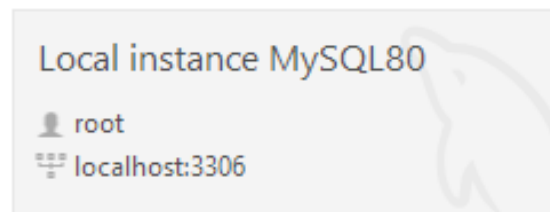


Final Project Prompts

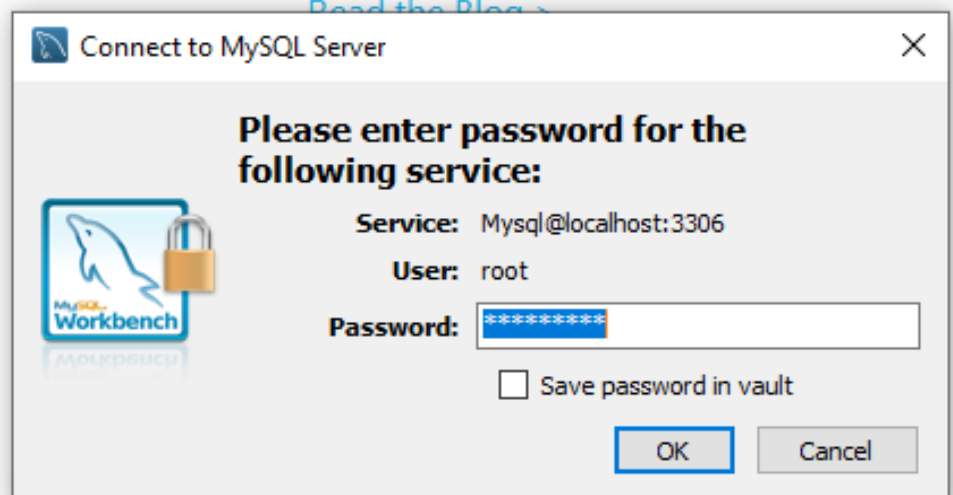
Prompt 1 - Create the Schema/Database

- 01.
02. Launch your local instance and login:

MySQL Connections



database vendors to your MySQL database.



03. When MySQL Workbench loads, the **Schemas** displays on the left-hand side of the window.



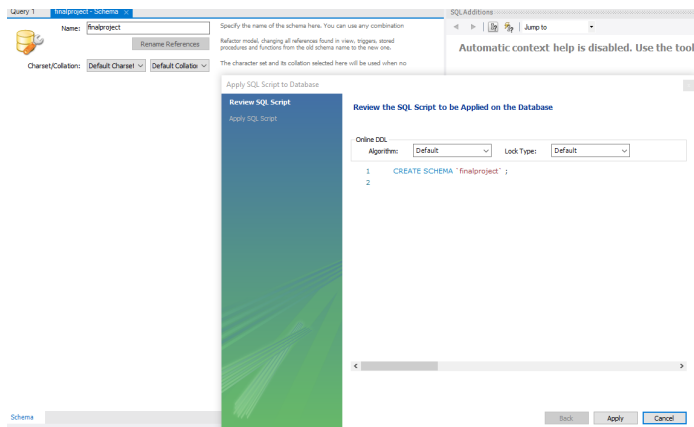
04. In the Schemas panel of MySQL, **right-click** and select **Create Schema**. Note, you may not have any existing schemas listed; that is OK.

05. Let's call it 'finalproject'. Type the name in and then click the Apply button at the lower-right corner of the window.

06. Click Apply at the next screen:

07. Then click Finish

08.



09.

10. Label this screen capture 'Prompt 1'

SCHEMAS

Filter objects

- ▼ finalproject
 - Tables
 - Views
 - Stored Procedures
 - Functions
- ▶ sakila
- ▶ sys
- ▶ world

Administration Schemas

Information

Schema: **finalproject**

1

<

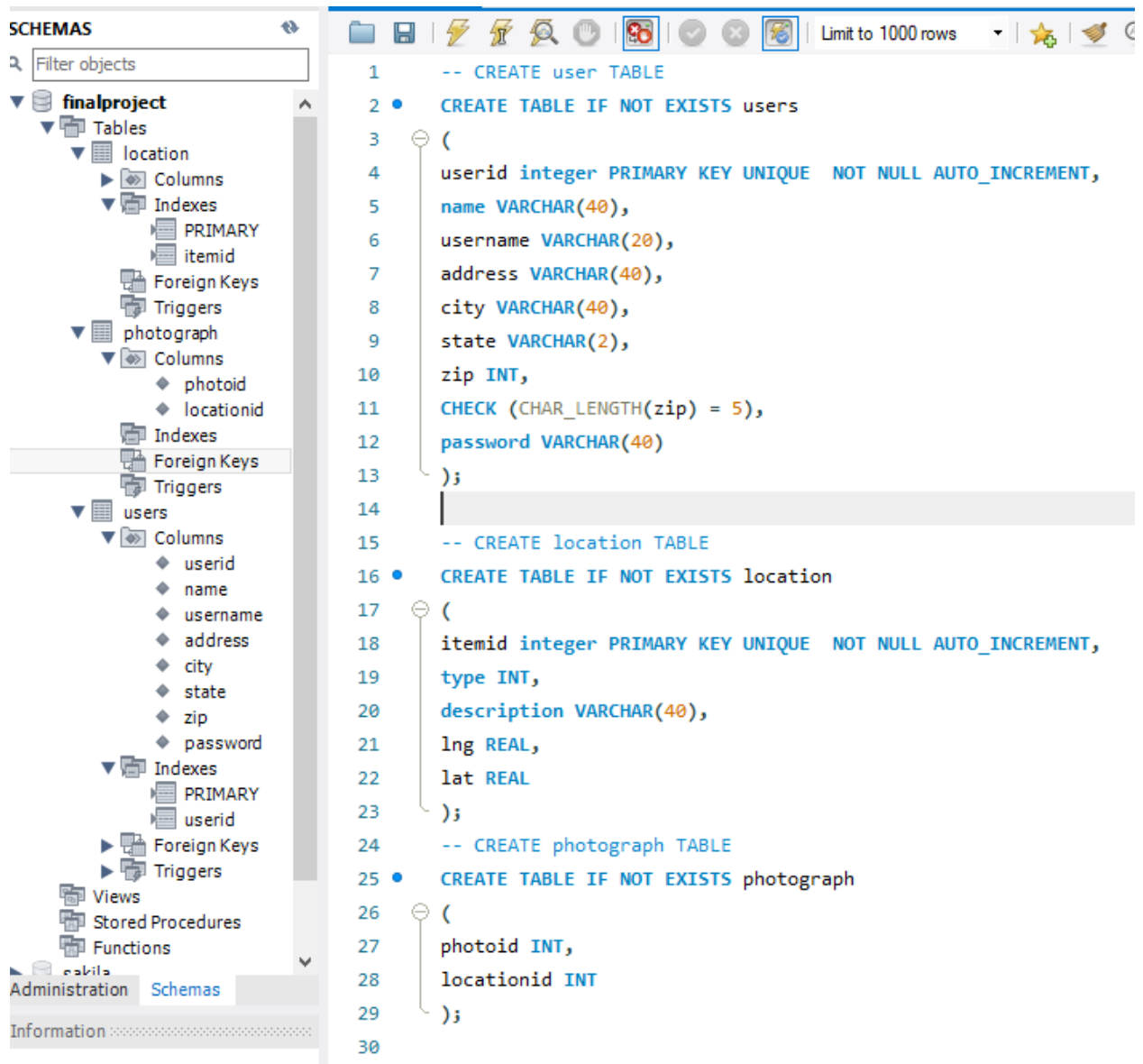
Output



Action Output

	#	Time	Action
✓	1	22:55:18	Apply changes to finalproject
!	2	22:57:09	Apply changes to finalproject

11. Prompt 2 - Create Tables



The screenshot displays a database management interface. On the left, a 'SCHEMAS' pane shows a tree view for a database named 'finalproject'. The tree includes folders for 'Tables', 'location', 'photograph', and 'users'. The 'users' folder is expanded, showing columns (userid, name, username, address, city, state, zip, password), indexes (PRIMARY, userid), foreign keys, and triggers. The 'location' folder is also expanded, showing columns (itemid, type, description, lng, lat) and indexes (PRIMARY, itemid). The 'photograph' folder is expanded, showing columns (photoid, locationid) and indexes (PRIMARY, photoid). The 'Administration' tab is selected at the bottom.

On the right, a SQL editor shows the following code:

```
1  -- CREATE user TABLE
2  CREATE TABLE IF NOT EXISTS users
3  (
4      userid integer PRIMARY KEY UNIQUE NOT NULL AUTO_INCREMENT,
5      name VARCHAR(40),
6      username VARCHAR(20),
7      address VARCHAR(40),
8      city VARCHAR(40),
9      state VARCHAR(2),
10     zip INT,
11     CHECK (CHAR_LENGTH(zip) = 5),
12     password VARCHAR(40)
13 );
14
15 -- CREATE location TABLE
16 CREATE TABLE IF NOT EXISTS location
17 (
18     itemid integer PRIMARY KEY UNIQUE NOT NULL AUTO_INCREMENT,
19     type INT,
20     description VARCHAR(40),
21     lng REAL,
22     lat REAL
23 );
24 -- CREATE photograph TABLE
25 CREATE TABLE IF NOT EXISTS photograph
26 (
27     photoid INT,
28     locationid INT
29 );
30
```

12. Prompt 3 - Alter Tables

```
-- Prompt 3 - Alter Tables
ALTER TABLE location MODIFY type INT NOT NULL;
ALTER TABLE location MODIFY description VARCHAR(40) NOT NULL;
ALTER TABLE location MODIFY lng REAL NOT NULL;
ALTER TABLE location MODIFY lat REAL NOT NULL;
ALTER TABLE users MODIFY name VARCHAR(40) NOT NULL;
ALTER TABLE users MODIFY userid VARCHAR(40) NOT NULL;
ALTER TABLE users MODIFY password VARCHAR(40) NOT NULL;
ALTER TABLE photograph MODIFY photoid INT NOT NULL;
ALTER TABLE photograph MODIFY locationid INT NOT NULL;
```

13. Prompt 4 - Create Index

```
42 -- Prompt 4 - Create Index
43 • CREATE UNIQUE INDEX id ON users (userid);
44 • CREATE UNIQUE INDEX photo_id ON users (photoid);
45
```

14. Prompt 5 - Enter Data

```
47 • INSERT INTO users (userid, name, username, address, city, state, zip, password)
48 VALUES
49 (1, "Bonnie Buntcake", "bbunt", "6709 Wonder Street", "Wonderbread", "OH", 46106, "eclectic"),
50 (2, "Sam Smarf", "ssmarf", "356 A Street", "Beefy", "PA", 19943, "swimming"),
51 (3, "Wendy Grog", "wgrog", "900 Star Street", "Mary", "MD", 21340, "wells"),
52 (4, "Joe Jogger", "jjogger", "183713 N North Street", "Norther", "WV", 51423, "tarts");
53
54 • SELECT * FROM users;
```

[illegible]

15. Prompt 6 - Count Rows

```
56 -- Prompt 6 - Count Rows
57 • SELECT count(*)
58 FROM users;
```

<

Result Grid | Filter Rows:

	count(*)
▶	4

16. Prompt 7 - Add Column

```
62 • ALTER TABLE photograph ADD COLUMN userid INT NOT NULL AFTER locationid;
```

<

Output

Action Output ▾

	#	Time	Action	
✓	23	00:01:29	ALTER TABLE location MODIFY lat REAL NOT NULL	0
✓	24	00:02:24	ALTER TABLE photograph MODIFY locationid INT NOT NULL	0
✓	25	00:20:04	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (1,"Bonnie Burtcake","bbunt", "6709 Wonder Street","Wo...	4
✓	26	00:22:16	SELECT * FROM users LIMIT 0, 1000	4
✓	27	00:26:01	SELECT count(*) FROM users LIMIT 0, 1000	1
✓	28	00:33:20	ALTER TABLE photograph ADD COLUMN userid INT NOT NULL AFTER locationid	0

–verify

```
62 • ALTER TABLE photograph ADD
63 • SELECT * FROM photograph;
```


<

Result Grid | Filter Rows:

	photoid	locationid	userid
--	---------	------------	--------

17. Prompt 8 - Issue with New Column

This statement will work in the program, however due to the constraints that have been established the primary key will take precedence and all that will be entered will register as the userid. So the photo will be taken by a user in the user FROM the user TABLE. The Constraint must be assigned to establish a connect between the user and the photograph table using a FOREIGN Key constraint in the users TABLE. The statement was originally an INTEGER then changed to a VARCHAR(40) now back to a VARCHAR(40) and the the






```
ALTER TABLE photograph
ADD FOREIGN KEY (userid)
REFERENCES users(userid);
```

-- Prompt 8 - Issue with New Column

```
ALTER TABLE photograph MODIFY userid VARCHAR(40) NOT NULL;
ALTER TABLE photograph
ADD FOREIGN KEY (userid)
REFERENCES users(userid);
```


18. Prompt 9 - Location and Photograph Table Updates

```
71 -- Prompt 9 - Location and Photograph Table Updates
72 • INSERT INTO location (type, description, lng, lat)
73 VALUES
74 (1, "Independence Hall", 794.35, 651.43),
75 (2, "6709 Wonder Street", 323.41, 412.22),
76 (1, "Sunrise", 221.45, 132.43),
77 (2, "356 A Street", 123.32, 222.43),
78 (1, "Mountains", 34.12, 87.99),
79 (2, "900 Star Street", 1071.9, 206.45),
80 (1, "Moonrise", 816.2, 111.2),
81 (2, "183714 N North Street", 76.11, 11.176);
82
83 • SELECT *
84 FROM location;
85
```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:    Export/Import					
itemid	type	description	lng	lat	
5	1	Mountains	34.12	87.99	
6	2	900 Star Street	1071.9	206.45	
7	1	Moonrise	816.2	111.2	
8	2	183714 N North Street	76.11	11.176	
NULL	NULL	NULL	NULL	NULL	

location 6 ×		
Output		
Action Output		
#	Time	Action
39	01:06:36	ALTER TABLE photograph ADD FOREIGN KEY (userid) REFERENCE
40	01:27:09	INSERT INTO locations (itemid,type,description,lng,lat) VALUES (1,"Ind
41	01:27:54	INSERT INTO location (itemid,type,description,lng,lat) VALUES (1,"Inde
42	01:30:28	INSERT INTO location (type,description,lng,lat) VALUES (1,"Independence

```

83 • SELECT *
84     FROM location;
85
86 • INSERT INTO photograph(photoid, locationid, userid)
87     VALUES
88     (1, 11, 1),
89     (2, 9, 1),
90     (3, 3, 3),
91     (4, 8, 4);
92
93 • SELECT *
94     FROM photograph;
95

```

Result Grid			
Filter Rows:			
Export:			
Wrap Cell Content:			
	photoid	locationid	userid
▶	1	11	1
	2	9	1
	3	3	3
	4	8	4

19. Prompt 10 - Users

```

95
96     -- Prompt 10 - Users
97 • SELECT name FROM users;
98

```

Result Grid	
Filter Rows:	
name	
▶	Bonnie Buntcake
	Sam Smarf
	Wendy Grog
	Joe Jogger

20. Prompt 11 - Who's Taking Pictures?

```
96      -- Prompt 10 - Users
97 •    SELECT DISTINCT name FROM users;
98
99      -- Prompt 11 - Who's Taking Pictures?
100 •   SELECT DISTINCT name
101      FROM users AS u, photograph AS p
102      WHERE u.userid IN(p.userid);
```

Result Grid		Filter Rows:	Export:
	name		
▶	Bonnie Buntcake		
	Wendy Grog		
	Joe Jogger		

21. Prompt 12 - Unique Names

```
103
104      -- Prompt 12 - Unique Names
105 •   SELECT DISTINCT name
106      FROM users AS u, photograph AS p
107      WHERE u.userid IN(p.userid);
```

Result Grid		Filter Rows:	Export:	Wr
	name			
▶	Bonnie Buntcake			
	Wendy Grog			
	Joe Jogger			

Bibliography

Database Management - Assignment: Database System. (2018, April 23). Retrieved from <https://study.com/academy/lesson/database-management-assignment-database-system.html>.

Github: https://github.com/mensahTribeWeb/DB_Fundamentals_Com_Sci_303.git