

# **HOTEL DATABASE MANAGEMENT SYSTEM**

## **FINAL PROJECT REPORT**

**Team Professor:** Abdullah Alenezi

**Team Members(Group 8) :**

Sravani Reddy Meda

Namratha Reddy Regalla

Sai Sruthi Thileti

Joshnavi Chenreddy

Ashwitha Eravelly

## **DATABASE OVERVIEW:**

### **OBJECTIVE:**

The Hotel Database Management System aims to provide a fully functional and offline automated system, with some online capabilities. Its primary objective is to offer a reliable and well-organized system with minimal errors. The system seeks customer feedback to demonstrate the hotel's commitment to its clients.

Customers have the convenience of creating an account from the comfort of their homes and logging in for their intended purposes. A centralized database has been implemented to address the challenges and flaws associated with the current manual approach while ensuring secure record-keeping. Executives can access information about room availability by logging in with their unique IDs.

The main goal of the Hotel Database Management System is to design and maintain comprehensive records of users and room details, including hotel bookings.

### **SCOPE:**

There are several reasons to consider implementing a new computer-based hotel management system. One of the key advantages of the proposed solution is the elimination of laborious tasks for both customers and hotel executives when searching for and booking hotel rooms. The system will streamline these processes, making them more efficient.

The suggested solution will also benefit administrative personnel or hotel executives by providing accurate databases for maintaining daily and historical records of clients. Additionally, the system will generate appropriate reports for customers upon check-out. Despite its advanced capabilities, the system is designed to be user-friendly, even for new users.

Once implemented, the hotel management system will enable remote access to the hotel database, allowing authorized individuals to retrieve information and perform various functions from anywhere. Customers will have the ability to search for available rooms, make reservations, and provide feedback or reviews on their experiences.

## **USER REQUIREMENTS:**

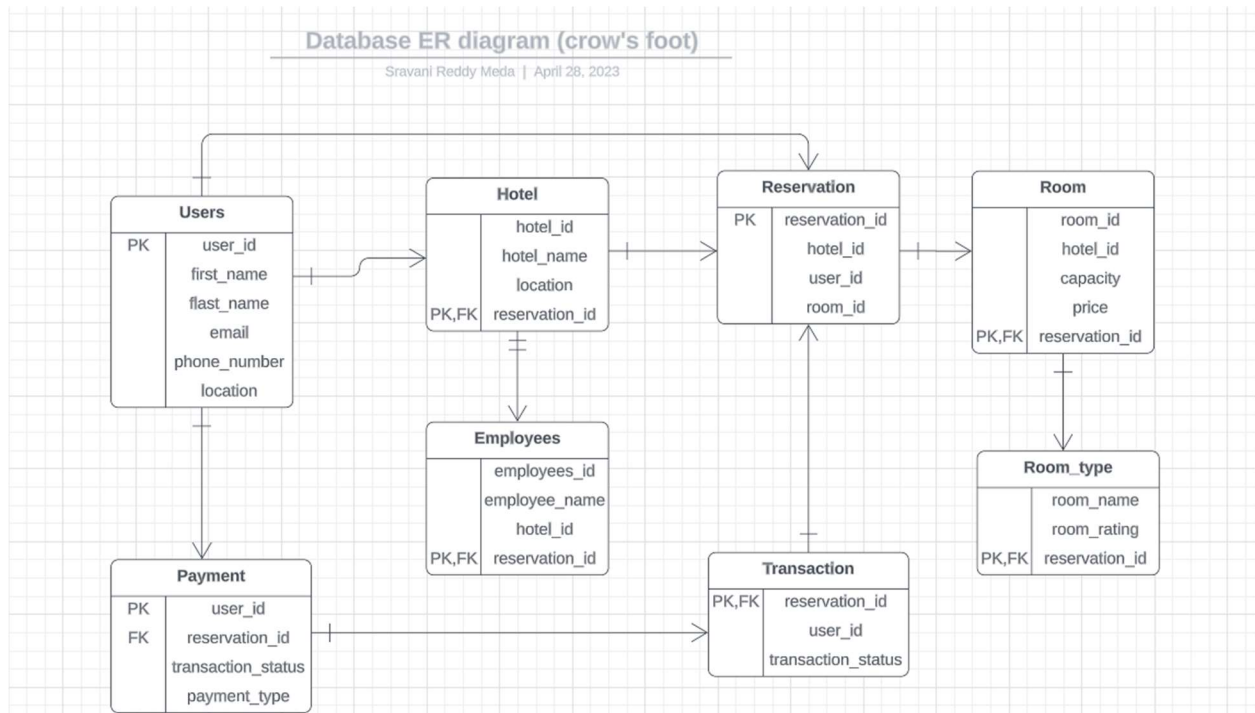
The hotel management system faces significant challenges when it comes to retrieving customer booking data. The system struggles to access and display all the necessary customer details along with their reservation information. As a result, it becomes difficult to track bookings, reservations, and guest information effectively. This hinders the hotel's ability to manage inventory and pricing efficiently.

Implementing a robust database can address these issues by providing a comprehensive view of customer records and their corresponding reservation details. With improved access to data, the hotel management system can track bookings more accurately, manage reservations effectively, and maintain updated guest information. Additionally, the database can offer valuable insights into customer trends and behaviors, enabling the hotel to make informed decisions and improve customer satisfaction.

## **BUSINESS RULES:**

1. Each customer has the option to make multiple bookings.
2. A single customer can only make one reservation at a time.
3. Employees have the flexibility to enter zero or more bookings.
4. Each reservation can only be entered by one employee.
5. Customers are allowed to make one or more payments.
6. Only one customer can make a payment at a given time.
7. Each reservation for a room can have one and only one associated payment.
8. A payment can be linked to one or more reservations for rooms.
9. Each reservation is assigned to a specific room.
10. A room can be part of one or more bookings but cannot be assigned to multiple room reservations simultaneously.
11. Room reservations can only be entered into the system by designated staff.

## ENTITY RELATIONSHIP DIAGRAM:



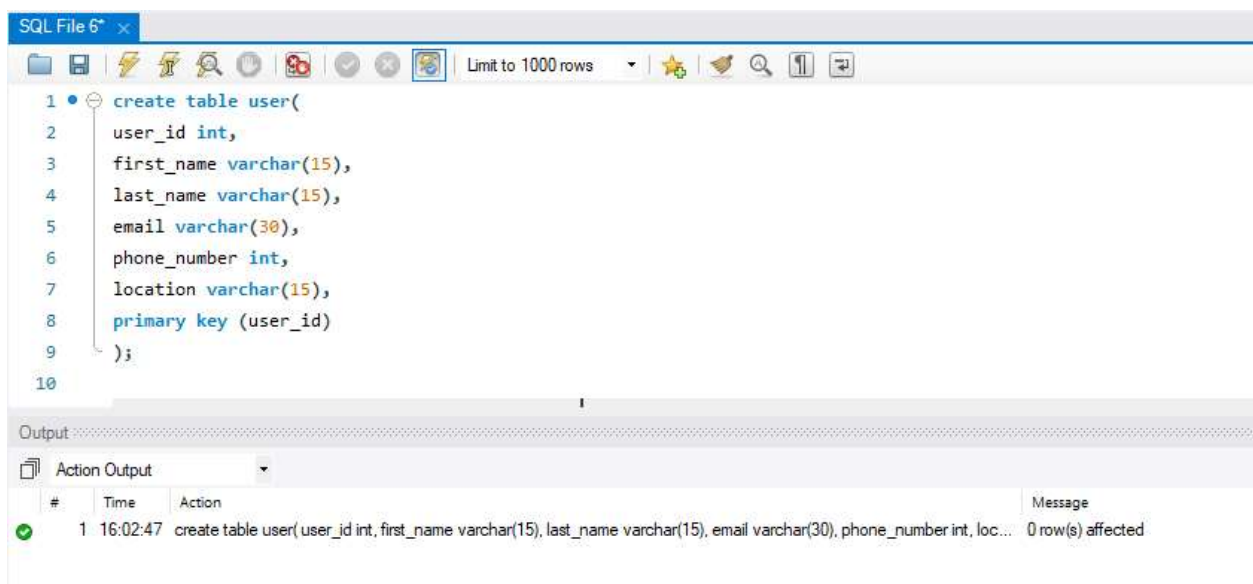
## DATA DICTIONARY:

TABLE NAME	ATTRIBUTE NAME	DATA TYPE	PRIMARY KEY REQUIRED	FOREIGN KEY REQUIRED
USERS	user_id first_name last_name email phone_number location	int varchar varchar Varchar int varchar	user_id	
RESERVATION	reservation_id hotel_id user_id room_id	int int int int	reservation_id	user id
room	room_id hotel_id Capacity Price reservation_id	Int Int Int Double Int		reservation_id



### Creating “User” table:

```
create table user(  
  user_id int,  
  first_name varchar(15),  
  last_name varchar(15),  
  email varchar(30),  
  phone_number int,  
  location varchar(15),  
  primary key (user_id)  
);
```



### Inserting Values Into “User” table:

Insert into user value(1001,'James','Robert','jRobert3@gmail.com',2345678,'Austin');

Insert into user value(1002,'Swetha','Goli','gswetha@gmail.com',1234567,'Austin');

Insert into user value(1003,'Liki','tha','likitha@gmail.com',0987654,'Arlington');

Insert into user value(1004,'Bindu','hh','Bindu@gmail.com',76654097,'Houston');

Insert into user value(1005,'John','Michael','John@gmail.com',8568946,'Irving');

Insert into user value(1006,'David','William','William@gmail.com',2309764,'cleveland');  
 Insert into user value(1007,'Richard','Barbara','Barbara@gmail.com',5568790,'Irving');  
 Insert into user value(1008,'Anabella','Susan','Susan@gmail.com',7468536,'Coppel');  
 Insert into user value(1009,'Jessica','Sarah','Sarah@gmail.com',6479075,'Coppel');  
 Insert into user value(1010,'Margaret','Donaldson','Donaldson@gmail.com',8753689,'Houston');  
 Insert into user value(1011,'Sandra','Mark','V22Sandra@gmail.com',3399775,'Houston');  
 Insert into user value(1012,'Ashley','Steven','Steven@gmail.com',3300998,'San Antonio');  
 Insert into user value(1013,'Amanda','Brian','Brian@gmail.com',9468027,'Laredo');  
 Insert into user value(1014,'Jeffrey','Laura','Laura@gmail.com',4498765,'Laredo');  
 Insert into user value(1015,'Timothy','George','Timothy@gmail.com',0685356,'Laredo');

SQL File 6\*

Limit to 1000 rows

```
16 Insert into user value(1006,'David','William','William@gmail.com',2309764,'cleveland');
17 Insert into user value(1007,'Richard','Barbara','Barbara@gmail.com',5568790,'Irving');
18 Insert into user value(1008,'Anabella','Susan','Susan@gmail.com',7468536,'Coppel');
19 Insert into user value(1009,'Jessica','Sarah','Sarah@gmail.com',6479075,'Coppel');
20 Insert into user value(1010,'Margaret','Donaldson','Donaldson@gmail.com',8753689,'Houston');
21 Insert into user value(1011,'Sandra','Mark','V22Sandra@gmail.com',3399775,'Houston');
22 Insert into user value(1012,'Ashley','Steven','Steven@gmail.com',3300998,'San Antonio');
23 Insert into user value(1013,'Amanda','Brian','Brian@gmail.com',9468027,'Laredo');
24 Insert into user value(1014,'Jeffrey','Laura','Laura@gmail.com',4498765,'Laredo');
```

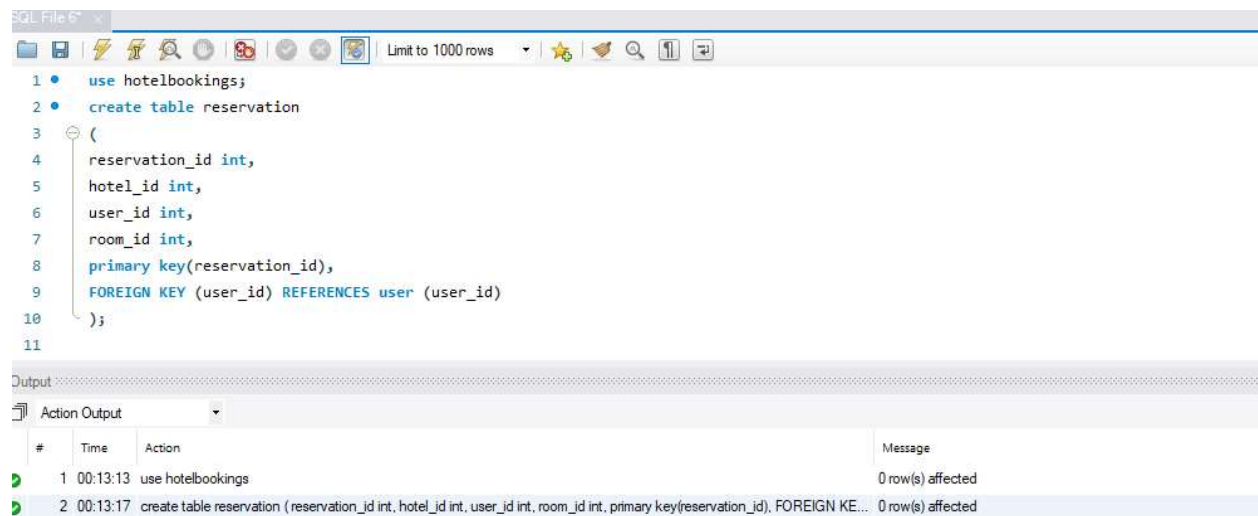
Output

Action Output

#	Time	Action	Message
3	16:03:38	Insert into user value(1002,'Swetha','Goli','gswetha@gmail.com',1234567,'Austin')	1 row(s) affected
4	16:03:41	Insert into user value(1003,'Liki','tha','likitha@gmail.com',0987654,'Arlington')	1 row(s) affected
5	16:03:43	Insert into user value(1004,'Bindu','hh','Bindu@gmail.com',76654097,'Houston')	1 row(s) affected
6	16:03:46	Insert into user value(1005,'John','Michael','John@gmail.com',8568946,'Irving')	1 row(s) affected
7	16:03:49	Insert into user value(1006,'David','William','William@gmail.com',2309764,'cleveland')	1 row(s) affected
8	16:03:51	Insert into user value(1007,'Richard','Barbara','Barbara@gmail.com',5568790,'Irving')	1 row(s) affected
9	16:03:54	Insert into user value(1008,'Anabella','Susan','Susan@gmail.com',7468536,'Coppel')	1 row(s) affected
10	16:03:58	Insert into user value(1009,'Jessica','Sarah','Sarah@gmail.com',6479075,'Coppel')	1 row(s) affected
11	16:04:00	Insert into user value(1010,'Margaret','Donaldson','Donaldson@gmail.com',8753689,'Houston')	1 row(s) affected
12	16:04:08	Insert into user value(1011,'Sandra','Mark','V22Sandra@gmail.com',3399775,'Houston')	1 row(s) affected
13	16:04:10	Insert into user value(1012,'Ashley','Steven','Steven@gmail.com',3300998,'San Antonio')	1 row(s) affected
14	16:04:13	Insert into user value(1013,'Amanda','Brian','Brian@gmail.com',9468027,'Laredo')	1 row(s) affected

## Creating “Reservation” table:

```
create table reservation  
(  
    reservation_id int,  
    hotel_id int,  
    user_id int,  
    room_id int,  
    primary key(reservation_id),  
    FOREIGN KEY (user_id) REFERENCES user (user_id)  
);
```



The screenshot shows a SQL IDE window titled "SQL File 6" with a toolbar and a "Limit to 1000 rows" dropdown. The SQL editor contains the following code:

```
1 • use hotelbookings;  
2 • create table reservation  
3 • (  
4 •     reservation_id int,  
5 •     hotel_id int,  
6 •     user_id int,  
7 •     room_id int,  
8 •     primary key(reservation_id),  
9 •     FOREIGN KEY (user_id) REFERENCES user (user_id)  
10 • );  
11
```

Below the editor is the "Output" panel, which is set to "Action Output". It displays a table with columns "#", "Time", "Action", and "Message".

#	Time	Action	Message
1	00:13:13	use hotelbookings	0 row(s) affected
2	00:13:17	create table reservation ( reservation_id int, hotel_id int, user_id int, room_id int, primary key(reservation_id), FOREIGN KE...	0 row(s) affected

## Inserting values into “Reservation” table:

Insert into reservation value(1789,888,1001,171);

Insert into reservation value(2637,888,1002,172);

Insert into reservation value(3227,888,1003,173);

Insert into reservation value(4778,888,1004,174);

Insert into reservation value(5222,888,1005,175);



Insert into reservation value(6082,888,1006,176);

Insert into reservation value(7752,888,1007,177);

Insert into reservation value(8767,888,1008,178);

Insert into reservation value(9279,888,1009,179);

Insert into reservation value(1089,888,1010,180);

Insert into reservation value(1112,888,1011,181);

Insert into reservation value(1211,888,1012,182);

Insert into reservation value(1397,888,1013,183);

Insert into reservation value(1467,888,1014,184);

Insert into reservation value(1513,888,1015,185);

The screenshot shows a SQL IDE window titled "SQL File 6" with a toolbar and a script containing 12 INSERT statements. Below the script is an "Output" window showing the execution results of the first seven statements.

SQL Script:

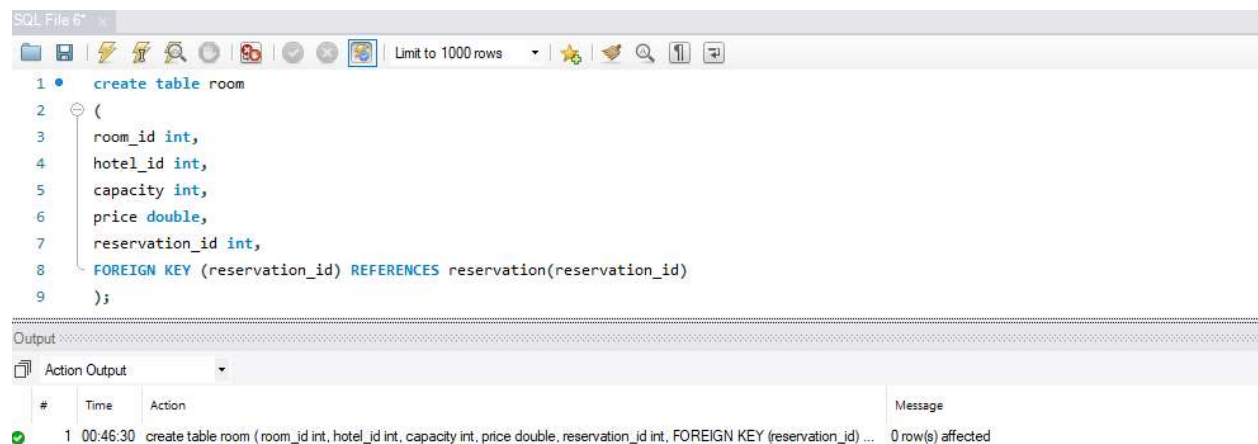
```
12 • Insert into reservation value(1789,888,1001,171);
13 • Insert into reservation value(2637,888,1002,172);
14 • Insert into reservation value(3227,888,1003,173);
15 • Insert into reservation value(4778,888,1004,174);
16 • Insert into reservation value(5222,888,1005,175);
17 • Insert into reservation value(6082,888,1006,176);
18 • Insert into reservation value(7752,888,1007,177);
19 • Insert into reservation value(8767,888,1008,178);
20 • Insert into reservation value(9279,888,1009,179);
21 • Insert into reservation value(1089,888,1010,180);
22 • Insert into reservation value(1112,888,1011,181);
23 • Insert into reservation value(1211,888,1012,182);
24 • Insert into reservation value(1397,888,1013,183);
25 • Insert into reservation value(1467,888,1014,184);
```

Output Window:

#	Time	Action	Message
1	00:30:09	create table reservation ( reservation_id int, hotel_id int, user_id int, room_id int, primary key(reservation_id), FOREIGN ...	0 row(s) affected
2	00:30:50	Insert into reservation value(1789,888,1001,171)	1 row(s) affected
3	00:30:53	Insert into reservation value(2637,888,1002,172)	1 row(s) affected
4	00:30:55	Insert into reservation value(3227,888,1003,173)	1 row(s) affected
5	00:31:01	Insert into reservation value(4778,888,1004,174)	1 row(s) affected
6	00:31:04	Insert into reservation value(5222,888,1005,175)	1 row(s) affected
7	00:31:08	Insert into reservation value(6082,888,1006,176)	1 row(s) affected

## Creating “room” table:

```
create table room  
(  
    room_id int,  
    hotel_id int,  
    capacity int,  
    price double,  
    reservation_id int,  
    FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)  
);
```



## Inserting values into “room” table:

Insert into room value(171,888,2,1000,1789);

Insert into room value(172,888,2,1000,2637);

Insert into room value(173,888,2,1000,3227);

Insert into room value(174,888,2,1000,4778);

Insert into room value(175,888,2,1000,5222);

Insert into room value(176,888,2,1000,6082);

Insert into room value(177,888,3,1500,7752);

Insert into room value(178,888,3,1500,8767);

Insert into room value(179,888,3,1500,9279);

Insert into room value(180,888,3,1500,1089);

Insert into room value(181,888,3,1500,1112);

Insert into room value(182,888,1,500,1211);

Insert into room value(183,888,1,500,1397);

Insert into room value(184,888,1,500,1467);

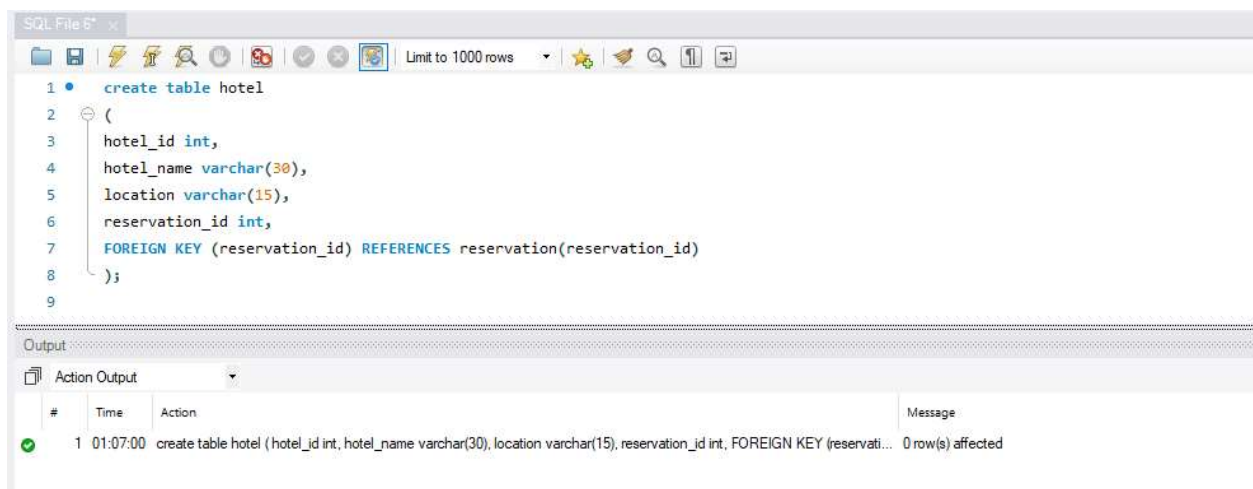
Insert into room value(185,888,1,500,1513);

The screenshot shows a SQL IDE window titled "SQL File 6" with a toolbar and a list of 13 SQL statements. The statements are numbered 1 through 13. The first 10 statements are "Insert into room value(...)" with various parameters. The last two statements are "Insert into room value(182,888,1,500,1211);". Below the statements is an "Output" window with a tab labeled "Action Output". The output window displays a table with 4 columns: #, Time, Action, and Message. It shows 10 rows of execution results, each with a green checkmark icon, a number, a time, the action, and the message "1 row(s) affected".

#	Time	Action	Message
3	00:54:42	Insert into room value(172,888,2,1000,2637)	1 row(s) affected
4	00:54:44	Insert into room value(173,888,2,1000,3227)	1 row(s) affected
5	00:54:47	Insert into room value(174,888,2,1000,4778)	1 row(s) affected
6	00:54:56	Insert into room value(175,888,2,1000,5222)	1 row(s) affected
7	00:54:59	Insert into room value(176,888,2,1000,6082)	1 row(s) affected
8	00:55:02	Insert into room value(177,888,3,1500,7752)	1 row(s) affected
9	00:55:04	Insert into room value(178,888,3,1500,8767)	1 row(s) affected
10	00:55:07	Insert into room value(179,888,3,1500,9279)	1 row(s) affected

## Creating “Hotel” table:

```
create table hotel  
(  
    hotel_id int,  
    hotel_name varchar(30),  
    location varchar(15),  
    reservation_id int,  
    FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)  
);
```



## Inserting values into “Hotel” table:

Insert into hotel value(888,'Brown Town','Irving',1789);

Insert into hotel value(888,'Brown Town','Irving',2637);

Insert into hotel value(888,'Brown Town','Irving',3227);

Insert into hotel value(888,'Brown Town','Irving',4778);

Insert into hotel value(888,'Brown Town','Irving',5222);

Insert into hotel value(888,'Brown Town','Irving',6082);

Insert into hotel value(888,'Brown Town','Irving',7752);

Insert into hotel value(888,'Brown Town','Irving',8767);

Insert into hotel value(888,'Brown Town','Irving',9279);

Insert into hotel value(888,'Brown Town','Irving',1089);

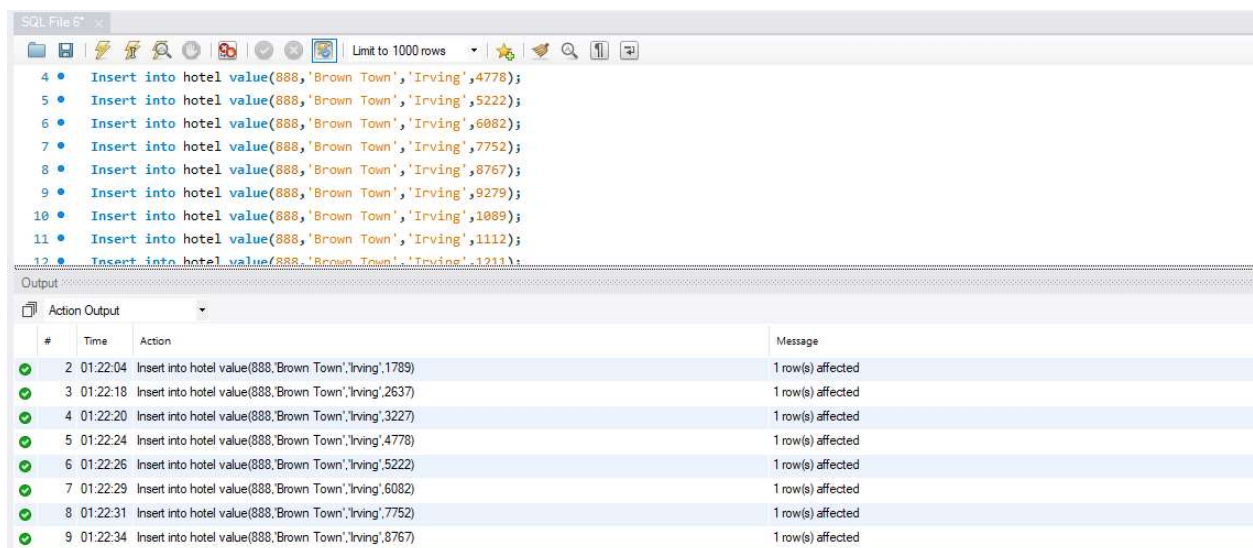
Insert into hotel value(888,'Brown Town','Irving',1112);

Insert into hotel value(888,'Brown Town','Irving',1211);

Insert into hotel value(888,'Brown Town','Irving',1397);

Insert into hotel value(888,'Brown Town','Irving',1467);

Insert into hotel value(888,'Brown Town','Irving',1513);



#	Time	Action	Message
2	01:22:04	Insert into hotel value(888,'Brown Town','Irving',1789)	1 row(s) affected
3	01:22:18	Insert into hotel value(888,'Brown Town','Irving',2637)	1 row(s) affected
4	01:22:20	Insert into hotel value(888,'Brown Town','Irving',3227)	1 row(s) affected
5	01:22:24	Insert into hotel value(888,'Brown Town','Irving',4778)	1 row(s) affected
6	01:22:26	Insert into hotel value(888,'Brown Town','Irving',5222)	1 row(s) affected
7	01:22:29	Insert into hotel value(888,'Brown Town','Irving',6082)	1 row(s) affected
8	01:22:31	Insert into hotel value(888,'Brown Town','Irving',7752)	1 row(s) affected
9	01:22:34	Insert into hotel value(888,'Brown Town','Irving',8767)	1 row(s) affected

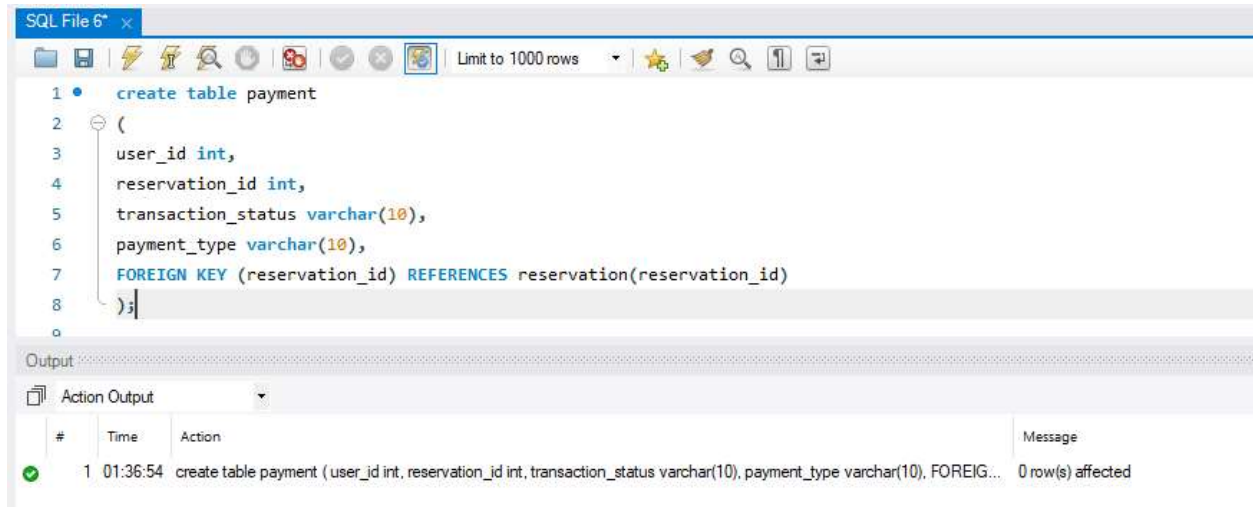
## Creating “Payment” table:

create table payment

(

user\_id int,

```
reservation_id int,  
transaction_status varchar(10),  
payment_type varchar(10),  
FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)  
);
```



### Inserting values into “Payment” table:

Insert into payment value(1789,1001,'success','debit card');

Insert into payment value(2637,1002,'success','debit card');

Insert into payment value(3227,1003,'success','credit card');

Insert into payment value(4778,1004,'success','google pay');

Insert into payment value(5222,1005,'success','cash');

Insert into payment value(6082,1006,'fail','credit card');

Insert into payment value(7752,1007,'fail','debit card');

Insert into payment value(8767,1008,'fail','paypal');

Insert into payment value(9279,1009,'success','apple pay');

Insert into payment value(1089,1010,'success','debit card');

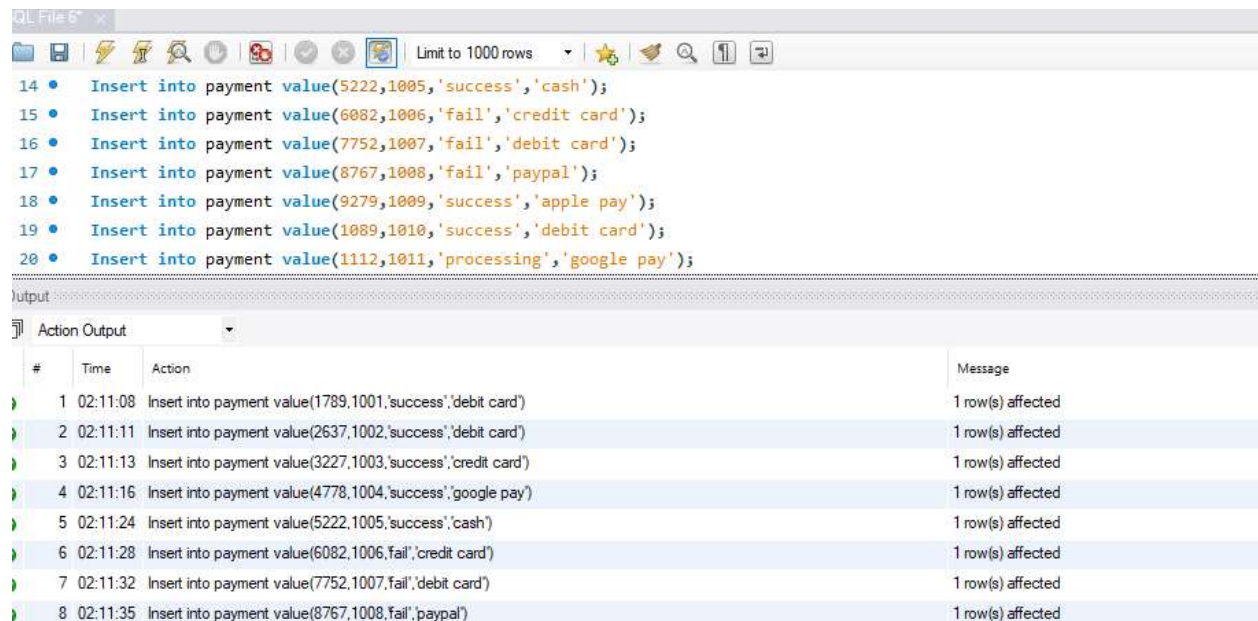
Insert into payment value(1112,1011,'processing','google pay');

Insert into payment value(1211,1012,'processing','debit card');

Insert into payment value(1397,1013,'fail','apple pay');

Insert into payment value(1467,1014,'fail','google pay');

Insert into payment value(1513,1015,'success','google pay');



The screenshot shows a SQL client window titled "QL File 6" with a toolbar and a list of 20 SQL insert statements. Below the statements is an "Output" section with a tab labeled "Action Output". This tab displays a table with 4 columns: "#", "Time", "Action", and "Message". The table contains 8 rows of execution results, each showing a successful insert operation with a timestamp and a message indicating "1 row(s) affected".

#	Time	Action	Message
1	02:11:08	Insert into payment value(1789,1001,'success','debit card')	1 row(s) affected
2	02:11:11	Insert into payment value(2637,1002,'success','debit card')	1 row(s) affected
3	02:11:13	Insert into payment value(3227,1003,'success','credit card')	1 row(s) affected
4	02:11:16	Insert into payment value(4778,1004,'success','google pay')	1 row(s) affected
5	02:11:24	Insert into payment value(5222,1005,'success','cash')	1 row(s) affected
6	02:11:28	Insert into payment value(6082,1006,'fail','credit card')	1 row(s) affected
7	02:11:32	Insert into payment value(7752,1007,'fail','debit card')	1 row(s) affected
8	02:11:35	Insert into payment value(8767,1008,'fail','paypal')	1 row(s) affected

## Creating “Transaction” table:

create table transaction

(

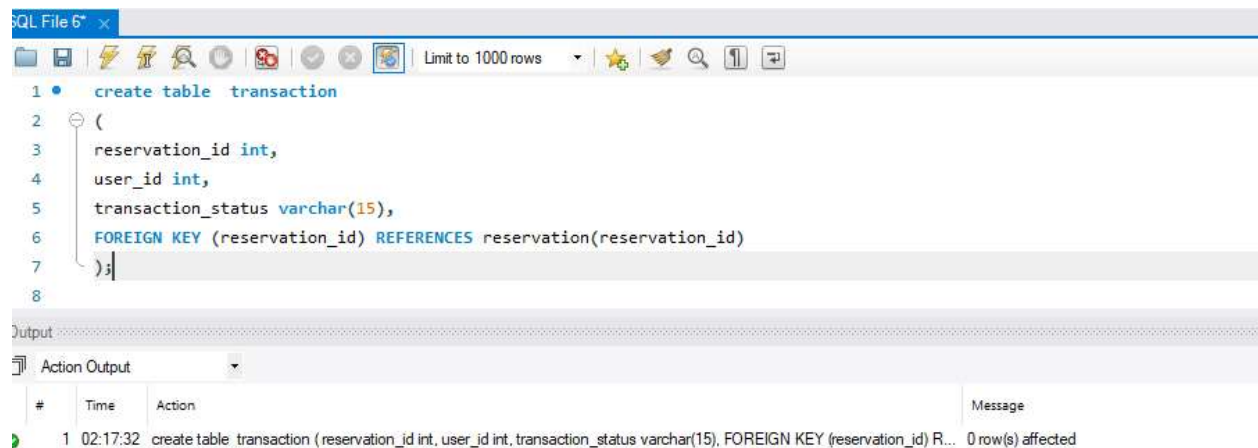
reservation\_id int,

user\_id int,

transaction\_status varchar(15),

FOREIGN KEY (reservation\_id) REFERENCES reservation(reservation\_id)

);



The screenshot shows a SQL editor window titled "SQL File 6" with a toolbar and a list of icons. The SQL code is as follows:

```
1 • create table transaction
2 (
3     reservation_id int,
4     user_id int,
5     transaction_status varchar(15),
6     FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)
7 )
8
```

Below the code editor is an "Output" section with a dropdown menu set to "Action Output". It displays a single row of execution results:

#	Time	Action	Message
1	02:17:32	create table transaction ( reservation_id int, user_id int, transaction_status varchar(15), FOREIGN KEY (reservation_id) R...	0 row(s) affected

### Inserting values into “Transaction” table:

Insert into transaction value(1789,1001,'success');

Insert into transaction value(2637,1002,'success');

Insert into transaction value(3227,1003,'success');

Insert into transaction value(4778,1004,'success');

Insert into transaction value(5222,1005,'success');

Insert into transaction value(6082,1006,'fail');

Insert into transaction value(7752,1007,'fail');

Insert into transaction value(8767,1008,'fail');

Insert into transaction value(9279,1009,'success');

Insert into transaction value(1089,1010,'success');

Insert into transaction value(1112,1011,'processing');

Insert into transaction value(1211,1012,'processing');

Insert into transaction value(1397,1013,'fail');

Insert into transaction value(1467,1014,'fail');



Insert into transaction value(1513,1015,'success');

The screenshot shows a SQL IDE window titled "SQL File 6". The main editor contains a list of 19 SQL statements, each preceded by a blue bullet point. The statements are: 11. Insert into transaction value(4778,1004,'success'); 12. Insert into transaction value(5222,1005,'success'); 13. Insert into transaction value(6082,1006,'fail'); 14. Insert into transaction value(7752,1007,'fail'); 15. Insert into transaction value(8767,1008,'fail'); 16. Insert into transaction value(9279,1009,'success'); 17. Insert into transaction value(1089,1010,'success'); 18. Insert into transaction value(1112,1011,'processing'); 19. Insert into transaction value(1211,1012,'processing');. Below the editor is an "Output" pane with a tab labeled "Action Output". It displays a table with 4 columns: #, Time, Action, and Message. The table contains 9 rows of data, corresponding to the first 9 statements in the editor. Each row has a green checkmark in the # column, a timestamp in the Time column, the SQL statement in the Action column, and the message "1 row(s) affected" in the Message column.

#	Time	Action	Message
1	02:20:17	Insert into transaction value(1789,1001,'success')	1 row(s) affected
2	02:20:48	Insert into transaction value(2637,1002,'success')	1 row(s) affected
3	02:20:51	Insert into transaction value(3227,1003,'success')	1 row(s) affected
4	02:20:53	Insert into transaction value(4778,1004,'success')	1 row(s) affected
5	02:20:56	Insert into transaction value(5222,1005,'success')	1 row(s) affected
6	02:20:59	Insert into transaction value(6082,1006,'fail')	1 row(s) affected
7	02:21:01	Insert into transaction value(7752,1007,'fail')	1 row(s) affected
8	02:21:05	Insert into transaction value(8767,1008,'fail')	1 row(s) affected
9	02:21:07	Insert into transaction value(9279,1009,'success')	1 row(s) affected

## Creating “Employees” table:

create table employees

(

employee\_id int,

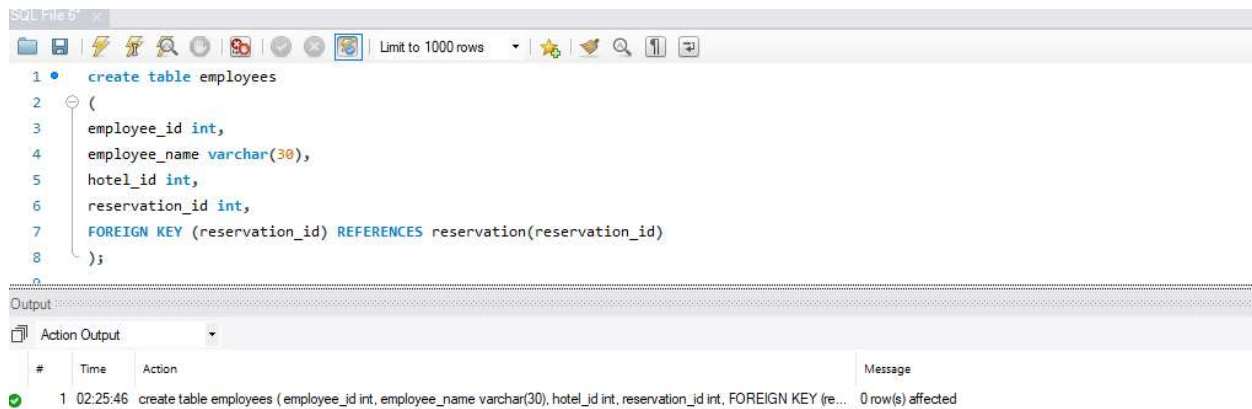
employee\_name varchar(30),

hotel\_id int,

reservation\_id int,

FOREIGN KEY (reservation\_id) REFERENCES reservation(reservation\_id)

);



```
1 • create table employees
2 (
3     employee_id int,
4     employee_name varchar(30),
5     hotel_id int,
6     reservation_id int,
7     FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)
8 );
```

Output

Action Output

#	Time	Action	Message
1	02:25:46	create table employees ( employee_id int, employee_name varchar(30), hotel_id int, reservation_id int, FOREIGN KEY (re...	0 row(s) affected

## Inserting the values into “Employees” table:

Insert into employees value(001,'jayanth',888,1789);

Insert into employees value(002,'yashwanth',888,2637);

Insert into employees value(003,'david',888,3227);

Insert into employees value(004,'varun',888,4778);

Insert into employees value(005,'koushik',888,5222);

Insert into employees value(006,'taylor',888,6082);

Insert into employees value(007,'Caralon',888,7752);

Insert into employees value(008,'Likitha',888,8767);

Insert into employees value(009,'Kavya',888,9279);

Insert into employees value(010,'Terrisa',888,1089);

Insert into employees value(011,'Bindu',888,1112);

Insert into employees value(012,'Satwika',888,1211);

Insert into employees value(013,'Varsha',888,1397);

Insert into employees value(014,'Swetha',888,1467);

Insert into employees value(015,'Namratha',888,1513);

The screenshot shows a SQL IDE window titled "SQL File 6\*". The main editor contains 11 INSERT statements for an "employees" table. The statements are numbered 14 through 24. Below the editor, the "Output" pane shows the "Action Output" for these statements. Each statement was executed successfully, resulting in "1 row(s) affected".

```

14 • Insert into employees value(004,'varun',888,4778);
15 • Insert into employees value(005,'koushik',888,5222);
16 • Insert into employees value(006,'taylor',888,6082);
17 • Insert into employees value(007,'Caralon',888,7752);
18 • Insert into employees value(008,'Likitha',888,8767);
19 • Insert into employees value(009,'Kavya',888,9279);
20 • Insert into employees value(010,'Terrisa',888,1089);
21 • Insert into employees value(011,'Bindu',888,1112);
22 • Insert into employees value(012,'Satwika',888,1211);
23 • Insert into employees value(013,'Varsha',888,1397);
24 • Insert into employees value(014,'Swetha',888,1467);

```

#	Time	Action	Message
6	02:38:16	Insert into employees value(005,'koushik',888,5222)	1 row(s) affected
7	02:38:19	Insert into employees value(006,'taylor',888,6082)	1 row(s) affected
8	02:38:22	Insert into employees value(007,'Caralon',888,7752)	1 row(s) affected
9	02:38:24	Insert into employees value(008,'Likitha',888,8767)	1 row(s) affected
10	02:38:32	Insert into employees value(009,'Kavya',888,9279)	1 row(s) affected
11	02:38:35	Insert into employees value(010,'Terrisa',888,1089)	1 row(s) affected
12	02:38:37	Insert into employees value(011,'Bindu',888,1112)	1 row(s) affected
13	02:38:40	Insert into employees value(012,'Satwika',888,1211)	1 row(s) affected

## Creating “Room\_type” table:

create table room\_type

(

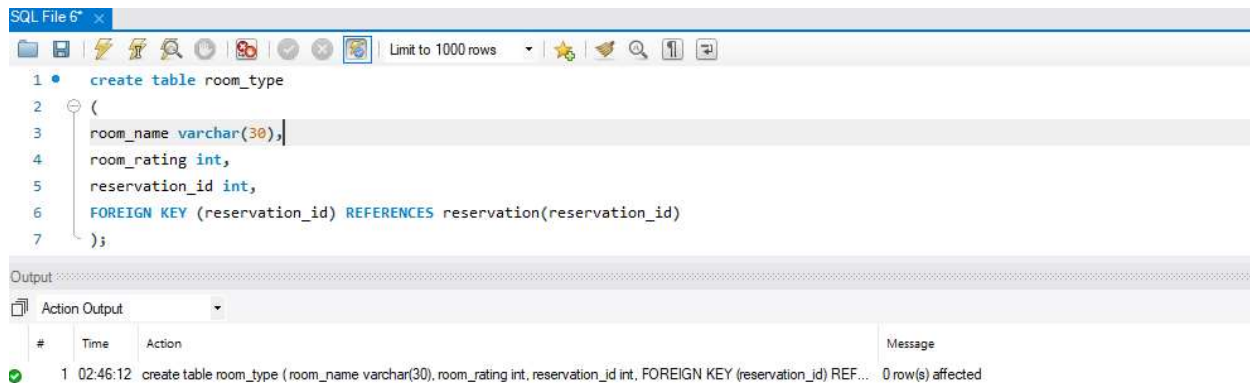
room\_name varchar(30),

room\_rating int,

reservation\_id int,

FOREIGN KEY (reservation\_id) REFERENCES reservation(reservation\_id)

);



The screenshot shows a SQL editor window titled "SQL File 6" with a toolbar and a list of rows. The SQL code is as follows:

```
1 • create table room_type
2 (
3     room_name varchar(30),
4     room_rating int,
5     reservation_id int,
6     FOREIGN KEY (reservation_id) REFERENCES reservation(reservation_id)
7 );
```

Below the editor is an "Output" section with a dropdown menu set to "Action Output". It displays a table with the following data:

#	Time	Action	Message
✓ 1	02:46:12	create table room_type (room_name varchar(30), room_rating int, reservation_id int, FOREIGN KEY (reservation_id) REF...	0 row(s) affected

### Inserting the value into “Room\_type” table:

Insert into room\_type value('deluxe',5,1789);

Insert into room\_type value('deluxe',4,2637);

Insert into room\_type value('deluxe',5,3227);

Insert into room\_type value('deluxe',4,4778);

Insert into room\_type value('vip',5,5222);

Insert into room\_type value('vip',5,6082);

Insert into room\_type value('vip',5,7752);

Insert into room\_type value('vip',5,8767);

Insert into room\_type value('deluxe',5,9279);

Insert into room\_type value('deluxe',5,1089);

Insert into room\_type value('Superior',3,1112);

Insert into room\_type value('Superior',3,1211);

Insert into room\_type value('Superior',4,1397);

Insert into room\_type value('Superior',4,1467);

Insert into room\_type value('Superior',4,1513);

SQL File 6\*

```

9 • Insert into room_type value('deluxe',5,1789);
10 • Insert into room_type value('deluxe',4,2637);
11 • Insert into room_type value('deluxe',5,3227);
12 • Insert into room_type value('deluxe',4,4778);
13 • Insert into room_type value('vip',5,5222);
14 • Insert into room_type value('vip',5,6082);
15 • Insert into room_type value('vip',5,7752);
16 • Insert into room_type value('vip',5,8767);
17 • Insert into room_type value('deluxe',5,9279);
18 • Insert into room_type value('deluxe',5,1089);

```

Output

Action Output

#	Time	Action	Message
5	03:07:41	Insert into room_type value('deluxe',4,4778)	1 row(s) affected
6	03:07:44	Insert into room_type value('vip',5,5222)	1 row(s) affected
7	03:07:48	Insert into room_type value('vip',5,6082)	1 row(s) affected
8	03:07:51	Insert into room_type value('vip',5,7752)	1 row(s) affected
9	03:07:53	Insert into room_type value('vip',5,8767)	1 row(s) affected
10	03:08:01	Insert into room_type value('deluxe',5,9279)	1 row(s) affected
11	03:08:03	Insert into room_type value('deluxe',5,1089)	1 row(s) affected
12	03:08:06	Insert into room_type value('deluxe',5,1112)	1 row(s) affected

## Data Retrieval and Simple Reports

### Select statements

- `SELECT room.*, SUM(price) OVER() AS price FROM room;`

SQL File 6\*

```

1 • SELECT room.*, SUM(price) OVER() AS price FROM room;
2

```

Result Grid

room_id	hotel_id	capacity	price	reservation_id	price
172	888	2	1000	2637	15500
173	888	2	1000	3227	15500
174	888	2	1000	4778	15500
175	888	2	1000	5222	15500
176	888	2	1000	6082	15500
177	888	3	1500	7752	15500
178	888	3	1500	8767	15500
179	888	3	1500	9279	15500
180	888	3	1500	1089	15500
181	888	3	1500	1112	15500
182	888	1	500	1211	15500

- `select * from user where length(phone_number)<=10;`

The screenshot shows a database IDE interface. At the top, a toolbar includes icons for file operations, a search icon, and a 'Limit to 1000 rows' dropdown. Below the toolbar, a SQL editor contains the following query:

```
1
2 • select * from user where length(phone_number)<=10;
```

The 'Result Grid' tab is active, displaying a table with 6 columns: `user_id`, `first_name`, `last_name`, `email`, `phone_number`, and `location`. The table contains 11 rows of data:

user_id	first_name	last_name	email	phone_number	location
1001	James	Robert	jRobert3@gmail.com	2345678	Austin
1002	Swetha	Goli	gswetha@gmail.com	1234567	Austin
1003	Liki	tha	likitha@gmail.com	987654	Arlington
1004	Bindu	hh	Bindu@gmail.com	76654097	Houston
1005	John	Michael	John@gmail.com	8568946	Irving
1006	David	William	William@gmail.com	2309764	devland
1007	Richard	Barbara	Barbara@gmail.com	5568790	Irving
1008	Anabella	Susan	Susan@gmail.com	7468536	Coppel
1009	Jessica	Sarah	Sarah@gmail.com	6479075	Coppel
1010	Margaret	Donaldson	Donaldson@gmail.com	8753689	Houston
1011	Sandra	Mark	V22Sandra@gmail.com	3399775	Houston

Below the result grid, the 'Output' tab is visible, showing the execution details:

#	Time	Action	Message
1	10:56:04	select * from user where length(phone_number)<=10 LIMIT 0, 1000	15 row(s) returned

- `WITH temporaryTable(avgVal) as  
(SELECT avg(Price)  
from room)  
SELECT *  
FROM room, temporaryTable  
WHERE room.price > temporaryTable.avgVal;`



SQL File 6

```

2 • WITH temporaryTable(avgVal) as
3   (SELECT avg(Price)
4    from room)
5   SELECT *
6   FROM room, temporaryTable
7   WHERE room.price > temporaryTable.avgVal;
8

```

Result Grid

room_id	hotel_id	capacity	price	reservation_id	avgVal
177	888	3	1500	7752	1033.3333333333333
178	888	3	1500	8767	1033.3333333333333
179	888	3	1500	9279	1033.3333333333333
180	888	3	1500	1089	1033.3333333333333
181	888	3	1500	1112	1033.3333333333333

Result 10

Output

Action Output

#	Time	Action	Message
1	10:53:49	WITH temporaryTable(avgVal) as (SELECT avg(Price) from room) SELECT * FROM room, temporary...	5 row(s) returned

- select room\_type.\*,instr(room\_name,'a') from room\_type where instr(room\_name,'s')>0;

SQL File 6

```

1 • select room_type.*,instr(room_name,'a') from room_type where instr(room_name,'s')>0;

```

Result Grid

room_name	room_rating	reservation_id	instr(room_name,'a')
Superior	3	1112	0
Superior	3	1211	0
Superior	4	1397	0
Superior	4	1467	0
Superior	4	1513	0

Result 8

Output

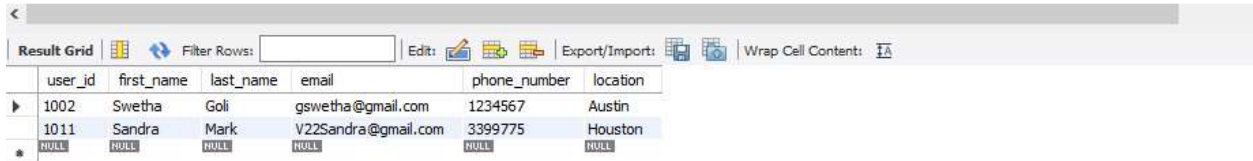
Action Output

#	Time	Action	Message
1	10:51:04	select room_type.*,instr(room_name,'a') from room_type where instr(room_name,'s')>0 LIMIT 0, 1000	5 row(s) returned

- `select * from user where regexp_like(first_name,'^S','i');`



The screenshot shows a text editor window titled "SQL File 6". The query `select * from user where regexp_like(first_name,'^S','i');` is entered on line 2. The editor includes a toolbar with icons for file operations and a "Limit to 1000 rows" dropdown.



The screenshot shows the "Result Grid" of the SQL query. It displays a table with 7 columns: user\_id, first\_name, last\_name, email, phone\_number, and location. Two rows are returned, corresponding to users with first names starting with 'S' (case-insensitive): Swetha and Sandra. The third row is a placeholder with NULL values.

user_id	first_name	last_name	email	phone_number	location
1002	Swetha	Goli	gswetha@gmail.com	1234567	Austin
1011	Sandra	Mark	V22Sandra@gmail.com	3399775	Houston
NULL	NULL	NULL	NULL	NULL	NULL



The screenshot shows the "Output" window with the "Action Output" tab selected. It displays a log entry for the execution of the SQL query, indicating that 2 rows were returned.

#	Time	Action	Message
1	10:59:20	select * from user where regexp_like(first_name,'^S','i') LIMIT 0, 1000	2 row(s) returned