Assignment 1: Database System

Sarah Gillard

CS303: Database Management

February 28, 2024

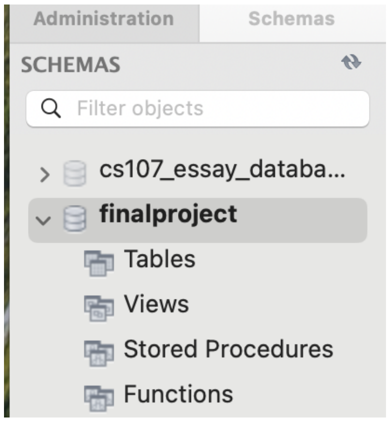
Project Overview

Part 1 of the final project for CS303 ensures that you understand and can carry out the necessary tasks in order to manage a sample database. It consists of the following steps:

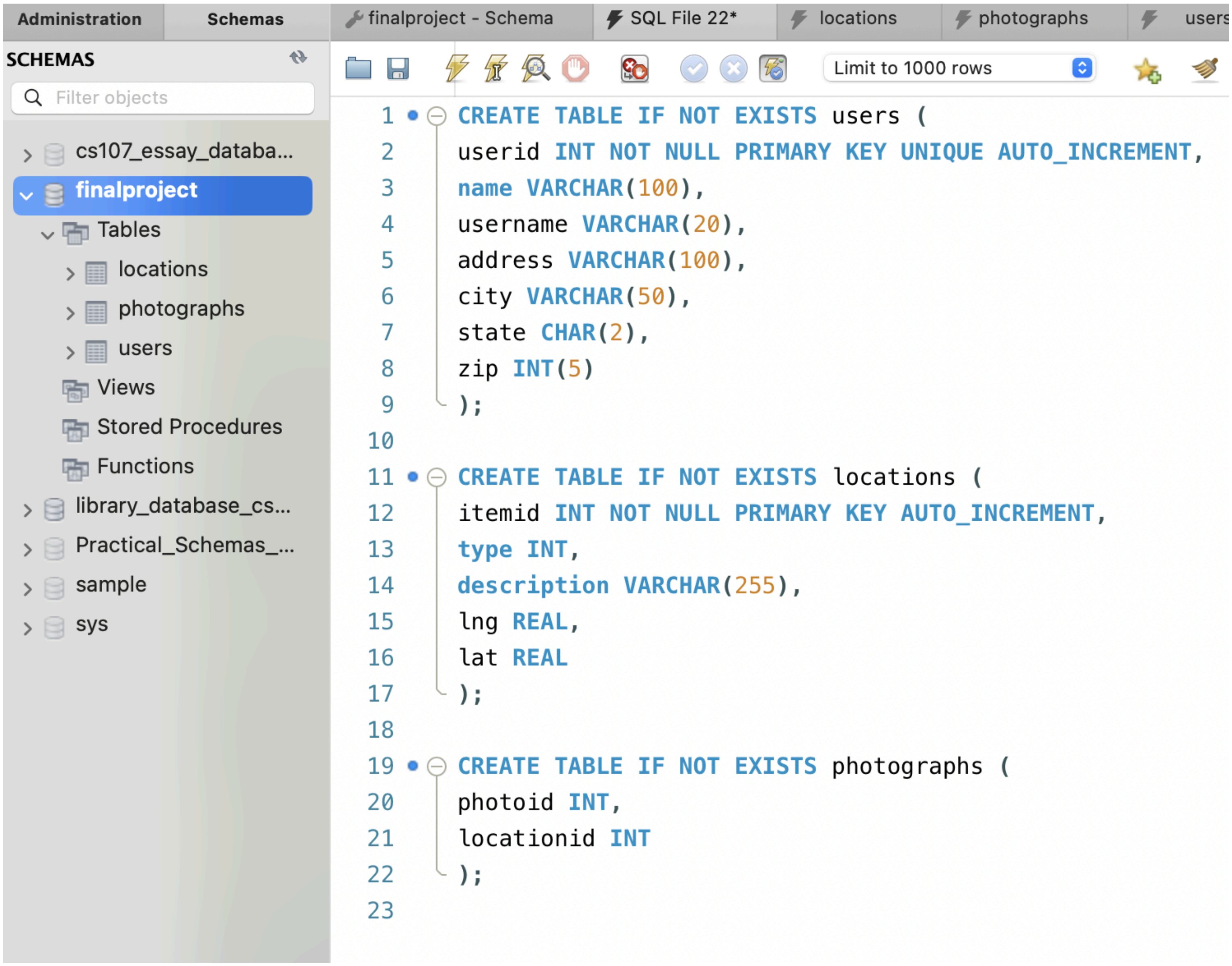
1. Selection of a database.
2. Identifying the requirements for the database based on the purpose for the database and the data to fill the database.
3. Creating a database design based on the requirements for the database: create the physical database and load data into the database.
4. Carrying out sample queries that would be performed on the actual database, then showing the results of the sample queries.

Screen captures associated with each prompt are provided in this document with labels indicating the prompt they are associated with.

Prompt 1: Create the Schema/Database

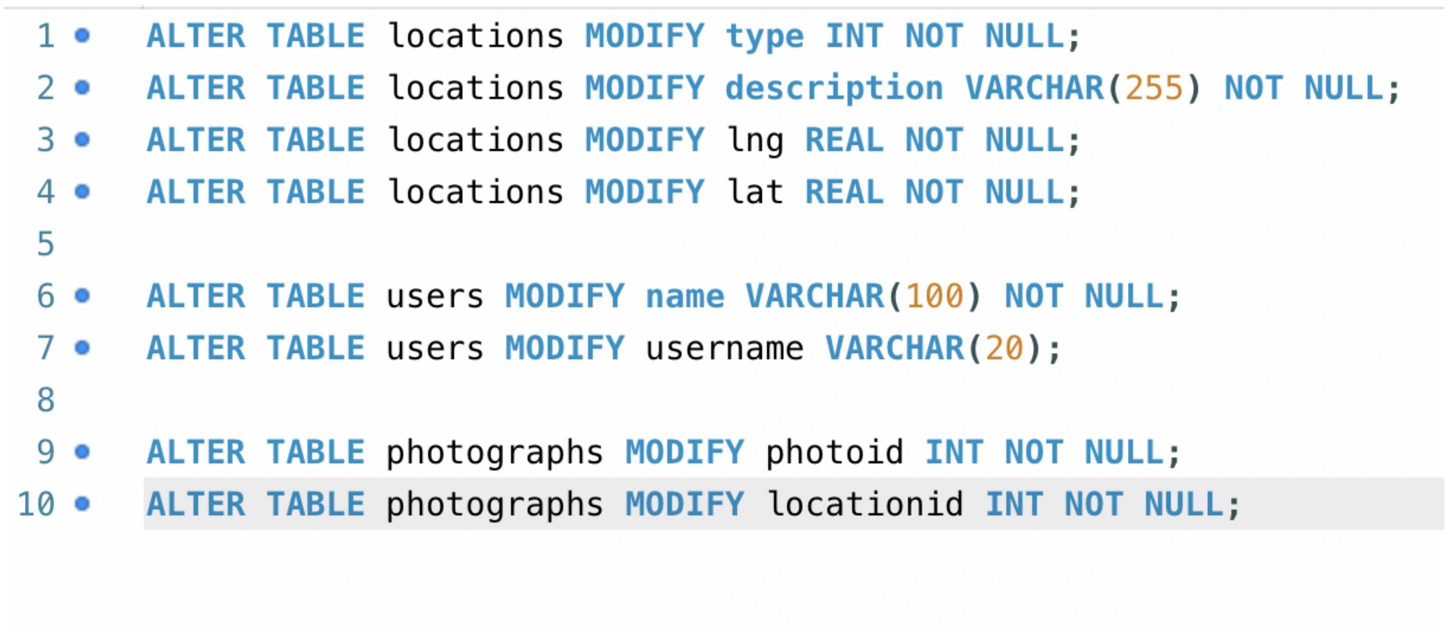


Prompt 2: Create Tables



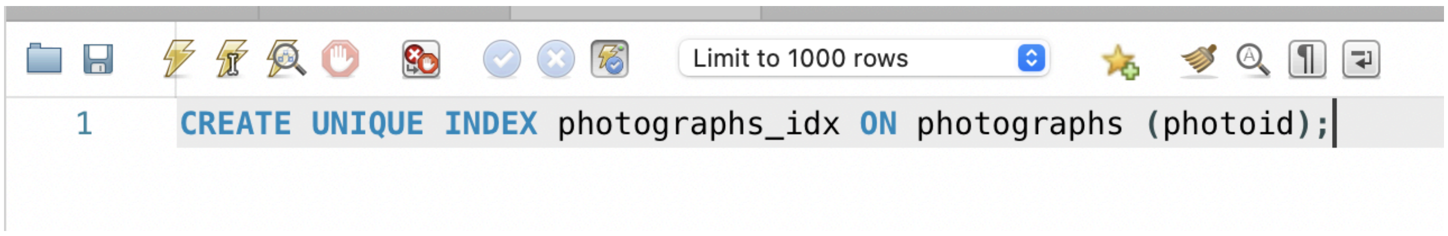
```sql
CREATE TABLE IF NOT EXISTS users (
userid INT NOT NULL PRIMARY KEY UNIQUE AUTO_INCREMENT,
name VARCHAR(100),
username VARCHAR(20),
address VARCHAR(100),
city VARCHAR(50),
state CHAR(2),
zip INT(5)
);

CREATE TABLE IF NOT EXISTS locations (
itemid INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
type INT,
description VARCHAR(255),
lng REAL,
lat REAL
);

CREATE TABLE IF NOT EXISTS photographs (
photoid INT,
locationid INT
);
```

Prompt 3: Alter Tables



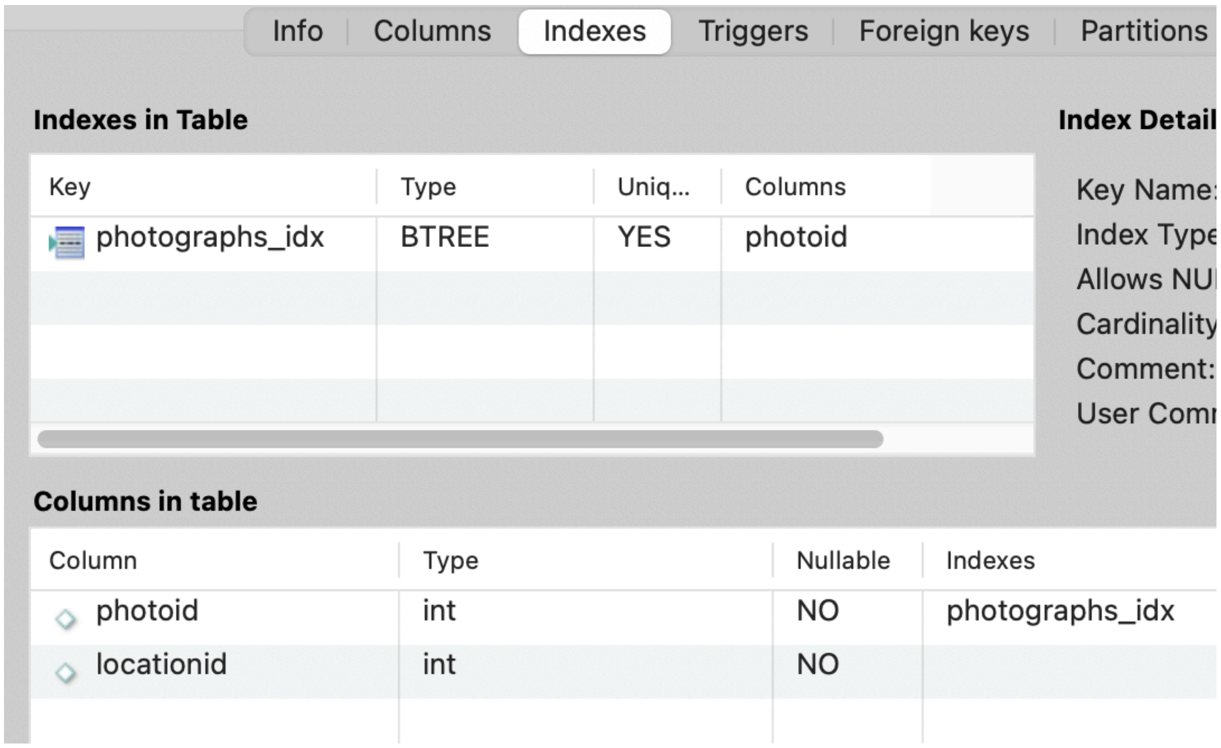
```
ALTER TABLE locations MODIFY type INT NOT NULL;
ALTER TABLE locations MODIFY description VARCHAR(255) NOT NULL;
ALTER TABLE locations MODIFY lng REAL NOT NULL;
ALTER TABLE locations MODIFY lat REAL NOT NULL;

ALTER TABLE users MODIFY name VARCHAR(100) NOT NULL;
ALTER TABLE users MODIFY username VARCHAR(20);

ALTER TABLE photographs MODIFY photoid INT NOT NULL;
ALTER TABLE photographs MODIFY locationid INT NOT NULL;
```

Prompt 4: Create Index



Limit to 1000 rows

```
CREATE UNIQUE INDEX photographs_idx ON photographs (photoid);
```



Indexes in Table

Key	Type	Uniq...	Columns
photographs_idx	BTREE	YES	photoid

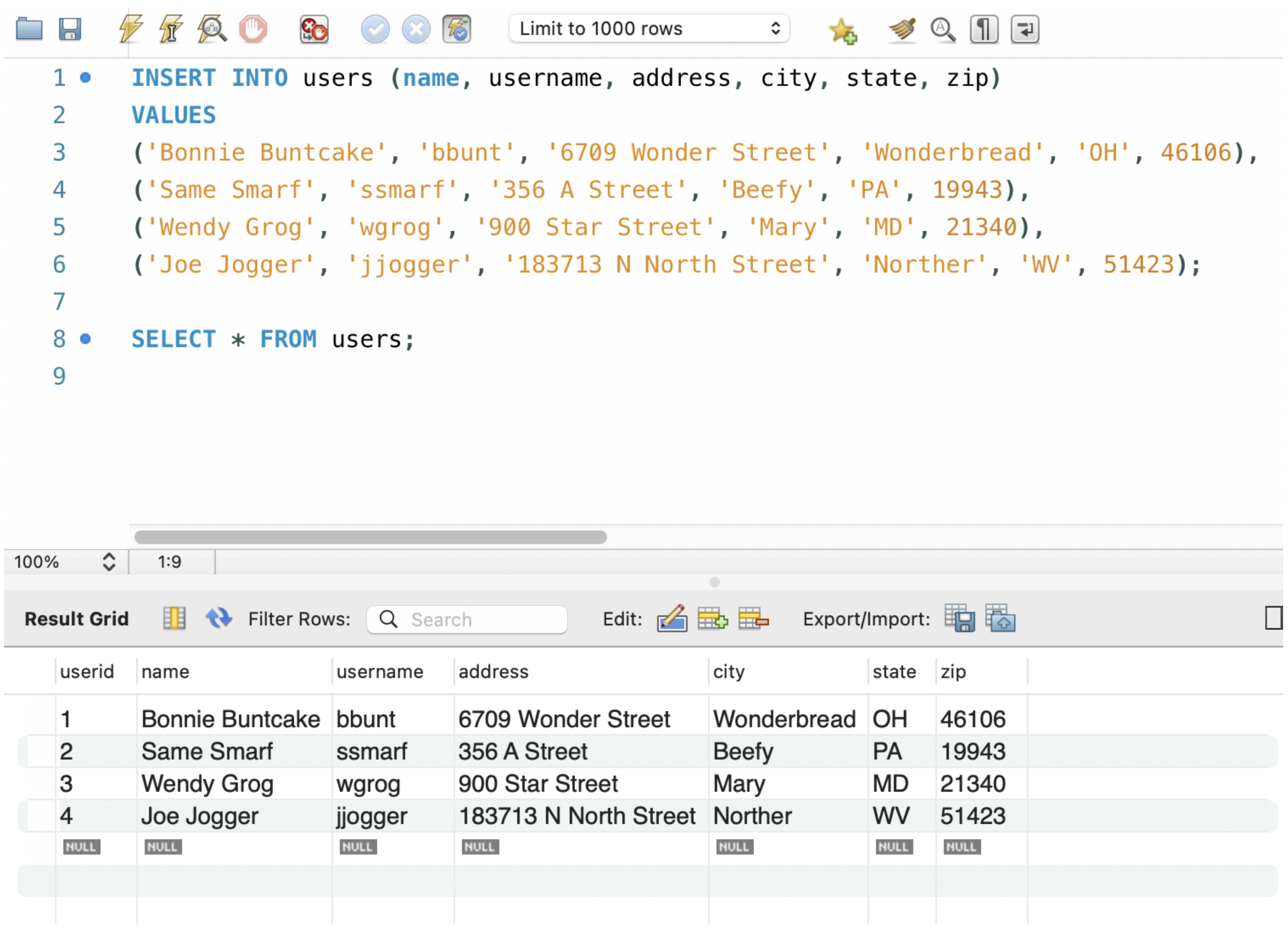
Index Detail

Key Name:
Index Type
Allows NU
Cardinality
Comment:
User Comm

Columns in table

Column	Type	Nullable	Indexes
photoid	int	NO	photographs_idx
locationid	int	NO	

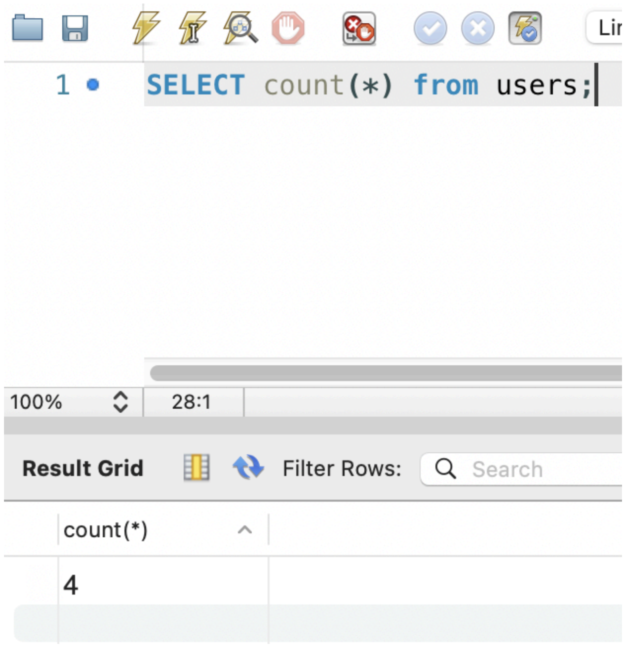
Prompt 5: Enter Data



```sql
INSERT INTO users (name, username, address, city, state, zip)
VALUES
('Bonnie Buntcake', 'bbunt', '6709 Wonder Street', 'Wonderbread', 'OH', 46106),
('Same Smarf', 'ssmarf', '356 A Street', 'Beefy', 'PA', 19943),
('Wendy Grog', 'wgrog', '900 Star Street', 'Mary', 'MD', 21340),
('Joe Jogger', 'jjogger', '183713 N North Street', 'Norther', 'WV', 51423);

SELECT * FROM users;
```

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

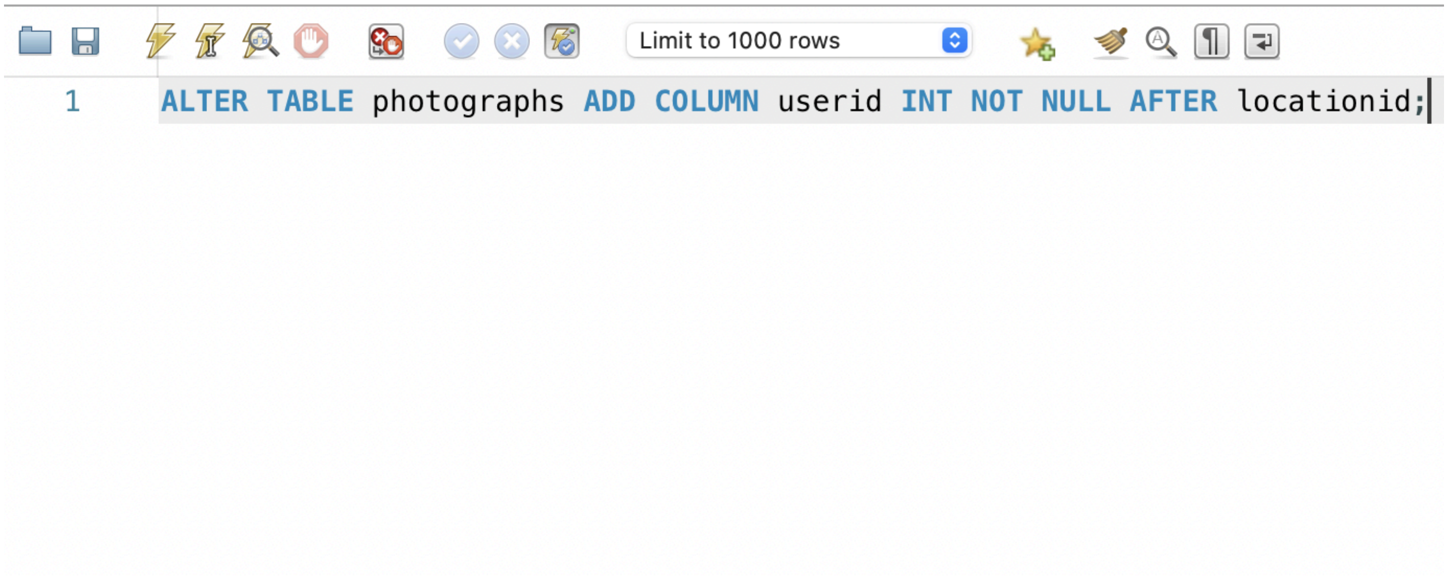
Prompt 6: Count Rows



```sql
SELECT count(*) from users;
```

count(*)
4

Prompt 7: Add Column



Limit to 1000 rows

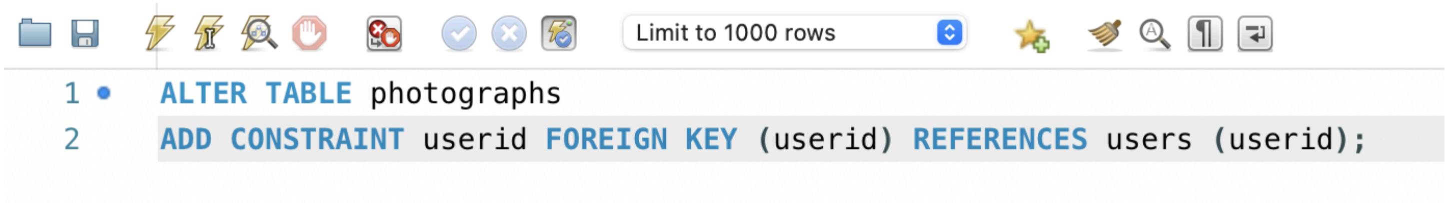
```sql
ALTER TABLE photographs ADD COLUMN userid INT NOT NULL AFTER locationid;
```

Prompt 8: Issue with New Column

To ensure data integrity we can set more constraints on the newly added “userid” column. It is already NOT NULL and this ensures that each photograph must have a linked user, preventing any potential inconsistencies due to missing or null values.

We will now make it a foreign key referencing the “userid” in the “users” table. Establishing a foreign key relationship between the “photographs” and “users” tables will help ensure data integrity and maintain consistency in the database. The use of this foreign key ensures that every entry in the “photographs” table is associated with a valid and existing user in the “users” table.

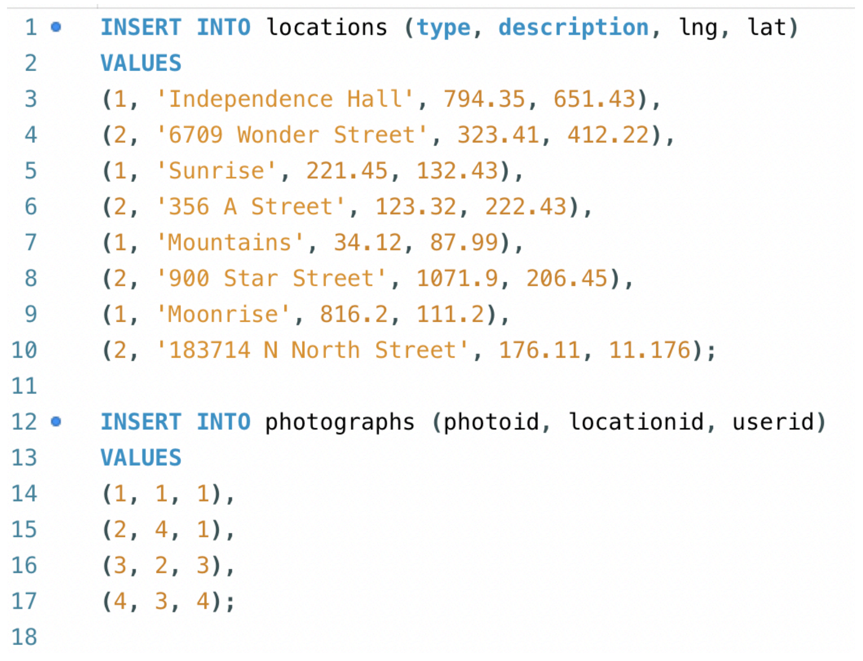
We should also ensure that the data type of the “userid” column in the “photographs” table matches the data type of the referenced “userid” column in the “users” table to avoid data type conflicts and promote uniformity.



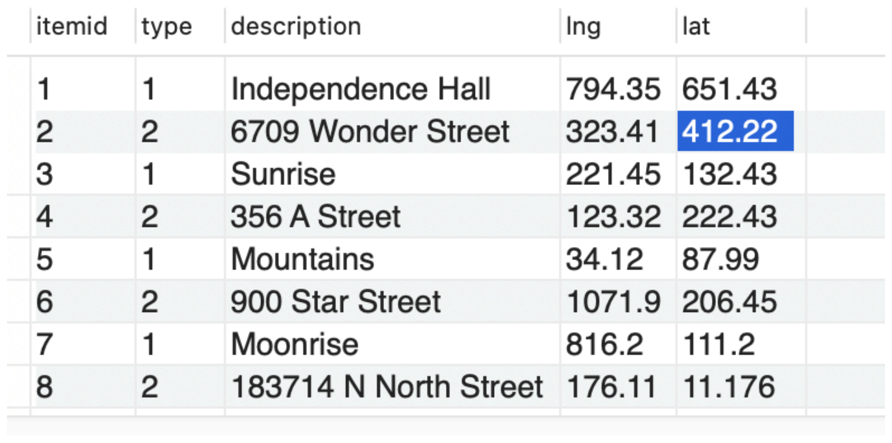
Limit to 1000 rows

```sql
ALTER TABLE photographs
ADD CONSTRAINT userid FOREIGN KEY (userid) REFERENCES users (userid);
```

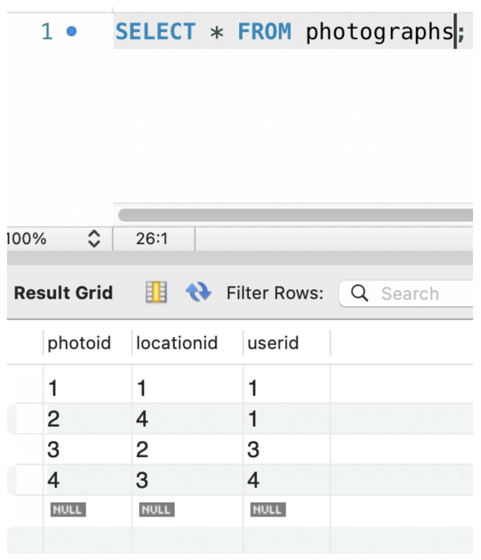
Prompt 9: Location and Photograph Table Updates



```sql
INSERT INTO locations (type, description, lng, lat)
VALUES
(1, 'Independence Hall', 794.35, 651.43),
(2, '6709 Wonder Street', 323.41, 412.22),
(1, 'Sunrise', 221.45, 132.43),
(2, '356 A Street', 123.32, 222.43),
(1, 'Mountains', 34.12, 87.99),
(2, '900 Star Street', 1071.9, 206.45),
(1, 'Moonrise', 816.2, 111.2),
(2, '183714 N North Street', 176.11, 11.176);

INSERT INTO photographs (photoid, locationid, userid)
VALUES
(1, 1, 1),
(2, 4, 1),
(3, 2, 3),
(4, 3, 4);

```



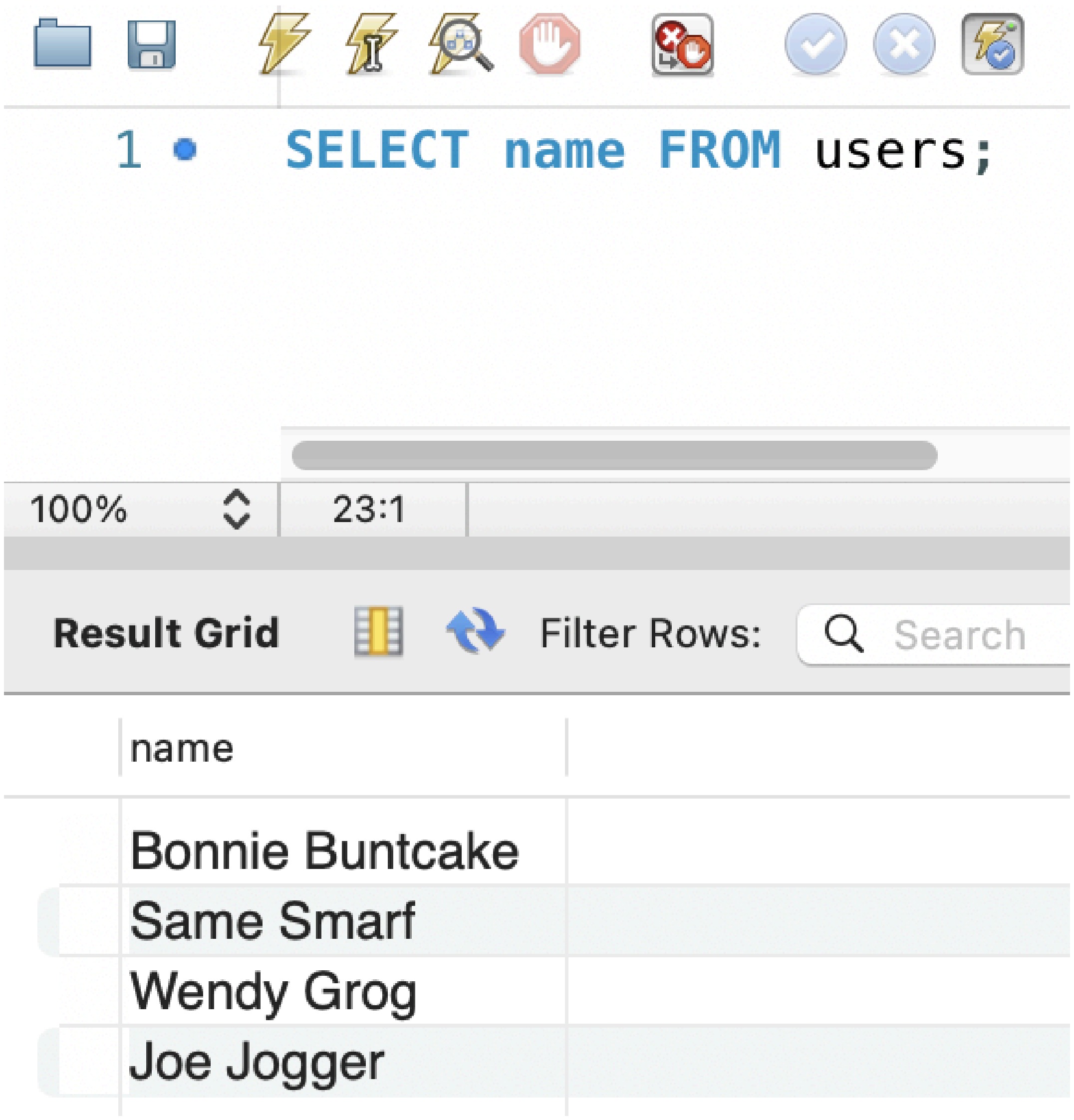
itemid	type	description	lng	lat
1	1	Independence Hall	794.35	651.43
2	2	6709 Wonder Street	323.41	412.22
3	1	Sunrise	221.45	132.43
4	2	356 A Street	123.32	222.43
5	1	Mountains	34.12	87.99
6	2	900 Star Street	1071.9	206.45
7	1	Moonrise	816.2	111.2
8	2	183714 N North Street	176.11	11.176



```sql
SELECT * FROM photographs;
```

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

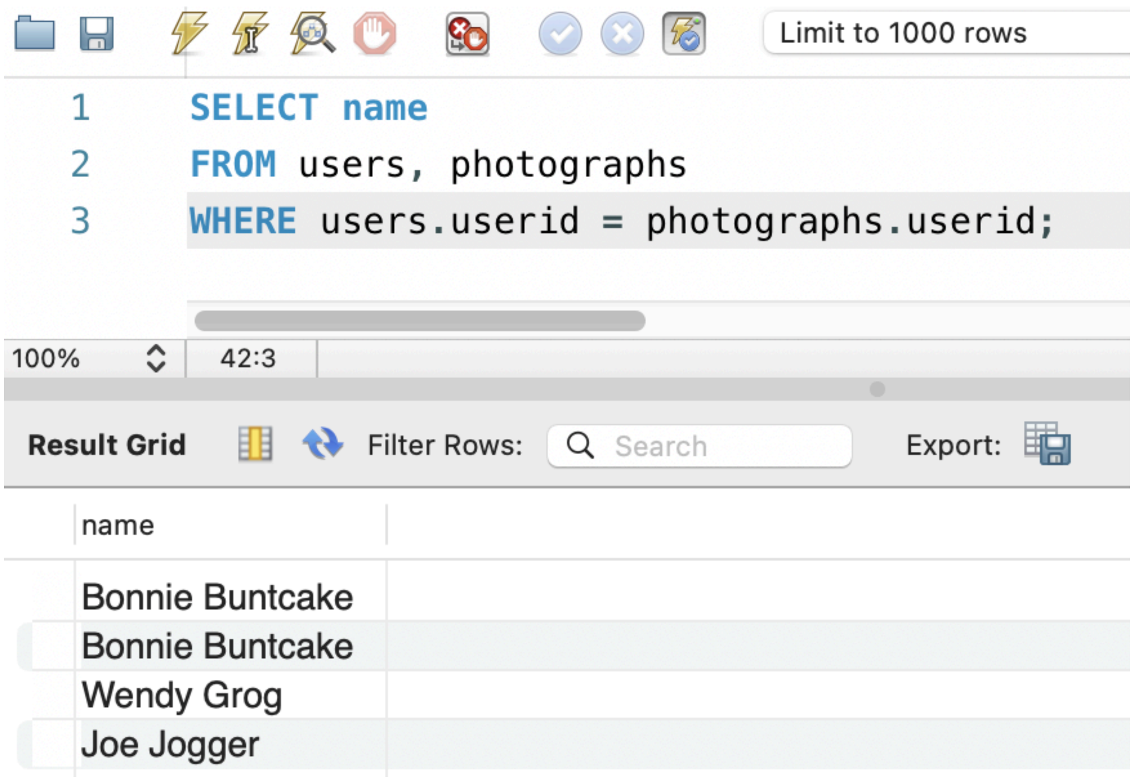
Prompt 10: Users



```
SELECT name FROM users;
```

```
name
Bonnie Buntcake
Same Smarf
Wendy Grog
Joe Jogger
```

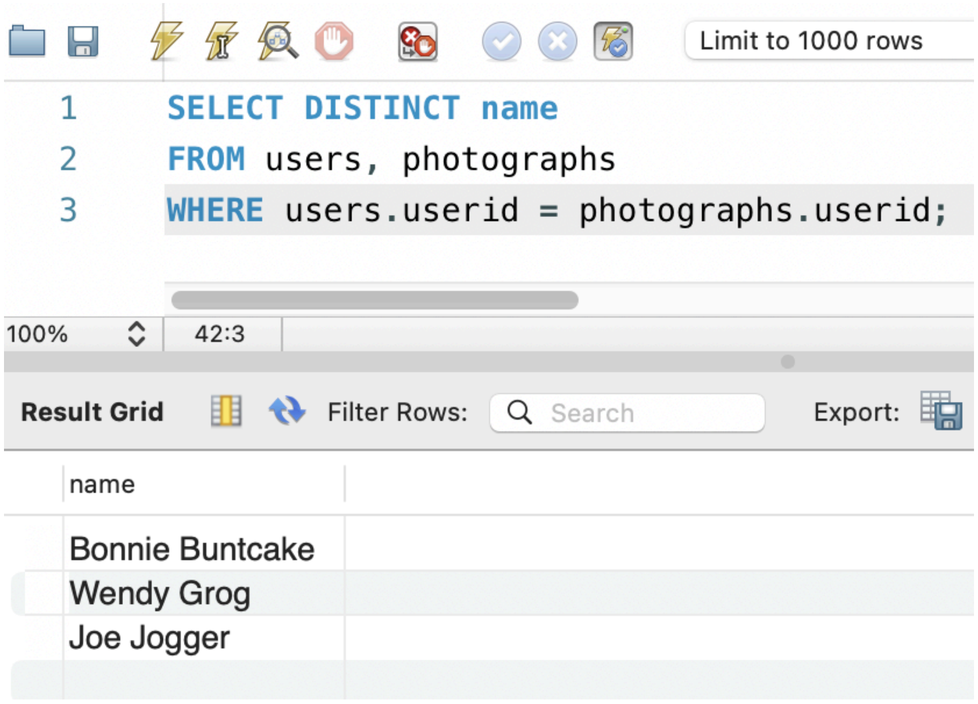
Prompt 11: Who’s Taking Pictures?



```
SELECT name
FROM users, photographs
WHERE users.userid = photographs.userid;
```

```
name
Bonnie Buntcake
Bonnie Buntcake
Wendy Grog
Joe Jogger
```

Prompt 12: Unique Names



```
SELECT DISTINCT name
FROM users, photographs
WHERE users.userid = photographs.userid;
```

```
name
Bonnie Buntcake
Wendy Grog
Joe Jogger
```