# Section 5: SQL Interactions in Each State + Output

## Structure Overview:

0. Initialize

1. Login

Event 1.1: Submit Login Form

2. Registration

Event 2.1: Submit Registration Form

3. Home (My Bookings)

On Page Load

Event 3.1: Cancel Booking as Consumer

Event 3.2: Accept Booking as Supplier

Event 3.3: Reject Booking as Supplier

Event 3.4: Cancel Booking as Supplier

4. Search

On Page Load

Event 4.1: Submit Search

5. Property Details

On Page Load

Event 5.1: Submit Review

Event 5.2: Submit Reply

Event 5.3: Delete Review

Event 5.4: Delete Reply

Event 5.5: Admin Delete Property

6. Booking Request Popup

Event 6.1: Submit Booking Request

7. My Properties

On Page Load

8. Add Property Popup

Event 8.1: Submit form

9. Edit Property Popup

Event 9.1: Submit form

10. Delete Property Confirmation Popup

Event 10.1: Confirm deletion

11. User Settings

On Page Load

Event 11.1: Submit form

Event 11.2: Cancel Membership

12. Administrator

On Page Load

Event 12.1: Click “delete member”

## 0. Initialize

We decided to load our static tables (booking\_status\_types, city\_districts, degree\_types, rental\_property\_types) into PHP to reduce complex queries. The first time data from one of these static tables, we load the table in and set a variable. Future requests for this data will see that the variable is set and not call the database again.

Action: Get data from static tables.

SQL Queries:

SELECT id, district\_name FROM city\_districts

SELECT \* FROM `rental\_property\_types`

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS

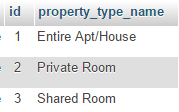
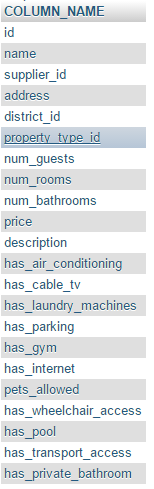
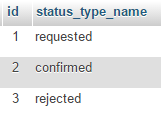
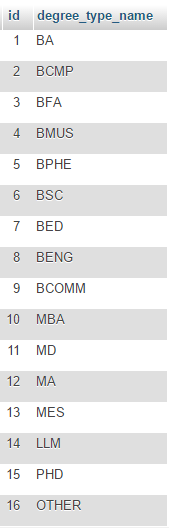
WHERE TABLE\_SCHEMA = "cisc332project" AND TABLE\_NAME = "rental\_properties"

SELECT id, status\_type\_name FROM booking\_status\_types

SELECT id, degree\_type\_name FROM degree\_types

SQL Output:

(rest cut off)

## 1. Login

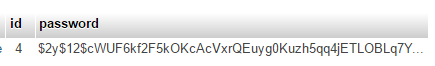
### Event 1.1: Submit Login Form

Action: Check if login valid. We will use a PHP function to hash the password string in order to keep it secure.

SQL Query:

SELECT id, password FROM users WHERE email = "davidthompson@teleworm.us"

Sample Output:



## 2. Registration

### Event 2.1: Submit Registration Form

Action: Check that email address is unique in database. The SELECT 1 will return 1 if the email exists in the database and nothing if it doesn’t. It requires less processing from the databases and transfers only the necessary information.

SQL Query:

SELECT 1 FROM users WHERE email = "12sak2@queensu.ca"

SQL Output:



Action: If email is unique, add user to database.

SQL Query:

INSERT INTO users (id, is\_admin, first\_name, last\_name, email, password, phone\_number, grad\_year, faculty\_id, degree\_type\_id, gender) VALUES

(NULL, “0”, "Shannon", "Klett", "12sak2@queensu.ca", "$2y$12$wwCrm0V8hc1bEvzGbfVAKugSlrRwxNW03hnEIo0HE/SNtFsjMp1cC", "+1 902-111-1101", "2016", "1", "2", "Female")

Sample Output:



## 3. Home (My Bookings)

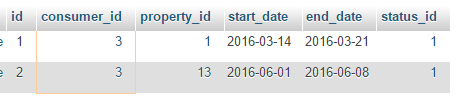
### On Page Load

Action: Get all bookings requested by user.

SQL Query:

SELECT \* FROM bookings WHERE consumer\_id = 3

Sample Output:



Action: Get all bookings on a user’s accommodations.

### Event 3.1: Cancel Booking as Consumer

Action: Get supplier email in order to notify them, and information about booking so email is informative.

SQL Query:

SELECT supplier.first\_name as supplier\_first, supplier.last\_name as supplier\_last, supplier.email as supplier\_email, consumer.first\_name as consumer\_first, consumer.last\_name as consumer\_last, rental\_properties.name as property\_name, start\_date, end\_date, status\_type\_name

FROM rental\_properties JOIN bookings ON property\_id = rental\_properties.id

JOIN users supplier ON supplier\_id = supplier.id

JOIN users consumer ON consumer\_id = consumer.id

JOIN booking\_status\_types ON status\_id = booking\_status\_types.id

WHERE bookings.id = 4

Sample Output:



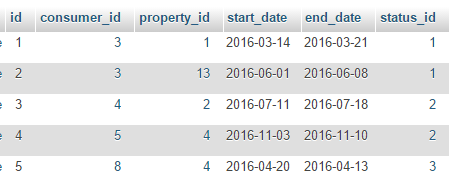
Action: Remove booking from database

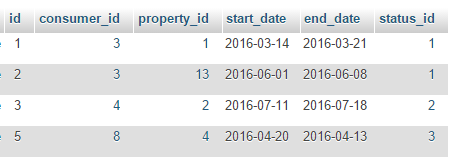
SQL Query:

DELETE FROM bookings

WHERE bookings.id = 4

Sample Output:

Before - 

After - 

### Event 3.2: Accept Booking as Supplier

Action: Status of booking updated in database

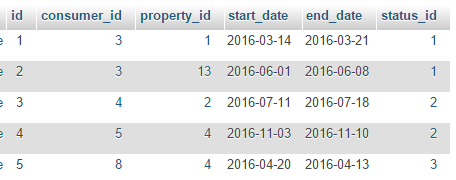
SQL Query:

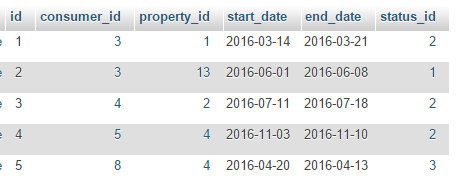
UPDATE bookings

SET status\_id = 2

WHERE id = 1 AND status\_id = 1

Sample Output:

Before -

After - 

Action: Get information to notify consumer

SQL Query:

SELECT first\_name, last\_name, email, rental\_properties.name as property\_name, start\_date, end\_date

FROM users JOIN bookings ON consumer\_id = users.id

JOIN rental\_properties ON property\_id = rental\_properties.id

WHERE bookings.id = 1

Sample Output:



### Event 3.3: Reject Booking as Supplier

Action: Status of booking updated in database (status must be pending)

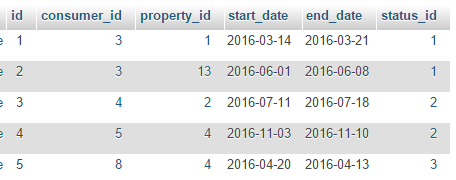
SQL Query:

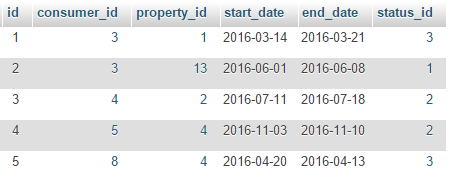
UPDATE bookings

SET status\_id = 3

WHERE id = 1 AND status\_id = 1

Sample Output:

Before -

After - 

Action: Get information to notify consumer (same SQL query as 3.2: Accept Booking as Supplier)

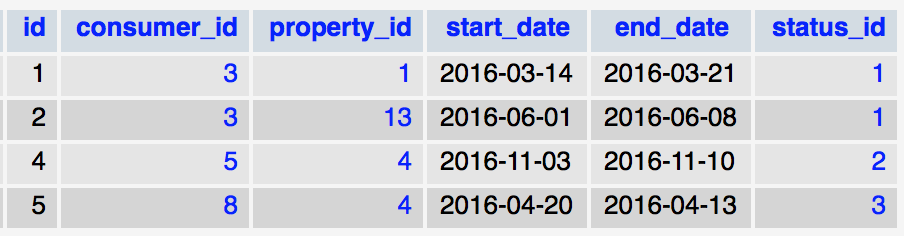
### Event 3.4: Cancel Booking as Supplier

Action: Booking removed from database and customer notified

SQL Query:

DELETE FROM bookings WHERE id = 3

Sample Output:



Action: Get information to notify consumer (same SQL query as 3.2: Accept Booking as Supplier)

## 4. Search

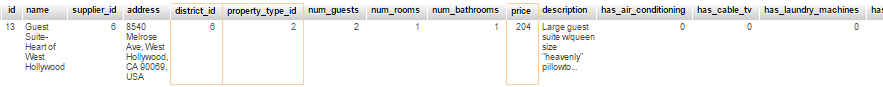
### Event 4.1: Submit Search

Action: Get list of properties that match search parameters. Users can search by any combination of: district, type, features, price.

SQL Query:

SELECT \* FROM rental\_properties WHERE property\_type\_id = 2 AND price <= 300 AND district\_id = 6

SQL Output:



## 5. Property Details

### On Page Load

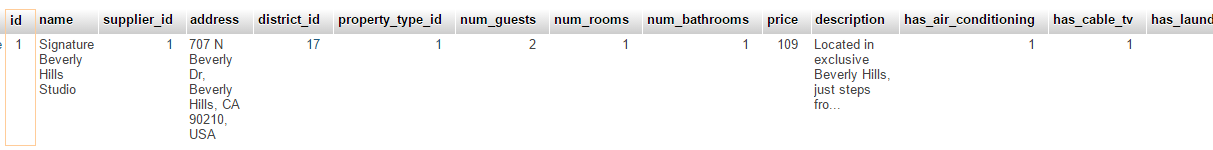
Action: Get property details

SQL Query:

SELECT \* FROM rental\_properties WHERE id = 1

Sample Output:

(The rest is cut off)



Action: Check if user has previously made a booking on property (then display ‘add review’ and delete review buttons)

SQL Query:

SELECT 1 as num\_bookings

FROM rental\_properties JOIN bookings ON property\_id = rental\_properties.id

JOIN users on supplier\_id = users.id

WHERE rental\_properties.id = 1 AND consumer\_id = 3 AND consumer\_id != supplier\_id

Sample Output:



Action: Check if user is the supplier (then show reply and delete reply buttons)

SQL Query:

SELECT COUNT(\*)

FROM rental\_properties

WHERE id = 1 AND supplier\_id = 1

Sample Output:



### Event 5.1: Submit Review

Action: Add review to database.

SQL Query:

INSERT INTO reviews (consumer\_id, property\_id, rating, comment) VALUES

(8, 4, 3, "It was alright")

Sample Output:



### Event 5.2: Submit Reply

Action: Add reply to database.

SQL Query:

UPDATE reviews

SET reply = "How could it improve?"

WHERE consumer\_id = 8 and property\_id = 5

Sample Output:



### Event 5.3: Delete Review

Action: Delete review from database.

SQL Query:

DELETE FROM reviews

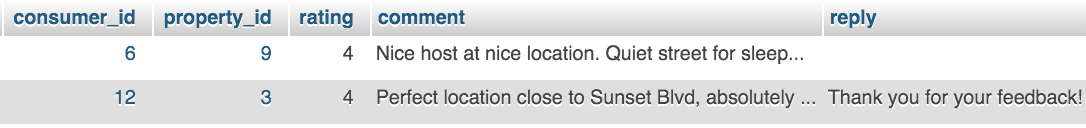
WHERE consumer\_id = 19 and property\_id = 14

Sample Output:

Before -

### 

After -



### Event 5.4: Delete Reply

Action: Delete reply from review in database.

SQL Query:

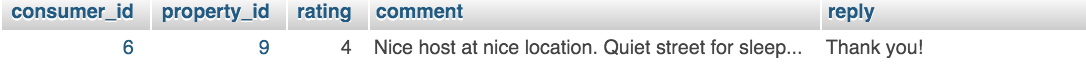
UPDATE reviews

SET reply = ""

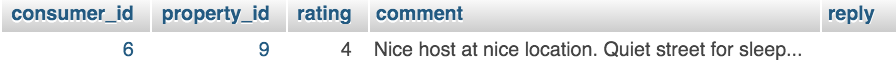
WHERE consumer\_id = 6 and property\_id = 9

Sample Output:

Before -



After -



### Event 5.5: Admin Delete Property

Action: As an admin, delete property and related data from database. Queries will be executed in a transaction.

SQL Queries:

DELETE FROM reviews WHERE property\_id = 2

DELETE FROM bookings WHERE property\_id = 2

DELETE FROM rental\_properties WHERE id = 2

Sample Output:

(2 is gone)



## 6. Booking Request Popup

### Event 6.1: Submit Booking Request

Action: Update database with new booking request.

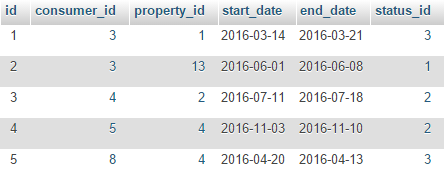
SQL Query:

INSERT INTO bookings (consumer\_id, property\_id, start\_date, end\_date, status\_id)

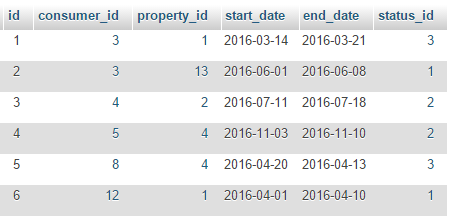
VALUES (12, 1, "2016-04-01", "2016-04-10", 1)

Sample Output:

Before-



After-



Action: Get information to notify supplier of request.

SQL Query:

SELECT supplier.first\_name as supplier\_first, supplier.last\_name as supplier\_last, supplier.email as supplier\_email,

consumer.first\_name as consumer\_first, consumer.last\_name as consumer\_last,

rental\_properties.name as property\_name, start\_date, end\_date

FROM rental\_properties JOIN bookings ON property\_id = rental\_properties.id

JOIN users supplier ON supplier\_id = supplier.id

JOIN users consumer ON consumer\_id = consumer.id

WHERE bookings.id = 1

Sample Output:



## 7. My Properties

User can see a list of their own properties, as well as buttons for adding a property, editing a property, and deleting one. These buttons will open a popup. We will pass information from the My Properties page to these popups, so they don’t require their own sql queries for page load.

### On Page Load

Action: Get all properties and details for current user.

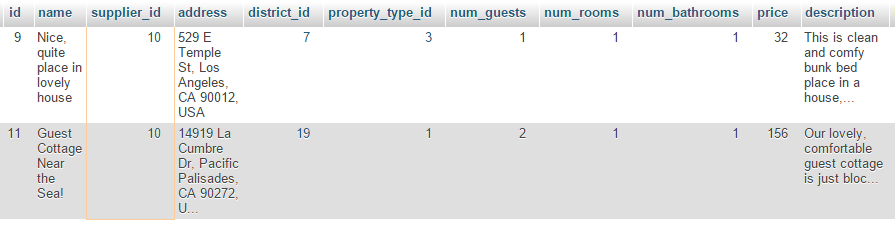
SQL Query:

SELECT \* FROM `rental\_properties`

WHERE supplier\_id = 1

SQL Output:

(rest cut off)



## 8. Add Property Popup

### Event 8.1: Submit form

Action: Add new (valid) property to database.

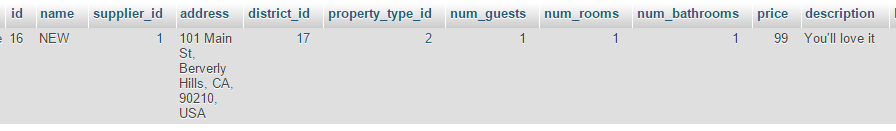
SQL Query:

INSERT INTO rental\_properties (name, supplier\_id, address, district\_id, property\_type\_id, num\_guests, num\_rooms, num\_bathrooms, price, description, has\_air\_conditioning, has\_cable\_tv, has\_laundry\_machines, has\_parking, has\_gym, has\_internet, pets\_allowed, has\_wheelchair\_access, has\_pool, has\_transport\_access, has\_private\_bathroom)

VALUES ("NEW", 1, "101 Main St, Beverly Hills, CA, 90210, USA", 17, 2, 1, 1, 1, 99, "You'll love it", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Sample Output:

(Rest cut off)



## 9. Edit Property Popup

Suppliers can see and edit their property information.

### Event 9.1: Submit form

Action: Update database with modifications for property. Users cannot change the property id.

SQL Query:

UPDATE rental\_properties

SET

name = "NEW Signature Beverly Hills Studio",

property\_type\_id = 2,

num\_guests = 1,

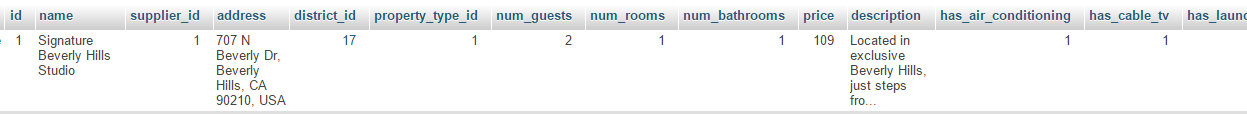
price = 120,

has\_air\_conditioning = 0,

has\_cable\_tv = 0

WHERE id = 1

Sample Output:

Before - 

After - 

## 10. Delete Property Confirmation Popup

This popup will appear when a user clicks “delete”. The user can confirm deletion of

the property, and all related information, or go back to the property details page. Delete

button is only visible if the current user is the supplier or an admin.

### Event 10.1: Confirm deletion

Action: Remove property from database. Current user id must be the supplier\_id of the property.

SQL Queries:

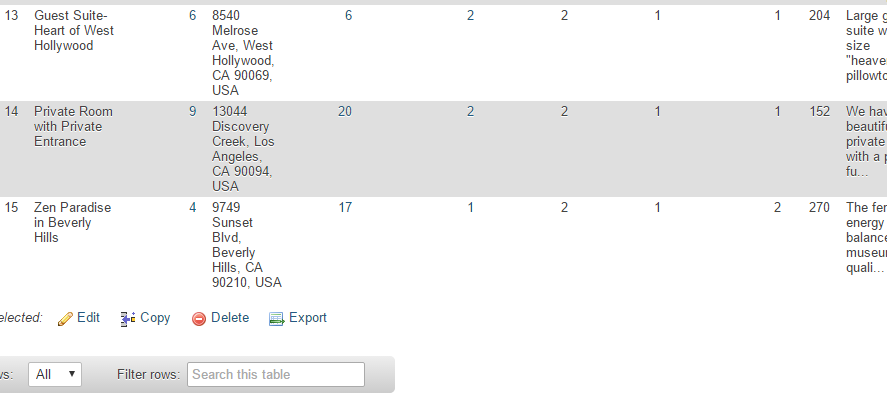
DELETE FROM reviews WHERE property\_id = 16

DELETE FROM bookings WHERE property\_id = 16

DELETE FROM rental\_properties WHERE id = 16

Sample Output:

(16 is gone)



Action: Get information to notify any consumers with pending or confirmed bookings. If admin deletes property, also notify the supplier.

SQL Query:

SELECT bookings.id as booking\_id, rental\_properties.name as property\_name, start\_date, end\_date, status\_type\_name,

consumers.first\_name as consumer\_first, consumers.last\_name as consumer\_last, consumers.email as consumer\_email,

suppliers.first\_name as supplier\_first, suppliers.last\_name as supplier\_last, suppliers.email as supplier\_email

FROM bookings JOIN rental\_properties ON property\_id = rental\_properties.id

JOIN booking\_status\_types ON status\_id = booking\_status\_types.id

JOIN users consumers ON consumer\_id = consumers.id

JOIN users suppliers ON supplier\_id = suppliers.id

WHERE (status\_id = 1 OR status\_id =2) AND bookings.id IN (SELECT bookings.id FROM bookings WHERE property\_id = 4)

SQL Output:



## 11. User Settings

On this page, users can see and edit all of the information in the users table except id and is\_admin. We will build up the update query based on the changes the user makes. Users can also cancel their membership to the site. This action will remove all their bookings, reviews, replies, and properties.

### On Page Load

Action: Get user information

SQL Query:

SELECT is\_admin, first\_name, last\_name, password, phone\_number, grad\_year, faculty\_name, degree\_type\_name, gender

FROM users JOIN faculties ON faculty\_id = faculties.id

JOIN degree\_types ON degree\_type\_id = degree\_types.id

WHERE users.id = 1

Sample Output:



### Event 11.1: Submit form

Action: Update users table with new information. Id of modified row must match that of the current user.

SQL Query:

UPDATE users

SET

first\_name = "S",

password = "Hj5545lmnk09wWzx51r",

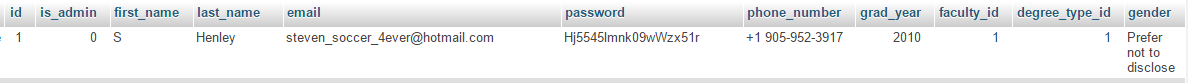
gender = "Prefer not to disclose"

WHERE id = 1

Sample Output:

Before - 

After -



### Event 11.2: Cancel Membership

Action: Remove user from database, all their bookings, reviews and replies, and properties. These queries will be executed in a transaction.

SQL Queries:

DELETE r FROM reviews r JOIN rental\_properties p ON r.property\_id = p.id

WHERE p.id = 23

DELETE b FROM bookings b JOIN rental\_properties p ON b.property\_id = p.id

WHERE p.id = 23

DELETE FROM bookings WHERE consumer\_id = 23

DELETE FROM rental\_properties WHERE supplier\_id = 23

DELETE FROM users WHERE id = 23

Sample Output:

(23 is gone)



## 12. Administrator

This page will have three tab views, Members, Bookings, and Reviews, but all data for the three views will be loaded when the page is loaded. The Members view will show a list of members and a delete button for each. The Bookings view will have a list of bookings and three buttons for sorting these bookings by property, supplier, or consumer. We will only do one query to get the bookings from the server, then do the sorting of this data on the frontend. The Reviews view will have a list of all reviews, and the user can sort by either the property or the supplier.

### On Page Load

Action: Get all members.

SQL Query:

SELECT id, is\_admin, first\_name, last\_name, email, phone\_number, grad\_year, faculty\_id, degree\_type\_id, gender

FROM `users`

ORDER BY last\_name

Sample Output:

(the rest is cut off)



Action: Get all bookings

SQL Query:

SELECT bookings.id as booking\_id, start\_date, end\_date, status\_type\_name,

consumers.id as consumer\_id, consumers.first\_name as consumer\_first, consumers.last\_name as consumer\_last,

suppliers.id as supplier\_id, suppliers.first\_name as supplier\_first, suppliers.last\_name as supplier\_last,

rental\_properties.id as property\_id, rental\_properties.name

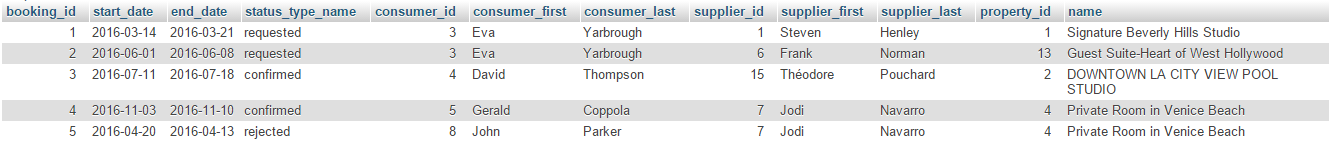
FROM bookings JOIN rental\_properties ON property\_id = rental\_properties.id

JOIN booking\_status\_types ON status\_id = booking\_status\_types.id

JOIN users consumers ON consumer\_id = consumers.id

JOIN users suppliers ON supplier\_id = suppliers.id

Sample Output:



Action: Get all reviews.

SQL Query:

SELECT rating, comment, reply,

consumers.id as consumer\_id, consumers.first\_name as consumer\_first, consumers.last\_name as consumer\_last,

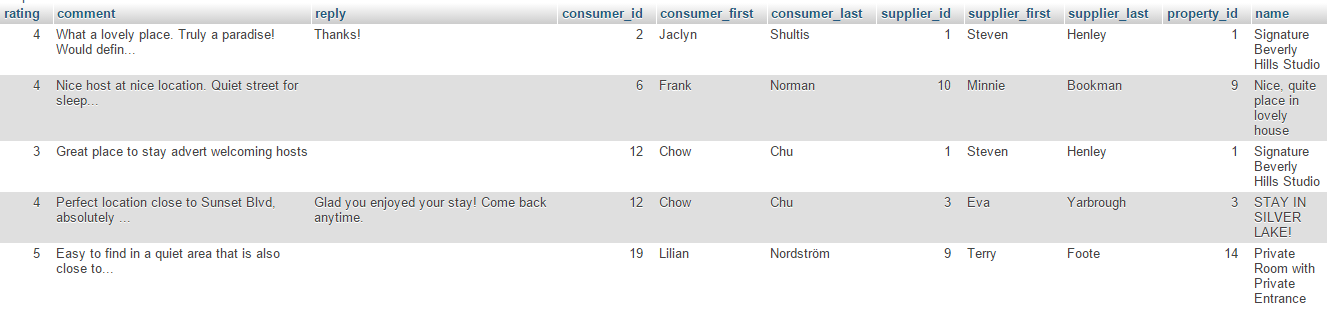
suppliers.id as supplier\_id, suppliers.first\_name as supplier\_first, suppliers.last\_name as supplier\_last,

rental\_properties.id as property\_id, rental\_properties.name

FROM reviews JOIN rental\_properties ON property\_id = rental\_properties.id

JOIN users consumers ON consumer\_id = consumers.id

JOIN users suppliers ON supplier\_id = suppliers.id

SQL Output:

### Event 12.1: Click “delete member”

Runs the same set of queries as 11.2: Cancel Membership.