

Hotel Management: Case Study

Database Schema:

Guests Table:

guest_id (Primary Key)

first_name

last_name

email

phone

Rooms Table:

room_id (Primary Key)

room_number

room_type

rate_per_night

Reservations Table:

reservation_id (Primary Key)

guest_id (Foreign Key)

room_id (Foreign Key)

check_in_date

check_out_date

total_cost

Billing Table:

billing_id (Primary Key)

reservation_id (Foreign Key)

billing_date

amount_paid

Creating a Table with values:

TABLE: Guests

guest_id	first_name	last_name	email	phone
1	John	Doe	johndoe@gmail.com	123-456-7890
2	Prajakta	Bhosale	prajakta@gmail.com	957-927-9654
3	Ruchita	Phalke	ruchita@gmail.com	702-092-4730
4	Avantika	Chavan	avantika@gmail.com	777-456-7850
5	Pratik	Kadam	pratik@gmail.com	241-456-7850
6	Ekta	darekar	ekta@gmail.com	856-235-9078

TABLE: Rooms

room_id	room_number	room_type	rate_per_night
1	102	Standard	100.00
2	103	Medium	150.00
3	104	deluxe	250.00
4	105	Standard	100.00
5	106	Standard	100.00
6	107	Standard	100.00
7	108	deluxe	250.00
8	109	deluxe	250.00
9	110	Medium	250.00
10	111	Medium	250.00

TABLE: **Reservations**

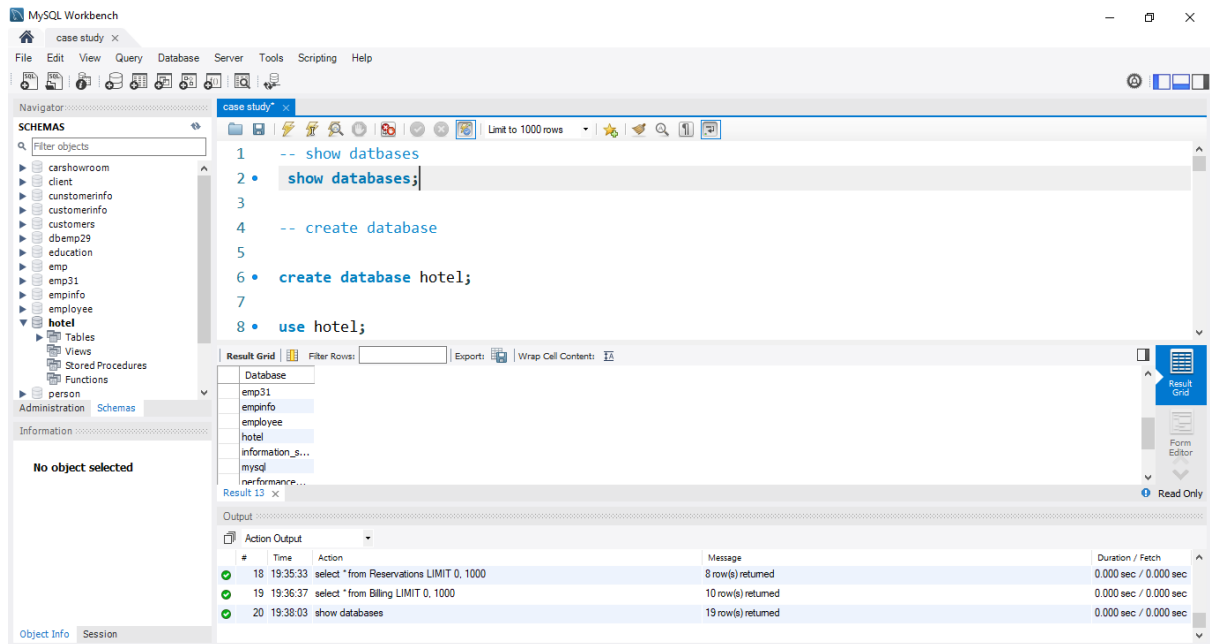
reservation_id	guest_id	room_id	check_in_date	check_out_date	total_cost
1	3	3	2023-07-12	2023-07-13	200.00
2	4	4	2023-08-21	2023-08-26	800.00
3	5	5	2023-09-10	2023-09-11	200.00
4	6	6	2023-09-02	2023-09-04	400.00
5	7	7	2023-07-10	2023-07-13	200.00
6	8	8	2023-09-13	2023-09-14	200.00
7	7	7	2023-09-18	2023-09-19	200.00

TABLE: **Billing**

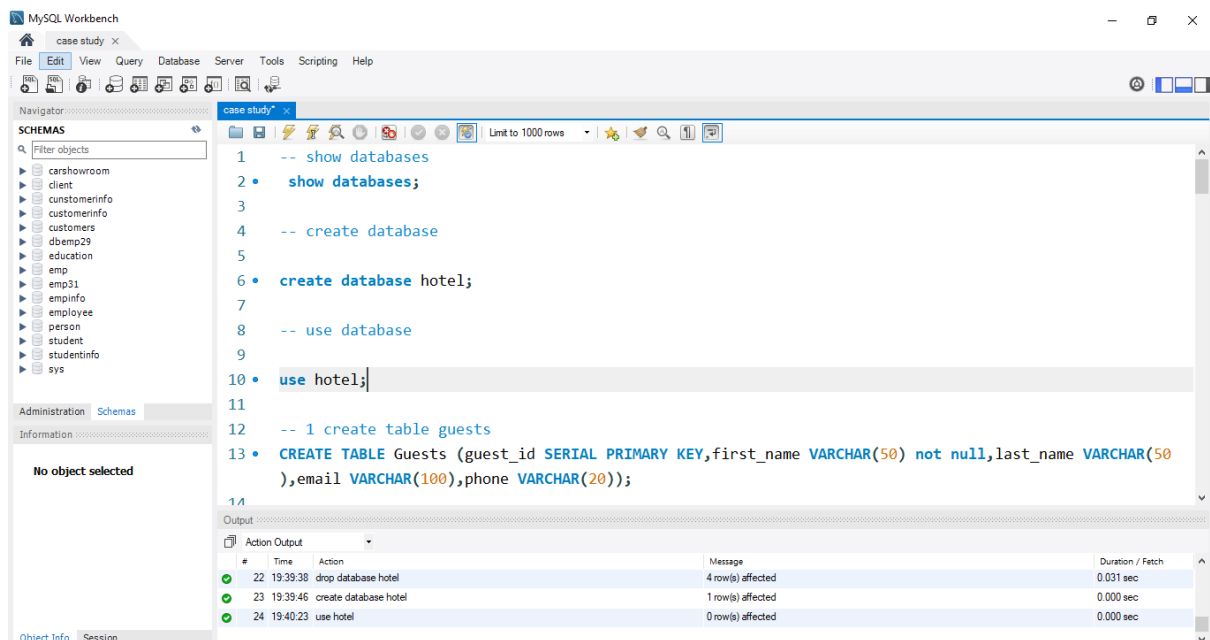
billing_id	reservation_id	billing date	amount paid
1	2	2023-09-15	300.00
2	3	2023-07-13	200.00
3	4	2023-08-26	800.00
4	5	2023-09-11	200.00
5	6	2023-09-04	400.00
6	7	2023-07-13	200.00
7	8	2023-07-13	400.00
8	9	2023-09-14	200.00
9	10	2023-09-13	200.00

SQL QUERIES

1. Show databases.



2. Create database hotel and use database hotel.



3. Alter table Guests use add column nationality.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'hotel' database selected. The main editor window contains the following SQL queries:

```
93 • select * from Billing;
94
95 -- alter table nationality in guests table
96 ALTER TABLE Guests ADD COLUMN nationality VARCHAR(50);
97
98 -- describe table Guests
99 desc Guests;
100
```

Below the queries, the 'Result Grid' shows the output of the 'desc Guests' query:

Field	Type	Null	Key	Default	Extra
guest_id	bigint unsigned	NO	PRI		auto_increment
first_name	varchar(50)	NO			
last_name	varchar(50)	YES			
email	varchar(100)	YES			
phone	varchar(20)	YES			
nationality	varchar(50)	YES			

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:39:13	use hotel	0 row(s) affected	0.000 sec
2	18:41:27	ALTER TABLE Guests ADD COLUMN nationality VARCHAR(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
3	18:41:59	desc Guests	6 row(s) returned	0.000 sec / 0.000 sec

4. Alter table Guests use drop column nationality.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'hotel' database selected. The main editor window contains the following SQL queries:

```
98 -- describe table Guests
99 desc Guests;
100
101 -- drop column(delete column)
102 alter table Guests drop column nationality;
103
104 -- describe table Guests
105 desc Guests;
```

Below the queries, the 'Result Grid' shows the output of the 'desc Guests' query:

Field	Type	Null	Key	Default	Extra
guest_id	bigint unsigned	NO	PRI		auto_increment
first_name	varchar(50)	NO			
last_name	varchar(50)	YES			
email	varchar(100)	YES			
phone	varchar(20)	YES			

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	18:41:59	desc Guests	6 row(s) returned	0.000 sec / 0.000 sec
4	18:46:35	alter table Guests drop column nationality	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
5	18:46:50	desc Guests	5 row(s) returned	0.000 sec / 0.000 sec

5. Alter table billing use Add column bill_no at first.

The screenshot shows the MySQL Workbench interface with the 'case study' database selected. The SQL editor contains the following queries:

```
104 -- describe table Guests
105 • desc Guests;
106
107 -- add new column at first
108 • alter table Billing add column bill_no int (10) first;
109
110 -- describe table Guests
111 • desc Billing;
```

The 'Result Grid' shows the structure of the 'Billing' table:

Field	Type	Null	Key	Default	Extra
bill_no	int	YES			
billing_id	bigint unsigned	NO	PRI		auto_increment
reservation_id	int	YES			
billing_date	date	YES			
amount_paid	decimal(10,2)	YES			

The 'Action Output' pane shows the results of the queries:

#	Time	Action	Message	Duration / Fetch
5	18:46:50	desc Guests	5 row(s) returned	0.000 sec / 0.000 sec
6	19:12:07	alter table Billing add column bill_no int (10) first	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future version.	0.016 sec
7	19:12:42	desc Billing	5 row(s) returned	0.016 sec / 0.000 sec

6. Alter table Reservations use rename column total_cost to total.

The screenshot shows the MySQL Workbench interface with the 'case study' database selected. The SQL editor contains the following queries:

```
116 -- describe table Guests
117 • desc Billing;
118
119 -- rename column name total_cost to total
120 • alter table Reservations rename column total_cost to total;
121
122 -- describe table Reservations
123 • desc Reservations;
```

The 'Result Grid' shows the structure of the 'Reservations' table:

Field	Type	Null	Key	Default	Extra
reservation_id	bigint unsigned	NO	PRI		auto_increment
guest_id	int	YES			
room_id	int	YES			
check_in_date	date	YES			
check_out_date	date	YES			
total	decimal(10,2)	YES			

The 'Action Output' pane shows the results of the queries:

#	Time	Action	Message	Duration / Fetch
10	19:17:54	select * from Reservations LIMIT 0, 1000	8 row(s) returned	0.015 sec / 0.000 sec
11	19:21:05	alter table Reservations rename column total_cost to total	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.156 sec
12	19:21:35	desc Reservations	6 row(s) returned	0.000 sec / 0.000 sec

7. Insert Data in Guests Table.

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
44 • INSERT INTO Guests (first_name, last_name, email, phone)
45 VALUES ('Prajakta', 'bhosale', 'prajakta@example.com', '957-927-9654'),
46 ('Ruchita', 'Phalke', 'ruchita@example.com', '702-092-4730'),('Avantika', 'Chavan',
47 'avantika@example.com', '777-456-7850'),('Pratik', 'Kadam', 'pratik@example.com', '241-456-5555'),
48 ('Ekta', 'darekar', 'ekta@example.com', '856-235-9078');
49 -- describe guests table
50 • select * from Guests;
```

The result grid displays the data inserted into the Guests table:

guest_id	first_name	last_name	email	phone
1	John	Doe	john.doe@example.com	123-456-7890
2	Prajakta	bhosale	prajakta@example.com	957-927-9654
3	Ruchita	Phalke	ruchita@example.com	702-092-4730
4	Avantika	Chavan	avantika@example.com	777-456-7850
5	Pratik	Kadam	pratik@example.com	241-456-5555
6	Ekta	darekar	ekta@example.com	856-235-9078
7	Prajakta	bhosale	prajakta@example.com	957-927-9654

The Output tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
14	19:25:23	desc Billing	4 row(s) returned	0.015 sec / 0.000 sec
15	19:30:08	INSERT INTO Guests (first_name, last_name, email, phone) VALUES ('Prajakta', 'bhosale', ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
16	19:30:11	select * from Guests LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec

7.1 Insert Data in Rooms Table.

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
56 • INSERT INTO Rooms (room_number, room_type, rate_per_night)
57 VALUES ('102', 'Standard', 100),('103', 'Medium', 150),('104', 'deluxe', 250),
58 ('105', 'Standard', 100),('106', 'Standard', 100),('107', 'Standard', 100),
59 ('108', 'deluxe', 250),('109', 'deluxe', 250),('110', 'Medium', 250),('111', 'Medium', 250);
60
61 -- show insert data in Rooms
62 • select * from Rooms;
63
```

The result grid displays the data inserted into the Rooms table:

room_id	room_number	room_type	rate_per_night
1	101	Standard	100.00
2	102	Standard	100.00
3	103	Medium	150.00
4	104	deluxe	250.00
5	105	Standard	100.00
6	106	Standard	100.00
7	107	Standard	100.00

The Output tab shows the execution results:

#	Time	Action	Message	Duration / Fetch
15	19:30:08	INSERT INTO Guests (first_name, last_name, email, phone) VALUES ('Prajakta', 'bhosale', ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.000 sec
16	19:30:11	select * from Guests LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
17	19:33:28	select * from Rooms LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec

7.2 Insert Data in Reservations Table.

The screenshot shows the MySQL Workbench interface with the following SQL queries in the editor:

```
71 • INSERT INTO Reservations (guest_id, room_id, check_in_date, check_out_date, total_cost)
72 VALUES (2, 2, '2023-09-15', '2023-09-18', 300);
73 • INSERT INTO Reservations (guest_id, room_id, check_in_date, check_out_date, total_cost)
74 VALUES (3, 3, '2023-07-12', '2023-07-13', 200),(4, 4, '2023-08-21', '2023-08-26', 800),
75 (5, 5, '2023-09-10', '2023-09-11', 200),(6, 6, '2023-09-02', '2023-09-04', 400),
76 (7, 7, '2023-07-10', '2023-07-13', 200),(8, 8, '2023-09-13', '2023-09-14', 200);
77
78 • select * from Reservations;
```

The Result Grid displays the data inserted into the Reservations table:

reservation_id	guest_id	room_id	check_in_date	check_out_date	total
1	1	1	2023-09-15	2023-09-18	300.00
2	2	2	2023-09-15	2023-09-18	300.00
3	3	3	2023-07-12	2023-07-13	200.00
4	4	4	2023-08-21	2023-08-26	800.00
5	5	5	2023-09-10	2023-09-11	200.00
6	6	6	2023-09-02	2023-09-04	400.00
7	7	7	2023-07-10	2023-07-13	200.00

The Output pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
16	19:30:11	select * from Guests LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
17	19:33:28	select * from Rooms LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
18	19:35:33	select * from Reservations LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

7.3 Insert Data in Billing Table.

The screenshot shows the MySQL Workbench interface with the following SQL queries in the editor:

```
87
88 • INSERT INTO Billing (reservation_id, billing_date, amount_paid)
89 VALUES (2, '2023-09-15', 300),(3, '2023-07-13', 200),(4, '2023-08-26', 800),
90 (5, '2023-09-11', 200),(6, '2023-09-04', 400),(7, '2023-07-13', 200),
91 (8, '2023-07-13', 400),(9, '2023-09-14', 200),(10, '2023-09-13', 200);
92
93 • select * from Billing;
94
```

The Result Grid displays the data inserted into the Billing table:

billing_id	reservation_id	billing_date	amount_paid
1	1	2023-09-18	300.00
2	2	2023-09-15	300.00
3	3	2023-07-13	200.00
4	4	2023-08-26	800.00
5	5	2023-09-11	200.00
6	6	2023-09-04	400.00
7	7	2023-07-13	200.00

The Output pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
17	19:33:28	select * from Rooms LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
18	19:35:33	select * from Reservations LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec
19	19:36:37	select * from Billing LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

8. Updating Guest Information:

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
-- Updating Guest Information:
UPDATE Guests
SET phone = '987-654-3210'
WHERE guest_id = 1;

select * from Guests;
```

The result grid displays the following data:

guest_id	first_name	last_name	email	phone
1	John	Doe	john.doe@example.com	987-654-3210
2	Prajakta	bhosale	prajakta@example.com	957-927-9654
3	Ruchita	Phalke	ruchita@example.com	702-092-4730
4	Avantika	Chavan	avantika@example.com	777-456-7850
5	Pratik	Kadam	pratik@example.com	241-456-5555
6	Ekta	darekar	ekta@example.com	856-235-9078

The Output tab shows the following messages:

#	Time	Action	Message	Duration / Fetch
52	20:12:55	SELECT SUM(total_cost) AS total_revenue FROM Reservations WHERE DATE_PART(...)	Error Code: 1305. FUNCTION hotel.DATE_PART does not exist	0.000 sec
53	20:13:42	UPDATE Guests SET phone = '987-654-3210' WHERE guest_id = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
54	20:14:08	select * from Guests LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

9. retrieve rooms whose rate per night is greater than 100

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
-- retrieve rooms whose rate per night is greater than 100
select * from rooms where rate_per_night > 100;
```

The result grid displays the following data:

room_id	room_number	room_type	rate_per_night
2	103	Medium	150.00
3	104	deluxe	250.00
7	108	deluxe	250.00
8	109	deluxe	250.00
9	110	Medium	250.00
10	111	Medium	250.00

The Output tab shows the following messages:

#	Time	Action	Message	Duration / Fetch
62	20:36:33	select * from Guests LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
63	20:36:44	select * from Rooms LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
64	20:40:18	select * from rooms where rate_per_night > 100 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

10. retrieve room type who guests stay in room

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is as follows:

```
137 • select * from rooms where rate_per_night > 100;
138
139 -- retrieve room type who guests stay in room
140
141 • select * from rooms where room_type = 'deluxe';
142
```

The result grid displays the following data:

room_id	room_number	room_type	rate_per_night
3	104	deluxe	250.00
7	108	deluxe	250.00
8	109	deluxe	250.00

The output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
63	20:36:44	select * from Rooms LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
64	20:40:18	select * from rooms where rate_per_night > 100 LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
65	20:44:49	select * from rooms where room_type = 'deluxe' LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

11. find total_cost in who reservation room id = 4

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is as follows:

```
140
141 • select * from rooms where room_type = 'deluxe';
142
143 -- find total_cost in who reservation room id = 4
144 • select * from Reservations where room_id = '4';
145
```

The result grid displays the following data:

reservation_id	guest_id	room_id	check_in_date	check_out_date	total_cost
2	4	4	2023-08-21	2023-08-26	800.00

The output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
67	20:46:33	select * from Billing LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
68	20:46:44	select * from Reservations LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
69	20:52:09	select * from Reservations where room_id = '4' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

12. write a query for guest first_name ends with a

The screenshot shows the MySQL Workbench interface with a query window open. The query is as follows:

```
-- write a query for guest fname ends with a
select * from Guests where first_name like '%a';
```

The result grid displays the following data:

guest_id	first_name	last_name	email	phone
2	Prajakta	bhosale	prajakta@example.com	957-927-9654
3	Ruchita	Phalke	ruchita@example.com	702-092-4730
4	Avantika	Chavan	avantika@example.com	777-456-7850
6	Ekta	darekar	ekta@example.com	856-235-9078
7	Prajakta	bhosale	prajakta@example.com	957-927-9654
8	Ruchita	Phalke	ruchita@example.com	702-092-4730
9	Avantika	Chavan	avantika@example.com	777-456-7850
11	Ekta	darekar	ekta@example.com	856-235-9078

The output pane shows the execution of the query, indicating that 8 rows were returned.

13. Find available rooms for a specific date range.

The screenshot shows the MySQL Workbench interface with a query window open. The query is as follows:

```
desc Reservations;

-- Find available rooms for a specific date range
SELECT room_id, room_number, room_type FROM Rooms WHERE room_id NOT IN (SELECT room_id FROM Reservations
WHERE ('2023-07-13' BETWEEN check_in_date AND check_out_date) OR ('2023-09-15' BETWEEN check_in_date AND check_out_date));
```

The result grid displays the following data:

room_id	room_number	room_type
1	102	Standard
2	103	Medium
4	105	Standard
5	106	Standard
6	107	Standard
8	109	deluxe
9	110	Medium
10	111	Medium

The output pane shows the execution of the query, indicating that 8 rows were returned.

14. retrieve all guest bill paid amount sorted in descending order.

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
148 • select * from Guests where first_name like 'P%';
149
150 -- retrieve all guest bill paid amount sorted in descending order
151
152 • select * from Billing order by amount_paid;
153
154 -- Finding the Total Amount Paid by a Guest:
```

The result grid displays the following data:

billing_id	reservation_id	billing_date	amount_paid
2	3	2023-07-13	200.00
4	5	2023-09-11	200.00
6	7	2023-07-13	200.00
8	9	2023-09-14	200.00
9	10	2023-09-13	200.00
1	2	2023-09-15	300.00
5	6	2023-09-04	400.00
7	8	2023-07-13	400.00
3	4	2023-08-26	800.00

The output pane shows the following action output:

#	Time	Action	Message	Duration / Fetch
4	19:04:07	select * from Reservations order by amount_paid LIMIT 0, 1000	Error Code: 1054. Unknown column 'amount_paid' in 'order clause'	0.015 sec
5	19:04:33	select * from Billing LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
6	19:04:56	select * from Billing order by amount_paid LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec

15. Find reservations for Ruchita using Alias.

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
131
132 -- Find reservations for Ruchita
133 • SELECT r.reservation_id, r.check_in_date, r.check_out_date, r.total_cost
134 FROM Reservations r
135 JOIN Guests g ON r.guest_id = g.guest_id
136 WHERE g.first_name = 'Ruchita' AND g.last_name = 'Phalke';
137
```

The result grid displays the following data:

reservation_id	check_in_date	check_out_date	total_cost
1	2023-07-12	2023-07-13	200.00

The output pane shows the following action output:

#	Time	Action	Message	Duration / Fetch
47	20:02:44	select * from Billing LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
48	20:03:41	SELECT room_id, room_number, room_type FROM Rooms WHERE room_id NOT IN (SE...	8 row(s) returned	0.000 sec / 0.000 sec
49	20:07:00	SELECT r.reservation_id, r.check_in_date, r.check_out_date, r.total_cost FROM Reserva...	1 row(s) returned	0.000 sec / 0.000 sec

16. Find 2nd bill amount_paid from billing table.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
170
171 -- find 2nd bill amount_paid from billing table
172
173 • select distinct(amount_paid) from Billing order by amount_paid desc limit 1,1;
174
175
```

The result grid shows a single row with the value 400.00.

The output pane shows the following actions and messages:

#	Time	Action	Message	Duration / Fetch
5	19:04:33	select * from Billing LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
6	19:04:56	select * from Billing order by amount_paid LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
7	19:11:52	select distinct(amount_paid) from Billing order by amount_paid desc limit 1,1	1 row(s) returned	0.000 sec / 0.000 sec

17. Find 2nd bill amount_paid from billing table using sub-queries.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
173 • select distinct(amount_paid) from Billing order by amount_paid desc limit 1,1;
174
175 -- find 2nd bill amount_paid from billing table using sub-queries
176 • select max(amount_paid) from Billing where amount_paid < (select max(amount_paid) from Billing);
177
178 -- write a query to find no.of guests stay in hotel each room_type
```

The result grid shows a single row with the value 400.00.

The output pane shows the following actions and messages:

#	Time	Action	Message	Duration / Fetch
19	19:41:18	select billing_id ,count(*) from Billing group by billing_id having count(*) > 1 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
20	19:41:24	select billing_id ,count(*) from Billing group by billing_id having count(*) LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec
21	19:46:33	select max(amount_paid) from Billing where amount_paid < (select max(amount_paid) from ...	1 row(s) returned	0.000 sec / 0.000 sec

18. write a query to find no.of guests stay in hotel each room_type.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' list with 'hotel' selected. The main editor window contains the following SQL code:

```
173 • select distinct(amount_paid) from Billing order by amount_paid desc limit 1,1;
174
175 -- write a query to find no.of guests stay in hotel each room_type
176 • select room_type,count(*) from Rooms group by room_type;
177
178 -- Find reservations for Ruchita
```

The 'Result Grid' shows the output of the query in table format:

room_type	count(*)
Standard	4
Medium	3
deluxe	3

The 'Action Output' pane at the bottom shows the execution details of the queries.

19. Write a query to find
max(amount_paid),min(amount_paid),sum(amount_paid),avg(amount_paid) each
from billing_id.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' list with 'hotel' selected. The main editor window contains the following SQL code:

```
177
178 /* write a query to find max(amount_paid),min(amount_paid),sum(amount_paid),
179 avg(amount_paid) each from billing_id */
180
181 • select billing_id,max(amount_paid),min(amount_paid),sum(amount_paid),
182 avg(amount_paid) from Billing group by billing_id;
```

The 'Result Grid' shows the output of the query in table format:

billing_id	max(amount_paid)	min(amount_paid)	sum(amount_paid)	avg(amount_paid)
1	300.00	300.00	300.00	300.000000
2	200.00	200.00	200.00	200.000000
3	800.00	800.00	800.00	800.000000
4	200.00	200.00	200.00	200.000000
5	400.00	400.00	400.00	400.000000
6	200.00	200.00	200.00	200.000000
7	400.00	400.00	400.00	400.000000
8	200.00	200.00	200.00	200.000000
9	200.00	200.00	200.00	200.000000

The 'Action Output' pane at the bottom shows the execution details of the queries.

20. write a query to find no.of guests stay in hotel each billing_id having count

MySQL Workbench interface showing a query to find the number of guests staying in a hotel for each billing_id. The query is:

```

182 avg(amount_paid) from Billing group by billing_id;
183
184 -- write a query to find no.of guests stay in hotel each billing_id having count
185 • select billing_id ,count(*) from Billing group by billing_id having count(*) > 1;
186
187 -- Find reservations for Ruchita

```

The result grid shows the following data:

billing_id	count(*)
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1

The output pane shows the execution of the query, indicating that 9 rows were returned.

21. Listing All Guests Who Have Checked In.

MySQL Workbench interface showing a query to list all guests who have checked in. The query is:

```

136 WHERE g.first_name = 'Ruchita' AND g.last_name = 'Phalke';
137
138 -- Listing All Guests Who Have Checked In:
139 • SELECT g.first_name, g.last_name, r.check_in_date, r.check_out_date
140 FROM Guests g
141 JOIN Reservations r ON g.guest_id = r.guest_id
142

```

The result grid shows the following data:

first_name	last_name	check_in_date	check_out_date
Ruchita	Phalke	2023-07-12	2023-07-13
Avantika	Chavan	2023-08-21	2023-08-26
Pratik	Kadam	2023-09-10	2023-09-11
Eka	darekar	2023-09-02	2023-09-04

The output pane shows the execution of the query, indicating that 4 rows were returned.

22. Finding the Total Amount Paid by a Guest.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' list with 'hotel' selected. The main editor window contains the following SQL query:

```
134
135 -- Finding the Total Amount Paid by a Guest:
136 • SELECT SUM(amount_paid) AS total_amount_paid
137 FROM Billing
138 WHERE reservation_id IN (SELECT reservation_id FROM Reservations WHERE guest_id = 1);
139
140
```

Below the query editor, the 'Result Grid' shows a single column named 'total_amount_paid' with a value of '1000'. The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
53	20:13:42	UPDATE Guests SET phone = '987-654-3210' WHERE guest_id = 1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
54	20:14:08	select * from Guests LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
55	20:16:54	SELECT SUM(amount_paid) AS total_amount_paid FROM Billing WHERE reservation_id IN (SELECT reservation_id FROM Reservations WHERE guest_id = 1);	1 row(s) returned	0.000 sec / 0.000 sec

23. Finding the Most Popular Room Type.

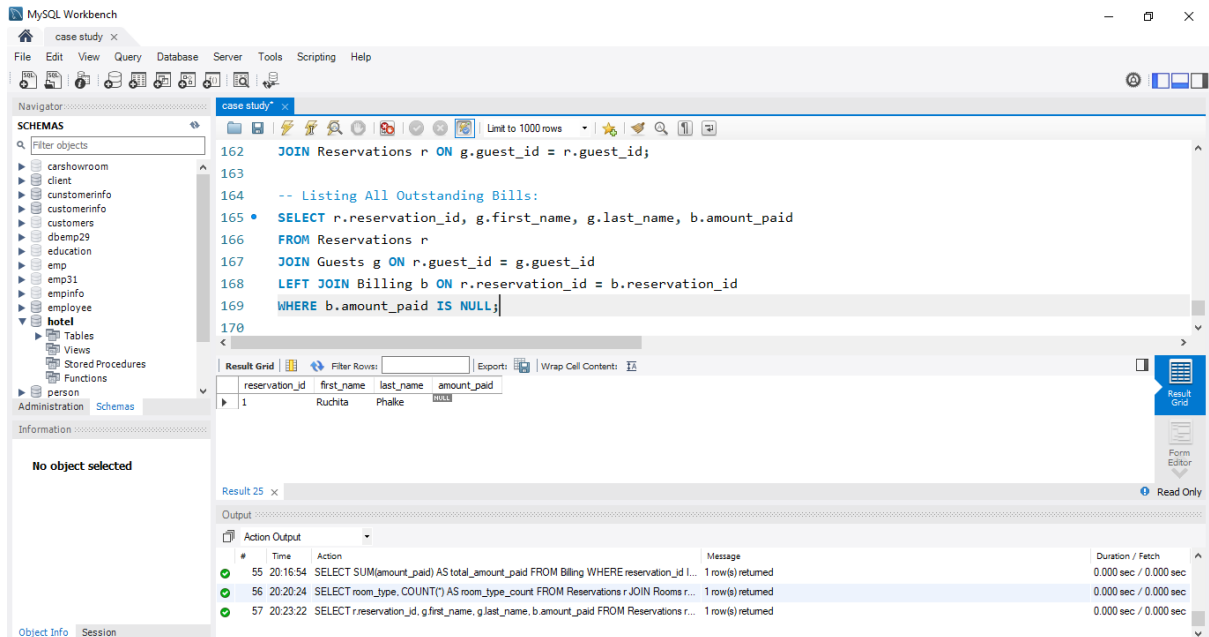
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' list with 'hotel' selected. The main editor window contains the following SQL query:

```
150
151 -- Finding the Most Popular Room Type:
152 • SELECT room_type, COUNT(*) AS room_type_count FROM Reservations r JOIN Rooms rm ON r.room_id = rm.room_id
153 GROUP BY room_type
154 ORDER BY room_type_count DESC
155 LIMIT 1;
156
157
158
```

Below the query editor, the 'Result Grid' shows two columns: 'room_type' and 'room_type_count'. The first row shows 'deluxe' with a count of '3'. The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
54	20:14:08	select * from Guests LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
55	20:16:54	SELECT SUM(amount_paid) AS total_amount_paid FROM Billing WHERE reservation_id IN (SELECT reservation_id FROM Reservations WHERE guest_id = 1);	1 row(s) returned	0.000 sec / 0.000 sec
56	20:20:24	SELECT room_type, COUNT(*) AS room_type_count FROM Reservations r JOIN Rooms rm ON r.room_id = rm.room_id GROUP BY room_type ORDER BY room_type_count DESC LIMIT 1;	1 row(s) returned	0.000 sec / 0.000 sec

24. Listing All Outstanding Bills.



The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is as follows:

```
162 JOIN Reservations r ON g.guest_id = r.guest_id;
163
164 -- Listing All Outstanding Bills:
165 • SELECT r.reservation_id, g.first_name, g.last_name, b.amount_paid
166 FROM Reservations r
167 JOIN Guests g ON r.guest_id = g.guest_id
168 LEFT JOIN Billing b ON r.reservation_id = b.reservation_id
169 WHERE b.amount_paid IS NULL;
170
```

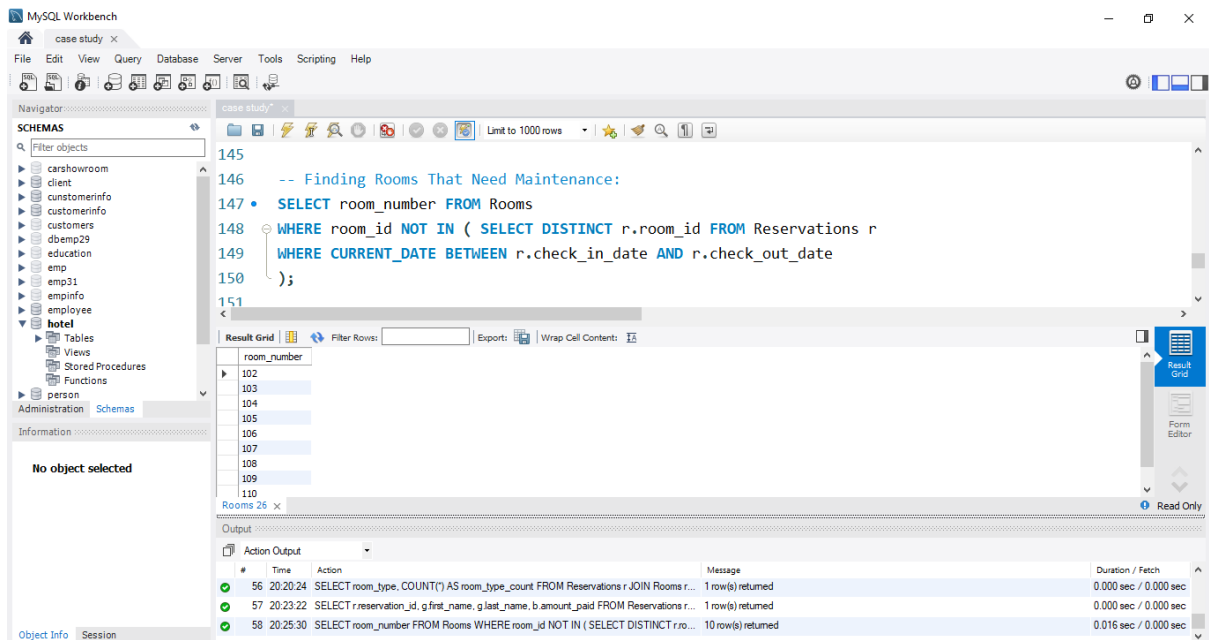
The results grid shows one row of data:

reservation_id	first_name	last_name	amount_paid
1	Ruchita	Phalke	0.00

The output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
55	20:16:54	SELECT SUM(amount_paid) AS total_amount_paid FROM Billing WHERE reservation_id L...	1 row(s) returned	0.000 sec / 0.000 sec
56	20:20:24	SELECT room_type, COUNT(*) AS room_type_count FROM Reservations r JOIN Rooms r...	1 row(s) returned	0.000 sec / 0.000 sec
57	20:23:22	SELECT r.reservation_id, g.first_name, g.last_name, b.amount_paid FROM Reservations r...	1 row(s) returned	0.000 sec / 0.000 sec

25. Finding Rooms That Need Maintenance.



The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is as follows:

```
145
146 -- Finding Rooms That Need Maintenance:
147 • SELECT room_number FROM Rooms
148 WHERE room_id NOT IN ( SELECT DISTINCT r.room_id FROM Reservations r
149 WHERE CURRENT_DATE BETWEEN r.check_in_date AND r.check_out_date
150 );
151
```

The results grid shows one row of data:

room_number
102

The output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
56	20:20:24	SELECT room_type, COUNT(*) AS room_type_count FROM Reservations r JOIN Rooms r...	1 row(s) returned	0.000 sec / 0.000 sec
57	20:23:22	SELECT r.reservation_id, g.first_name, g.last_name, b.amount_paid FROM Reservations r...	1 row(s) returned	0.000 sec / 0.000 sec
58	20:25:30	SELECT room_number FROM Rooms WHERE room_id NOT IN (SELECT DISTINCT r.ro...	10 row(s) returned	0.016 sec / 0.000 sec

26. Calculating Total Number of Reservations

The screenshot shows the MySQL Workbench interface with a query editor and a results pane. The query editor contains the following SQL code:

```
213 LEFT JOIN Billing b ON r.reservation_id = b.reservation_id
214 WHERE b.amount_paid IS NULL;
215
216 -- Calculating Total Number of Reservations:
217 • SELECT COUNT(*) AS total_reservations FROM Reservations;
218
219
```

The results pane shows the output of the query, which is a single row with the value 6 for the column total_reservations.

#	Time	Action	Message	Duration / Fetch
1	22:59:31	use hotel	0 row(s) affected	0.000 sec
2	23:01:59	select * from Reservations LIMIT 0, 1000	6 row(s) returned	0.016 sec / 0.000 sec
3	23:02:18	SELECT COUNT(*) AS total_reservations FROM Reservations LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

27. Listing All Room Types:

The screenshot shows the MySQL Workbench interface with a query editor and a results pane. The query editor contains the following SQL code:

```
217 • SELECT COUNT(*) AS total_reservations FROM Reservations;
218
219 -- Listing All Room Types
220 • SELECT DISTINCT room_type FROM Rooms;
221
222
```

The results pane shows the output of the query, which is a single column with three rows: Standard, Medium, and deluxe.

#	Time	Action	Message	Duration / Fetch
2	23:01:59	select * from Reservations LIMIT 0, 1000	6 row(s) returned	0.016 sec / 0.000 sec
3	23:02:18	SELECT COUNT(*) AS total_reservations FROM Reservations LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
4	23:05:19	SELECT DISTINCT room_type FROM Rooms LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

28. Finding Guests by Last Name

The screenshot shows the MySQL Workbench interface. The left sidebar displays a schema tree with various tables like 'carshowroom', 'client', 'customerinfo', etc. The main editor window contains the following SQL code:

```
220 • SELECT DISTINCT room_type FROM Rooms;
221
222 -- 28 Finding Guests by Last Name:
223 • SELECT * FROM Guests
224 WHERE last_name = 'Bhosale';
225
```

The 'Result Grid' shows the results of the query:

guest_id	first_name	last_name	email	phone
2	Prajakta	bhosale	prajakta@example.com	957-927-9654
7	Prajakta	bhosale	prajakta@example.com	957-927-9654
NULL	NULL	NULL	NULL	NULL

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	14:01:20	use hotel	0 row(s) affected	0.000 sec
2	14:01:41	SELECT * FROM Guests WHERE last_name = 'Bhosale' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

29. Finding Available Rooms for a Date Range and Room Type:

The screenshot shows the MySQL Workbench interface. The left sidebar displays a schema tree. The main editor window contains the following SQL code:

```
226 -- Finding Available Rooms for a Date Range and Room Type:
227 • SELECT r.room_number, r.room_type FROM Rooms r LEFT JOIN Reservations res ON r.room_id = res.room_id
228 WHERE res.reservation_id IS NULL OR ('2023-09-13' NOT BETWEEN res.check_in_date AND res.check_out_date)
229 AND r.room_type = 'Deluxe';
230
231 -- Calculating the Average Length of Stay for All Reservations:
232
```

The 'Result Grid' shows the results of the query:

room_number	room_type
102	Standard
103	Medium
104	deluxe
108	deluxe
110	Medium
111	Medium

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
2	14:01:41	SELECT * FROM Guests WHERE last_name = 'Bhosale' LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
3	14:05:22	select * from Reservations LIMIT 0, 1000	6 row(s) returned	0.015 sec / 0.000 sec
4	14:06:32	SELECT r.room_number, r.room_type FROM Rooms r LEFT JOIN Reservations res ON r.r...	6 row(s) returned	0.000 sec / 0.000 sec

30. Finding the Most Recent Check-In Date for a Guest

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is as follows:

```
235
236 -- Finding the Most Recent Check-In Date for a Guest:
237 * SELECT g.first_name, g.last_name, MAX(r.check_in_date) AS most_recent_check_in
238 FROM Guests g
239 JOIN Reservations r ON g.guest_id = r.guest_id
240 GROUP BY g.first_name, g.last_name];
241
```

The result grid displays the following data:

first_name	last_name	most_recent_check_in
Ruchita	Phalke	2023-09-13
Avantika	Chavan	2023-08-21
Pratik	Kadam	2023-09-10
Ekta	darekar	2023-09-02
Prajakta	bhosale	2023-07-10

The output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
4	14:06:32	SELECT r.room_number, r.room_type FROM Rooms r LEFT JOIN Reservations res ON r.r...	6 row(s) returned	0.000 sec / 0.000 sec
5	14:08:51	SELECT AVG(DATEDIFF(day, check_in_date, check_out_date)) AS avg_length_of_stay ...	Error Code: 1582 Incorrect parameter count in the call to native function 'DATEDIFF'	0.000 sec
6	14:09:10	SELECT g.first_name, g.last_name, MAX(r.check_in_date) AS most_recent_check_in FR...	5 row(s) returned	0.015 sec / 0.000 sec

31. Checking for Overlapping Reservations:

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is as follows:

```
241
242 -- Checking for Overlapping Reservations:
243 * SELECT r1.reservation_id, r1.guest_id, r1.check_in_date, r1.check_out_date FROM Reservations r1, Reservations r2
244 WHERE r1.reservation_id != r2.reservation_id AND r1.room_id = r2.room_id
245 AND ((r1.check_in_date BETWEEN r2.check_in_date AND r2.check_out_date)
246 OR (r1.check_out_date BETWEEN r2.check_in_date AND r2.check_out_date));
247
```

The result grid displays the following data:

reservation_id	guest_id	check_in_date	check_out_date
----------------	----------	---------------	----------------

The output pane shows the following messages:

#	Time	Action	Message	Duration / Fetch
5	14:08:51	SELECT AVG(DATEDIFF(day, check_in_date, check_out_date)) AS avg_length_of_stay ...	Error Code: 1582 Incorrect parameter count in the call to native function 'DATEDIFF'	0.000 sec
6	14:09:10	SELECT g.first_name, g.last_name, MAX(r.check_in_date) AS most_recent_check_in FR...	5 row(s) returned	0.015 sec / 0.000 sec
7	14:10:32	SELECT r1.reservation_id, r1.guest_id, r1.check_in_date, r1.check_out_date FROM Res...	0 row(s) returned	0.000 sec / 0.000 sec