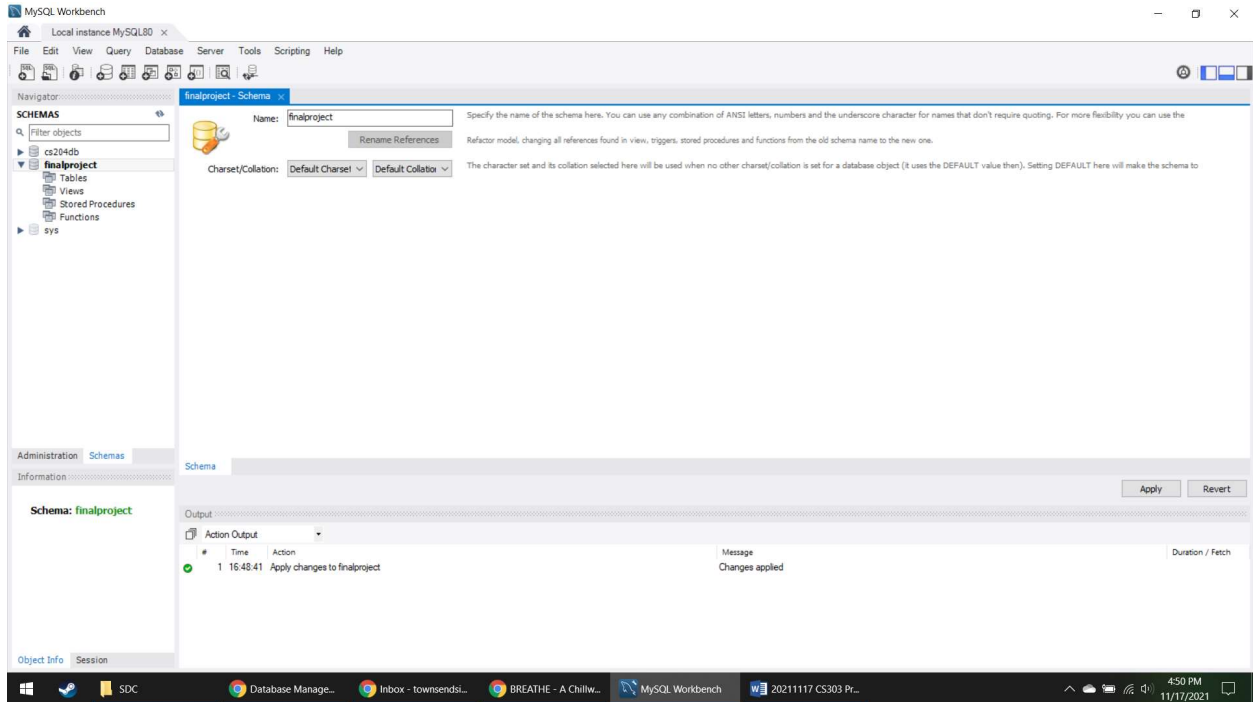


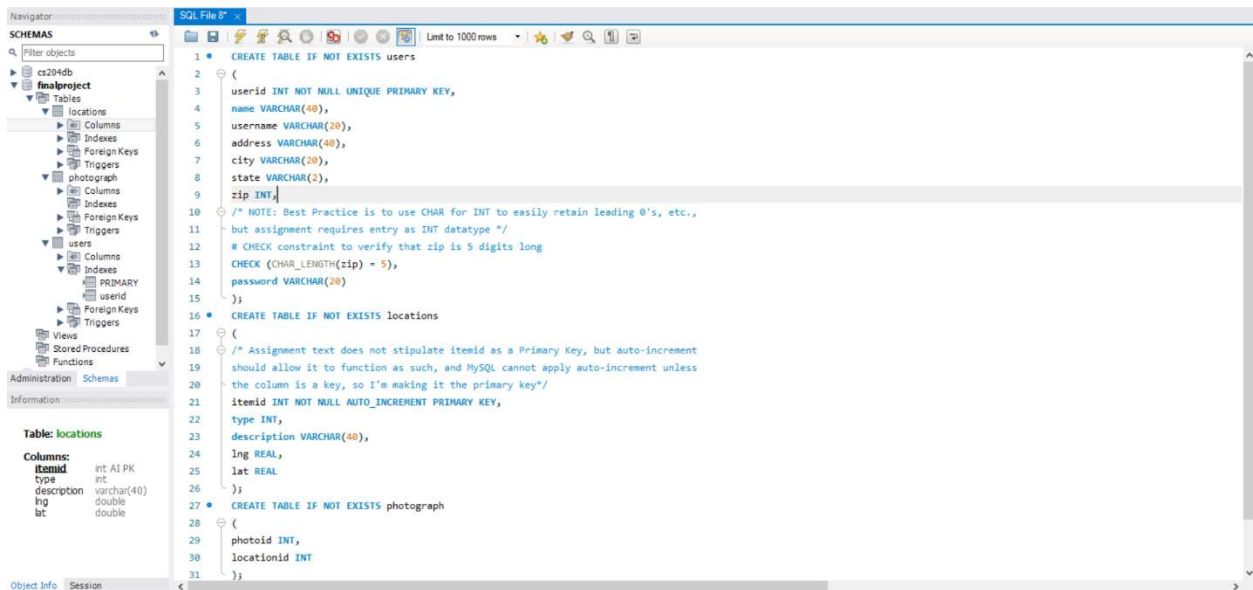
CompSci 303 Project

NOTE: All SQL queries are provided in their entirety in the screenshots for each project prompt, inside of the required screenshots demonstrating the outcome of the query.

Prompt 1: Create the Schema/Database



Prompt 2: Create Tables



Prompt 3: Alter Tables

The screenshot shows the SQL Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'finalproject' expanded, showing tables 'locations', 'users', and 'photograph'. The main pane shows a batch of 9 ALTER TABLE statements. The bottom pane shows the 'Output' window with the 'Action Output' tab selected, displaying the execution results of the statements.

Table: locations

Column	DataType	Nullable	PK
locationid	int	NOT NULL	PK
type	int	NOT NULL	
description	varchar(40)	NOT NULL	
lng	double	NOT NULL	
lat	double	NOT NULL	

Batch of SQL Statements:

- ALTER TABLE locations MODIFY type INT NOT NULL;
- ALTER TABLE locations MODIFY description VARCHAR(40) NOT NULL;
- ALTER TABLE locations MODIFY lng REAL NOT NULL;
- ALTER TABLE locations MODIFY lat REAL NOT NULL;
- ALTER TABLE users MODIFY name VARCHAR(40) NOT NULL;
- ALTER TABLE users MODIFY username VARCHAR(20) NOT NULL;
- ALTER TABLE users MODIFY password VARCHAR(20) NOT NULL;
- ALTER TABLE photograph MODIFY photoid INT NOT NULL;
- ALTER TABLE photograph MODIFY locationid INT NOT NULL;

Action Output:

#	Time	Action	Message	Duration / Fetch
11	17:54:20	ALTER TABLE locations MODIFY lat REAL NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
12	17:54:20	ALTER TABLE users MODIFY name VARCHAR(40) NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
13	17:54:20	ALTER TABLE users MODIFY username VARCHAR(20) NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
14	17:54:20	ALTER TABLE users MODIFY password VARCHAR(20) NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
15	17:54:20	ALTER TABLE photograph MODIFY photoid INT NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
16	17:54:20	ALTER TABLE photograph MODIFY locationid INT NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec

Prompt 4: Create Index

The screenshot shows the SQL Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'finalproject' expanded, showing tables 'locations', 'users', and 'photograph'. The 'photograph' table is selected, and its 'Indexes' folder is expanded, showing a new index 'photograph_pid'. The main pane shows a single SQL statement: 'CREATE UNIQUE INDEX photograph_pid ON photograph (photoid)'. The bottom pane shows the 'Output' window with the 'Action Output' tab selected, displaying the execution results of the statement.

Index: photograph_pid

Definition:

Type	Unique	Visible	Columns
B TREE	Yes	Yes	photoid

Batch of SQL Statements:

- CREATE UNIQUE INDEX photograph_pid ON photograph (photoid)

Action Output:

#	Time	Action	Message	Duration / Fetch
12	17:54:20	ALTER TABLE users MODIFY name VARCHAR(40) NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
13	17:54:20	ALTER TABLE users MODIFY username VARCHAR(20) NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
14	17:54:20	ALTER TABLE users MODIFY password VARCHAR(20) NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
15	17:54:20	ALTER TABLE photograph MODIFY photoid INT NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
16	17:54:20	ALTER TABLE photograph MODIFY locationid INT NOT NULL	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
17	17:55:55	CREATE UNIQUE INDEX photograph_pid ON photograph (photoid)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec

Prompt 5: Enter Data

Navigator

SCHEMAS

Filter objects

cs204db

finalproject

locations

Columns

Indexes

Foreign Keys

Triggers

photograph

Columns

photoid

locationid

Indexes

photograph_pid

Foreign Keys

Triggers

users

Columns

userid

name

username

address

city

Administration

Schemas

Information

Index: photograph_pid

Definition:

Type: B-TREE

Unique: Yes

Visible: Yes

Columns: photoid

Object Info

Session

SQL File 8" SQL File 9" SQL File 10" SQL File 11" SQL File 12"

Limit to 1000 rows

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

Result Grid

userid	name	username	address	city	state	zip	password
1	Bonnie Buntcake	bbunt	6709 Wonder Street	Wonderbread	OH	46106	edlectc
2	Sam Smarf	ssmarf	356 A Street	Beefy	PA	19943	swimming
3	Wendy Grog	wgrog	900 Star Street	Mary	MD	21340	wells
4	Joe Jogger	jjogger	183713 N North Street	Norther	WV	51423	tarts

users 2 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
23	18:07:37	DELETE FROM users WHERE userid > 0	4 row(s) affected	0.000 sec
24	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (1, "Bonnie Bunt...	1 row(s) affected	0.000 sec
25	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (2, "Sam Smarf"...	1 row(s) affected	0.000 sec
26	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (3, "Wendy Grog"...	1 row(s) affected	0.000 sec
27	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (4, "Joe Jogger"...	1 row(s) affected	0.000 sec
28	18:07:43	SELECT * FROM users LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Prompt 6: Count Rows

Navigator

SCHEMAS

Filter objects

cs204db

finalproject

locations

Columns

Indexes

Foreign Keys

Triggers

photograph

Columns

photoid

locationid

Indexes

photograph_pid

Foreign Keys

Triggers

users

Columns

userid

name

username

address

city

Administration

Schemas

Information

Index: photograph_pid

Definition:

Type: B-TREE

Unique: Yes

Visible: Yes

Columns: photoid

Object Info

Session

SQL File 8" SQL File 9" SQL File 10" SQL File 11" SQL File 12"

Limit to 1000 rows

```
1 * SELECT COUNT(*)
2 FROM users;
```

Result Grid

COUNT(*)
4

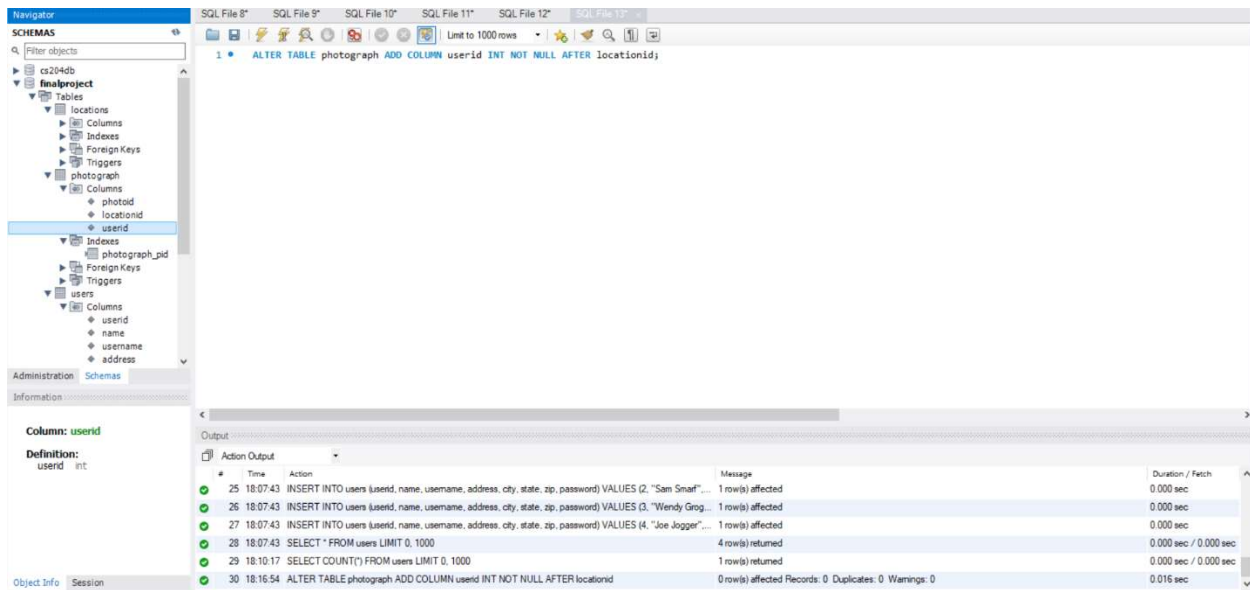
Result 1 x

Output

Action Output

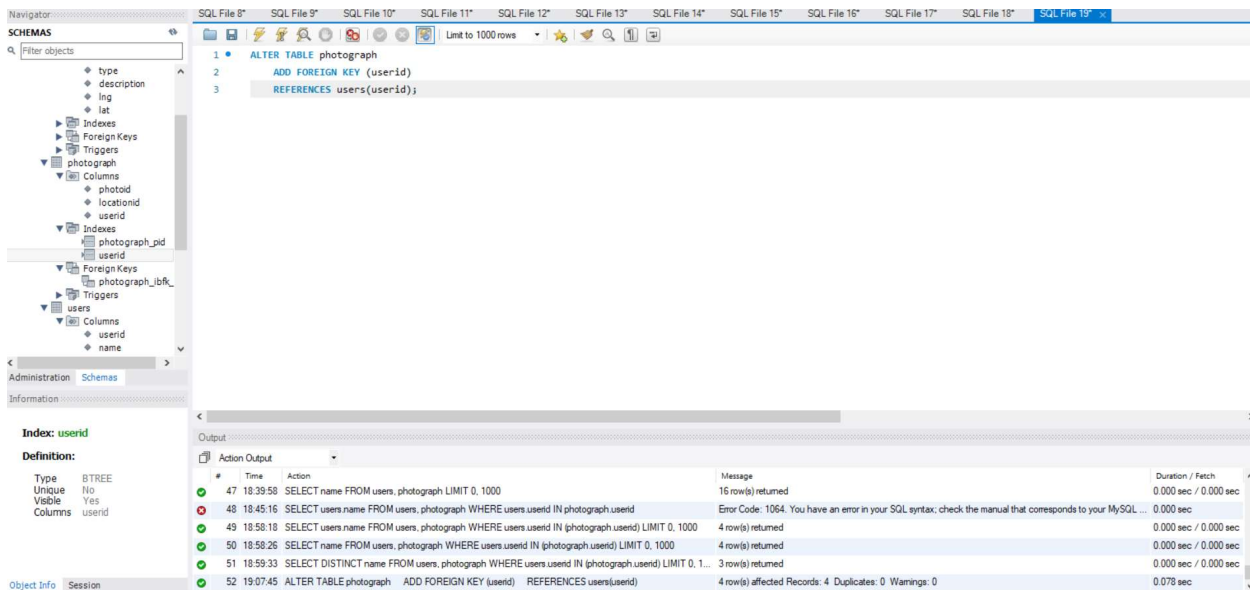
#	Time	Action	Message	Duration / Fetch
24	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (1, "Bonnie Bunt...	1 row(s) affected	0.000 sec
25	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (2, "Sam Smarf"...	1 row(s) affected	0.000 sec
26	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (3, "Wendy Grog"...	1 row(s) affected	0.000 sec
27	18:07:43	INSERT INTO users (userid, name, username, address, city, state, zip, password) VALUES (4, "Joe Jogger"...	1 row(s) affected	0.000 sec
28	18:07:43	SELECT * FROM users LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
29	18:10:17	SELECT COUNT(*) FROM users LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Prompt 7: Add Column



Prompt 8: Issue with New Column

The column will technically work just fine, but 'userid' represents a relationship which we did not incorporate into the database. When we add a picture to the 'photograph' table, that picture was taken by a user from the 'users' table, who is uniquely identified by their 'userid'. 'userid' in 'users' is a primary key, so its usage in 'photographs' makes it a foreign key, depicting the relationship between photo and user. To fix this, we would use ALTER TABLE to provide a FOREIGN KEY:



Prompt 9: Location and Photograph Table Updates

SQL File 8

```

1 • INSERT INTO locations (type, description, lng, lat) VALUES (1, "Independence Hall", 794.35, 651.43);
2 • INSERT INTO locations (type, description, lng, lat) VALUES (2, "6709 Wonder Street", 323.41, 412.22);
3 • INSERT INTO locations (type, description, lng, lat) VALUES (1, "Sunrise", 221.45, 132.43);
4 • INSERT INTO locations (type, description, lng, lat) VALUES (2, "356 A Street", 123.32, 222.43);
5 • INSERT INTO locations (type, description, lng, lat) VALUES (1, "Mountains", 34.12, 87.99);
6 • INSERT INTO locations (type, description, lng, lat) VALUES (2, "900 Star Street", 1071.9, 206.45);
7 • INSERT INTO locations (type, description, lng, lat) VALUES (1, "Moonrise", 816.2, 111.2);
8 • INSERT INTO locations (type, description, lng, lat) VALUES (2, "183714 N North Street", 176.11, 11.176);
9 • SELECT *
10 FROM locations;

```

itemid	type	description	lng	lat
1	1	Independence Hall	794.35	651.43
2	2	6709 Wonder Street	323.41	412.22
3	1	Sunrise	221.45	132.43
4	2	356 A Street	123.32	222.43
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	176.11	11.176

Column: userid
Definition: userid int

^^ locations updates ^^

SQL File 8*

```

1 • # Assignment directs me to fill in the locationid, so I used 4, 3, 2, 1
2 • INSERT INTO photograph (photoid, locationid, userid) VALUES (1, 4, 1);
3 • INSERT INTO photograph (photoid, locationid, userid) VALUES (2, 3, 1);
4 • INSERT INTO photograph (photoid, locationid, userid) VALUES (3, 2, 3);
5 • INSERT INTO photograph (photoid, locationid, userid) VALUES (4, 1, 4);
6 • SELECT *
7 FROM photograph;

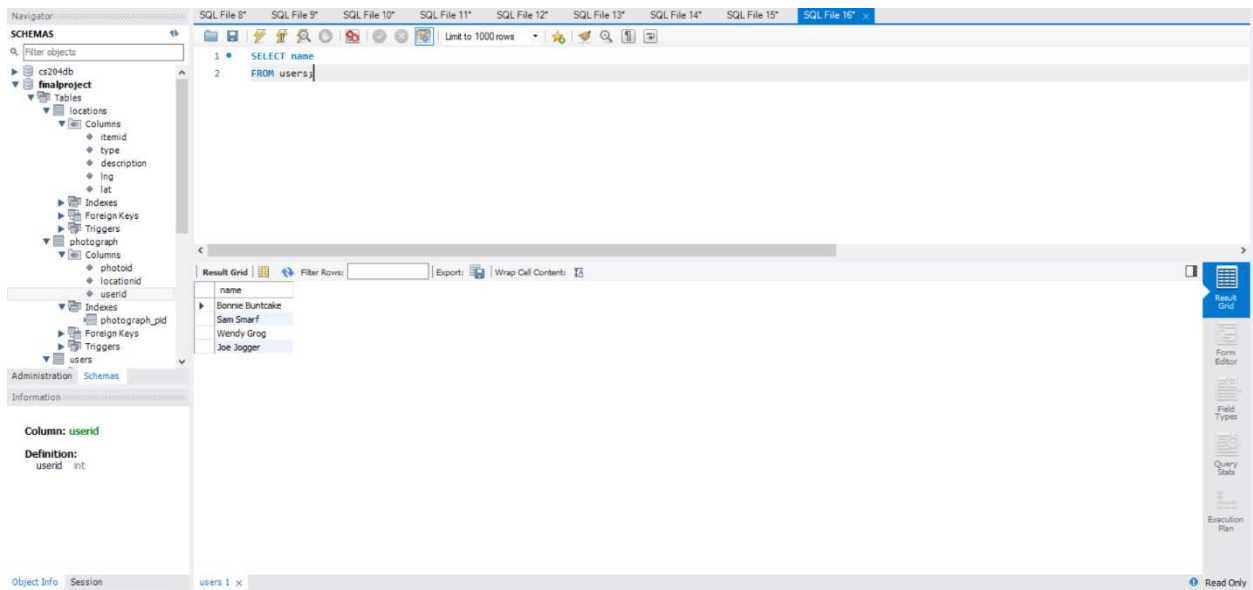
```

photoid	locationid	userid
1	4	1
2	3	1
3	2	3
4	1	4

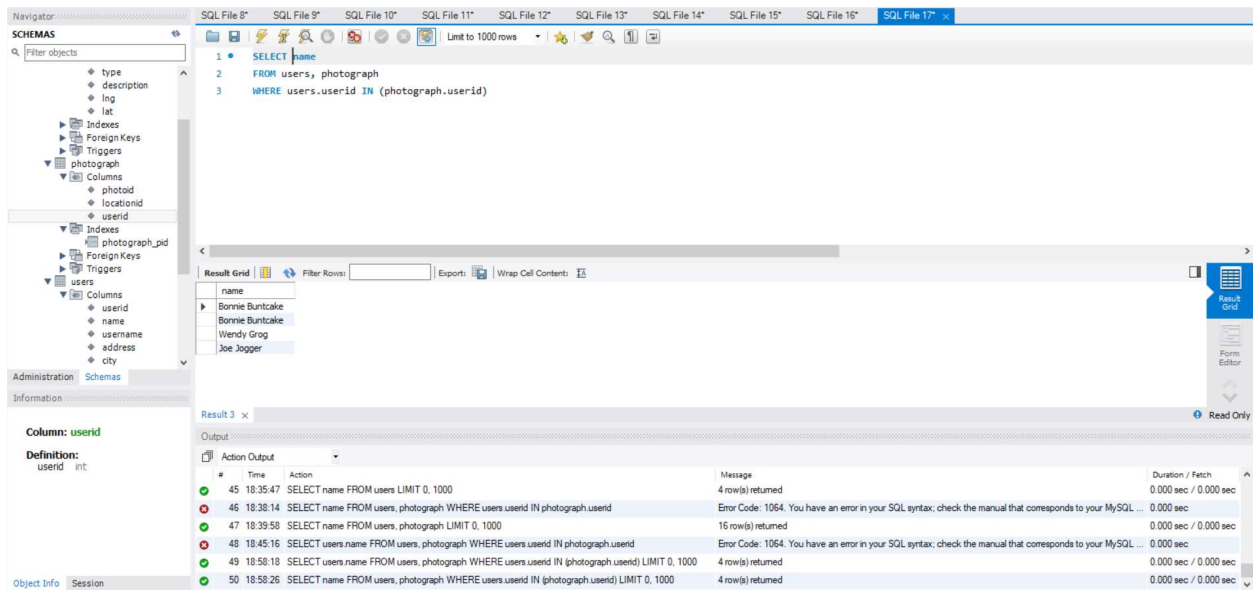
Column: userid
Definition: userid int

^^ photograph updates ^^

Prompt 10: Users



Prompt 11: Who's Taking Pictures?



Prompt 12: Unique Names

Navigator

Filter objects

- type
- description
- ing
- lat
- Indexes
- Foreign Keys
- Triggers
- photograph
 - Columns
 - photoId
 - locationId
 - userId
 - Indexes
 - photograph_pid
 - Foreign Keys
 - Triggers
- users
 - Columns
 - userId
 - name
 - username
 - address
 - city

Administration Schemas

Information

Column: **userId**

Definition:

userId int

Object Info Session

SQL File 8* SQL File 9* SQL File 10* SQL File 11* SQL File 12* SQL File 13* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 18*

1. SELECT DISTINCT name

2. FROM users, photograph

3. WHERE users.userId IN (photograph.userId)

Result Grid

name
Bonnie Buntcake
Wendy Grog
Joe Jogger

Result 1 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
46	18:38:14	SELECT name FROM users, photograph WHERE users.userId IN photograph.userId	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL ...	0.000 sec
47	18:39:58	SELECT name FROM users, photograph LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
48	18:45:16	SELECT users.name FROM users, photograph WHERE users.userId IN photograph.userId	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL ...	0.000 sec
49	18:58:18	SELECT users.name FROM users, photograph WHERE users.userId IN (photograph.userId) LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
50	18:58:26	SELECT name FROM users, photograph WHERE users.userId IN (photograph.userId) LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
51	18:59:33	SELECT DISTINCT name FROM users, photograph WHERE users.userId IN (photograph.userId) LIMIT 0, 1...	3 row(s) returned	0.000 sec / 0.000 sec

asdf