_Watched, EPISODE, TITLE, APPLE\_FRAME, AURORA\_BOREALIS, BARN, BEACH, BOAT, BRIDGE, BUILDING, BUSHES, CABIN, CACTUS, CIRCLE\_FRAME, and CIRRUS.
- SQL Editor:** Contains the following SQL query:

```
1 USE bob_ross;
2
3 SELECT episode, title FROM `bob_ross` WHERE title LIKE '%TREE%'
4 UNION ALL
5 SELECT episode, title FROM `bob_ross`s WHERE title LIKE '%TREES%'
6 ORDER BY title;
7
8
```
- Result Grid:** Displays the query results in a table with columns 'episode' and 'title':

episode	title
S12E04	""BRIGHT AUTUMN TREES""
S12E04	""BRIGHT AUTUMN TREES""
S20E08	""THE OLD OAK TREE""
- Action Output:** Shows the execution log with the following entries:

#	Time	Action	Message	Duration
2	15:30:30	SELECT EPISODE FROM bob_ross WHERE ROCKS = 1	77 row(s) returned	0.000 se
3	15:36:57	USE bob_ross	0 row(s) affected	0.000 se
4	15:36:57	SELECT episode, title FROM `bob_ross` WHERE title LIKE '%TREE%' UNION ALL SELECT episode, title FROM ...	3 row(s) returned	0.000 se

**Subqueries** – **SELECT** to choose headers from the columns; followed by **FROM** the SCHEMA Finally, **WHERE** data I wanted to pull from the column.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' list with 'CONIFER' selected. The main editor shows a SQL query in 'SQL File 9':

```
1 • USE bob_ross;
2
3 • SELECT TITLE, BUSHES, CABIN, CLOUDS, CONIFER, EPISODE
4 • FROM bob_ross WHERE TITLE IN (SELECT TITLE
5 • FROM bob_ross WHERE BUSHES = 1 AND CLOUDS = 1
6 • AND CONIFER = 1 AND CABIN = 1);
```

Below the query editor, the 'Result Grid' displays the following data:

	TITLE	BUSHES	CABIN	CLOUDS	CONIFER	EPISODE
▶	""SHADES OF GREY""	1	1	1	1	S02E04
	""HIGH CHATEAU""	1	1	1	1	S06E09
	""COUNTRY CABIN""	1	1	1	1	S11E02
	""STEEP MOUNTAINS""	1	1	1	1	S12E06
	""FROZEN SOLITUDE""	1	1	1	1	S13E02
	""EVENING AT SUNSET""	1	1	1	1	S13E04
	""PERFECT WINTER'S DAY""	1	1	1	1	S29E11
	""BEFORE THE SNOWFALL""	1	1	1	1	S31E02

The bottom panel shows the 'Output' tab with 'Action Output' selected. It displays the execution log:

#	Time	Action	Message
✗ 41	18:07:52	SELECT TITLE, BUSHES, CABIN, CLOUDS, CONIFER, EPISODE FROM bob_ross WHERE TITLE IN (SELEC...	Error Code: 1064. You have an error in your SQL syntax; check the manual that
✓ 42	18:07:58	USE bob_ross	0 row(s) affected
✓ 43	18:07:58	SELECT TITLE, BUSHES, CABIN, CLOUDS, CONIFER, EPISODE FROM bob_ross WHERE TITLE IN (SELEC...	8 row(s) returned

Order by Operations -  
SELECT column headers, FROM Schema, WHERE table, GROUP BY, combined or aggregated by Title, HAVING only the item, ORDER BY ascending.

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

avengers

bob\_ross

Tables

bob\_ross

Columns

Already\_Watched

EPISODE

TITLE

APPLE\_FRAME

AURORA\_BOREALI

BARN

BEACH

BOAT

BRIDGE

BUILDING

BUSHES

CABIN

CACTUS

CIRCLE\_FRAME

CIRRUS

Administration

Schemas

Information

Column: EPISODE

Collation: utf8mb4\_0900\_ai\_ci

Definition: EPISODE text

SQL File 8\*

SQL File 9\*

1 USE bob\_ross;

2

3 SELECT TITLE, CABIN, SNOW, NIGHT, MOUNTAINS, MOON, WINTER, EPISODE

4 FROM `bob\_ross` WHERE NIGHT > 0 GROUP BY TITLE HAVING MOON > 0

5 ORDER BY TITLE ASC;

6

7

8

9

10

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	TITLE	CABIN	SNOW	NIGHT	MOUNTAINS	MOON	WINTER	EPISODE
▶	***BLUE MOON***	0	0	1	0	1	0	S03E02
	***EVENING'S PEACE***	0	1	1	0	1	1	S19E12
	***WINTER MOON***	1	1	1	1	1	1	S01E06

bob\_ross32

Output

Action Output

#	Time	Action	Message
✖ 77	18:20:04	SELECT TITLE, CABIN, SNOW, NIGHT, MOUNTAINS, MOON, WINTER, EPISODE FROM `bob_ross` WHER...	Error Code: 1064. You have an error in your SQL syntax;
✔ 78	18:20:10	USE bob_ross	0 row(s) affected
✔ 79	18:20:10	SELECT TITLE, CABIN, SNOW, NIGHT, MOUNTAINS, MOON, WINTER, EPISODE FROM `bob_ross` WHER...	3 row(s) returned

## Associations (foreign keys)

An association defines relationships between two objects based on common attributes. It can be one-to-one or one-to-many.

The screenshot shows the DBeaver SQL editor interface. On the left, the 'SCHEMAS' panel displays a tree view of databases, with 'bob\_ross' selected and its 'Tables' folder expanded. The 'Table: bob\_ross\_paintings' is highlighted. The main editor area shows a SQL query: `SELECT EPISODE FROM bob_ross UNION SELECT TITLE FROM bob_ross;`. Below the query, the 'Result Grid' shows the execution results. The first result set, 'Result 6', contains 804 rows of episode numbers (S01E01 to S01E12). The second result set, 'Result 7', contains an error message: 'Error Code: 1054. Unknown column 'TITLE' in 'field list''. The error message is circled in red.

File Edit View Query Database Server Tools Scripting Help

SQL File 8\* SQL File 10\* x

Don't Limit

1  
2  
3  
4 • `SELECT EPISODE FROM bob_ross`  
5 `UNION`  
6 `SELECT TITLE FROM bob_ross;`

Result Grid Filter Rows: Export: Wrap Cell Content: A

EPISODE  
S01E01  
S01E02  
S01E03  
S01E04  
S01E05  
S01E06  
S01E07  
S01E08  
S01E09  
S01E10  
S01E11  
S01E12  
S01E13

Result 6 x

Output

Action Output

#	Time	Action	Message
✓ 91	20:28:59	SELECT EPISODE FROM bob_ross UNION SELECT TITLE FROM bob_ross	804 row(s) returned
✗ 92	20:29:25	SELECT EPISODE FROM bob_ross_paintings UNION SELECT TITLE FROM bob_ross_paintings	Error Code: 1054. Unknown column 'TITLE' in 'field list'
✓ 93	20:29:34	SELECT EPISODE FROM bob_ross UNION SELECT TITLE FROM bob_ross	804 row(s) returned

Object Info Session

Joins and/or multiple table joins – joined both tables 1) bob.ross and 2) bob\_ross\_paintings.

The screenshot displays the SQL Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows a tree view with 'bob\_ross' and 'bob\_ross\_paintings' tables. The 'Columns' of 'bob\_ross\_paintings' are listed, including 'youtube\_src'. The main pane shows a SQL query in 'SQL File 10' that performs a LEFT JOIN between 'bob\_ross' and 'bob\_ross\_paintings' on the 'TITLE' column. The query is as follows:

```
1
2 SELECT `bob_ross`.EPISODE, `bob_ross`.TITLE, bob_ross_paintings.painting_title,
3 bob_ross_paintings.season, bob_ross_paintings.episode, bob_ross_paintings.youtube_src
4 FROM bob_ross
5 LEFT JOIN bob_ross_paintings
6 ON `bob_ross`.TITLE = bob_ross_paintings.painting_title
7 ORDER BY bob_ross_paintings.painting_title ASC;
```

Below the query, the 'Result Grid' shows the data returned by the query. The columns are EPISODE, TITLE, painting\_title, season, episode, and youtube\_src. The data is as follows:

EPISODE	TITLE	painting_title	season	episode	youtube_src
S01E01	***A WALK IN THE WOODS***	NULL	NULL	NULL	NULL
S01E02	***MT. MCKINLEY***	NULL	NULL	NULL	NULL
S01E03	***EBONY SUNSET***	NULL	NULL	NULL	NULL
S01E04	***WINTER MIST***	NULL	NULL	NULL	NULL
S01E05	***QUIET STREAM***	NULL	NULL	NULL	NULL
S01E06	***WINTER MOON***	NULL	NULL	NULL	NULL
S01E07	***AUTUMN MOUNTAINS***	NULL	NULL	NULL	NULL
S01E08	***PEACEFUL VALLEY***	NULL	NULL	NULL	NULL
S01E09	***SEASCAPE***	NULL	NULL	NULL	NULL
S01E10	***MOUNTAIN LAKE***	NULL	NULL	NULL	NULL
S01E11	***WINTER GLOW***	NULL	NULL	NULL	NULL
S01E12	***SNOWFALL***	NULL	NULL	NULL	NULL
S01E13	***FINAL REFLECTIONS***	NULL	NULL	NULL	NULL

At the bottom, the 'Output' pane shows the execution log. The log indicates that the query returned 404 row(s) and that the 'youtube\_src' column is of type text.

Column: youtube\_src  
Collation: utf8mb4\_0900\_ai\_ci  
Definition: youtube\_src text

Result 8 x

Output

Action Output

#	Time	Action	Message
93	20:29:34	SELECT EPISODE FROM bob_ross UNION SELECT TITLE FROM bob_ross	804 row(s) returned
94	20:35:31	SELECT `bob_ross`.EPISODE, `bob_ross`.TITLE, bob_ross_paintings.painting_title, bob_ross_paintings.seas...	403 row(s) returned
95	20:36:13	SELECT `bob_ross`.EPISODE, `bob_ross`.TITLE, bob_ross_paintings.painting_title, bob_ross_paintings.seas...	403 row(s) returned