

# renting\_db

creating database with Relation Database Management System MySQL



In this phase, all the SQL statements of the 20 tables will be copied and displayed with a screenshot of the result.

## Create Database:

```
DROP DATABASE IF EXISTS renting_db;
```

```
CREATE DATABASE IF NOT EXISTS renting_db;
```

```
USE renting_db;
```

```
DROP TABLE IF EXISTS country, city, property_commission, property_type,  
neighbourhood, host, property, facility, room_type, room, amenities, guest_commission,  
guest, property_review, guest_review, cancelation, reservation, payment_status, voucher,  
payment;
```

# voucher

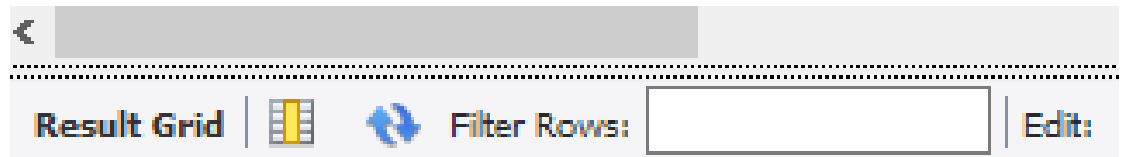
## Creating voucher table:

```
CREATE TABLE voucher  
(voucherID INTEGER PRIMARY KEY  
  AUTO_INCREMENT,  
  voucher VARCHAR(20),  
  discount DECIMAL(4,2),  
  updated_at DATETIME DEFAULT  
  now());
```

## SQL query



```
1 • SELECT voucherID, voucher, discount  
2   FROM renting_db.voucher  
3   WHERE discount > 0.24;
```



|   | voucherID | voucher | discount |
|---|-----------|---------|----------|
| ▶ | 16        | SUMM    | 0.25     |
|   | 17        | SPEC1   | 0.30     |
|   | 18        | SPEC2   | 0.32     |
|   | 19        | VIP1    | 0.35     |
|   | 20        | VIP2    | 0.40     |
|   | SUM       | SUM     | SUM      |

# payment\_status

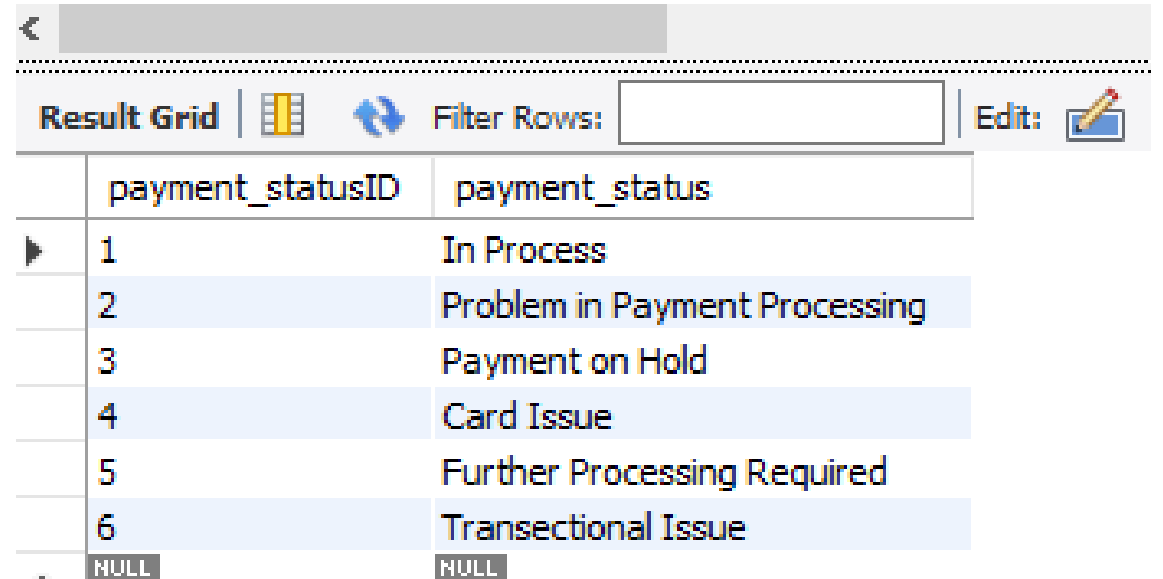
Creating payment\_status table:

```
CREATE TABLE payment_status  
(payment_statusID INTEGER  
PRIMARY KEY AUTO_INCREMENT,  
payment_status VARCHAR(50),  
updated_at DATETIME DEFAULT  
now());
```

SQL query



```
1 • SELECT payment_statusID, payment_status  
2 FROM renting_db.payment_status  
3 LIMIT 6;
```



|   | payment_statusID | payment_status                |
|---|------------------|-------------------------------|
| ▶ | 1                | In Process                    |
|   | 2                | Problem in Payment Processing |
|   | 3                | Payment on Hold               |
|   | 4                | Card Issue                    |
|   | 5                | Further Processing Required   |
|   | 6                | Transectional Issue           |
| ⌵ | NULL             | NULL                          |

# cancelation

Creating cancelation table:

```
CREATE TABLE cancelation  
(cancelationID INTEGER PRIMARY  
KEY AUTO_INCREMENT,  
cancel_charge DECIMAL(5,2),  
updated_at DATETIME DEFAULT  
now() );
```

SQL query



```
1 • SELECT cancelationID,  
2    cancel_charge  
3    FROM renting_db.cancelation  
4    WHERE cancel_charge > .70;
```

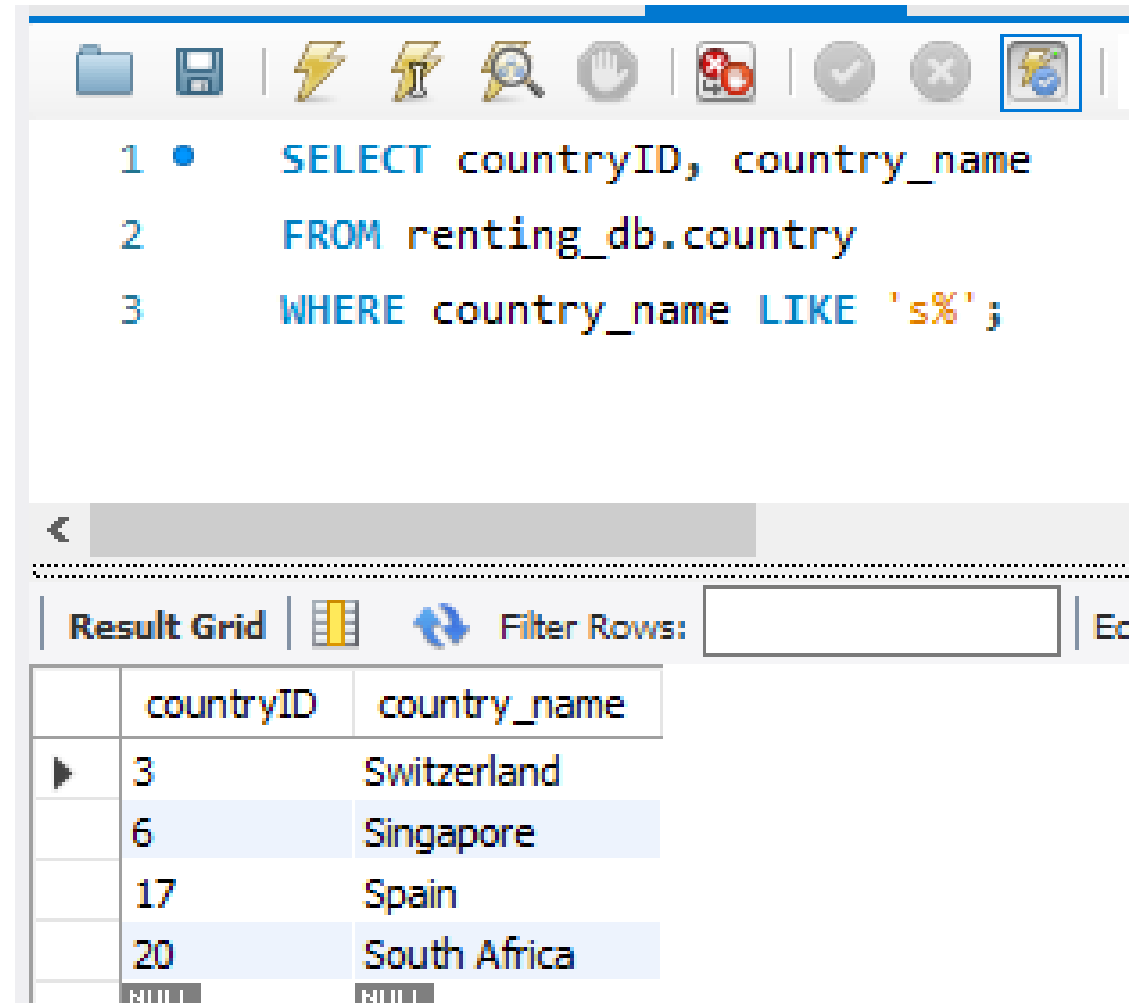
| <            |               |               |
|--------------|---------------|---------------|
| Result Grid  |               |               |
| Filter Rows: |               |               |
|              | cancelationID | cancel_charge |
| ▶            | 15            | 0.75          |
|              | 16            | 0.80          |
|              | 17            | 0.85          |
|              | 18            | 0.90          |
|              | 19            | 0.95          |
|              | 20            | 1.00          |

# country

Creating country table:

```
CREATE TABLE country  
(countryID INTEGER PRIMARY KEY  
AUTO_INCREMENT,  
country_name VARCHAR(70) NOT  
NULL,  
updated_at DATETIME DEFAULT  
now());
```

SQL query



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and navigation. The query text is as follows:

```
1 • SELECT countryID, country_name  
2 FROM renting_db.country  
3 WHERE country_name LIKE 's%';
```

Below the query editor, the results are displayed in a table. The table has two columns: **countryID** and **country\_name**. The results show four rows of data, with the first row highlighted.

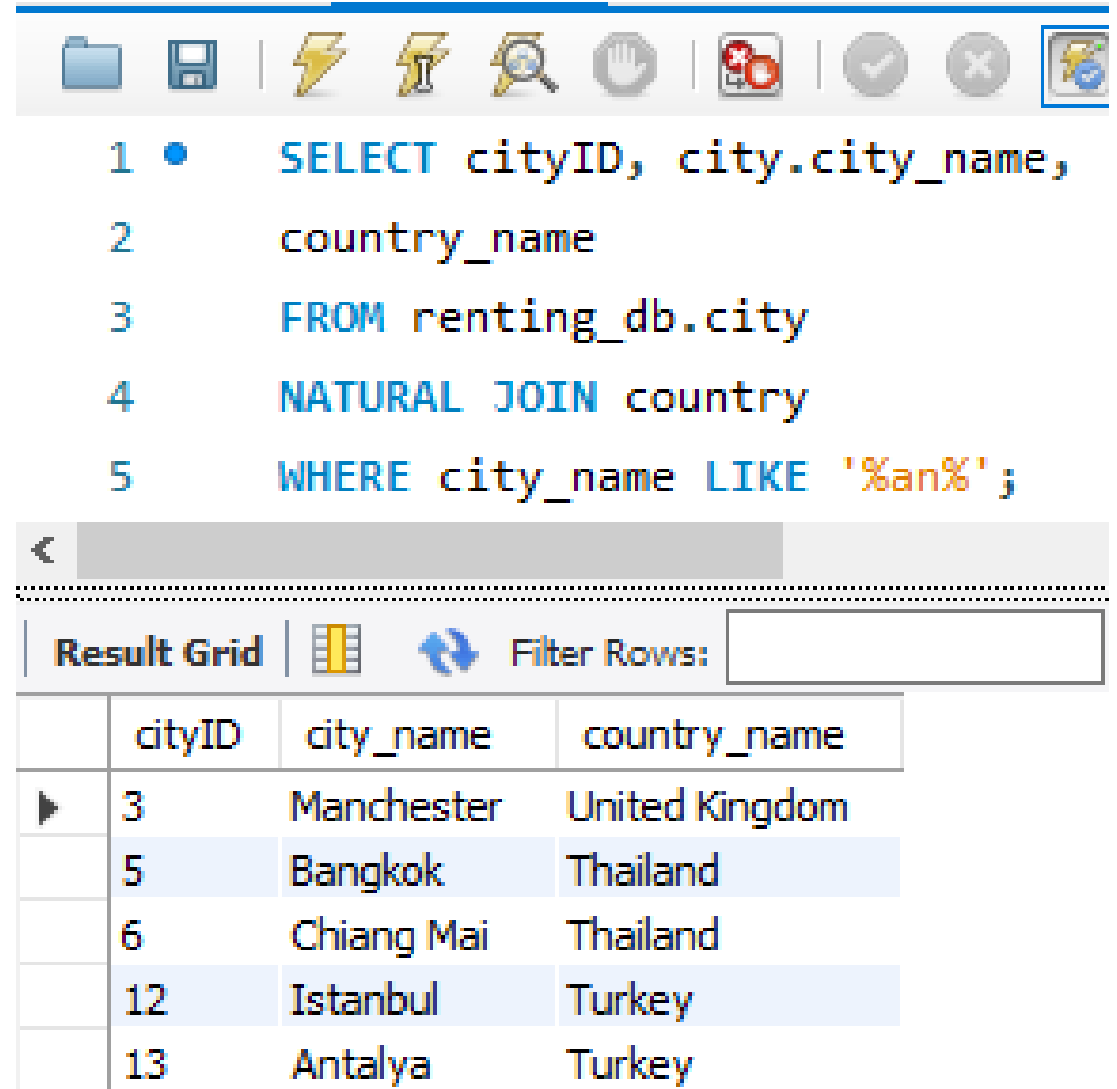
|   | countryID | country_name |
|---|-----------|--------------|
| ▶ | 3         | Switzerland  |
|   | 6         | Singapore    |
|   | 17        | Spain        |
|   | 20        | South Africa |

# city

Creating city table:

```
CREATE TABLE city  
(cityID INTEGER PRIMARY KEY  
AUTO_INCREMENT, countryID  
INTEGER, city_name VARCHAR(70)  
NOT NULL, FOREIGN KEY (countryID)  
REFERENCES country (countryID),  
updated_at DATETIME DEFAULT  
now());
```

SQL query



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and navigation. The query is as follows:

```
1 • SELECT cityID, city.city_name,  
2     country_name  
3     FROM renting_db.city  
4     NATURAL JOIN country  
5     WHERE city_name LIKE '%an%';
```

Below the query editor is a "Result Grid" section. It includes a "Filter Rows:" input field and a table displaying the results of the query. The table has four columns: cityID, city\_name, and country\_name. The results are as follows:

|   | cityID | city_name  | country_name   |
|---|--------|------------|----------------|
| ▶ | 3      | Manchester | United Kingdom |
|   | 5      | Bangkok    | Thailand       |
|   | 6      | Chiang Mai | Thailand       |
|   | 12     | Istanbul   | Turkey         |
|   | 13     | Antalya    | Turkey         |

# property\_commission

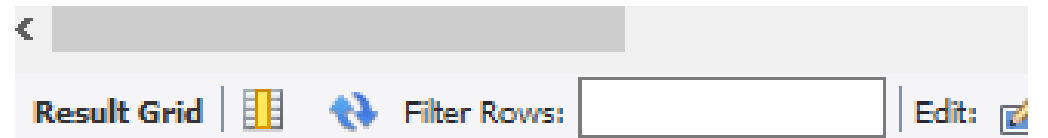
Creating property\_commission table:

```
CREATE TABLE property_commission  
(property_commissionID INTEGER  
PRIMARY KEY AUTO_INCREMENT,  
property_commission DECIMAL(4,2)  
NOT NULL,  
updated_at DATETIME DEFAULT  
now());
```

## SQL query



```
1 • SELECT property_commissionID,  
2   property_commission  
3   FROM renting_db.property_commission  
4   WHERE property_commission BETWEEN  
5     0.15 AND 0.20;
```




|   | property_commissionID | property_commission |
|---|-----------------------|---------------------|
| ▶ | 11                    | 0.15                |
|   | 12                    | 0.16                |
|   | 13                    | 0.17                |
|   | 14                    | 0.18                |
|   | 15                    | 0.19                |
|   | 16                    | 0.20                |
|   | 17004                 | 17004               |

# property\_type


Creating property\_type table:

```
CREATE TABLE property_type  
(property_typeID INTEGER PRIMARY  
KEY AUTO_INCREMENT,  
property_type VARCHAR(30),  
updated_at DATETIME DEFAULT  
now());
```

## SQL query



```
1 • SELECT property_typeID,  
2     property_type  
3     FROM renting_db.property_type  
4     WHERE property_type LIKE '%n%';
```



|   | property_typeID | property_type        |
|---|-----------------|----------------------|
| ▶ | 1               | Apartment            |
|   | 2               | Serviced Apartment   |
|   | 5               | Town House           |
|   | 6               | Bed and Breakfast    |
|   | 7               | Bungalow             |
|   | 9               | Hostel Accommodation |
|   | 16              | Cabin                |

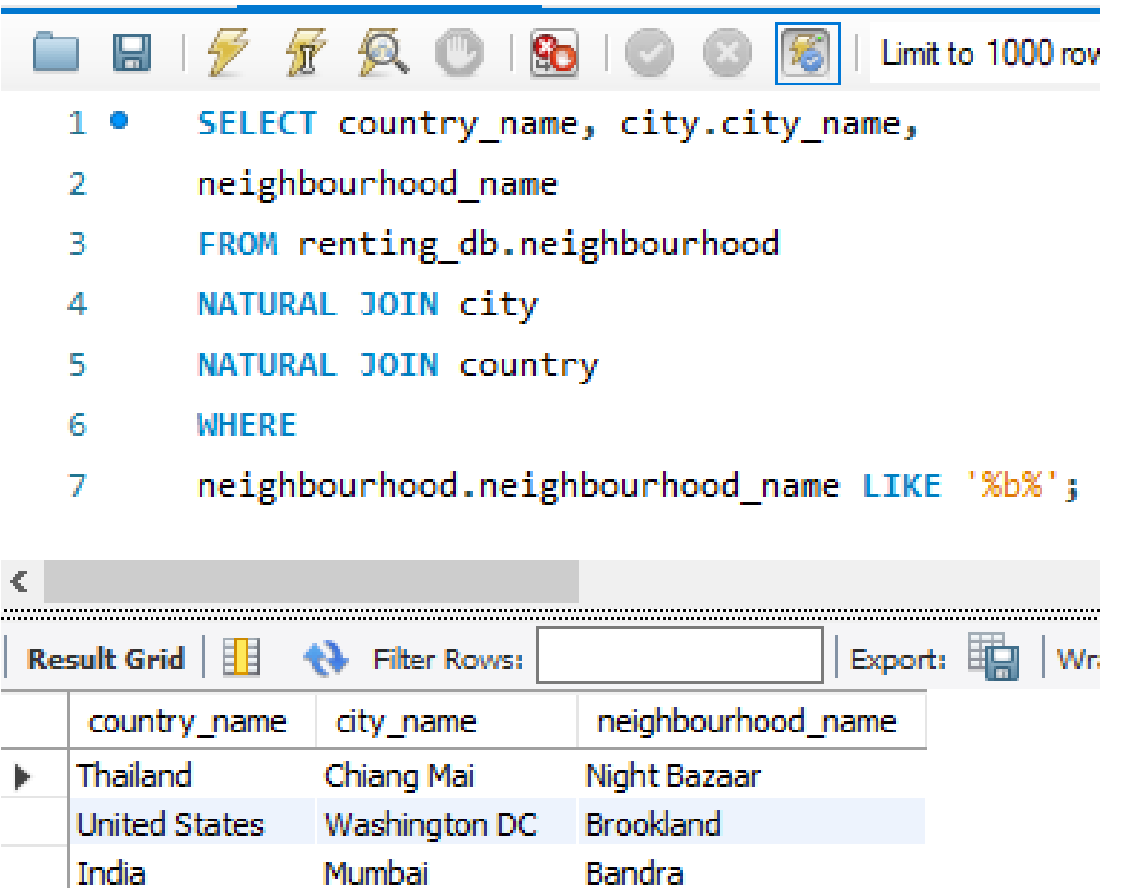


# neighbourhood

Creating neighbourhood table:

```
CREATE TABLE neighbourhood  
(neighbourhoodID INTEGER PRIMARY  
KEY AUTO_INCREMENT,  
cityID INTEGER,  
neighbourhood_name VARCHAR(30),  
FOREIGN KEY (cityID) REFERENCES  
city (cityID),  
updated_at DATETIME DEFAULT  
now());
```

SQL query



```
1 • SELECT country_name, city.city_name,  
2     neighbourhood_name  
3     FROM renting_db.neighbourhood  
4     NATURAL JOIN city  
5     NATURAL JOIN country  
6     WHERE  
7     neighbourhood.neighbourhood_name LIKE '%b%';
```

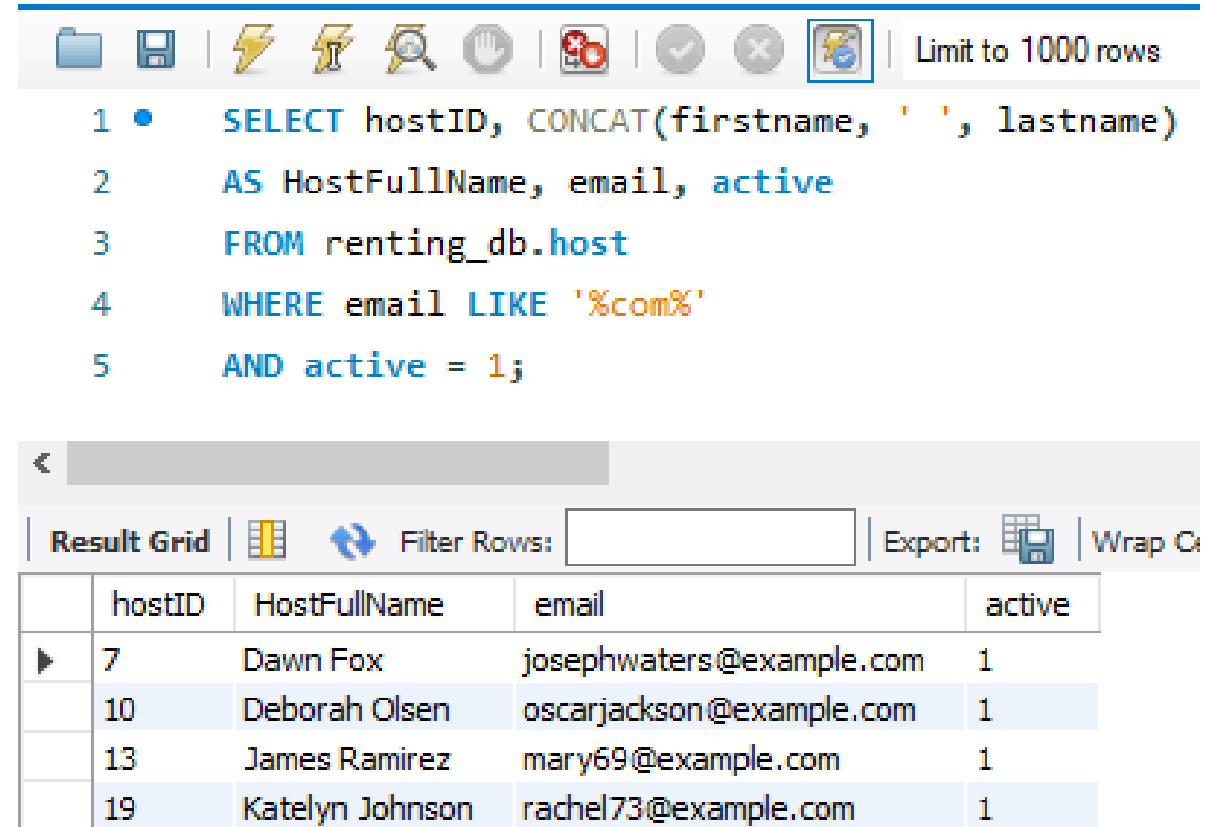
|   | country_name  | city_name     | neighbourhood_name |
|---|---------------|---------------|--------------------|
| ▶ | Thailand      | Chiang Mai    | Night Bazaar       |
|   | United States | Washington DC | Brookland          |
|   | India         | Mumbai        | Bandra             |

# host

## Creating host table:

```
CREATE TABLE host  
(hostID INTEGER PRIMARY KEY  
  AUTO_INCREMENT,  
  firstname VARCHAR(30),  
  lastname VARCHAR(30),  
  address VARCHAR(100),  
  email VARCHAR(30),  
  phone VARCHAR(20),  
  password CHAR(8),  
  member_since DATETIME,  
  active BOOLEAN,  
  updated_at DATETIME DEFAULT now());
```

## SQL query



The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons for file operations, execution, and viewing. Below the toolbar, the SQL query is displayed in a monospaced font. The query is a SELECT statement that retrieves hostID, concatenates the first and last names into HostFullName, and includes email and active status. It filters for records where the email ends with '.com' and the active status is 1. Below the query, there is a result grid showing the output of the query. The grid has five columns: hostID, HostFullName, email, and active. It displays four rows of data, each representing a host record.

```
1 • SELECT hostID, CONCAT(firstname, ' ', lastname)  
2   AS HostFullName, email, active  
3   FROM renting_db.host  
4   WHERE email LIKE '%com%'  
5   AND active = 1;
```

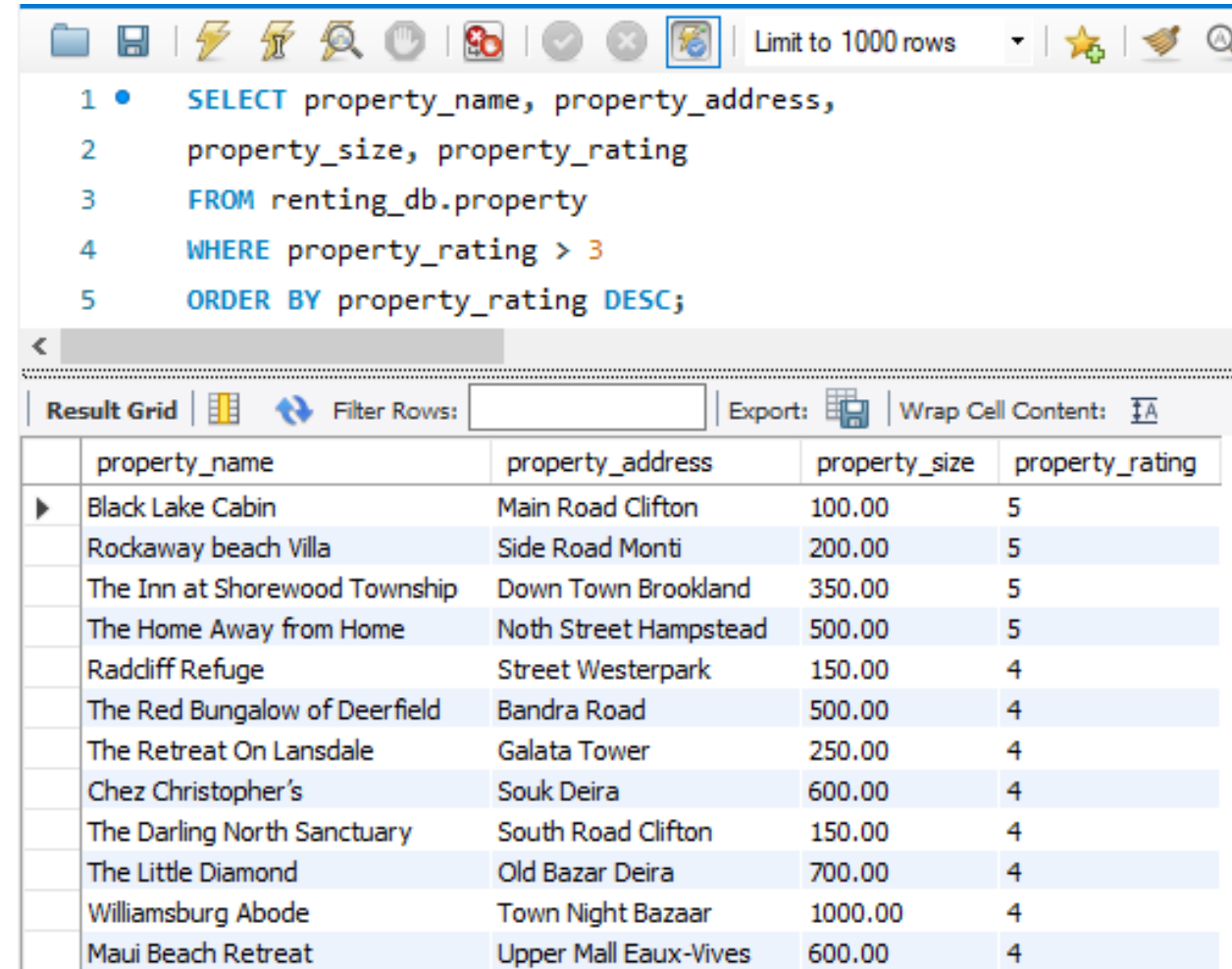
|   | hostID | HostFullName    | email                    | active |
|---|--------|-----------------|--------------------------|--------|
| ▶ | 7      | Dawn Fox        | josephwaters@example.com | 1      |
|   | 10     | Deborah Olsen   | oscarjackson@example.com | 1      |
|   | 13     | James Ramirez   | mary69@example.com       | 1      |
|   | 19     | Katelyn Johnson | rachel73@example.com     | 1      |

# property

## Creating property table:

```
CREATE TABLE property
(propertyID INTEGER PRIMARY KEY
AUTO_INCREMENT,hostID
INTEGER,property_typeID
INTEGER,neighbourhoodID
INTEGER,property_commissionID
INTEGER,property_name
VARCHAR(50),property_address
VARCHAR(100),property_size
DECIMAL(8,2),property_rating INTEGER,qty_room
INTEGER,FOREIGN KEY (property_commissionID)
REFERENCES property_commission
(property_commissionID),FOREIGN KEY (hostID)
REFERENCES host (hostID),FOREIGN KEY
(property_typeID) REFERENCES property_type
(property_typeID),FOREIGN KEY (neighbourhoodID)
REFERENCES neighbourhood
(neighbourhoodID),updated_at DATETIME DEFAULT
now());
```

## SQL query



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT property_name, property_address,
2     property_size, property_rating
3     FROM renting_db.property
4     WHERE property_rating > 3
5     ORDER BY property_rating DESC;
```

Below the query editor, there is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The results are displayed in a table with the following columns: property\_name, property\_address, property\_size, and property\_rating. The table contains 14 rows of data, sorted by property\_rating in descending order.

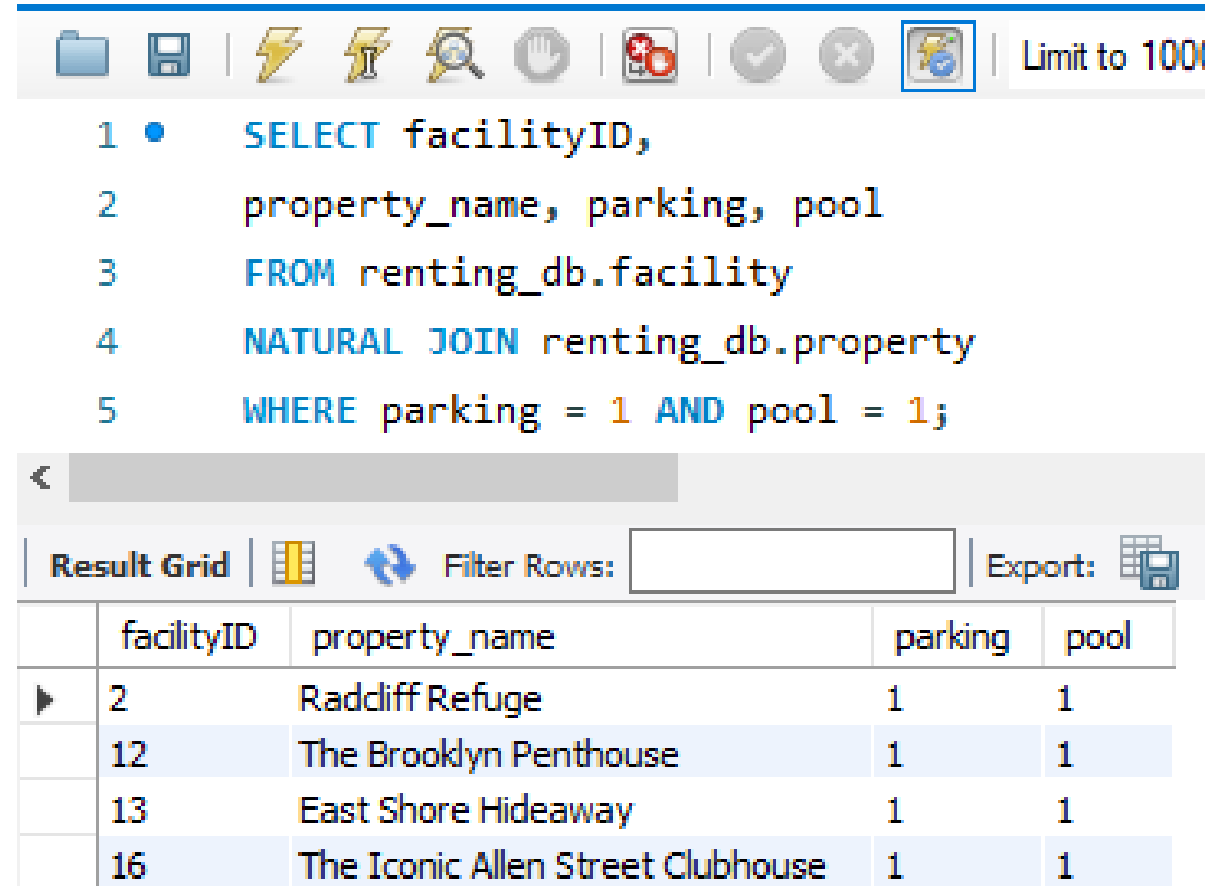
|   | property_name                 | property_address      | property_size | property_rating |
|---|-------------------------------|-----------------------|---------------|-----------------|
| ▶ | Black Lake Cabin              | Main Road Clifton     | 100.00        | 5               |
|   | Rockaway beach Villa          | Side Road Monti       | 200.00        | 5               |
|   | The Inn at Shorewood Township | Down Town Brookland   | 350.00        | 5               |
|   | The Home Away from Home       | Noth Street Hampstead | 500.00        | 5               |
|   | Raddliff Refuge               | Street Westerpark     | 150.00        | 4               |
|   | The Red Bungalow of Deerfield | Bandra Road           | 500.00        | 4               |
|   | The Retreat On Lansdale       | Galata Tower          | 250.00        | 4               |
|   | Chez Christopher's            | Souk Deira            | 600.00        | 4               |
|   | The Darling North Sanctuary   | South Road Clifton    | 150.00        | 4               |
|   | The Little Diamond            | Old Bazar Deira       | 700.00        | 4               |
|   | Williamsburg Abode            | Town Night Bazaar     | 1000.00       | 4               |
|   | Maui Beach Retreat            | Upper Mall Eaux-Vives | 600.00        | 4               |

# facility

Creating facility table:

```
CREATE TABLE facility  
(facilityID INTEGER PRIMARY KEY  
  AUTO_INCREMENT,  
  propertyID INTEGER,  
  parking BOOLEAN,  
  pool BOOLEAN,  
  FOREIGN KEY (propertyID)  
  REFERENCES property (propertyID),  
  updated_at DATETIME DEFAULT  
  now() );
```

SQL query



The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons for file operations, execution, and viewing. Below the toolbar, the SQL query is displayed in a text area. The query is a SELECT statement that retrieves facilityID, property\_name, parking, and pool from the facility table, joined with the property table on a natural join, and filtered by parking = 1 and pool = 1. Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Export' button. The result grid itself is a table with 5 columns: an index column, facilityID, property\_name, parking, and pool. It contains 4 rows of data.

```
1 • SELECT facilityID,  
2     property_name, parking, pool  
3     FROM renting_db.facility  
4     NATURAL JOIN renting_db.property  
5     WHERE parking = 1 AND pool = 1;
```

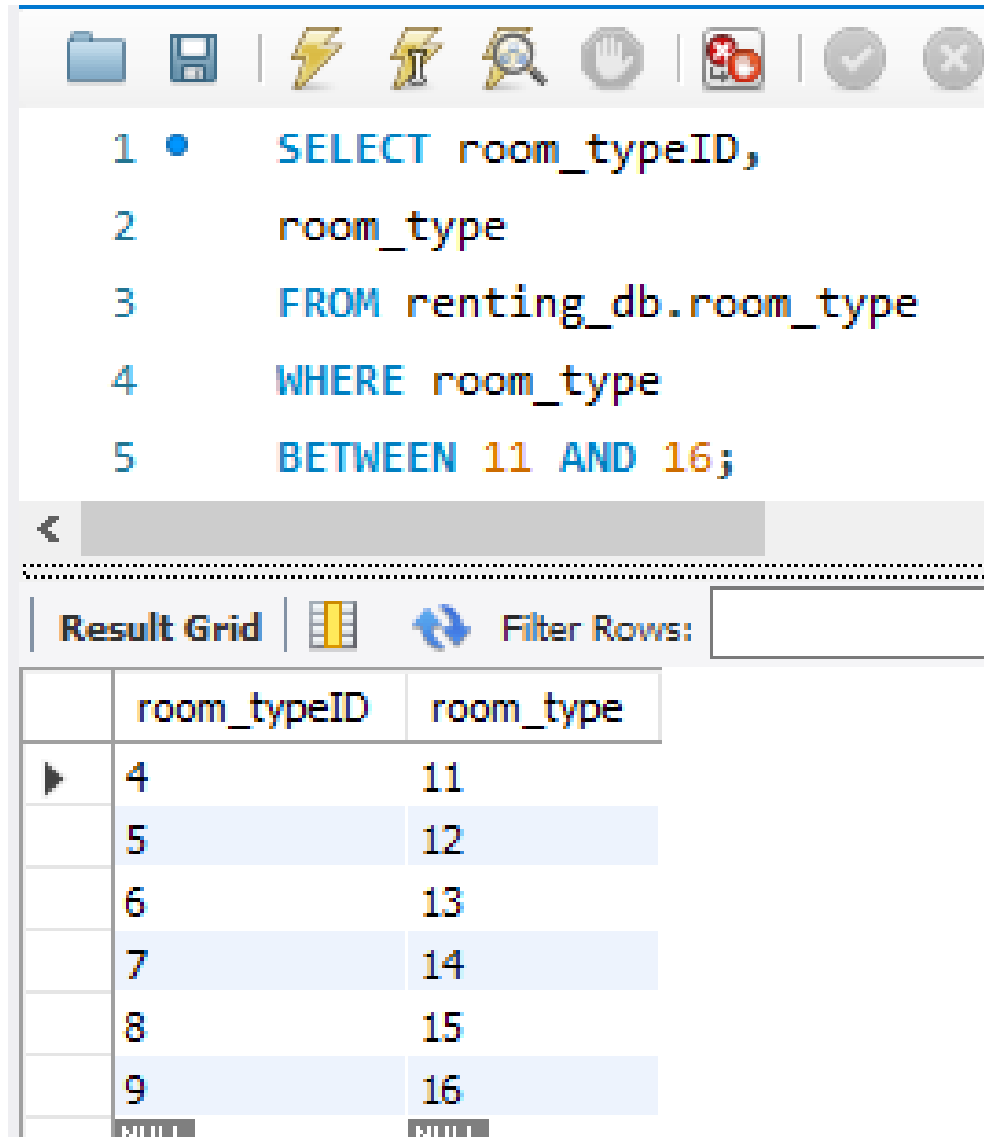
|   | facilityID | property_name                     | parking | pool |
|---|------------|-----------------------------------|---------|------|
| ▶ | 2          | Raddiff Refuge                    | 1       | 1    |
|   | 12         | The Brooklyn Penthouse            | 1       | 1    |
|   | 13         | East Shore Hideaway               | 1       | 1    |
|   | 16         | The Iconic Allen Street Clubhouse | 1       | 1    |

# room\_type

Creating room\_type table:

```
CREATE TABLE room_type  
(room_typeID INTEGER PRIMARY  
KEY AUTO_INCREMENT,room_type  
INTEGER,updated_at DATETIME  
DEFAULT now() );
```

SQL query



The screenshot shows a SQL query editor window. The query is a SELECT statement that retrieves room\_typeID and room\_type from the renting\_db.room\_type table, filtered by room\_type values between 11 and 16. Below the query editor, the results are displayed in a grid format. The grid has two columns: room\_typeID and room\_type. The results show six rows of data, with room\_typeID values ranging from 4 to 9 and room\_type values ranging from 11 to 16.

```
1 • SELECT room_typeID,  
2     room_type  
3     FROM renting_db.room_type  
4     WHERE room_type  
5     BETWEEN 11 AND 16;
```

< [Progress Bar]

Result Grid | [Grid Icon] | Filter Rows: [Input Field]

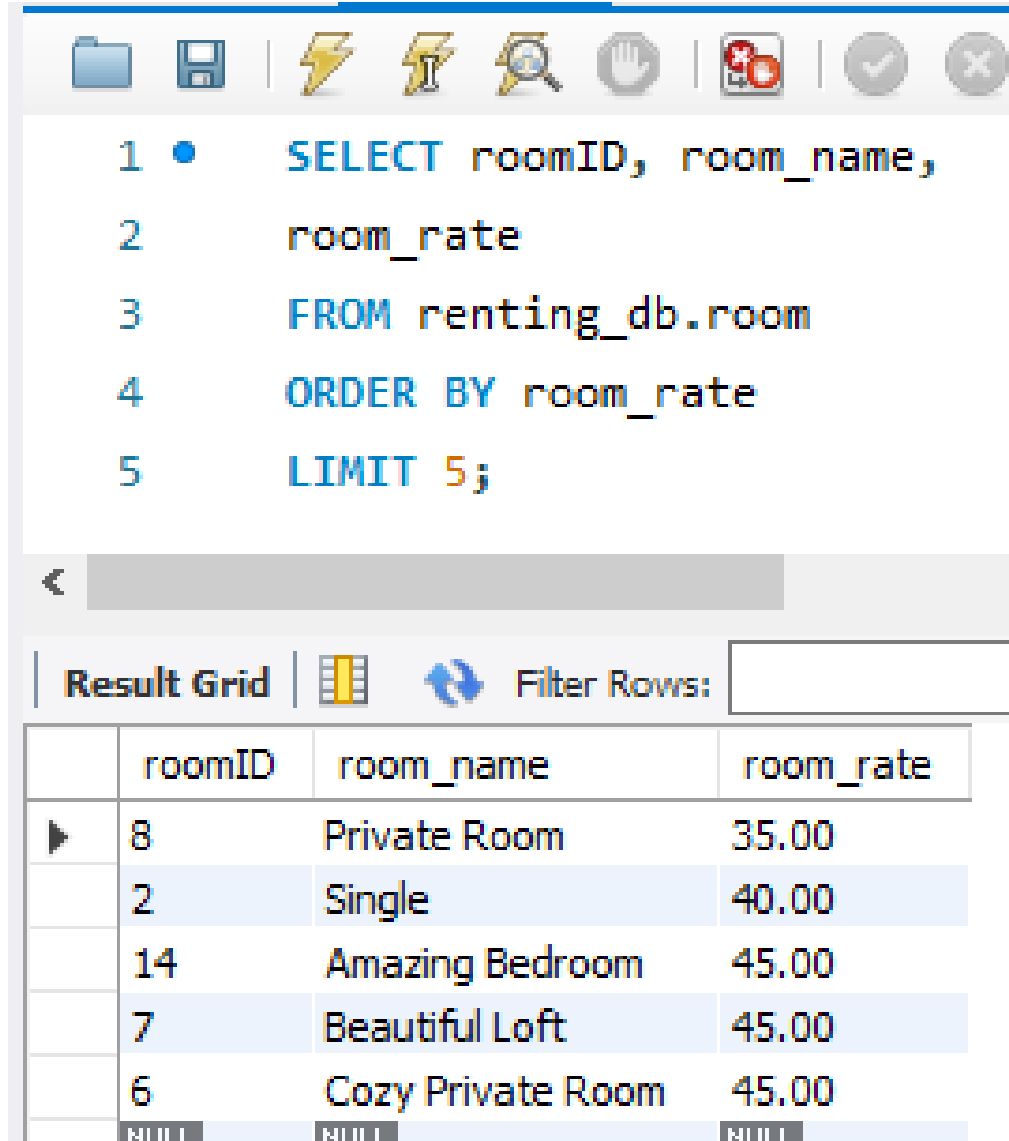
|   | room_typeID | room_type |
|---|-------------|-----------|
| ▶ | 4           | 11        |
|   | 5           | 12        |
|   | 6           | 13        |
|   | 7           | 14        |
|   | 8           | 15        |
|   | 9           | 16        |

# room

Creating room table:

```
CREATE TABLE room
(roomID INTEGER PRIMARY KEY
AUTO_INCREMENT,propertyID
INTEGER,room_typeID
INTEGER,room_name
VARCHAR(30),room_number
INTEGER,room_size INTEGER,room_rate
DECIMAL(6,2),room_availability
BOOLEAN,FOREIGN KEY (room_typeID)
REFERENCES room_type
(room_typeID),FOREIGN KEY (propertyID)
REFERENCES property
(propertyID),updated_at DATETIME
DEFAULT now() );
```

SQL query



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, search, and window management. The query is as follows:

```
1 • SELECT roomID, room_name,
2     room_rate
3     FROM renting_db.room
4     ORDER BY room_rate
5     LIMIT 5;
```

Below the query editor is a 'Result Grid' tab. It displays a table with 4 columns: roomID, room\_name, and room\_rate. The first column is unlabeled but contains row numbers. The table contains 5 rows of data, which are the top 5 rooms ordered by rate.

|   | roomID | room_name         | room_rate |
|---|--------|-------------------|-----------|
| ▶ | 8      | Private Room      | 35.00     |
|   | 2      | Single            | 40.00     |
|   | 14     | Amazing Bedroom   | 45.00     |
|   | 7      | Beautiful Loft    | 45.00     |
|   | 6      | Cozy Private Room | 45.00     |
|   | NULL   | NULL              | NULL      |

# amenities

## Creating amenities table:

```
CREATE TABLE amenities  
(amenitiesID INTEGER PRIMARY KEY  
AUTO_INCREMENT,roomID  
INTEGER,WiFi BOOLEAN,heating  
BOOLEAN,TV BOOLEAN,towel  
BOOLEAN,shampoo  
BOOLEAN,FOREIGN KEY (roomID)  
REFERENCES room  
(roomID),updated_at DATETIME  
DEFAULT now() );
```

## SQL query



```
1 • SELECT amenitiesID, room_name,  
2 WiFi FROM renting_db.amenities  
3 NATURAL JOIN renting_db.room  
4 WHERE WiFi = 1  
5 LIMIT 5;
```

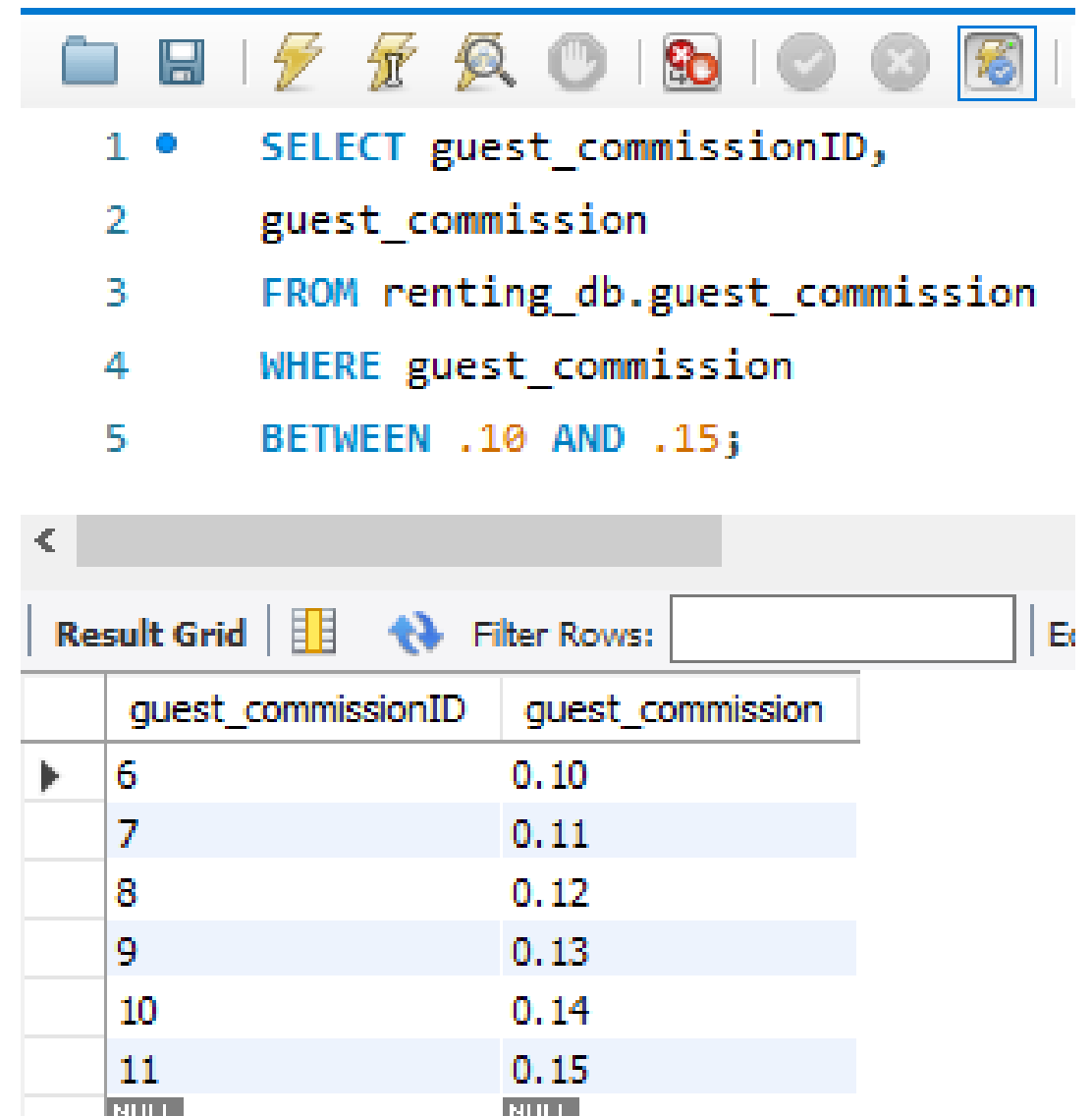
|   | amenitiesID | room_name         | WiFi |
|---|-------------|-------------------|------|
| ▶ | 3           | Double            | 1    |
|   | 6           | Cozy Private Room | 1    |
|   | 7           | Beautiful Loft    | 1    |
|   | 8           | Private Room      | 1    |
|   | 9           | Bright Brooklyn   | 1    |

# guest\_commission

Creating guest\_commission table:

```
CREATE TABLE guest_commission  
(guest_commissionID INTEGER  
PRIMARY KEY AUTO_INCREMENT,  
guest_commission DECIMAL(4,2),  
updated_at DATETIME DEFAULT  
now() );
```

SQL query



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and search. The query is as follows:

```
1 • SELECT guest_commissionID,  
2     guest_commission  
3     FROM renting_db.guest_commission  
4     WHERE guest_commission  
5     BETWEEN .10 AND .15;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query. The results are shown in a table with two columns: 'guest\_commissionID' and 'guest\_commission'. The table contains six rows of data, with the first row highlighted.

|   | guest_commissionID | guest_commission |
|---|--------------------|------------------|
| ▶ | 6                  | 0.10             |
|   | 7                  | 0.11             |
|   | 8                  | 0.12             |
|   | 9                  | 0.13             |
|   | 10                 | 0.14             |
|   | 11                 | 0.15             |

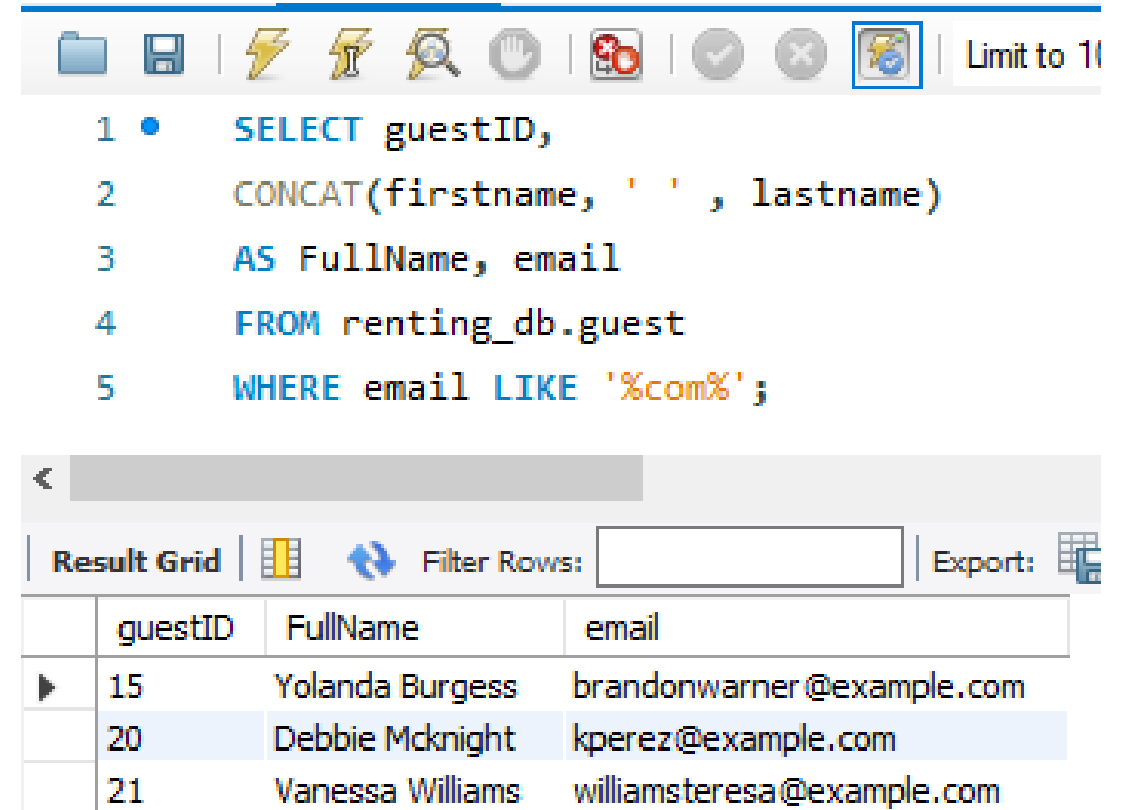


# guest

## Creating guest table:

```
CREATE TABLE guest
(guestID INTEGER PRIMARY KEY
AUTO_INCREMENT,guest_commissionID
INTEGER,firstname VARCHAR(30),lastname
VARCHAR(30),address
VARCHAR(100),email VARCHAR(30),phone
VARCHAR(20), password
CHAR(8),member_since
DATETIME,guest_level INTEGER, active
BOOLEAN, guest_rating INTEGER,FOREIGN
KEY (guest_commissionID) REFERENCES
guest_commission(guest_commissionID),
updated_at DATETIME DEFAULT now());
```

## SQL query



The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons for file operations, execution, and navigation. Below the toolbar, the SQL query is displayed in a text area, numbered 1 through 5. The query is a SELECT statement that retrieves guestID, concatenates the first and last names into a column named 'FullName', and includes the email address. It filters the results from the 'renting\_db.guest' table where the email contains the substring 'com'. Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Export' button. The result grid itself is a table with three columns: 'guestID', 'FullName', and 'email'. It contains three rows of data, with the second row highlighted in blue.

```
1 • SELECT guestID,
2   CONCAT(firstname, ' ', lastname)
3   AS FullName, email
4   FROM renting_db.guest
5   WHERE email LIKE '%com%';
```

|   | guestID | FullName         | email                      |
|---|---------|------------------|----------------------------|
| ▶ | 15      | Yolanda Burgess  | brandonwarner@example.com  |
|   | 20      | Debbie Mcknight  | kperez@example.com         |
|   | 21      | Vanessa Williams | williamsteresa@example.com |

# property\_review




Creating property\_review table:

```
CREATE TABLE property_review  
(property_reviewID INTEGER PRIMARY  
KEY AUTO_INCREMENT,propertyID  
INTEGER,guestID INTEGER,service  
INTEGER,amenities INTEGER,location  
INTEGER,FOREIGN KEY (guestID)  
REFERENCES guest (guestID),FOREIGN  
KEY (propertyID) REFERENCES property  
(propertyID),updated_at DATETIME  
DEFAULT now());
```

SQL query



```
1 • SELECT property_name,  
2     service, amenities, location  
3     FROM renting_db.property_review  
4     NATURAL JOIN renting_db.property  
5     WHERE service = 5 AND amenities = 5  
6     AND location = 5;
```

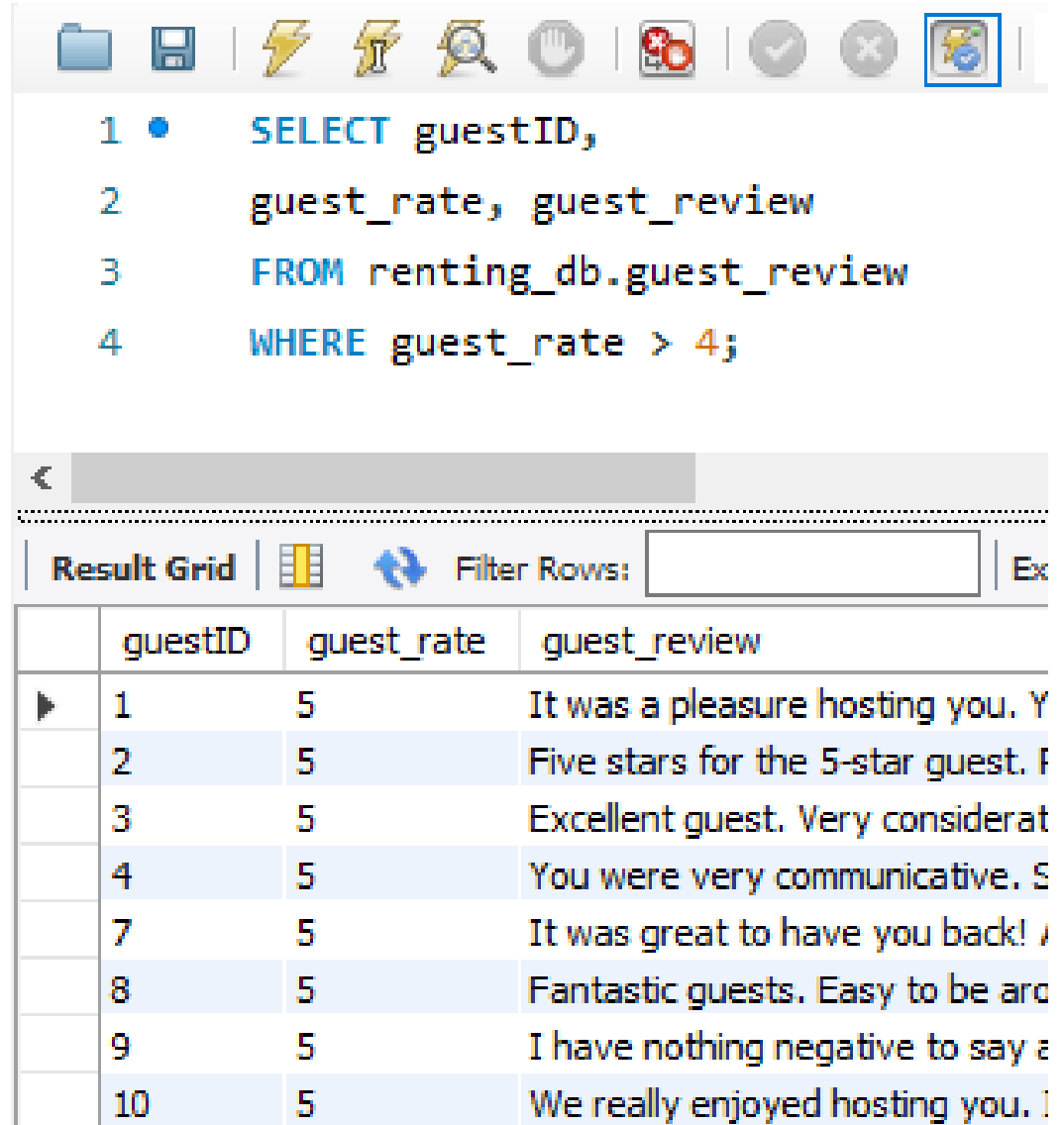
| <   |                               |         |           |          |
|---|-------------------------------|---------|-----------|----------|
| Result Grid     Filter Rows: <input type="text"/>   Export:  |                               |         |           |          |
|   | property_name                 | service | amenities | location |
| ▶   | Black Lake Cabin              | 5       | 5         | 5        |
|   | The Red Bungalow of Deerfield | 5       | 5         | 5        |
|   | The Brooklyn Penthouse        | 5       | 5         | 5        |

# guest\_review

Creating guest\_review table:

```
CREATE TABLE guest_review  
(guest_reviewID INTEGER PRIMARY  
KEY AUTO_INCREMENT, hostID  
INTEGER, guestID  
INTEGER, guest_rate  
INTEGER, guest_review  
TEXT, FOREIGN KEY (hostID)  
REFERENCES host (hostID), FOREIGN  
KEY (guestID) REFERENCES guest  
(guestID), updated_at DATETIME  
DEFAULT now());
```

## SQL query



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and search. The query is as follows:

```
1 • SELECT guestID,  
2     guest_rate, guest_review  
3     FROM renting_db.guest_review  
4     WHERE guest_rate > 4;
```

Below the query editor, the results are displayed in a table. The table has a header row with columns: guestID, guest\_rate, and guest\_review. There are 10 rows of data, all with a guest\_rate of 5. The first row is highlighted with a blue background.

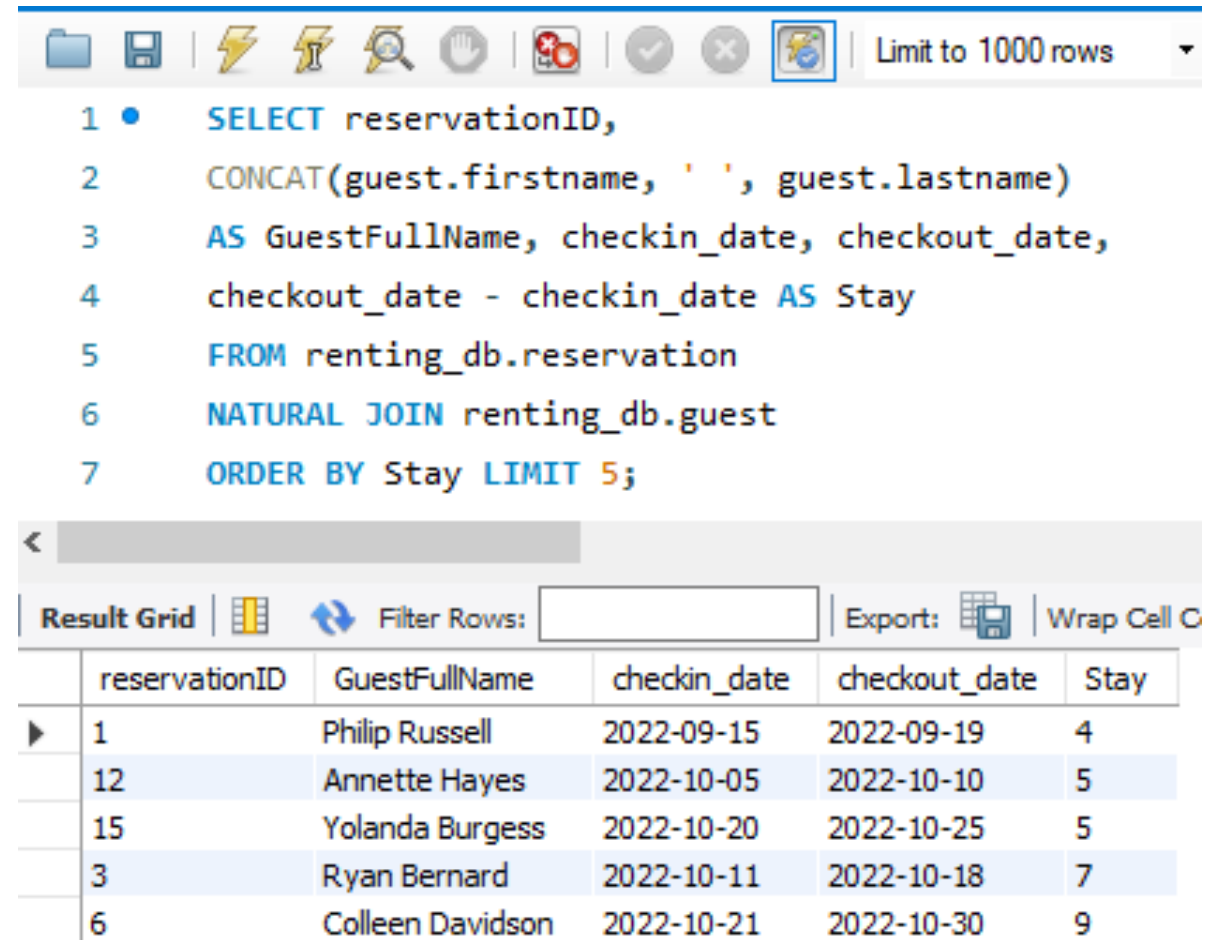
|   | guestID | guest_rate | guest_review                       |
|---|---------|------------|------------------------------------|
| ▶ | 1       | 5          | It was a pleasure hosting you. Y   |
|   | 2       | 5          | Five stars for the 5-star guest. F |
|   | 3       | 5          | Excellent guest. Very considerat   |
|   | 4       | 5          | You were very communicative. S     |
|   | 7       | 5          | It was great to have you back! /   |
|   | 8       | 5          | Fantastic guests. Easy to be arc   |
|   | 9       | 5          | I have nothing negative to say a   |
|   | 10      | 5          | We really enjoyed hosting you. I   |

# reservation

## Creating reservation table:

```
CREATE TABLE reservation
(reservationID INTEGER PRIMARY KEY
AUTO_INCREMENT,roomID INTEGER, guestID
INTEGER,hostID INTEGER,cancelationID
INTEGER,reservation_date
DATETIME,checkin_date DATE,checkout_date
DATE,free_cancelation_date DATE,FOREIGN
KEY (roomID) REFERENCES room
(roomID),FOREIGN KEY (guestID) REFERENCES
guest (guestID),FOREIGN KEY (hostID)
REFERENCES host (hostID),FOREIGN KEY
(cancelationID) REFERENCES cancelation
(cancelationID),CHECK (reservation_date <=
free_cancelation_date <= checkin_date <
checkout_date),updated_at DATETIME
DEFAULT now());
```

## SQL query



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and search. The query text is as follows:

```
1 • SELECT reservationID,
2   CONCAT(guest.firstname, ' ', guest.lastname)
3   AS GuestFullName, checkin_date, checkout_date,
4   checkout_date - checkin_date AS Stay
5   FROM renting_db.reservation
6   NATURAL JOIN renting_db.guest
7   ORDER BY Stay LIMIT 5;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query in a table format. The table has six columns: reservationID, GuestFullName, checkin\_date, checkout\_date, and Stay. The results are limited to 5 rows.

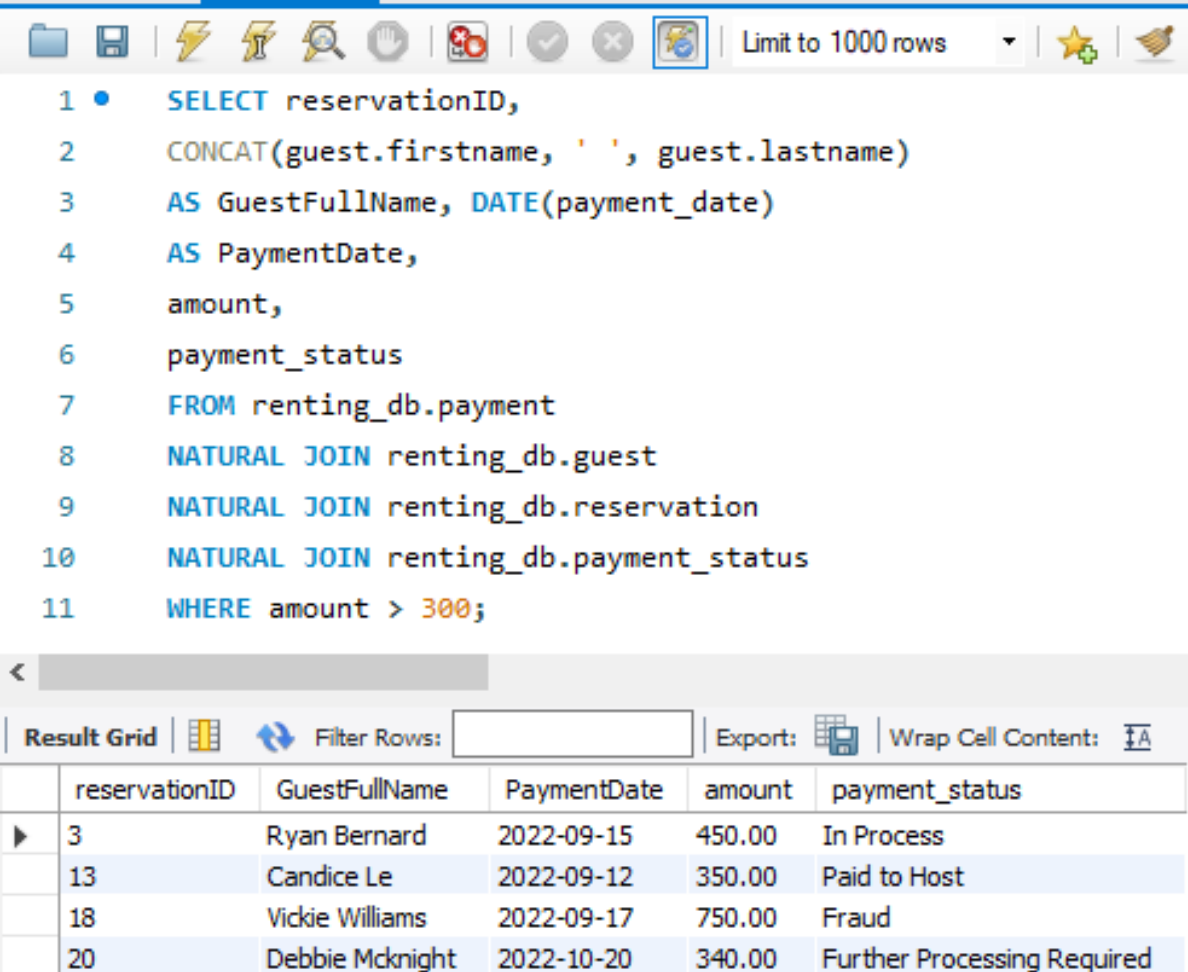
|   | reservationID | GuestFullName    | checkin_date | checkout_date | Stay |
|---|---------------|------------------|--------------|---------------|------|
| ▶ | 1             | Philip Russell   | 2022-09-15   | 2022-09-19    | 4    |
|   | 12            | Annette Hayes    | 2022-10-05   | 2022-10-10    | 5    |
|   | 15            | Yolanda Burgess  | 2022-10-20   | 2022-10-25    | 5    |
|   | 3             | Ryan Bernard     | 2022-10-11   | 2022-10-18    | 7    |
|   | 6             | Colleen Davidson | 2022-10-21   | 2022-10-30    | 9    |

# payment

## Creating payment table:

```
CREATE TABLE payment
(paymentID INTEGER PRIMARY KEY
AUTO_INCREMENT, payment_statusID
INTEGER, reservationID INTEGER,
voucherID INTEGER, amount
DECIMAL(8,2), payment_date DATETIME,
FOREIGN KEY (payment_statusID)
REFERENCES payment_status
(payment_statusID), FOREIGN KEY
(reservationID) REFERENCES reservation
(reservationID), FOREIGN KEY (voucherID)
REFERENCES voucher
(voucherID), updated_at DATETIME
DEFAULT now() );
```

## SQL query



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT reservationID,
2   CONCAT(guest.firstname, ' ', guest.lastname)
3   AS GuestFullName, DATE(payment_date)
4   AS PaymentDate,
5   amount,
6   payment_status
7   FROM renting_db.payment
8   NATURAL JOIN renting_db.guest
9   NATURAL JOIN renting_db.reservation
10  NATURAL JOIN renting_db.payment_status
11  WHERE amount > 300;
```

Below the query, the results are displayed in a table with the following columns: reservationID, GuestFullName, PaymentDate, amount, and payment\_status. The results show four rows of data.

|   | reservationID | GuestFullName   | PaymentDate | amount | payment_status              |
|---|---------------|-----------------|-------------|--------|-----------------------------|
| ▶ | 3             | Ryan Bernard    | 2022-09-15  | 450.00 | In Process                  |
|   | 13            | Candice Le      | 2022-09-12  | 350.00 | Paid to Host                |
|   | 18            | Vickie Williams | 2022-09-17  | 750.00 | Fraud                       |
|   | 20            | Debbie Mcknight | 2022-10-20  | 340.00 | Further Processing Required |