



Mini Project – Hotel Management System

Department of Computer Engineering

Faculty of Engineering

University of Ruhuna

Sri Lanka

On 9th of April 2024

In completing an assignment for the module

EE4350 Database System

By

Group No: 45

MUTHUKUMARI H.M.S. -EG/2021/4685

NADHA M.J. -EG/2021/4686

NAFLA M.N.P. -EG/2021/4687

Contents

Chapter 1 – Requirement Analysis	4
Introduction	4
Functional Requirements.....	5
Data Requirements	7
Chapter 2 – Conceptual Design (ER Diagram).....	9
Conceptual design (UML Diagram).....	10
Relationships of ER Diagram.....	11
Recursive Relationship:.....	11
Chapter 3 – Implementation.....	12
Schema Creation	12
Table Definitions	13
Inserting Data for Tables	18
Update Operations.....	23
Delete Operations.....	28
Chapter 4 – Transaction	33
Simple Queries	33
Complex Queries.....	36
Chapter 5 – Tuning	44

Table of Figures

Figure 1: ER Diagram.....	9
Figure 2: UML Diagram	10

1. Chapter 1 – Requirement Analysis

Introduction

Databases and data management systems are crucial in our daily lives, especially in industries like hospitality. Hotels, being pivotal in the travel and tourism sector, heavily rely on efficient management systems to ensure smooth operations and exceptional guest experiences.

This mini-project aims to address the complexities of managing hotel operations through a comprehensive database system. Well-organized hotel management systems are essential for ensuring guest satisfaction and business success.

Our project focuses on developing a database management system tailored for the hotel industry. Drawing inspiration from the rich cultural heritage and diverse landscapes of our surroundings, we aim to create a system that streamlines hotel operations, enhances guest services, and ultimately contributes to the overall quality of stay for guests.

By studying the organizational and operational characteristics of hotels, we've gathered the necessary information to design a system that meets the unique requirements of hotel management. This system will assist hotel staff in managing room reservations, tracking guest preferences, handling administrative tasks, and providing personalized services.

We believe that implementing this database system will not only improve the efficiency of hotel operations but also enhance the overall guest experience. From front desk personnel to housekeeping staff, everyone involved in hotel management will benefit from this user-friendly and comprehensive system.

Our project adheres to both the functional and data requirements of the hotel industry. Functional requirements define the operations and activities the system should perform, while data requirements identify the necessary information needed to build the system, including attributes of the ER diagram, columns, and records.

In summary, our goal is to develop a hotel management system that revolutionizes the way hotels operate, ensuring seamless experiences for both guests and staff alike.

Functional Requirements

Staff Management:

- The system should be capable of managing information about hotel staff, including their names, positions, salary and contact details.
- It should facilitate the addition, updating, and deletion of staff details, ensuring an up-to-date record of all employees.
- The system should provide insights into the availability and status of each staff member, indicating whether they are on duty, on leave, or unavailable for any reason.

Room Management:

- The system should manage information regarding different types of rooms available in the hotel, including their type ID, maximum guest capacity, total number of rooms, room description, and room price.
- The system should provide insights into the availability status of each room type, indicating the number of rooms available for booking and occupancy.

Room Management:

- The system should manage information about individual rooms within the hotel, including their room number, room type, rate, bed type, and cleaning status.
- The system should allow for the assignment of room types to specific room numbers, ensuring proper categorization and organization.
- It should provide insights into the cleaning status of each room, indicating whether it is clean and available for occupancy or in need of cleaning.

Reservation Management:

- The system should manage reservations made by guests, including guest ID, check-in date, check-out date, number of guests, and room number.
- It should facilitate the creation, modification, and cancellation of reservations, ensuring accurate and up-to-date booking records.
- The system should validate reservation details, ensuring that the check-in date is before the check-out date and that the selected room is available for the specified dates.

Payment Management:

- The system should manage payment information associated with guest reservations, including payment ID, guest ID, payment method, payment date, and total amount.
- It should facilitate the recording, updating, and tracking of payments made by guests for their reservations.
- The system should support various payment methods, such as cash, credit card, or online payment services, allowing guests flexibility in settling their bills.

Invoice Management:

- The system should manage invoice information for guest reservations, including guest ID, issue date, due date, tax amount, and total amount.
- It should facilitate the generation, updating, and tracking of invoices for each guest reservation, ensuring accurate billing and payment processing.
- The system should automatically calculate tax amounts based on applicable tax rates and include them in the invoice total.

Guest Management:

- The system should manage guest information, including guest ID, first name, last name, date of birth, age, email address, street address, city, province, and guest password.
- It should facilitate the creation, updating, and deletion of guest profiles, ensuring accurate and up-to-date guest records.
- The system should validate guest information, including ensuring that email addresses are in the correct format and that passwords meet security requirements.

Contact Number Management:

- The system should manage contact number information for guests, including guest ID and contact number.
- The system should validate contact numbers to ensure they are in the correct format and meet any specified criteria.

Cancellation Management:

- The system should manage cancellation information for guest reservations, including cancellation ID, guest ID, payment ID, reason for cancellation, refund amount, and cancellation date.

Data Requirements

Data requirements are the data that provide a detailed description of the data model that the system must use to fulfil its functional requirements. There are many entities in Er diagram and the attributes of those entities shows the data we need to build this database of a Hotel Management System. So, the data requirements can be listed as follows.

List of entities

1. Staff
2. Room Type
3. Room
4. Reservation
5. Payment
6. Invoice
7. Guest
8. Contact Number
9. Cancellation

Attribute List

Entity	Attributes (data Requirements)
Staff	Staff ID
	Staff Name
	Position
	Salary
	Hire date
Room Type	Type ID
	Maximum Guest
	Total Rooms
	Room Description
	Room Price
Room	Room Number
	Room Type
	Rate
	Bed Type
	Cleaning Status
Reservation	Guest ID
	Check In Date
	Check Out Date
	Number of Guests

	Room Number
Payment	Payment ID
	Guest ID
	Payment Method
	Payment Date
	Total Amount
Invoice	Guest ID
	Issue Date
	Due Date
	Tax Amount
	Total Amount
Guest	Guest ID
	First Name
	Last Name
	Date of Birth
	Age
	Email
	Street
	City
	Province
	Guest Password
Contact Number	Guest ID
	Contact Number
Cancellation	Cancellation ID
	Guest ID
	Payment ID
	Reason
	Refund Amount
	Cancellation Date

Chapter 2 – Conceptual Design (ER Diagram)

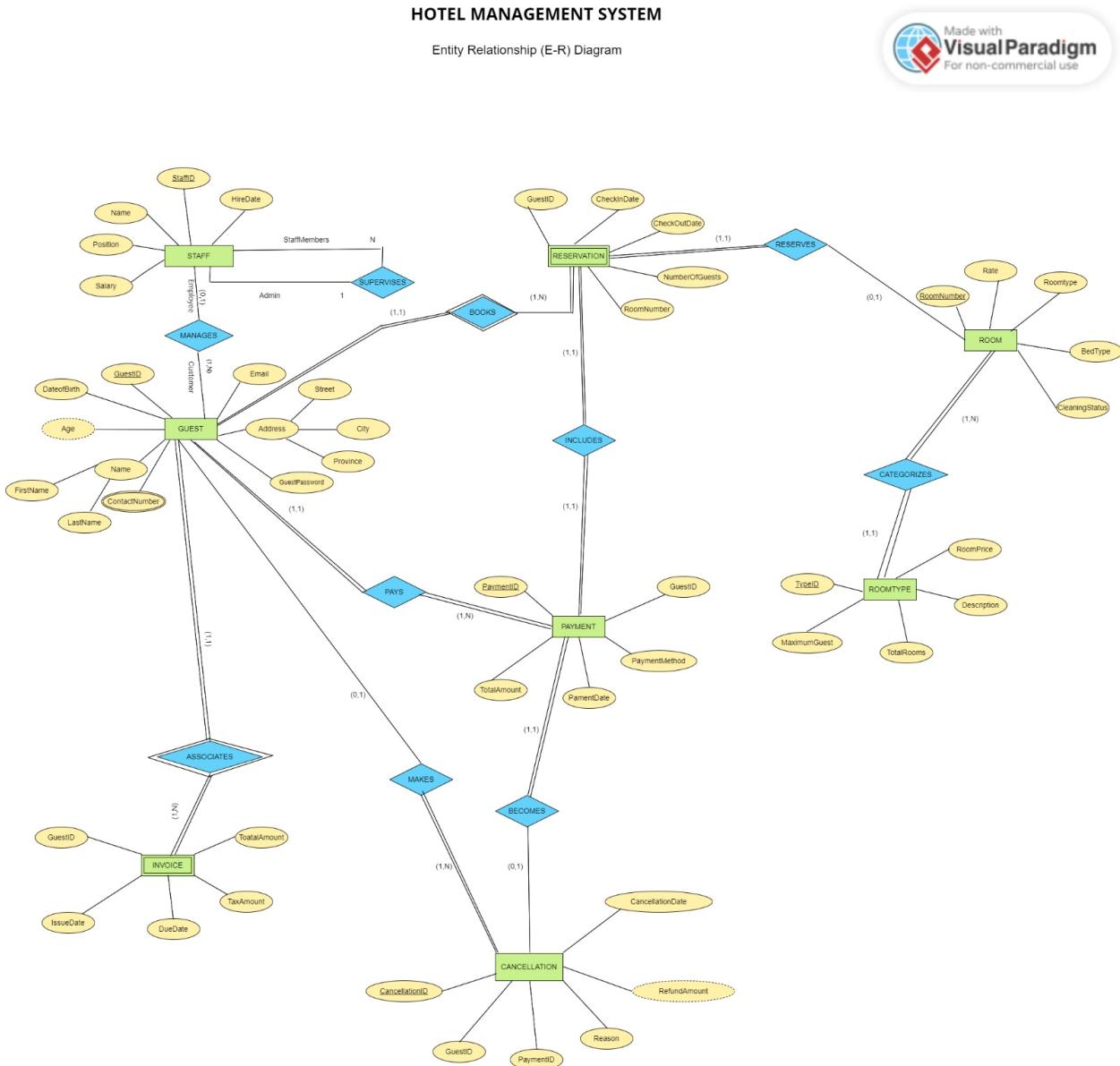


Figure 1: ER Diagram

Made with
VisualParadigm
For non-commercial use

Conceptual design (UML Diagram)

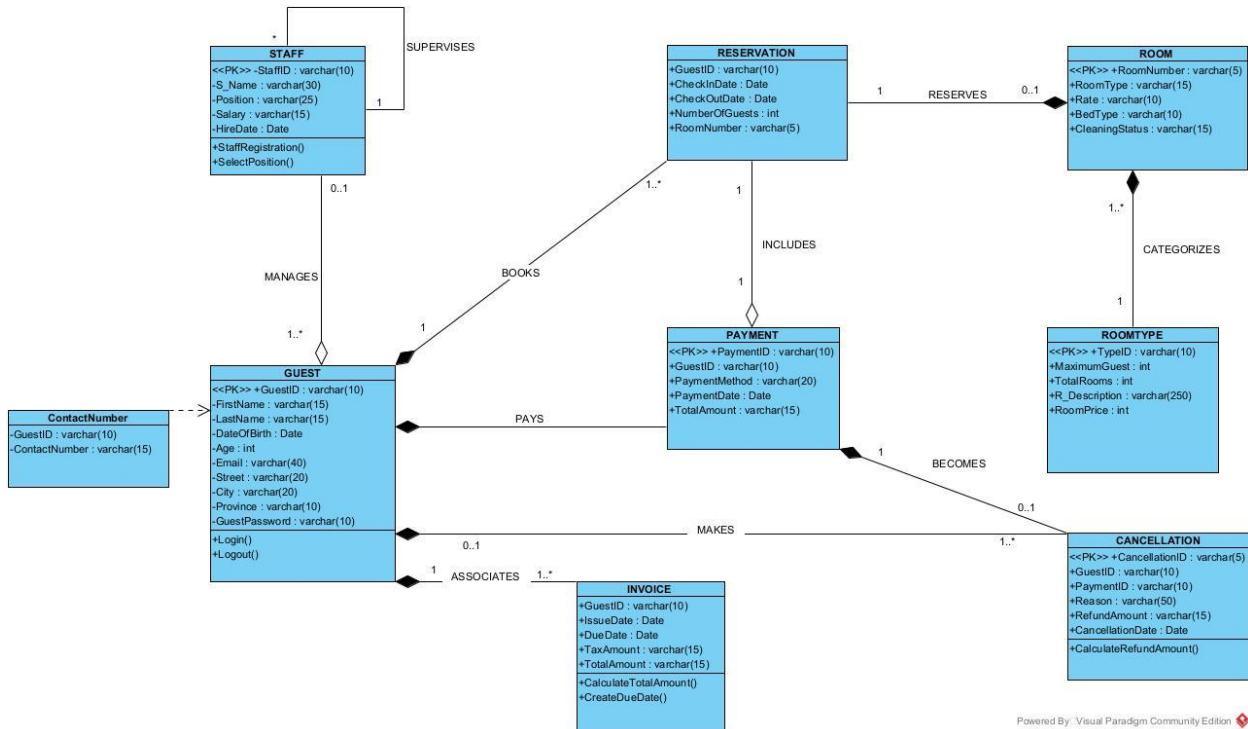


Figure 2: UML Diagram

Relationships of ER Diagram

- Guest-Reservation Relationship: A guest can make multiple reservations, but each reservation is made by one guest.
- Reservation-Rooms Relationship: A reservation can include multiple rooms, and each room can be included in multiple reservations.
- Room-Type of Rooms Relationship: Each room belongs to one type, and each type can have multiple rooms.
- Service Requested by Guests Relationship: Each service is requested by one guest, but a guest can request multiple services.
- Bill-Paid by Guest Relationship: Each bill is paid by one guest, but a guest can pay multiple bills.
- Bill-Service Charges Relationship: A bill includes charges for multiple services, and each service charge can be included in multiple bills.

Recursive Relationship:

The recursive relationship for the STAFF attributes with the SUPERVISES relationship can be described as follows:

Entity: Staff Attribute:

- StaffID (Primary key)
- Name
- Position
- Salary
- Hiredate

Relationships:

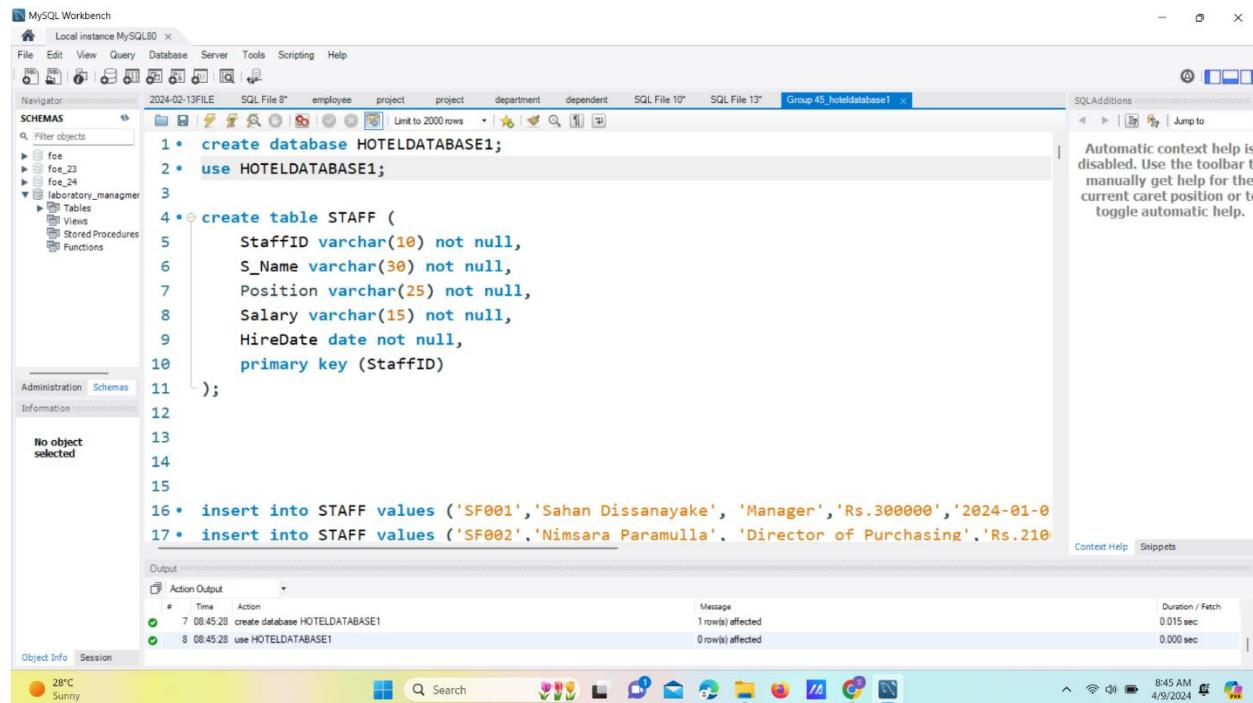
- Admin (foreign key referencing StaffID) - This field represents the admin of the current staff member.
- Staff Members - This field represents the Staff members who are working in the hotel
- Explanation: In this recursive relationship, the Staff entity relates to itself through the SUPERVISES relationship. Each staff member can act as both an admin and a staff member.

This recursive relationship allows for the representation of hierarchical structures within the staff organization, where staff members can admin and can themselves act as staff members for other staff members. This can be applied to both regular staff members and administrative staff.

Chapter 3 – Implementation

Schema Creation

Creating Database



The screenshot shows the MySQL Workbench interface. In the center, there is a SQL editor window containing the following SQL code:

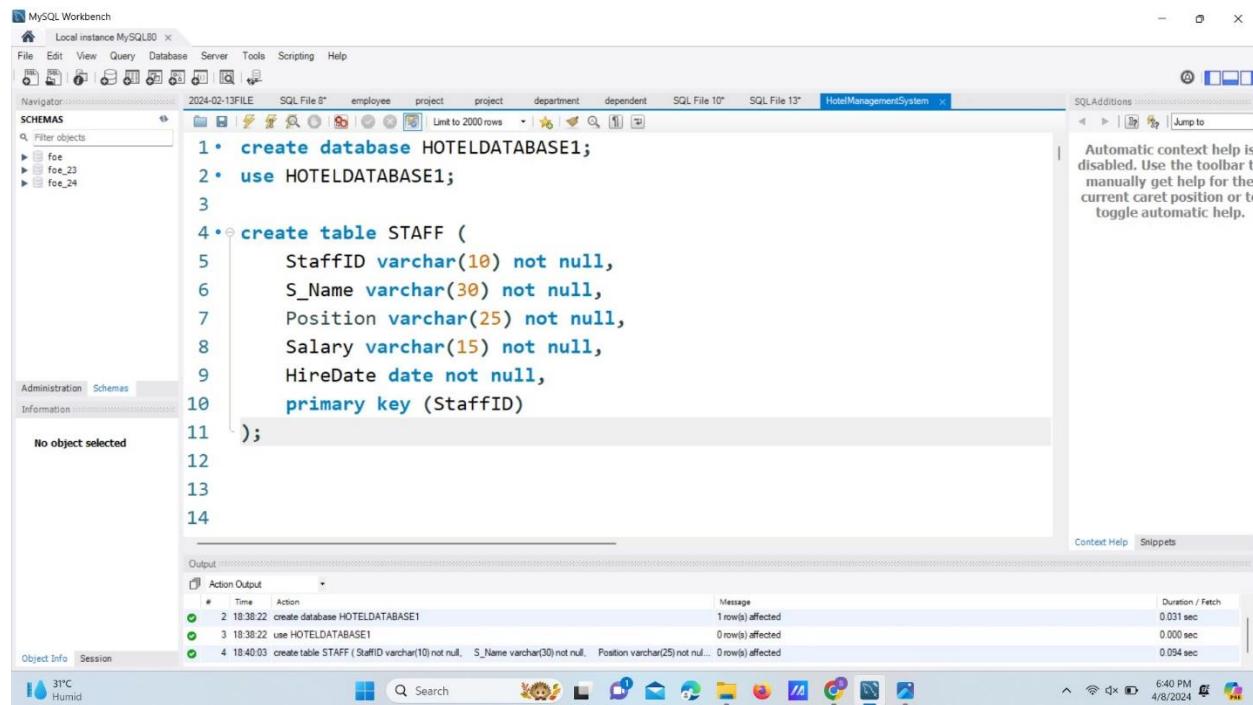
```
1 • create database HOTELDATABASE1;
2 • use HOTELDATABASE1;
3
4 • create table STAFF (
5     StaffID varchar(10) not null,
6     S_Name varchar(30) not null,
7     Position varchar(25) not null,
8     Salary varchar(15) not null,
9     HireDate date not null,
10    primary key (StaffID)
11 );
12
13
14
15
16 • insert into STAFF values ('SF001','Sahan Dissanayake', 'Manager', 'Rs.300000', '2024-01-01');
17 • insert into STAFF values ('SF002','Nimsara Paramulla', 'Director of Purchasing', 'Rs.210000', '2024-01-01');
```

Below the SQL editor, the 'Output' pane shows the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
1	7 08:45:28	create database HOTELDATABASE1	1 row(s) affected	0.015 sec
2	8 08:45:28	use HOTELDATABASE1	0 row(s) affected	0.000 sec

Table Definitions

Staff Table



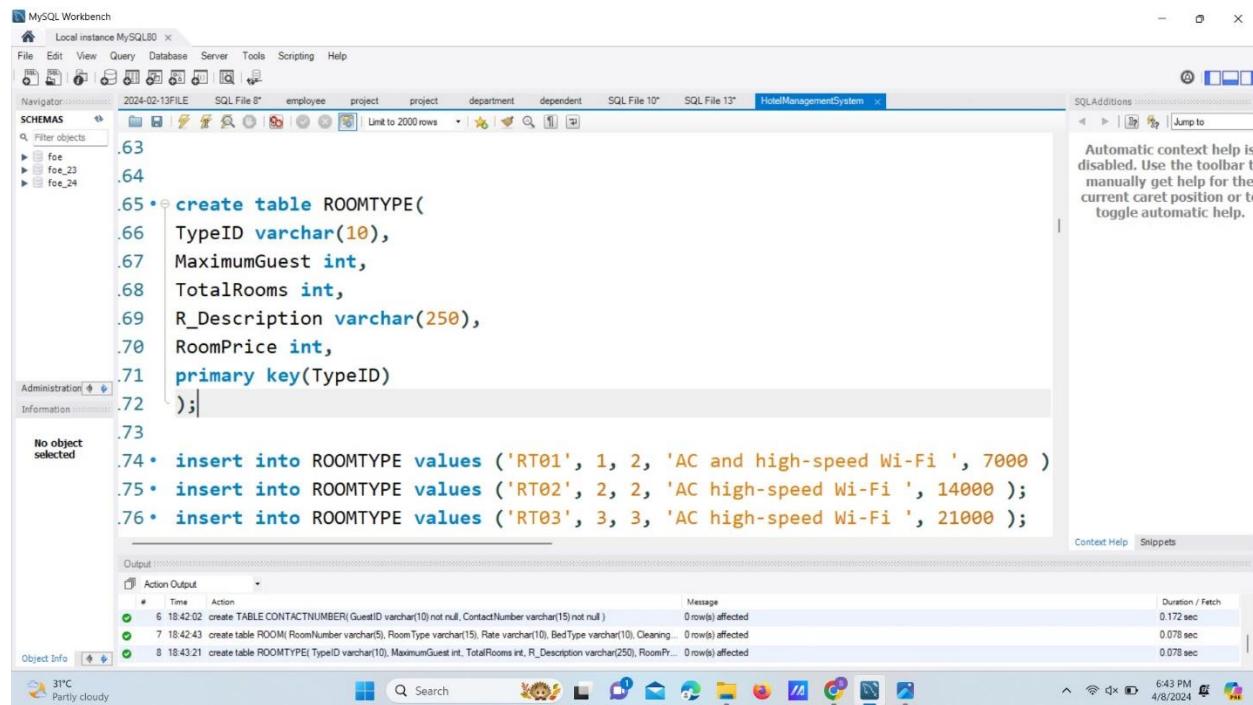
The screenshot shows the MySQL Workbench interface with the SQL tab active. The code pane displays the creation of the STAFF table:

```
1 • create database HOTELDATABASE1;
2 • use HOTELDATABASE1;
3
4 • create table STAFF (
5     StaffID varchar(10) not null,
6     S_Name varchar(30) not null,
7     Position varchar(25) not null,
8     Salary varchar(15) not null,
9     HireDate date not null,
10    primary key (StaffID)
11 );
12
13
14
```

The output pane shows the execution results:

Action	Time	Message	Duration / Fetch
create database HOTELDATABASE1	2 18:38:22	1 row(s) affected	0.031 sec
use HOTELDATABASE1	3 18:38:22	0 row(s) affected	0.000 sec
create table STAFF (StaffID varchar(10) not null, S_Name varchar(30) not null, Position varchar(25) not null, Salary varchar(15) not null, HireDate date not null, primary key (StaffID));	4 18:40:03	0 row(s) affected	0.094 sec

Room Type Table



The screenshot shows the MySQL Workbench interface with the SQL tab active. The code pane displays the creation of the ROOMTYPE table and three insert statements:

```
.63
.64
.65 • create table ROOMTYPE(
.66    TypeID varchar(10),
.67     MaximumGuest int,
.68     TotalRooms int,
.69     R_Description varchar(250),
.70     RoomPrice int,
.71     primary key(TypeID)
.72 );
.73
.74 • insert into ROOMTYPE values ('RT01', 1, 2, 'AC and high-speed Wi-Fi ', 7000 )
.75 • insert into ROOMTYPE values ('RT02', 2, 2, 'AC high-speed Wi-Fi ', 14000 );
.76 • insert into ROOMTYPE values ('RT03', 3, 3, 'AC high-speed Wi-Fi ', 21000 );
```

The output pane shows the execution results:

Action	Time	Message	Duration / Fetch
create TABLE CONTACTNUMBER(GuestID varchar(10) not null, ContactNumber varchar(15) not null)	6 18:42:02	0 row(s) affected	0.172 sec
create table ROOM(RoomNumber varchar(5), RoomType varchar(15), Rate varchar(10), BedType varchar(10), Cleaning ... 0 row(s) affected	7 18:42:43	0 row(s) affected	0.078 sec
create table ROOMTYPE(TypeID varchar(10),MaximumGuest int, TotalRooms int, R_Description varchar(250), RoomPr... 0 row(s) affected	8 18:43:21	0 row(s) affected	0.078 sec

Room Table

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code being run is:

```
.27
.28 • create table ROOM(
.29     RoomNumber varchar(5),
.30     RoomType varchar(15),
.31     Rate varchar(10),
.32     BedType varchar(10),
.33     CleaningStatus varchar(15),
.34     primary key(RoomNumber)
.35 );
.36
.37
.38 • insert into ROOM values ('RN01', 'AC', '8.2', 'Quad', 'Clean' );
.39 • insert into ROOM values ('RN02', 'Non AC', '6.5', 'Quad', 'Clean' );
.40 • insert into ROOM values ('RN03', 'AC', '9.3', 'Triple', 'Clean' );
```

The output pane shows the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
5	18:41:24	create table GUEST (GuestID varchar(10) not null, FirstName varchar(15) not null, LastName varchar(15) not null, Date...	0 row(s) affected	0.078 sec
6	18:42:02	create TABLE CONTACTNUMBER(GuestID varchar(10) not null, ContactNumber varchar(15) not null)	0 row(s) affected	0.172 sec
7	18:42:43	create table ROOM(RoomNumber varchar(5), RoomType varchar(15), Rate varchar(10), BedType varchar(10), Cleaning...	0 row(s) affected	0.078 sec

Reservation Table

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code being run is:

```
.93 • DELETE FROM ROOMTYPE
.94 WHERE TypeID = 'RT06';
.95
.96
.97 • create table RESERVATION (
.98     GuestID varchar(10) not null,
.99     CheckInDate date not null,
.100    CheckOutDate date,
.101    NumberOfGuests int,
.102    RoomNumber varchar(5)
.103 );
.104
.105
.106 • ALTER table RESERVATION
```

The output pane shows the results of the executed statements:

#	Time	Action	Message	Duration / Fetch
7	18:42:43	create table ROOM(RoomNumber varchar(5), RoomType varchar(15), Rate varchar(10), BedType varchar(10), Cleaning...	0 row(s) affected	0.078 sec
8	18:43:21	create table ROOMTYPE(TypeID varchar(10), MaximumGuest int, TotalRooms int, _R_Description varchar(250), RoomPr...	0 row(s) affected	0.078 sec
9	18:43:14	create table RESERVATION (GuestID varchar(10) not null, CheckInDate date not null, CheckOutDate date, NumberOf...	0 row(s) affected	0.062 sec

Payment Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the SQL editor tab, the following SQL code is being run:

```
!71 WHERE GuestID = 'GT005';
!72
!73
!74 • create table PAYMENT(
!75     PaymentID varchar(10) not null,
!76     GuestID varchar(10) not null,
!77     PaymentMethod varchar(20) not null,
!78     PaymentDate date,
!79     TotalAmount varchar(15),
!80     primary key(PaymentID)
!81
!82 );
!83
!84 • ALTER TABLE PAYMENT
```

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

Action	Time	Action	Message	Duration / Fetch
9	18:44:14	create table RESERVATION (GuestID varchar(10) not null, CheckInDate date not null, CheckOutDate date, NumberOf...)	0 row(s) affected	0.062 sec
10	18:44:50	create table INVOICE(GuestID varchar(10) not null, IssueDate date, DueDate date, TaxAmount varchar(15), TotalAmou...)	0 row(s) affected	0.125 sec
11	18:45:36	create table PAYMENT(PaymentID varchar(10) not null, GuestID varchar(10) not null, PaymentMethod varchar(20) not ...)	0 row(s) affected	0.062 sec

Invoice Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the SQL editor tab, the following SQL code is being run:

```
!35
!36 • create table INVOICE(
!37     GuestID varchar(10) not null,
!38     IssueDate date,
!39     DueDate date,
!40     TaxAmount varchar(15),
!41     TotalAmount varchar(15)
!42
!43 );
!44
!45 • ALTER TABLE INVOICE
!46 ADD constraint FK_INVOICE_GUEST foreign key (GuestID) REFERENCES GUEST(GuestID)
!47
!48 • insert into INVOICE values ('GT001', '2024-03-01', '2024-03-03', 'Rs.4000', '100')
```

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

Action	Time	Action	Message	Duration / Fetch
8	18:43:21	create table ROOMTYPE(TypeID varchar(10), MaximumGuest int, TotalRooms int, R_Description varchar(250), RoomPr...	0 row(s) affected	0.078 sec
9	18:44:14	create table RESERVATION (GuestID varchar(10) not null, CheckInDate date not null, CheckOutDate date, NumberOf...)	0 row(s) affected	0.062 sec
10	18:44:50	create table INVOICE(GuestID varchar(10) not null, IssueDate date, DueDate date, TaxAmount varchar(15), TotalAmou...)	0 row(s) affected	0.125 sec

Guest Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the SQL editor tab, the following SQL code is being run:

```
43
44 • create table GUEST (
45   GuestID varchar(10) not null,
46   FirstName varchar(15) not null,
47   LastName varchar(15) not null,
48   DateOfBirth date ,
49   Age int ,
50   Email varchar(40),
51   Street varchar(20),
52   City varchar(20),
53   Province varchar(20),
54   GuestPassword varchar(10),
55   primary key(GuestID)
56 );
```

The 'Output' pane shows the results of the execution:

Action	Time	Action	Message	Duration / Fetch
use HOTELDATABASE	10:38:22		0 row(s) affected	0.000 sec
create table STAFF	18:40:03	(StaffID varchar(10) not null, S_Name varchar(30) not null, Position varchar(25) not null, Sal...)	0 row(s) affected	0.094 sec
create table GUEST	18:41:24	(GuestID varchar(10) not null, FirstName varchar(15) not null, LastName varchar(15) not null, Date...)	0 row(s) affected	0.078 sec

Contact Number Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the SQL editor tab, the following SQL code is being run:

```
88
89 • create TABLE CONTACTNUMBER(
90   GuestID varchar(10) not null,
91   ContactNumber varchar(15) not null
92 );
93
94 • INSERT INTO CONTACTNUMBER (GuestID, ContactNumber)
95   VALUES ('GT001' , '0774711139'), ('GT001' , '0711777544' ),
96   ('GT002' , '0785524796' ), ('GT002' , '0725464569' ),
97   ('GT003' , '0701256935' ), ('GT003' , '0725465567' ),
98   ('GT004' , '0754125569' ), ('GT004' , '0725465098' ),
99   ('GT005' , '0761211456' ), ('GT005' , '0725467649'),
.00   ('GT006' , '0746558921'), ('GT006' , '0725465561' ),
.01   ('GT007' , '0701215588'), ('GT007' , '0725465701'),
```

The 'Output' pane shows the results of the execution:

Action	Time	Action	Message	Duration / Fetch
create table STAFF	18:40:03	(StaffID varchar(10) not null, S_Name varchar(30) not null, Position varchar(25) not null, Sal...)	0 row(s) affected	0.094 sec
create table GUEST	18:41:24	(GuestID varchar(10) not null, FirstName varchar(15) not null, LastName varchar(15) not null, Date...)	0 row(s) affected	0.078 sec
create TABLE CONTACTNUMBER	18:42:02	(GuestID varchar(10) not null, ContactNumber varchar(15) not null)	0 row(s) affected	0.172 sec

Cancellation Table

The screenshot shows the MySQL Workbench interface with the following details:

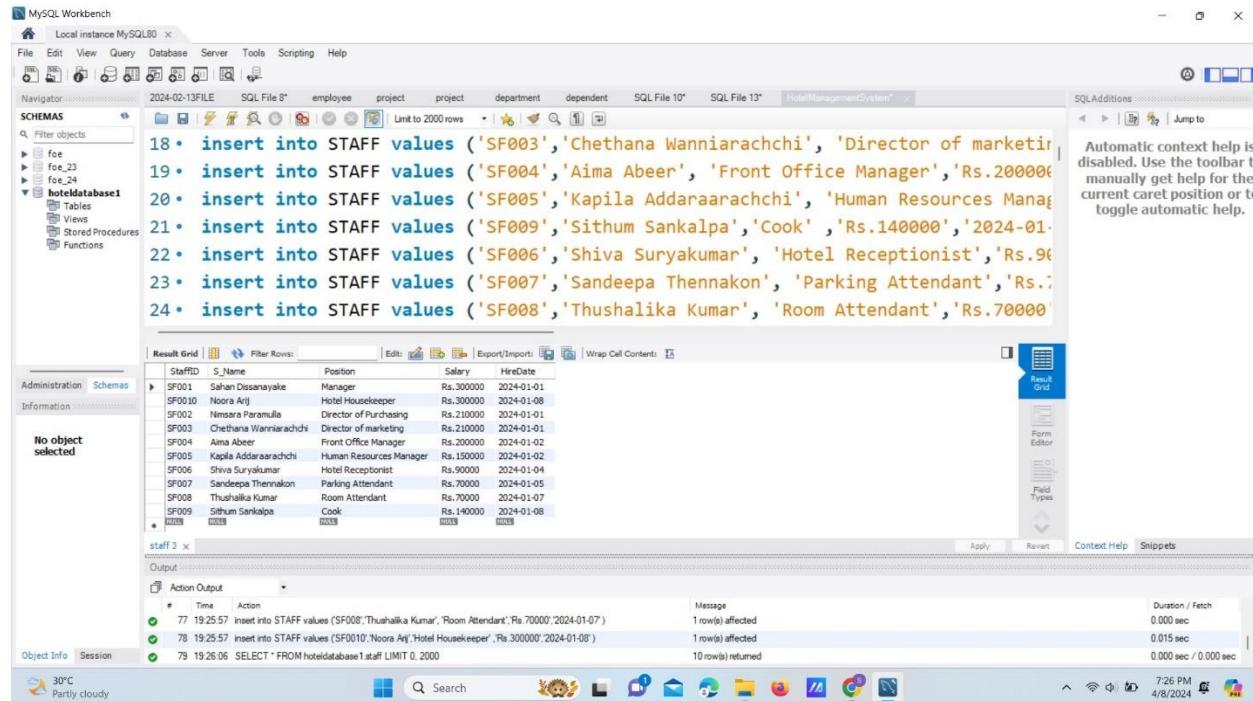
- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, employee, project, department, dependent, SQL File 10*, SQL File 13*, HotelManagementSystem.
- Code Editor:** SQL pane showing the creation of the CANCELLATION table.

```
|:10
|:11 • SELECT * FROM hoteldatabase.PAYMENT;
|:12
|:13 • create table CANCELLATION (
|:14     CancellationID varchar(5),
|:15     GuestID varchar(10) not null,
|:16     PaymentID varchar(10) not null,
|:17     Reason varchar(50),
|:18     RefundAmount varchar(15),
|:19     CancellationDate date,
|:20     primary key(CancellationID)
|:21 );|
|:22
|:23 • ALTER TABLE CANCELLATION
```
- Output Panel:** Action Output table showing the execution of three statements:

Action	Time	Message	Duration / Fetch
create table INVOICE	10 18:44:50	GuestID varchar(10) not null, IssueDate date, DueDate date, TaxAmount varchar(15), TotalAmou... 0 row(s) affected	0.125 sec
create table PAYMENT	11 18:45:36	PaymentID varchar(10) not null, GuestID varchar(10) not null, PaymentMethod varchar(20) not ... 0 row(s) affected	0.062 sec
create table CANCELLATION	12 18:49:11	(CancellationID varchar(5), GuestID varchar(10) not null, PaymentID varchar(10) not null, ... 0 row(s) affected	0.110 sec
- System Bar:** Shows the date (4/8/2024), time (6:48 PM), and weather (31°C Partly cloudy).

Inserting Data for Tables

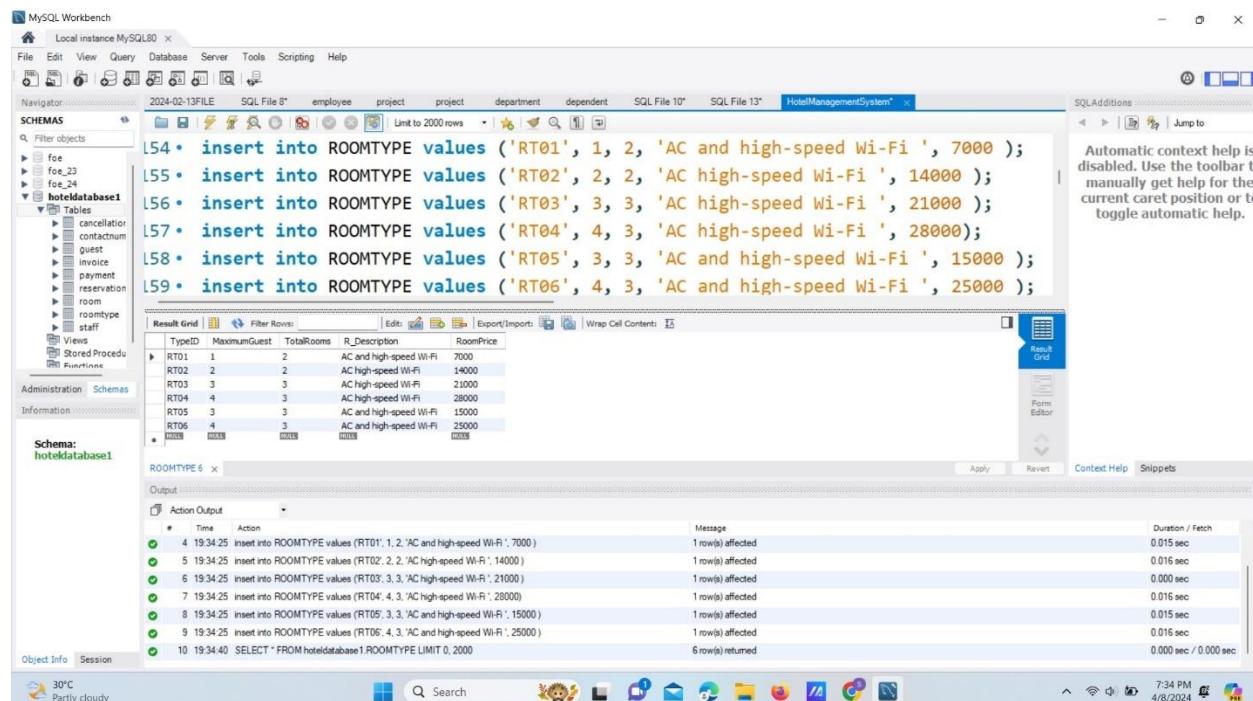
Staff Table



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- SQL Editor:** Contains 24 SQL INSERT statements for the STAFF table, each adding a new staff member with their name, position, salary, and hire date.
- Result Grid:** Displays the inserted data in a table format. The columns are StaffID, S_Name, Position, Salary, and HrDate. The data includes entries for Sahan Dissanayake, Noora Arji, Chethana Wanniarachchi, Aima Abeer, Kapila Addaraarachchi, Sithum Sankalpa, Shiva Suryakumar, Sandeepa Thennakon, and Thushalika Kumar.
- Action Output:** Shows the log of actions taken, including the insertion of rows and a SELECT statement.
- System Bar:** Shows the system tray with icons for battery, signal, and time (7:26 PM).

Room Type Table



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- SQL Editor:** Contains 10 SQL INSERT statements for the ROOMTYPE table, each adding a room type with its ID, maximum guests, total rooms, description, and price.
- Result Grid:** Displays the inserted data in a table format. The columns are TypeID, MaximumGuest, TotalRooms, R_Description, and RoomPrice. The data includes entries for RT01 through RT06.
- Action Output:** Shows the log of actions taken, including the insertion of rows and a SELECT statement.
- System Bar:** Shows the system tray with icons for battery, signal, and time (7:34 PM).

Room Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the Navigator pane, under the 'hoteldatabase1' schema, the 'Tables' node is expanded, showing tables like 'Room', 'roomtype', and 'staff'. The 'Room' table is currently selected. The SQL Editor pane contains a series of INSERT statements for the 'ROOM' table:

```
L18 • insert into ROOM values ('RN01', 'AC', '8.2', 'Quad', 'Clean');
L19 • insert into ROOM values ('RN02', 'Non AC', '6.5', 'Quad', 'Clean');
L20 • insert into ROOM values ('RN03', 'AC', '9.3', 'Triple', 'Clean');
L21 • insert into ROOM values ('RN04', 'AC', '10.0', 'Double', 'Clean');
L22 • insert into ROOM values ('RN05', 'Non AC', '8.9', 'Triple', 'Clean');
L23 • insert into ROOM values ('RN06', 'AC', '9.0', 'Single', 'Clean');
L24 • insert into ROOM values ('RN07', 'Non AC', '7.8', 'Triple', 'Clean');
L25 • insert into ROOM values ('RN08', 'AC', '8.7', 'Quad', 'Clean');
L26 • insert into ROOM values ('RN09', 'AC', '9.8', 'Single', 'Clean');
```

The Result Grid pane displays the inserted data:

RoomNumber	RoomType	Rate	BedType	CleaningStatus
RN01	AC	8.2	Quad	Clean
RN02	Non AC	6.5	Double	Clean
RN03	Non AC	6.5	Quad	Clean
RN04	AC	9.3	Triple	Clean
RN05	AC	10.0	Double	Clean
RN06	AC	8.9	Triple	Clean
RN07	Non AC	7.8	Single	Clean
RN08	AC	9.0	Quad	Clean
RN09	AC	8.7	Single	Clean
RN010	AC	9.8	Triple	Clean

The Output pane shows the execution log:

- 54 19:20:54 Insert into ROOM values ('RN010', 'Non AC', '7.8', 'Double', 'Clean') Message 1 row(s) affected Duration / Fetch 0.016 sec
- 55 19:21:22 SELECT * FROM hoteldatabase1.ROOM LIMIT 0, 2000 Error Code: 1049 Unknown database 'hoteldatabase1' 0.000 sec
- 56 19:21:33 SELECT * FROM hoteldatabase1.ROOM LIMIT 0, 2000 10 row(s) returned 0.000 sec / 0.000 sec

Reservation Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the Navigator pane, under the 'hoteldatabase1' schema, the 'Tables' node is expanded, showing tables like 'RESERVATION', 'roomtype', and 'staff'. The 'RESERVATION' table is currently selected. The SQL Editor pane contains a series of INSERT statements for the 'RESERVATION' table:

```
L192 • insert into RESERVATION values ('GT002', '2024-03-02', '2024-03-05', 4, 'RN02');
L193 • insert into RESERVATION values ('GT003', '2024-03-04', '2024-03-05', 2, 'RN04');
L194 • insert into RESERVATION values ('GT004', '2024-03-02', '2024-03-05', 3, 'RN03');
L195 • insert into RESERVATION values ('GT005', '2024-03-05', '2024-03-06', 4, 'RN01');
L196 • insert into RESERVATION values ('GT006', '2024-03-04', '2024-03-06', 3, 'RN05');
L197 • insert into RESERVATION values ('GT007', '2024-03-02', '2024-03-06', 2, 'RN010');
L198 • insert into RESERVATION values ('GT008', '2024-03-07', '2024-03-08', 4, 'RN01');
L199 • insert into RESERVATION values ('GT009', '2024-03-10', '2024-03-12', 3, 'RN07');
```

The Result Grid pane displays the inserted data:

GuestID	CheckInDate	CheckOutDate	NumberOfGuests	RoomNumber
GT001	2024-03-01	2024-03-03	4	RN01
GT002	2024-03-02	2024-03-05	4	RN02
GT003	2024-03-04	2024-03-05	2	RN04
GT004	2024-03-02	2024-03-05	3	RN03
GT005	2024-03-05	2024-03-06	4	RN01
GT006	2024-03-04	2024-03-06	3	RN05
GT007	2024-03-02	2024-03-06	2	RN010
GT008	2024-03-07	2024-03-08	4	RN01
GT009	2024-03-10	2024-03-12	3	RN07
GT010	2024-03-13	2024-03-15	4	RN08

The Output pane shows the execution log:

- 57 19:46:22 Insert into RESERVATION values ('GT0010', '2024-03-13', '2024-03-15', 4, 'RN08') Message 1 row(s) affected Duration / Fetch 0.016 sec
- 58 19:46:35 SELECT * FROM hoteldatabase1.RESERVATION LIMIT 0, 2000 10 row(s) returned 0.000 sec / 0.000 sec

Payment Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. The 'Tables' section of the Navigator pane shows the 'reservation' table. The SQL editor pane displays the following SQL code for inserting data into the 'PAYMENT' table:

```
268 • insert into PAYMENT values ('GP001', 'GT001', 'Cash Payment', '2024-03-01', 'Rs.28000');
269 • insert into PAYMENT values ('GP002', 'GT002', 'Credit Card', '2024-03-02', 'Rs.14000');
270 • insert into PAYMENT values ('GP003', 'GT004', 'Cash Payment', '2024-03-02', 'Rs.14000');
271 • insert into PAYMENT values ('GP004', 'GT007', 'Debit Card', '2024-03-02', 'Rs.21000');
272 • insert into PAYMENT values ('GP005', 'GT003', 'Credit Card', '2024-03-04', 'Rs.28000');
273 • insert into PAYMENT values ('GP006', 'GT006', 'Credit Card', '2024-03-04', 'Rs.21000');
274 • insert into PAYMENT values ('GP007', 'GT005', 'Cash Payment', '2024-03-05', 'Rs.28000');
275 • insert into PAYMENT values ('GP008', 'GT008', 'Cash Payment', '2024-03-07', 'Rs.28000');
276 • insert into PAYMENT values ('GP009', 'GT009', 'Cash Payment', '2024-03-10', 'Rs.21000');
277 • insert into PAYMENT values ('GP0010', 'GT010', 'Credit Card', '2024-03-13', 'Rs.28000');
```

The Result Grid pane shows the following data for the 'PAYMENT' table:

PaymentID	GuestID	PaymentMethod	PaymentDate	TotalAmount
GP001	GT001	Cash Payment	2024-03-01	Rs.28000
GP0010	GT0010	Credit Card	2024-03-13	Rs.28000
GP002	GT002	Credit Card	2024-03-02	Rs.14000
GP003	GT004	Cash Payment	2024-03-02	Rs.14000
GP004	GT007	Debit Card	2024-03-02	Rs.21000
GP005	GT003	Credit Card	2024-03-04	Rs.28000
GP006	GT006	Credit Card	2024-03-04	Rs.21000
GP007	GT005	Cash Payment	2024-03-05	Rs.28000
GP008	GT008	Cash Payment	2024-03-07	Rs.28000
GP009	GT009	Cash Payment	2024-03-10	Rs.21000
GP0010	GT010	Credit Card	2024-03-13	Rs.28000

Invoice Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. The 'Tables' section of the Navigator pane shows the 'reservation' table. The SQL editor pane displays the following SQL code for inserting data into the 'INVOICE' table:

```
227 ADD constraint FK_INVOICE_GUEST foreign key (GuestID) REFERENCES GUEST(GuestID) ON DELETE
228
229 • insert into INVOICE values ('GT001', '2024-03-01', '2024-03-03', 'Rs.4000', 'Rs.32000')
230 • insert into INVOICE values ('GT002', '2024-03-02', '2024-03-04', 'Rs.4000', 'Rs.32000')
231 • insert into INVOICE values ('GT003', '2024-03-04', '2024-03-06', 'Rs.4000', 'Rs.32000')
232 • insert into INVOICE values ('GT004', '2024-03-02', '2024-03-04', 'Rs.2000', 'Rs.16000')
233 • insert into INVOICE values ('GT005', '2024-03-05', '2024-03-07', 'Rs.4000', 'Rs.32000')
234 • insert into INVOICE values ('GT006', '2024-03-04', '2024-03-06', 'Rs.3000', 'Rs.24000')
235 • insert into INVOICE values ('GT007', '2024-03-02', '2024-03-04', 'Rs.3000', 'Rs.24000')
236 • insert into INVOICE values ('GT008', '2024-03-07', '2024-03-09', 'Rs.4000', 'Rs.32000')
237 • insert into INVOICE values ('GT009', '2024-03-10', '2024-03-12', 'Rs.3000', 'Rs.24000')
238 • insert into INVOICE values ('GT0010', '2024-03-13', '2024-03-15', 'Rs.4000', 'Rs.32000')
```

The Result Grid pane shows the following data for the 'INVOICE' table:

GuestID	IssueDate	DueDate	TaxAmount	TotalAmount
GT001	2024-03-01	2024-03-03	Rs.4000	Rs.32000
GT002	2024-03-02	2024-03-04	Rs.4000	Rs.32000
GT003	2024-03-04	2024-03-06	Rs.4000	Rs.32000
GT004	2024-03-02	2024-03-04	Rs.2000	Rs.16000
GT005	2024-03-05	2024-03-07	Rs.4000	Rs.32000
GT006	2024-03-04	2024-03-06	Rs.3000	Rs.24000
GT007	2024-03-02	2024-03-04	Rs.3000	Rs.24000
GT008	2024-03-07	2024-03-09	Rs.4000	Rs.32000
GT009	2024-03-10	2024-03-12	Rs.3000	Rs.24000
GT0010	2024-03-13	2024-03-15	Rs.4000	Rs.32000

Guest Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. The SQL editor tab contains the following SQL code:

```
63 • insert into GUEST values ('GT003', 'Nadhanan', 'Jabeerkish', '2001-05-20', 23, 'r
64 • insert into GUEST values ('GT004', 'Nafla', 'Naleef', '2000-03-17', 24, 'nafla
65 • insert into GUEST values ('GT005', 'Niwarthana', 'Madushani', '1996-04-08', 28,
66 • insert into GUEST values ('GT006', 'Gagani', 'Wedamulla', '1997-08-15', 27, 'gaga
67 • insert into GUEST values ('GT007', 'Mithurshan', 'Rajalingum', '1998-11-09', 26
68 • insert into GUEST values ('GT008', 'Sadun', 'Chamara', '1999-12-24', 25, 'sadun
69 • insert into GUEST values ('GT009', 'Minosh', 'Ahamed', '1999-01-20', 25, 'minosh
70 • insert into GUEST values ('GT010', 'Anne', 'Fernando', '2002-11-06', 22, 'anne1
71
```

The results grid displays the following data:

GuestID	FirstName	LastName	DateOfBirth	Age	Email	Street	City	Province	GuestPassword
GT001	Sampika	Kothalawala	1996-05-25	28	sampika.kothalawala@gmail.com	Lake Road	Anuradhapura	North Central	s123
GT010	Anne	Fernando	2002-11-06	22	annefernando@gmail.com	Villa Street	Puttalam	North Western	f455
GT003	Sandeepa	Muthukumar	2002-01-11	22	sandeepamuthukumar@gmail.com	Albon Road	Rathnapura	Uva	m128
GT003	Nadhanan	Jabeerkish	2001-05-20	23	nadhjan.jabeerkish@gmail.com	Armour Street	Kandy	Central	n222
GT004	Nafla	Naleef	2000-03-17	24	naflaaleef@gmail.com	Berke Street	Galle	Southern	n345
GT005	Niwarthana	Madushani	1996-04-08	28	niwarthanamadushani@gmail.com	Church Street	Colombo	Western	m000
GT006	Gagani	Wedamulla	1997-08-15	27	gaganawedamulla@gmail.com	Darley Road	Badulla	Uva	g454

The output pane shows the following action output:

#	Time	Action	Message	Duration / Fetch
1	19:28:28	SELECT * FROM hoteldatabase1.GUEST LIMIT 0, 2000	10 row(s) returned	0.000 sec / 0.000 sec

Contact Number Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. The SQL editor tab contains the following SQL code:

```
94 • INSERT INTO CONTACTNUMBER (GuestID, ContactNumber)
95 VALUES ('GT001', '0774711139'), ('GT001', '0711777544'),
96 ('GT002', '0785524796'), ('GT002', '0725464569'),
97 ('GT003', '0701256935'), ('GT003', '0725465567'),
98 ('GT004', '0754125569'), ('GT004', '0725465098'),
99 ('GT005', '0761211456'), ('GT005', '0725467649'),
100 ('GT006', '0746558921'), ('GT006', '0725465561'),
101 ('GT007', '0701215588'), ('GT007', '0725465701'),
102 ('GT008', '0785566987'), ('GT008', '0725465000'),
103 ('GT009', '0725465569'), ('GT009', '0725465090').
```

The results grid displays the following data:

GuestID	ContactNumber
GT001	0774711139
GT001	0711777544
GT002	0785524796
GT002	0725464569
GT003	0701256935
GT003	0725465567

The output pane shows the following action output:

#	Time	Action	Message	Duration / Fetch
2	19:30:27	INSERT INTO CONTACTNUMBER (GuestID, ContactNumber) VALUES ('GT001', '0774711139'), ('GT001', '0711777544')	20 rows affected Records: 20 Duplicates: 0 Warnings: 0	0.000 sec
3	19:31:00	SELECT * FROM hoteldatabase1.CONTACTNUMBER LIMIT 0, 2000	20 row(s) returned	0.000 sec / 0.000 sec

Cancellation Table

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: 2024-02-13FILE SQL File 8* employee project project department dependent SQL File 10* SQL File 13* HotelManagementSystem*

Schemas: Filter objects

hoteldatabase1 Tables: cancellation contractroom guest invoice payment reservation room roomtype staff Views Stored Procedures Functions

304 • ALTER TABLE CANCELLATION

305 ADD constraint FK_CANCELLATION_GUEST foreign key (GuestID) REFERENCES GUEST(GuestID) ON DELETE CASCADE ON UPDATE CASCADE

306 ADD constraint FK_CANCELLATION_PAYMENT foreign key (PaymentID) REFERENCES PAYMENT(PaymentID) ON DELETE CASCADE ON UPDATE CASCADE

307

308 • insert into CANCELLATION values ('C001', 'GT007', 'GP004', 'Weather Conditions', 'Rs.23000', '2024-03-03')

309 • insert into CANCELLATION values ('C002', 'GT005', 'GP007', 'Change the hotel', 'Rs.31000', '2024-03-03')

310 • insert into CANCELLATION values ('C003', 'GT009', 'GP009', 'Weather Conditions', 'Rs.23000', '2024-03-03')

311 • insert into CANCELLATION values ('C004', 'GT001', 'GP001', 'Weather Conditions', 'Rs.28000', '2024-03-01')

312 • insert into CANCELLATION values ('C005', 'GT002', 'GP002', 'Change the hotel', 'Rs.14000', '2024-03-02')

313 • insert into CANCELLATION values ('C006', 'GT004', 'GP003', 'Weather Conditions', 'Rs.14000', '2024-03-02')

314

315 -- Updating data in CANCELLATION table

Table: reservation

Columns: GuestID, CheckInDate, CheckOutDate, NumberOfGuests, RoomNumber

Result Grid | Filter Rows: | Edit: | Wrap Cell Content: |

CancellationID	GuestID	PaymentID	Reason	RefundAmount	CancellationDate
C001	GT007	GP004	Weather Conditions	Rs.23000	2024-03-03
C002	GT005	GP007	Change the hotel	Rs.31000	2024-03-03
C003	GT009	GP009	Weather Conditions	Rs.23000	2024-03-03
C004	GT001	GP001	Weather Conditions	Rs.28000	2024-03-01
C005	GT002	GP002	Change the hotel	Rs.14000	2024-03-02
C006	GT004	GP003	Weather Conditions	Rs.14000	2024-03-02

CANCELLATION 11 | Output | Action Output |

Object Info Session

30°C Partly cloudy

7:52 PM 4/8/2024

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Update Operations

Staff Table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `hoteldatabase1` with tables: cancellation, contractum, guest, invoice, payment, reservation, room, roomtype, staff.
- SQL Editor:** Contains the following SQL code:


```

28
29 -- Updating data in STAFF table
30 • UPDATE STAFF
31 SET S_Name = 'Amima Abishan', Position='Office Manager'
32 WHERE StaffID = 'SF004';
33
34 • UPDATE STAFF
35 SET S_Name = 'Sandeepa Hettigoda', Salary='Rs.80000'
36 WHERE StaffID = 'SF007';
      
```
- Result Grid:** Displays the updated data in the STAFF table. The table has columns: StaffID, S_Name, Position, Salary, HrDate. The data includes rows for StaffID SF001 to SF008, with the last two rows being the updates made in the SQL code.
- Output:** Shows the execution results of the UPDATE statements.

Room Type Table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `hoteldatabase1` with tables: cancellation, contractum, guest, invoice, payment, reservation, room, roomtype, staff.
- SQL Editor:** Contains the following SQL code:


```

180 • SELECT * FROM hoteldatabase1.ROOMTYPE;
181
182 -- Updating data in ROOMTYPE table
183 • UPDATE ROOMTYPE
184 SET R_Description='Non AC and high-speed Wi-Fi'
185 WHERE TypeID = 'RT05';
186
187 • UPDATE ROOMTYPE
188 SET R_Description='Non AC and high-speed Wi-Fi'
189 WHERE TypeID = 'RT06';
190
191 -- Deleting one row from the ROOMTYPE table
      
```
- Result Grid:** Displays the data in the ROOMTYPE table. The table has columns: TypeID, MaximumGuest, TotalRooms, R_Description, RoomPrice. The data includes rows for TypeID RT01 to RT06, with the last two rows being the updates made in the SQL code.
- Output:** Shows the execution results of the UPDATE and DELETE statements.

Room Table

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase
- Tables:** reservation
- SQL Editor:** Contains the following code:

```
L50 -- Updating data in ROOM table
L51 • UPDATE ROOM
L52 SET Rate = 8.8
L53 WHERE RoomNumber = 'RN01';
L54
L55 • UPDATE ROOM
L56 SET Rate = 8.7
L57 WHERE RoomNumber = 'RN07';
L58
L59 -- Deleting one row from the ROOM table
```
- Result Grid:** Shows the current state of the reservation table with 10 rows of data.

RoomNumber	RoomType	Rate	BedType	CleaningStatus
RN01	AC	8.8	Quad	Clean
RN10	Non AC	7.8	Double	Clean
RN02	Non AC	6.5	Quad	Clean
RN03	AC	9.3	Triple	Clean
RN04	AC	10.0	Double	Clean
RN05	Non AC	8.9	Triple	Clean
RN06	AC	9.0	Single	Clean
RN07	Non AC	8.7	Triple	Clean
RN08	AC	8.7	Quad	Clean

- Message Bar:** Displays a message about automatic context help being disabled.
- System Tray:** Shows the date and time as 8:49 PM on 4/8/2024.

Reservation Table

The screenshot shows the MySQL Workbench interface with the following details:

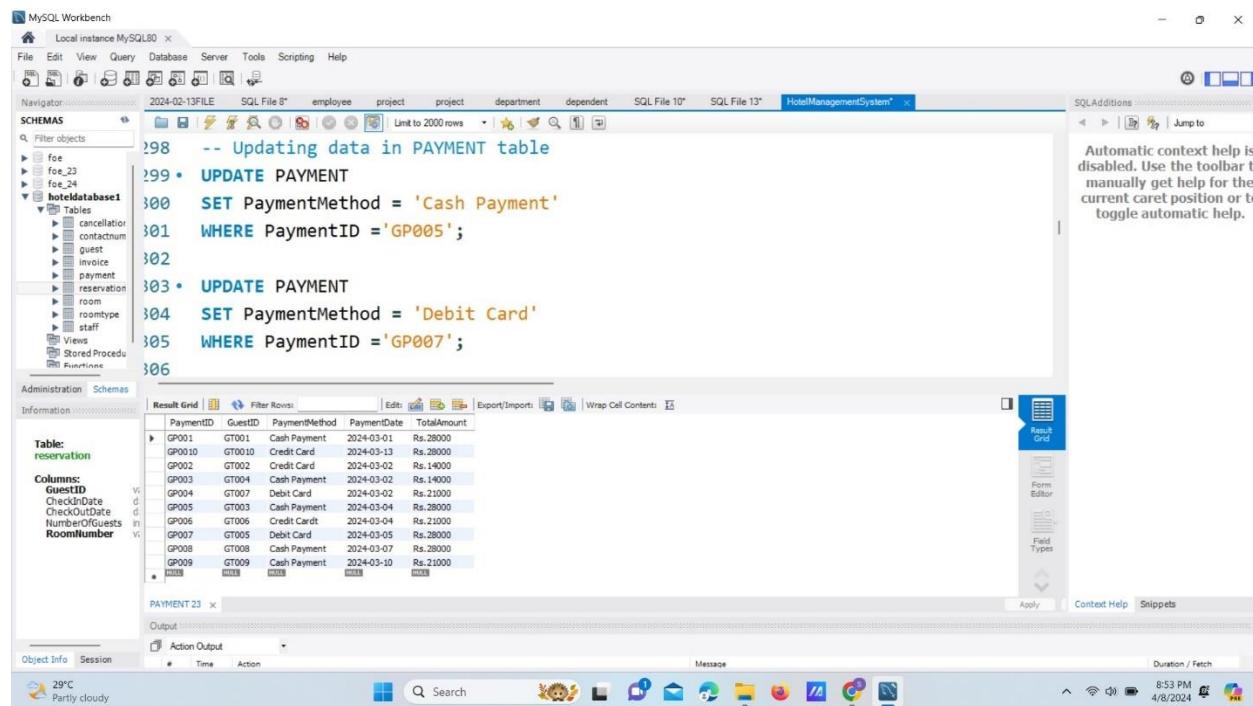
- Schemas:** hoteldatabase
- Tables:** reservation
- SQL Editor:** Contains the following code:

```
222
223 -- Updating data in RESERVATION table
224 • UPDATE RESERVATION
225 SET CheckInDate='2024-03-04', CheckOutDate='2024-03-05'
226 WHERE GuestID = 'GT005';
227
228 • UPDATE RESERVATION
229 SET CheckInDate='2024-03-09', CheckOutDate='2024-03-11'
230 WHERE GuestID = 'GT009';
```
- Result Grid:** Shows the current state of the reservation table with 10 rows of data.

GuestID	CheckInDate	CheckOutDate	NumberOfGuests	RoomNumber
GT001	2024-03-01	2024-03-03	4	RN01
GT002	2024-03-02	2024-03-05	4	RN02
GT003	2024-03-04	2024-03-05	2	RN04
GT004	2024-03-02	2024-03-05	3	RN03
GT005	2024-03-04	2024-03-05	4	RN01
GT006	2024-03-04	2024-03-06	3	RN05
GT007	2024-03-02	2024-03-06	2	RN010
GT008	2024-03-07	2024-03-08	4	RN01
GT009	2024-03-09	2024-03-11	3	RN07
GT0010	2024-03-13	2024-03-15	4	RN08

- Message Bar:** Displays a message about automatic context help being disabled.
- System Tray:** Shows the date and time as 8:52 PM on 4/8/2024.

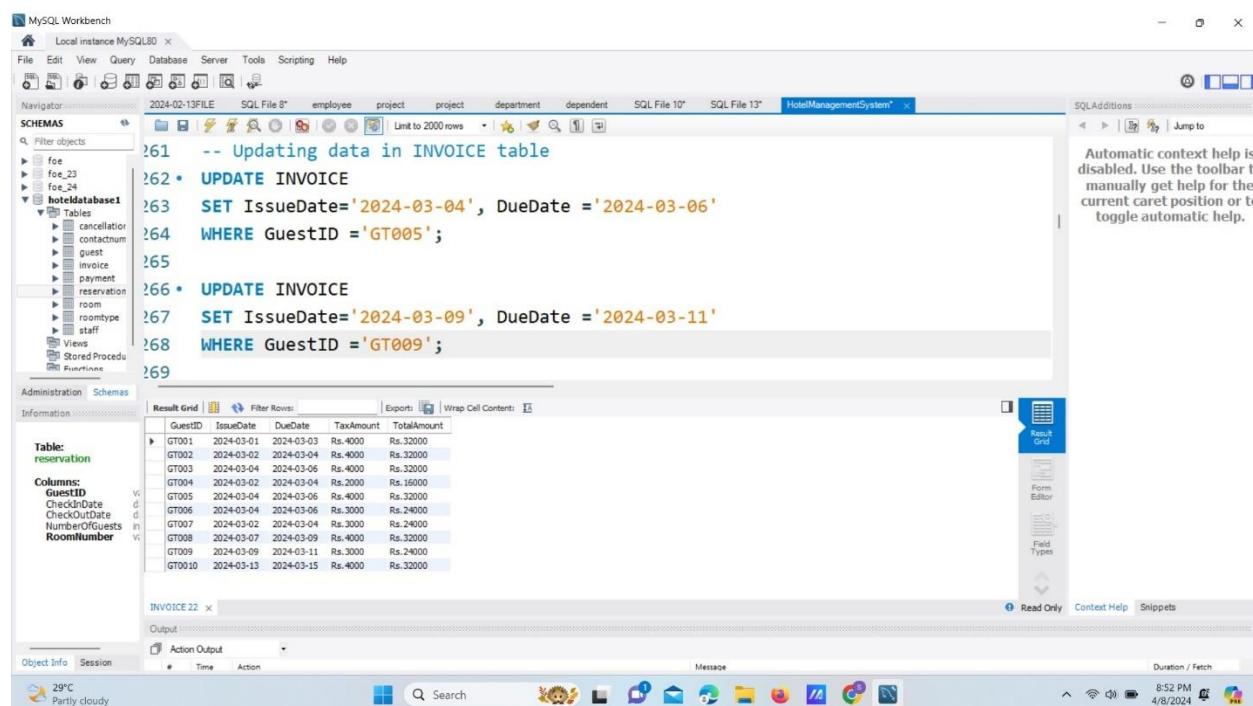
Payment Table



The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. The 'Payment' table is open, displaying 10 rows of payment records. The columns are: PaymentID, GuestID, PaymentMethod, PaymentDate, and TotalAmount. The data is as follows:

PaymentID	GuestID	PaymentMethod	PaymentDate	TotalAmount
GP001	GT001	Cash Payment	2024-03-01	Rs.28000
GP0010	GT0010	Credit Card	2024-03-13	Rs.28000
GP002	GT002	Credit Card	2024-03-02	Rs.14000
GP003	GT004	Cash Payment	2024-03-02	Rs.14000
GP004	GT007	Debit Card	2024-03-02	Rs.21000
GP005	GT003	Cash Payment	2024-03-04	Rs.28000
GP006	GT006	Credit Card	2024-03-04	Rs.21000
GP007	GT005	Debit Card	2024-03-05	Rs.28000
GP008	GT008	Cash Payment	2024-03-07	Rs.28000
GP009	GT009	Cash Payment	2024-03-10	Rs.21000

Invoice Table



The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. The 'Invoice' table is open, displaying 10 rows of invoice records. The columns are: GuestID, IssueDate, DueDate, TaxAmount, and TotalAmount. The data is as follows:

GuestID	IssueDate	DueDate	TaxAmount	TotalAmount
GT001	2024-03-01	2024-03-03	Rs.4000	Rs.32000
GT002	2024-03-01	2024-03-04	Rs.4000	Rs.32000
GT003	2024-03-04	2024-03-04	Rs.4000	Rs.32000
GT004	2024-03-01	2024-03-04	Rs.2000	Rs.16000
GT005	2024-03-04	2024-03-04	Rs.4000	Rs.32000
GT006	2024-03-04	2024-03-05	Rs.3000	Rs.24000
GT007	2024-03-01	2024-03-04	Rs.3000	Rs.24000
GT008	2024-03-07	2024-03-09	Rs.4000	Rs.32000
GT009	2024-03-09	2024-03-11	Rs.3000	Rs.24000
GT0010	2024-03-13	2024-03-15	Rs.4000	Rs.32000

Guest Table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure with the `hoteldatabase1` database selected.
- SQL Editor:** Displays the following SQL code for updating the `GUEST` table:


```

74 -- Updating data in GUEST table
75 • UPDATE GUEST
76 SET FirstName = 'Nadha', LastName='Jabeer'
77 WHERE GuestID = 'GT003';
78
79 • UPDATE GUEST
80 SET FirstName = 'Mithukshan', LastName='Rajaaperu'
81 WHERE GuestID = 'GT007';
82
      
```
- Result Grid:** Shows the updated data in the `GUEST` table. The table includes columns: GuestID, FirstName, LastName, DateOfBirth, Age, Email, Street, City, Province, GuestPassword. The updated rows are:

GuestID	FirstName	LastName	DateOfBirth	Age	Email	Street	City	Province	GuestPassword
GT001	Sasanka	Kothalawala	1996-05-25	28	sasankakothalawala@gmail.com	Lake Road	Anuradhapura	North Central	s123
GT002	Anni	Fernando	2002-11-06	22	annefernando@gmail.com	Wall Street	Puttalam	North Western	f455
GT003	Sandeepa	Muthukumari	2002-01-11	22	sandeepamuthukumari@gmail.com	Albion Road	Ratnapura	Sabaragamuwa	m128
GT004	Nadha	Jabeer	2001-05-20	23	nadhajabeer@gmail.com	Armour Street	Kurunegala	North Western	n222
GT005	Nirwantha	Modushani	1996-04-08	28	nirwanthanamodushani@gmail.com	Barber Street	Galle	South	n345
GT006	Gaganri	Wediawella	1997-08-15	27	gaganwediawella@gmail.com	Church Street	Colombo	Western	m000
GT007	Mithukshan	Rajaaperu	1998-11-09	26	mithurshanaajalin@gmail.com	Darley Road	Badulla	Uva	g454
GT008	Sadun	Chamara	1999-12-24	25	sadunchamara@gmail.com	Cheiku Street	Jaffna	Northern	m909
GT009	Aruna	Abeywardena	2000-01-01	25	arunaabeywardena@gmail.com	Deans Road	Kandy	Central	c333
- Action Output:** Shows the results of the update operations:
 - 102 20:02:22 UPDATE GUEST SET FirstName = 'Mithukshan', LastName='Rajaaperu' WHERE GuestID = 'GT007'
 - 103 20:02:26 SELECT * FROM hoteldatabase1.GUEST LIMIT 0, 2000

Contact Number Table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure with the `hoteldatabase1` database selected.
- SQL Editor:** Displays the following SQL code for updating the `CONTACTNUMBER` table:


```

106
107 -- Updating data in table
108 • UPDATE CONTACTNUMBER
109 SET ContactNumber = '0711777588'
110 WHERE GuestID = 'GT001';
111
112 • UPDATE CONTACTNUMBER
113 SET ContactNumber = '0713767599'
114 WHERE GuestID = 'GT002';
      
```
- Result Grid:** Shows the updated data in the `CONTACTNUMBER` table. The table includes columns: GuestID, ContactNumber. The updated rows are:

GuestID	ContactNumber
GT001	0711777588
GT002	0713767599
GT003	0701256693
GT004	0725465567
GT005	0754125569
- Action Output:** Shows the results of the update operations:
 - 110 20:40:04 INSERT INTO CONTACTNUMBER (GuestID, ContactNumber) VALUES ('GT001', '0774711139'), ('GT001', '0711777588')
 - 111 20:40:15 UPDATE CONTACTNUMBER SET ContactNumber = '0711777588' WHERE GuestID = 'GT001' and ContactNumber = '0711777588'
 - 112 20:40:21 SELECT * FROM hoteldatabase1.CONTACTNUMBER LIMIT 0, 2000

Cancellation Table

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: 2024-02-13FILE SQL File 8* employee project project department dependent SQL File 10* SQL File 13* HotelManagementSystem*

SCHEMAS: Schemas Filter objects

hoteldatabase1 Tables cancellation contract guest invoice payment reservation room roomtype staff Views Stored Procedure Functions

333
334 -- Updating data in CANCELLATION table
335 • UPDATE CANCELLATION
336 SET Reason='Change the hotel' , CancellationDate='2024-03-02'
337 WHERE CancellationID ='C003';
338
339 • UPDATE CANCELLATION
340 SET Reasons='Change the destination' , CancellationDate='2024-03-01'
341 WHERE CancellationID ='C005';

Result Grid Filter Rows: Edit: Export/Import Wrap Cell Contents

Table: reservation

Columns: GuestID CheckInDate CheckOutDate NumberOfGuests RoomNumber

CancellationID	GuestID	PaymentID	Reason	RefundAmount	CancellationDate
C001	GT007	GP004	Weather Conditions	Rs.23000	2024-03-03
C002	GT005	GP007	Change the hotel	Rs.31000	2024-03-03
C003	GT009	GP009	Change the hotel	Rs.23000	2024-03-02
C004	GT001	GP001	Weather Conditions	Rs.28000	2024-03-01
C005	GT002	GP002	Change the destination	Rs.14000	2024-03-01
C006	GT004	GP003	Weather Conditions	Rs.14000	2024-03-02

CANCELLATION 24 x

Output: Action Output Time Action

Object Info Session

Message

29°C Partly cloudy

8:54 PM 4/8/2024

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Delete Operations

Staff Table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the `hoteldatabase1` schema with tables: cancellation, contractum, guest, invoice, payment, reservation, room, roomtype, staff.
- SQL Editor:** Contains the following SQL code:

```
34 • UPDATE STAFF
35 SET S_Name = 'Sandeepa Hettigoda', Salary='Rs.80000'
36 WHERE StaffID = 'SF007';
37
38 -- Deleting one row from the STAFF table
39 • DELETE FROM STAFF
40 WHERE StaffID = 'SF002';
```
- Result Grid:** Displays the `STAFF` table with 10 rows of data. The row where `StaffID = 'SF002'` is highlighted.
- Action Output:** Shows the execution of the delete query:

```
133 20:56:25 DELETE FROM STAFF WHERE StaffID = 'SF002'
134 20:56:30 SELECT * FROM hoteldatabase1.staff LIMIT 0, 2000
```
- Output:** Shows the message "1 row(s) affected" and "9 row(s) returned".

Room Type Table

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the `hoteldatabase1` schema with tables: cancellation, contractum, guest, invoice, payment, reservation, room, roomtype, staff.
- SQL Editor:** Contains the following SQL code:

```
189 WHERE TypeID = 'RT06';
190
191 -- Deleting one row from the ROOMTYPE table
192 • DELETE FROM ROOMTYPE
193 WHERE TypeID = 'RT06';
194
195
196 • create table RESERVATION (
```
- Result Grid:** Displays the `ROOMTYPE` table with 5 rows of data. The row where `TypeID = 'RT06'` is highlighted.
- Action Output:** Shows the execution of the delete query:

```
141 20:59:31 DELETE FROM ROOMTYPE WHERE TypeID = 'RT06'
142 20:59:43 SELECT * FROM hoteldatabase1.ROOMTYPE LIMIT 0, 2000
```
- Output:** Shows the message "1 row(s) affected" and "5 row(s) returned".

Room Table

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, room
- SQL Editor:** Contains the following code:

```
L56 SET Rate = '8.7'
L57 WHERE RoomNumber = 'RN07';
L58
L59 -- Deleting one row from the ROOM table
L60 • DELETE FROM ROOM
L61 WHERE RoomNumber = 'RN010';
```
- Result Grid:** Shows the data for the ROOM table.
- Action Output:** Displays the results of the DELETE query:

Time	Action	Message	Duration / Fetch
139	20:58:54	DELETE FROM ROOM WHERE RoomNumber = RN010	1 row(s) affected 0.016 sec
140	20:58:58	SELECT * FROM hoteldatabase1.ROOM LIMIT 0, 2000	9 row(s) returned 0.000 sec / 0.000 sec

Reservation Table

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation
- SQL Editor:** Contains the following code:

```
31
32 • DELETE FROM RESERVATION
33 WHERE GuestID = 'GT005';
34
35
36 • create table INVOICE(
37 GuestID varchar(10) not null
```
- Result Grid:** Shows the data for the RESERVATION table.
- Action Output:** Displays the results of the DELETE query:

Time	Action	Message	Duration / Fetch
143	21:00:15	DELETE FROM RESERVATION WHERE GuestID = GT005	0 row(s) affected 0.000 sec
144	21:00:20	SELECT * FROM hoteldatabase1.RESERVATION LIMIT 0, 2000	8 row(s) returned 0.000 sec / 0.000 sec

Payment Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the SQL editor, the following SQL code is run:

```
06
07 -- Deleting one row from the PAYMENT table
08 • DELETE FROM PAYMENT
09 WHERE PaymentID = 'GP007';
10
11 • SELECT * FROM hoteldatabase1.PAYMENT;
12
```

The results show the PAYMENT table with 10 rows of data. The last row, which corresponds to the deleted record, is highlighted.

PaymentID	GuestID	PaymentMethod	PaymentDate	TotalAmount
GP001	GT001	Cash Payment	2024-03-01	Rs.28000
GP002	GT002	Credit Card	2024-03-02	Rs.14000
GP003	GT004	Cash Payment	2024-03-02	Rs.14000
GP004	GT007	Debit Card	2024-03-02	Rs.21000
GP005	GT003	Cash Payment	2024-03-04	Rs.28000
GP006	GT006	Credit Card	2024-03-04	Rs.21000
GP007	GT008	Cash Payment	2024-03-07	Rs.28000
GP008	GT009	Cash Payment	2024-03-10	Rs.21000
GP009	GT005	Cash Payment	2024-03-10	Rs.21000

The Action Output pane shows the execution of the DELETE query and the SELECT query, both completed successfully.

Invoice Table

The screenshot shows the MySQL Workbench interface with the 'HotelManagementSystem' database selected. In the SQL editor, the following SQL code is run:

```
67 SET IssueDate='2024-03-09', DueDate ='2024-03-11'
68 WHERE GuestID = 'GT009';
69
70 • DELETE FROM INVOICE
71 WHERE GuestID = 'GT005';
72
73
```

The results show the INVOICE table with 10 rows of data. The last row, which corresponds to the deleted record, is highlighted.

GuestID	IssueDate	DueDate	TaxAmount	TotalAmount
GT001	2024-03-01	2024-03-03	Rs.4000	Rs.32000
GT002	2024-03-02	2024-03-04	Rs.4000	Rs.32000
GT003	2024-03-04	2024-03-06	Rs.4000	Rs.32000
GT004	2024-03-02	2024-03-04	Rs.2000	Rs.16000
GT005	2024-03-04	2024-03-06	Rs.3000	Rs.24000
GT006	2024-03-02	2024-03-04	Rs.3000	Rs.24000
GT007	2024-03-09	2024-03-11	Rs.3000	Rs.24000
GT008	2024-03-07	2024-03-09	Rs.4000	Rs.32000
GT009	2024-03-09	2024-03-11	Rs.3000	Rs.24000
GT010	2024-03-13	2024-03-15	Rs.4000	Rs.32000

The Action Output pane shows the execution of the SET statement, the DELETE query, and the SELECT query, all completed successfully.

Guest Table

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation
- Columns:** GuestID, CheckInDate, CheckOutDate, NumberOfGuests, RoomNumber
- SQL Editor:** Contains the following SQL code:

```
83 -- Deleting one row from the GUEST table
84 • DELETE FROM GUEST
85 WHERE GuestID = 'GT005';
86
87
88
```
- Result Grid:** Shows a table with columns: GuestID, FirstName, LastName, DateOfBirth, Age, Email, Street, City, Province, GuestPassword. The data includes rows for GT001 through GT009.
- Action Output:** Displays the results of the DELETE query:
 - Time: 20:57:29 Action: DELETE FROM GUEST WHERE GuestID = GT005; Message: 1 row(s) affected Duration / Fetch: 0.000 sec
 - Time: 20:57:33 Action: SELECT * FROM hoteldatabase1.GUEST LIMIT 0, 2000 Message: 9 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec
- System Bar:** Shows the date and time as 4/8/2024 at 8:57 PM.

Contact Number Table

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation
- Columns:** GuestID, CheckInDate, CheckOutDate, NumberOfGuests, RoomNumber
- SQL Editor:** Contains the following SQL code:

```
114 WHERE GuestID = 'GT002';
115
116 -- Deleting one row from the CONTACTNUMBER table
117 • DELETE FROM CONTACTNUMBER
118 WHERE GuestID = 'GT005';
119
```
- Result Grid:** Shows a table with columns: GuestID, ContactNumber. The data includes rows for GT001 through GT009.
- Action Output:** Displays the results of the DELETE query:
 - Time: 20:58:00 Action: DELETE FROM CONTACTNUMBER WHERE GuestID = 'GT005' Message: 0 row(s) affected Duration / Fetch: 0.000 sec
 - Time: 20:58:04 Action: SELECT * FROM hoteldatabase1.CONTACTNUMBER LIMIT 0, 2000 Message: 54 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec
- System Bar:** Shows the date and time as 4/8/2024 at 8:58 PM.

Cancellation Table

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80 ×

SCHEMAS: Filter objects

- foe
- foe_23
- foe_24
- hoteldatabase1

Tables: cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff

Views

Stored Procedures

Functions

42

43 • DELETE FROM CANCELLATION

44 WHERE CancellationID = 'C002';

45

46 • SELECT * FROM hoteldatabase1.CANCELLATION;

47

48

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

CancellationID	GuestID	PaymentID	Reason	RefundAmount	CancellationDate
C001	GT007	GP004	Weather Conditions	Rs.23000	2024-03-03
C003	GT009	GP009	Change the hotel	Rs.23000	2024-03-02
C004	GT001	GP001	Weather Conditions	Rs.28000	2024-03-01
C005	GT002	GP002	Change the destination	Rs.14000	2024-03-01
C006	GT004	GP003	Weather Conditions	Rs.14000	2024-03-02
...

Table: reservation

Columns: GuestID, CheckInDate, CheckOutDate, NumberOfGuests, RoomNumber

CANCELLATION 33 x

Output

Action Output

Time	Action	Message	Duration / Fetch
149 21:02:17	DELETE FROM CANCELLATION WHERE CancellationID = 'C002'	0 row(s) affected	0.000 sec
150 21:02:20	SELECT * FROM hoteldatabase1.CANCELLATION LIMIT 0, 2000	5 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

29°C Partly cloudy

Search

9:02 PM 4/8/2024

Chapter 4 – Transaction

Simple Queries

Select Operation

MySQL Workbench - Local instance MySQL80

```

54 -- Simple Queries
55
56 -- 1). Select Operation: Retrieve all information about STAFF
57 • SELECT * FROM hoteldatabase1.staff;
58
59 -- 2). Project Operation: Retrieve only StaffID, S_Name and Posit:
60 • SELECT StaffID, S_Name, Position FROM hoteldatabase1.STAFF;

```

Result Grid

StaffID	S_Name	Position	Salary	HireDate
SF001	Sahan Disanayake	Manager	Rs.300000	2024-01-01
SF002	Noora Arj	Hotel Housekeeper	Rs.300000	2024-01-08
SF003	Chethana Wanniarachchi	Director of marketing	Rs.210000	2024-01-01
SF004	Amina Abshan	Office Manager	Rs.200000	2024-01-02
SF005	Kapila Addararachchi	Human Resources Manager	Rs.150000	2024-01-02
SF006	Shiva Suryakumar	Hotel Receptionist	Rs.90000	2024-01-04
SF007	Sandeepa Hettigoda	Parking Attendant	Rs.80000	2024-01-05
SF008	Thushalika Kumar	Room Attendant	Rs.70000	2024-01-07
SF009	Sithum Sankalpa	Cook	Rs.140000	2024-01-08

Action Output

Time	Action	Message	Duration / Fetch
1 21:04:45	SELECT * FROM hoteldatabase1.staff LIMIT 0, 2000	9 row(s) returned	0.000 sec / 0.000 sec

Project Operation

MySQL Workbench - Local instance MySQL80

```

57 • SELECT * FROM hoteldatabase1.staff;
58
59 -- 2). Project Operation: Retrieve only StaffID, S_Name and Posit:
60 • SELECT StaffID, S_Name, Position FROM hoteldatabase1.staff;
61
62 -- 3). Cartesian Product Operation: Retrieve a Cartesian product
63 • SELECT * FROM ROOM R00MTVPF;

```

Result Grid

StaffID	S_Name	Position
SF001	Sahan Disanayake	Manager
SF002	Noora Arj	Hotel Housekeeper
SF003	Chethana Wanniarachchi	Director of marketing
SF004	Amina Abshan	Office Manager
SF005	Kapila Addararachchi	Human Resources Manager
SF006	Shiva Suryakumar	Hotel Receptionist
SF007	Sandeepa Hettigoda	Parking Attendant
SF008	Thushalika Kumar	Room Attendant
SF009	Sithum Sankalpa	Cook

Action Output

Time	Action	Message	Duration / Fetch
1 21:04:45	SELECT * FROM hoteldatabase1.staff LIMIT 0, 2000	9 row(s) returned	0.000 sec / 0.000 sec
2 21:05:19	SELECT StaffID, S_Name, Position FROM hoteldatabase1.staff LIMIT 0, 2000	9 row(s) returned	0.000 sec / 0.000 sec

Product Operation

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: hoteldatabase1

Tables: room, roomtype, staff, reservation, guest, payment, invoice, cancellation, contactnum, project, employee, department, dependent

SQL File 8*, SQL File 10*, SQL File 13*, HotelManagementSystem*

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

62 -- 3). Cartesian Product Operation: Retrieve a Cartesian product

63 • SELECT * FROM ROOM, ROOMTYPE;

64

RoomNumber	RoomType	Rate	BedType	CleaningStatus	TypeID	MaximumGuest	TotalRooms	R_Description	RoomPrice
RN01	AC	8.8	Quad	Clean	RT05	3	3	Non AC and high-speed Wi-Fi	15000
RN01	AC	8.8	Quad	Clean	RT04	4	3	AC high-speed Wi-Fi	28000
RN01	AC	8.8	Quad	Clean	RT03	3	3	AC high-speed Wi-Fi	21000
RN01	AC	8.8	Quad	Clean	RT02	2	2	AC high-speed Wi-Fi	14000
RN01	AC	8.8	Quad	Clean	RT01	1	2	AC and high-speed Wi-Fi	7000
RN02	Non AC	6.5	Quad	Clean	RT05	3	3	Non AC and high-speed Wi-Fi	15000
RN02	Non AC	6.5	Quad	Clean	RT04	4	3	AC high-speed Wi-Fi	28000
RN02	Non AC	6.5	Quad	Clean	RT03	3	3	AC high-speed Wi-Fi	21000
RN02	Non AC	6.5	Quad	Clean	RT02	2	2	AC high-speed Wi-Fi	14000
RN02	Non AC	6.5	Quad	Clean	RT01	1	2	AC and high-speed Wi-Fi	7000
RN03	AC	9.3	Triple	Clean	RT05	3	3	Non AC and high-speed Wi-Fi	15000
RN03	AC	9.3	Triple	Clean	RT04	4	3	AC high-speed Wi-Fi	28000
RN03	AC	9.3	Triple	Clean	RT03	3	3	AC high-speed Wi-Fi	21000
RN03	AC	9.3	Triple	Clean	RT02	2	2	AC high-speed Wi-Fi	14000
RN03	AC	9.3	Triple	Clean	RT01	1	2	AC and high-speed Wi-Fi	7000
RN04	AC	10.0	Double	Clean	RT05	3	3	Non AC and high-speed Wi-Fi	15000
RN04	AC	10.0	Double	Clean	RT04	4	3	AC high-speed Wi-Fi	28000
RN04	AC	10.0	Double	Clean	RT03	3	3	AC high-speed Wi-Fi	21000
RN04	AC	10.0	Double	Clean	RT02	2	2	AC high-speed Wi-Fi	14000
RN04	AC	10.0	Double	Clean	RT01	1	2	AC and high-speed Wi-Fi	7000
RN05	Non AC	8.9	Triple	Clean	RT05	3	3	Non AC and high-speed Wi-Fi	15000
RN05	Non AC	8.9	Triple	Clean	RT04	4	3	AC high-speed Wi-Fi	28000
RN05	Non AC	8.9	Triple	Clean	RT03	3	3	AC high-speed Wi-Fi	21000
RN05	Non AC	8.9	Triple	Clean	RT02	2	2	AC and high-speed Wi-Fi	14000

Result 38 x

Output: Action Output

Object Info Session

29°C Partly cloudy

Search Bar: Search

Message Bar: Duration / Fetch

9:05 PM 4/8/2024

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Create a User View

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: hoteldatabase1

Tables: room, roomtype, staff, reservation, guest, payment, invoice, cancellation, contactnum, project, employee, department, dependent

SQL File 8*, SQL File 10*, SQL File 13*, HotelManagementSystem*

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

64

65 -- 4). -- Creating a User View: Create a view named "Passenger_Details"

66 • CREATE VIEW STAFF_Details AS

67 SELECT StaffID, S_Name, Position

68 FROM STAFF;

69

StaffID	S_Name	Position
SF001	Sahan Disanayake	Manager
SF002	Mihara Arj	Hotel Housekeeper
SF003	Cethana Wamarachchi	Director of marketing
SF004	Anura Abhan	Office Manager
SF005	Kapila Adharanadhi	Human Resources Manager
SF006	Shiva Suryakumar	Hotel Receptionist
SF007	Sandeepa Hettigoda	Parking Attendant
SF008	Thushalika Kumar	Room Attendant
SF009	Sithum Sanjala	Cook

STAFF_Details 37 x

Output: Action Output

Object Info Session

29°C Partly cloudy

Search Bar: Search

Message Bar: Duration / Fetch

0.000 sec / 0.000 sec

0.016 sec

0.000 sec / 0.000 sec

9:07 PM 4/8/2024

SQLAdditions: Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Renaming Operation

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, hoteldatabase.
- SQL Editor:** Contains the following SQL code:

```
73 -- 5). Renaming Operation: Rename the "RoomType" column in the ROOM table to Type;
74 ALTER TABLE ROOM
75 RENAME COLUMN RoomType TO Type;
76 select * from ROOM;
```
- Result Grid:** Shows the results of the SELECT query, displaying columns: RoomNumber, Type, Rate, BedType, CleaningStatus. The data includes rows for RN01 through RN09.
- Action Output:** Shows the execution history of the queries:
 - 5 21:07:23 select * from STAFF_Details LIMIT 0, 2000
 - 6 21:09:06 ALTER TABLE ROOM RENAME COLUMN RoomType TO Type
 - 7 21:09:09 select * from ROOM LIMIT 0, 2000
- System Bar:** Shows the date (4/8/2024), time (9:09 PM), and weather (29°C, Partly cloudy).

Aggregation Function

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** Local instance MySQL80, hoteldatabase.
- SQL Editor:** Contains the following SQL code:

```
76 select * from ROOM;
77
78 -- 6). -- Aggregation Function: Calculate the average salary of staff
79 SELECT AVG(RoomPrice) AS Average_RoomPrice FROM ROOMTYPE;
```
- Result Grid:** Shows the result of the AVG query, displaying a single row with Average_RoomPrice: 17000.0000.
- Action Output:** Shows the execution history of the queries:
 - 6 21:09:06 ALTER TABLE ROOM RENAME COLUMN RoomType TO Type
 - 7 21:09:09 select * from ROOM LIMIT 0, 2000
 - 8 21:12:33 SELECT AVG(RoomPrice) AS Average_RoomPrice FROM ROOMTYPE LIMIT 0, 2000
- System Bar:** Shows the date (4/8/2024), time (9:13 PM), and weather (29°C, Partly cloudy).

Use of LIKE Keyword

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Schemas (hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff), Views, Stored Procedure, Functions.
- SQL Editor:** SQL File 8*, SQL File 10*, SQL File 13*, HotelManagementSystem*. The code is:

```
79 • SELECT AVG(RoomPrice) AS Average_RoomPrice FROM ROOMTYPE;
80
81 -- 7). Use of LIKE Keyword: Retrieve all Guests whose first name s
82 • SELECT * FROM hoteldatabase1.guest WHERE FirstName LIKE 'S%';
83
84
85
```
- Result Grid:** Shows a table with columns GuestID, FirstName, LastName, DateOfBirth, Age, Email, Street, City, Province, GuestPassword. Data rows:

GuestID	FirstName	LastName	DateOfBirth	Age	Email	Street	City	Province	GuestPassword
GT001	Sasanka	Kothalawala	1996-05-25	28	sasankakothalawala@gmail.com	Lake Road	Anuradhapura	North Central	s123
GT002	Sandeepa	Muthukumari	2002-01-11	22	sandeepamuthukumari@gmail.com	Albion Road	Ratnapura	Sabaragamuwa	m128
GT008	Sadun	Chamara	1999-12-24	25	sadunchamara@gmail.com	Deans Road	Kandy	Central	c333
- Output:** Action Output pane showing log entries for the executed queries.
- Bottom:** Taskbar with various icons, system status, and a message bar indicating "Tomorrow's high Near record".

Complex Queries Union Operation

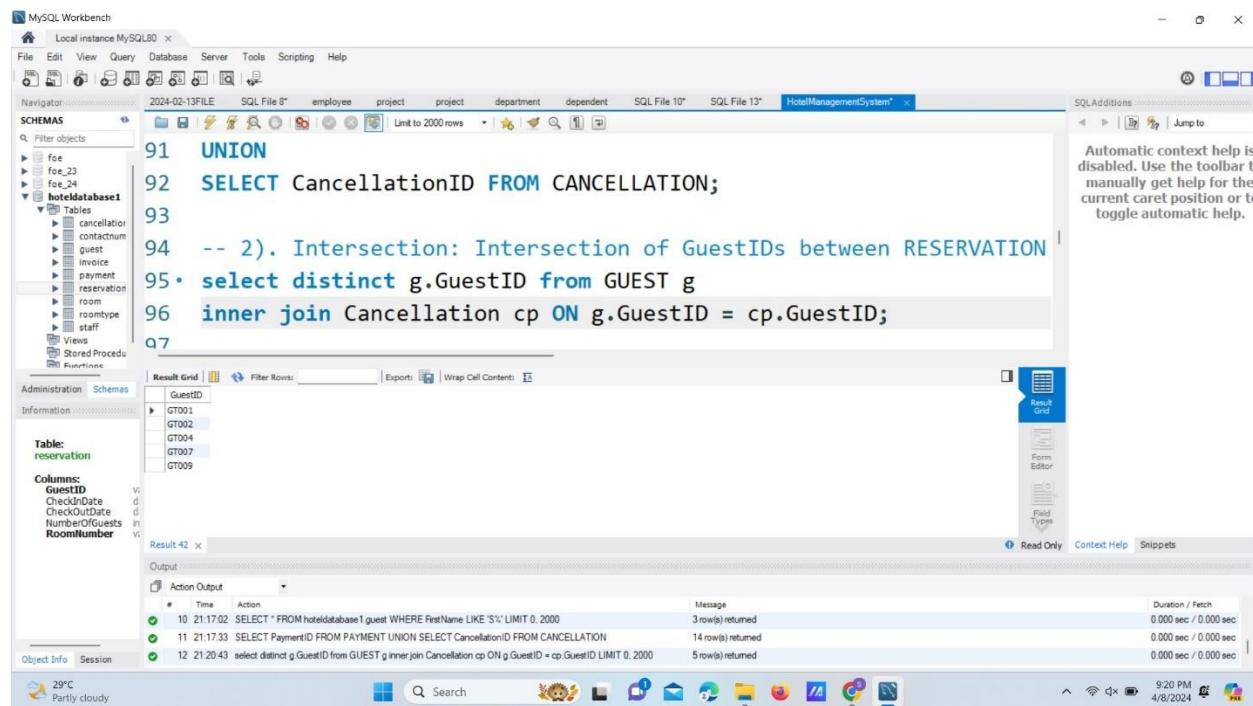
The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Schemas (hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff), Views, Stored Procedure, Functions.
- SQL Editor:** SQL File 8*, SQL File 10*, SQL File 13*, HotelManagementSystem*. The code is:

```
88
89 -- 1). Union: Retrieve all passengers' names and emails.
90 • SELECT PaymentID FROM PAYMENT
91 UNION
92 SELECT CancellationID FROM CANCELLATION;
93
94 -- 2). Intersection: Intersection of GuestIDs between RESERVATION
```
- Result Grid:** Shows a table with column PaymentID. Data rows:

PaymentID
GP001
GP0010
GP002
GP005
GP003
GP006
GP004
GP008
GP009
GP004
- Output:** Action Output pane showing log entries for the executed queries.
- Bottom:** Taskbar with various icons, system status, and a message bar indicating "Tomorrow's high Near record".

Intersection Operation



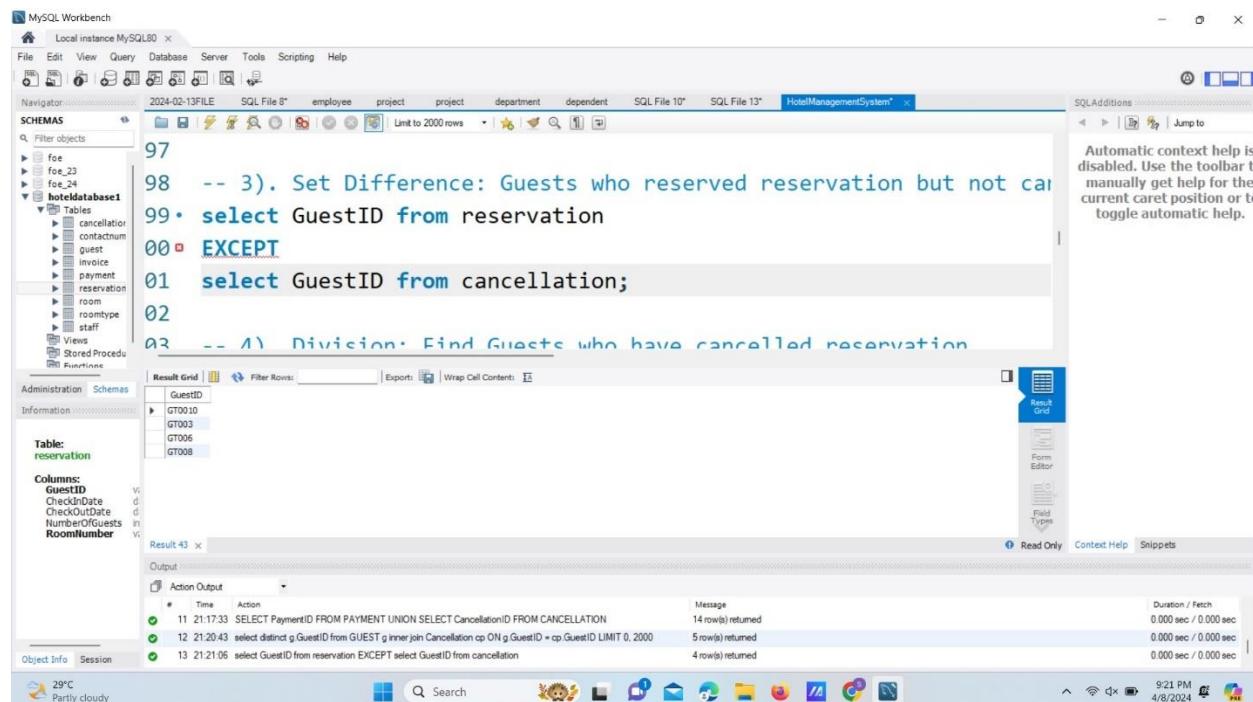
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, cancellation, guest, payment
- Query:**

```
91 UNION
92 SELECT CancellationID FROM CANCELLATION;
93
94 -- 2). Intersection: Intersection of GuestIDs between RESERVATION
95 select distinct g.GuestID from GUEST g
96 inner join Cancellation cp ON g.GuestID = cp.GuestID;
97
```

- Result Grid:** Shows GuestID values: GT001, GT002, GT004, GT007, GT009.
- Action Output:** Shows three log entries with their respective times, actions, messages, and duration/fetch times.
- System Bar:** Shows the date as 4/8/2024 and the time as 9:21 PM.

Set Difference Operation



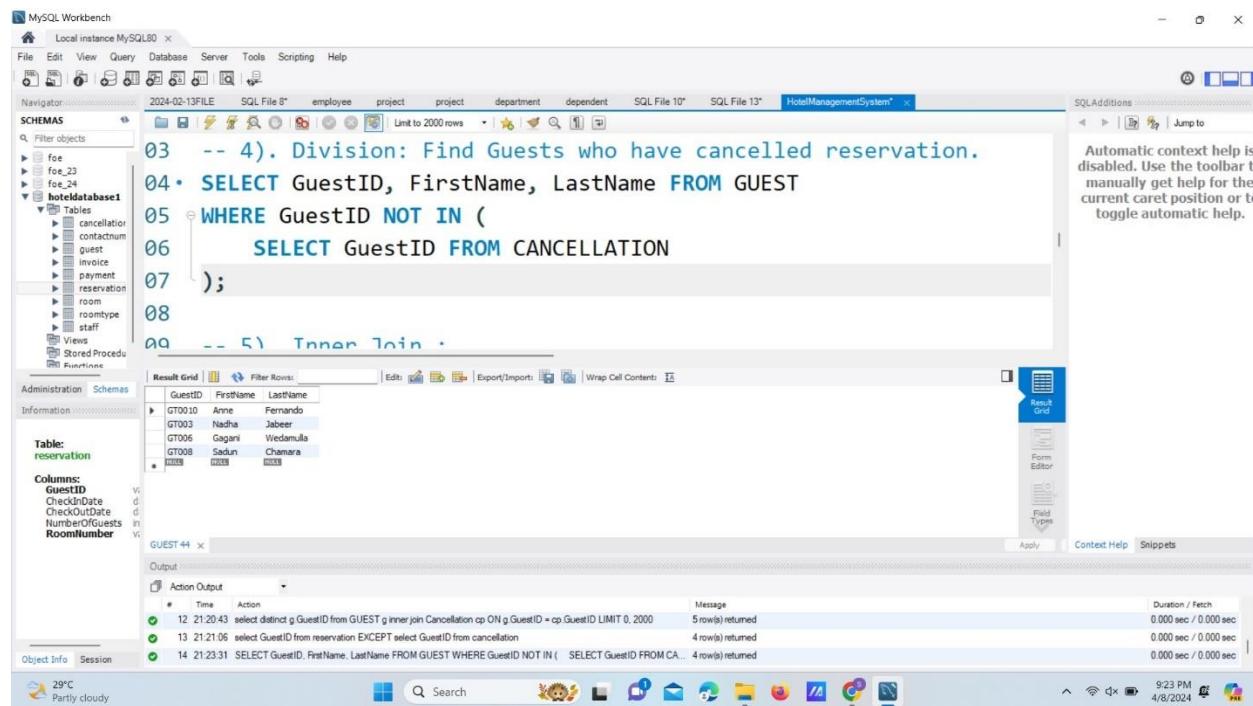
The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, cancellation, guest, payment
- Query:**

```
97
98 -- 3). Set Difference: Guests who reserved reservation but not cancellation
99 select GuestID from reservation
00 EXCEPT
01 select GuestID from cancellation;
02
03 -- 1) Division: Find Guests who have cancelled reservation
```

- Result Grid:** Shows GuestID values: GT0010, GT003, GT006, GT008.
- Action Output:** Shows four log entries with their respective times, actions, messages, and duration/fetch times.
- System Bar:** Shows the date as 4/8/2024 and the time as 9:21 PM.

Division Operation



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: hoteldatabase1

Tables: cancellation, contractum, guest, invoice, payment, reservation, room, roomtype, staff

Views: reservation

Stored Procedures: HotelManagementSystem

SQL Editor:

```
03 -- 4). Division: Find Guests who have cancelled reservation.
04 • SELECT GuestID, FirstName, LastName FROM GUEST
05 WHERE GuestID NOT IN (
06     SELECT GuestID FROM CANCELLATION
07 );
08
09 -- 5) Inner Join :
```

Result Grid:

GuestID	FirstName	LastName
GT0010	Anni	Fernando
GT003	Nadha	Jabeer
GT005	Gagari	Vedamulla
GT008	Sadun	Chamara

Action Output:

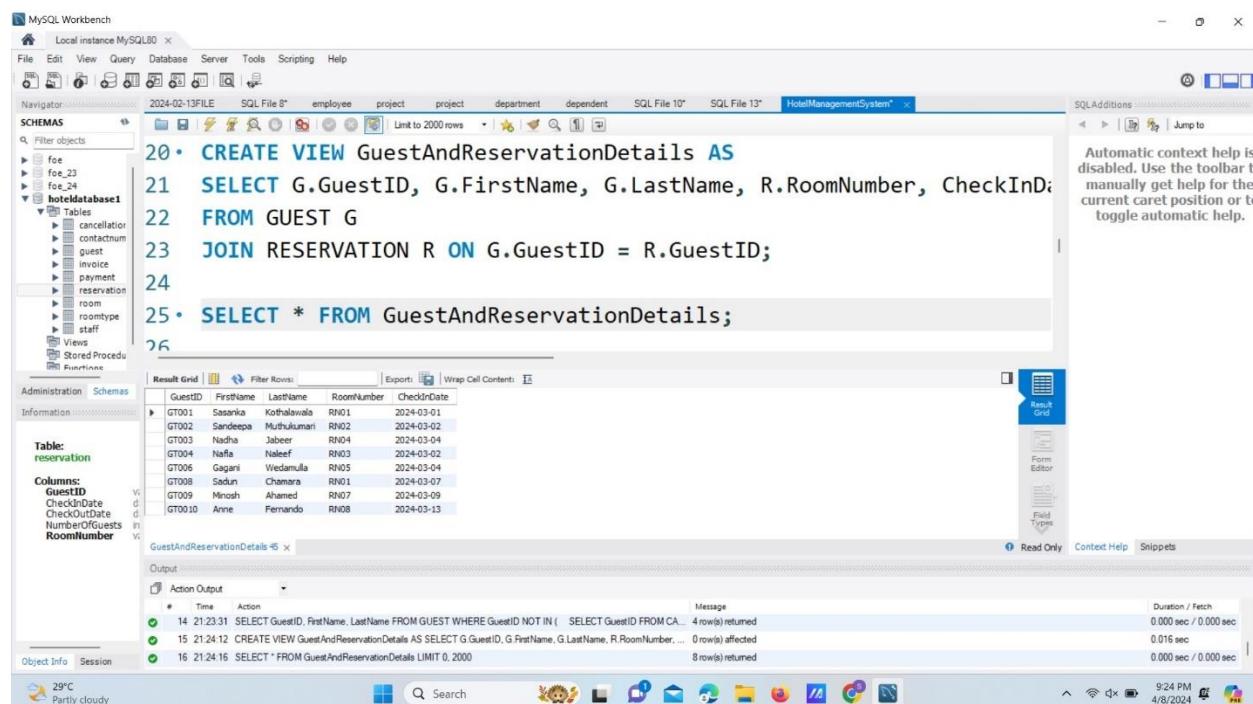
Time	Action	Message	Duration / Fetch
12 21:20:43	select distinct g GuestID from GUEST g inner join Cancellation cp ON g.GuestID = cp.GuestID LIMIT 0, 2000	5 row(s) returned	0.000 sec / 0.000 sec
13 21:21:06	select GuestID from reservation EXCEPT select GuestID from cancellation	4 row(s) returned	0.000 sec / 0.000 sec
14 21:23:31	SELECT GuestID, FirstName, LastName FROM GUEST WHERE GuestID NOT IN (SELECT GuestID FROM CA...)	4 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

29°C Partly cloudy

9:23 PM 4/8/2024

Inner Join Operation



MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: hoteldatabase1

Tables: cancellation, contractum, guest, invoice, payment, reservation, room, roomtype, staff

Views: reservation

Stored Procedures: HotelManagementSystem

SQL Editor:

```
20 • CREATE VIEW GuestAndReservationDetails AS
21   SELECT G.GuestID, G.FirstName, G.LastName, R.RoomNumber, CheckInDate
22   FROM GUEST G
23   JOIN RESERVATION R ON G.GuestID = R.GuestID;
24
25 • SELECT * FROM GuestAndReservationDetails;
```

Result Grid:

GuestID	FirstName	LastName	RoomNumber	CheckInDate
GT001	Sasanka	Kothalawala	RN01	2024-03-01
GT002	Sandeepa	Muthukumari	RN02	2024-03-02
GT003	Nadha	Jabeer	RN04	2024-03-04
GT004	Nafya	Naleef	RN03	2024-03-02
GT005	Gagari	Vedamulla	RN05	2024-03-04
GT008	Sadun	Chamara	RN01	2024-03-07
GT009	Minosh	Ahamed	RN07	2024-03-09
GT010	Anni	Fernando	RN08	2024-03-13

Action Output:

Time	Action	Message	Duration / Fetch
14 21:23:31	SELECT GuestID, FirstName, LastName FROM GUEST WHERE GuestID NOT IN (SELECT GuestID FROM CA...)	4 row(s) returned	0.000 sec / 0.000 sec
15 21:24:12	CREATE VIEW GuestAndReservationDetails AS SELECT G.GuestID, G.FirstName, G.LastName, R.RoomNumber, ...	0 row(s) affected	0.016 sec
16 21:24:16	SELECT * FROM GuestAndReservationDetails LIMIT 0, 2000	8 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

29°C Partly cloudy

9:24 PM 4/8/2024

Natural Join Operation

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code being run is:

```
438 • select * from cancellation natural join guest;
439
440 -- 2nd Method by using user view
441 CREATE VIEW GuestAndPayment AS
442 SELECT G.GuestID, G.FirstName, G.LastName, P.PaymentMethod
443 FROM GUEST G
444 NATURAL JOIN PAYMENT P;
445
446 • SELECT * FROM GuestAndPayment;
```

The results grid displays the following data:

GuestID	FirstName	LastName	PaymentMethod
GT001	Sasanki	Kothalawala	Cash Payment
GT002	Annie	Fernando	Credit Card
GT003	Sandeepa	Muthukumar	Credit Card
GT004	Nadha	Jabeer	Cash Payment
GT005	Nafisa	Naleef	Cash Payment
GT006	Gagari	Wedanulla	Credit Card
GT007	Mithulshan	Rajaaperu	Debit Card
GT008	Sadun	Chomara	Cash Payment
GT009	Minosh	Ahamed	Cash Payment

The status bar at the bottom right shows the date and time as 4/8/2024 9:34 PM.

Outer Join Operation

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code being run is:

```
503 -- 10). Outer join
504 -- Creating a view to combine guest and cancellation information using a full outer join
505 • CREATE VIEW Guest_Cancellation AS
506 SELECT COALESCE(G.GuestID, 'No Guest') AS GuestID, COALESCE(G.FirstName, 'No Guest') AS FirstName, COALESCE(G.LastName, 'No Guest') AS LastName,
507 C.CancellationID, C.Reason
508 FROM GUEST G
509 LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID
510 UNION
511 SELECT COALESCE(G.GuestID, 'No Guest') AS GuestID, COALESCE(G.FirstName, 'No Guest') AS FirstName, COALESCE(G.LastName, 'No Guest') AS LastName,
512 C.CancellationID, C.Reason
513 FROM GUEST G
514 RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID;
515
516 • select * from Guest_Cancellation;
```

The results grid displays the following data:

GuestID	FirstName	LastName	CancellationID	Reason
GT001	Sasanki	Kothalawala	C004	Weather Conditions
GT002	Annie	Fernando	NULL	NULL
GT003	Sandeepa	Muthukumar	C005	Change the destination
GT004	Nadha	Jabeer	NULL	NULL
GT005	Nafisa	Naleef	C006	Weather Conditions
GT006	Gagari	Wedanulla	NULL	NULL
GT007	Mithulshan	Rajaaperu	C001	Weather Conditions
GT008	Sadun	Chomara	NULL	NULL
GT009	Minosh	Ahamed	C003	Change the hotel

The status bar at the bottom right shows the date and time as 4/8/2024 9:29 PM.

Left Outer Join Operation

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, cancellation
- SQL Editor:** Contains the following code:

```
50 • CREATE VIEW GuestAndCancellation AS
51   SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
52   FROM GUEST G
53   LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID;
54
55 • SELECT * FROM GuestAndCancellation;
```
- Result Grid:** Shows the results of the query in the SQL editor. The table has columns GuestID, FirstName, LastName, and Reason. The data includes rows for GT001 through GT009.
- Output:** Shows the execution log with three entries related to the view creation and selection.

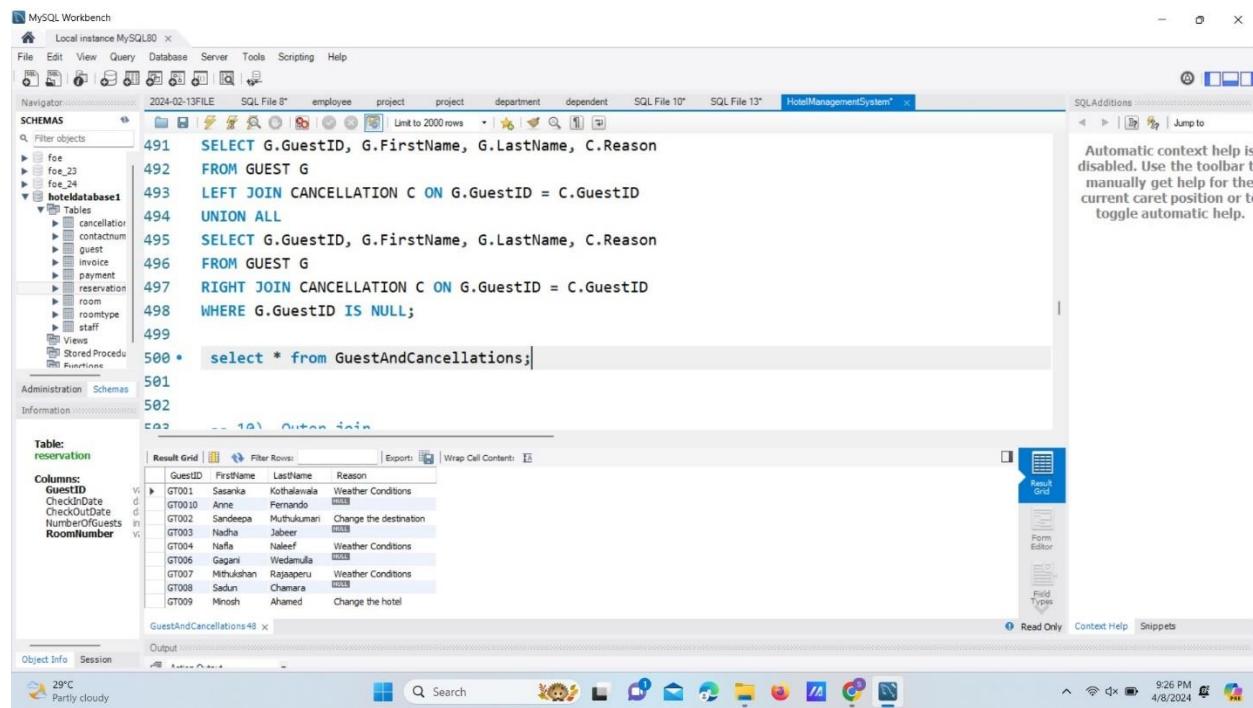
Right Outer Join Operation

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, room
- SQL Editor:** Contains the following code:

```
66 • CREATE VIEW RoomAndReservation AS
67   SELECT R.RoomNumber, R.Type, RES.NumberOfGuests
68   FROM ROOM R
69   RIGHT JOIN RESERVATION RES ON R.RoomNumber = RES.RoomNumber;
70
71 • SELECT * FROM RoomAndReservation;
```
- Result Grid:** Shows the results of the query in the SQL editor. The table has columns RoomNumber, Type, and NumberOfGuests. The data includes rows for RN01 through RN08.
- Output:** Shows the execution log with three entries related to the view creation and selection. It includes a note about a table not existing.

Full Outer Join Operation



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (foe, foe_23, foe_24, hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff, Views, Stored Procedu, Functions).
- SQL Editor:** SQL File 8*, project, project, department, dependent, SQL File 10*, SQL File 13*, HotelManagementSystem*. The code is as follows:

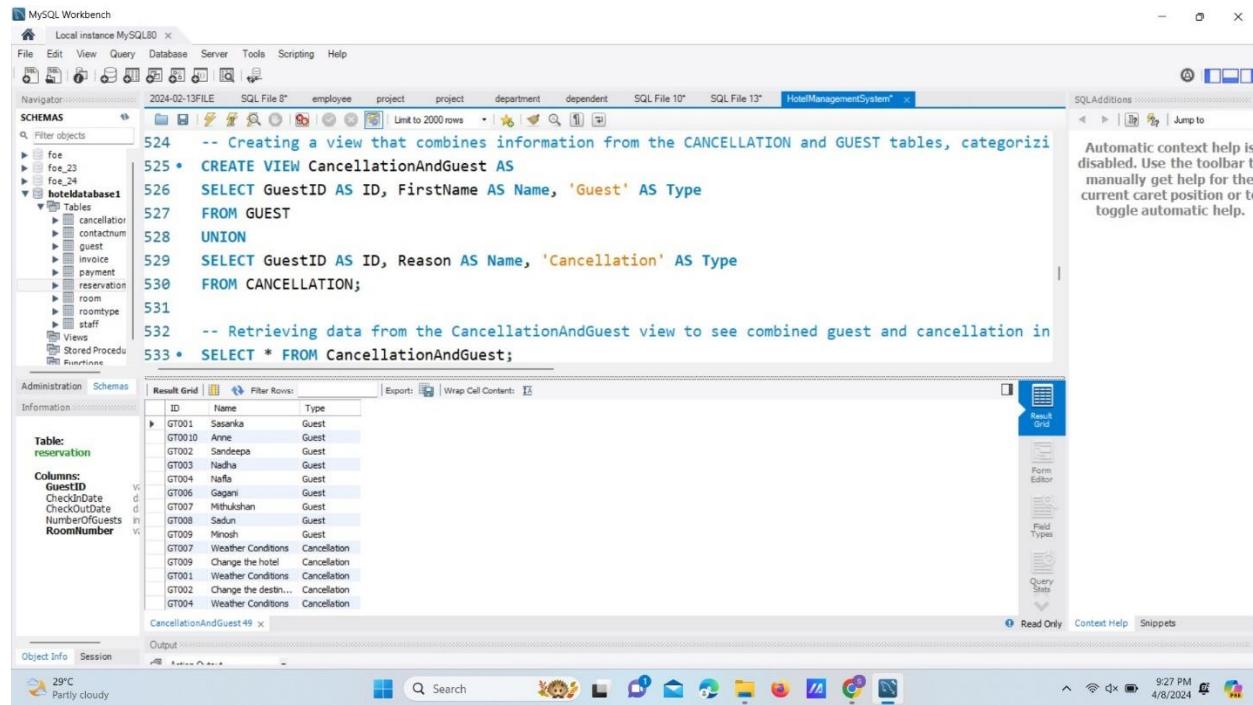
```
491 SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
492 FROM GUEST G
493 LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID
494 UNION ALL
495 SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
496 FROM GUEST G
497 RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID
498 WHERE G.GuestID IS NULL;
499
500 * select * from GuestAndCancellations;
```

The result grid shows the combined data from both tables:

GuestID	FirstName	LastName	Reason
GT001	Sasanka	Kothakewala	Weather Conditions
GT010	Anne	Fernando	
GT002	Sandeepa	Muthujuman	Change the destination
GT003	Nadha	Jithu	
GT004	Nafya	Naleef	Weather Conditions
GT005	Gagan	Wedanulla	
GT007	Mithukshan	Rajaaperu	Weather Conditions
GT008	Sadun	Chamara	
GT009	Minosh	Ahamed	Change the hotel

Output: 10 rows found.

Outer Union Operation



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (foe, foe_23, foe_24, hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff, Views, Stored Procedu, Functions).
- SQL Editor:** SQL File 8*, project, project, department, dependent, SQL File 10*, SQL File 13*, HotelManagementSystem*. The code is as follows:

```
524 -- Creating a view that combines information from the CANCELLATION and GUEST tables, categorizing by reason
525 * CREATE VIEW CancellationAndGuest AS
526   SELECT GuestID AS ID, FirstName AS Name, 'Guest' AS Type
527   FROM GUEST
528   UNION
529   SELECT GuestID AS ID, Reason AS Name, 'Cancellation' AS Type
530   FROM CANCELLATION;
531
532 -- Retrieving data from the CancellationAndGuest view to see combined guest and cancellation in
533 * SELECT * FROM CancellationAndGuest;
```

The result grid shows the combined data from both tables:

ID	Name	Type
GT001	Sasanka	Guest
GT010	Anne	Guest
GT002	Sandeepa	Guest
GT003	Nadha	Guest
GT004	Nafya	Guest
GT005	Gagan	Guest
GT007	Mithukshan	Guest
GT008	Sadun	Guest
GT009	Minosh	Guest
GT007	Weather Conditions	Cancellation
GT009	Change the hotel	Cancellation
GT001	Weather Conditions	Cancellation
GT002	Change the destin...	Cancellation
GT004	Weather Conditions	Cancellation

Output: 13 rows found.

Nested Query with Intersection Operation

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

```
529 |
530 |
531 -- 12). This query retrieves guest information along with their cancellation details for guests
532 • SELECT g.GuestID, g.FirstName, g.LastName, c.CancellationID, c.Reason
533   FROM GUEST g
534     INNER JOIN CANCELLATION c ON g.GuestID = c.GuestID
535   WHERE g.GuestID IN (SELECT GuestID FROM RESERVATION WHERE RoomNumber IN (SELECT RoomNumber FROM
536 |
537 )
```

The result grid shows the following data:

GuestID	FirstName	LastName	CancellationID	Reason
GT001	Sasanka	Kothalawala	C004	Weather Conditions
GT004	Nafisa	Naleef	C006	Weather Conditions

Output pane:

- Action Output: 7 21:35:26 SELECT g.GuestID, g.FirstName, g.LastName, c.CancellationID, c.Reason FROM GUEST g INNER JOIN CANCEL... Error Code: 1054: Unknown column 'RoomType' in 'where clause'
- Action Output: 8 21:35:43 SELECT g.GuestID, g.FirstName, g.LastName, c.CancellationID, c.Reason FROM GUEST g INNER JOIN CANCEL... 2 row(s) returned

Nested Query with Set Operation

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

```
538 -- 13).
539 • SELECT G.GuestID, G.FirstName, G.LastName
540   FROM GUEST G
541   WHERE NOT EXISTS (
542     SELECT *
543       FROM RESERVATION R
544         LEFT JOIN CANCELLATION C ON R.GuestID = C.GuestID
545         WHERE G.GuestID = R.GuestID
546         AND C.CancellationID IS NOT NULL
547   );
548
```

The result grid shows the following data:

GuestID	FirstName	LastName
GT0010	Anne	Fernando
GT003	Nadha	Jabeer
GT006	Gagari	Wedemulla
GT007	Mithukshan	Rajaaperu
GT008	Sadun	Chamara
NULL	NULL	NULL

Output pane:

- Action Output: 8 21:35:43 SELECT g.GuestID, g.FirstName, g.LastName, c.CancellationID, c.Reason FROM GUEST g INNER JOIN CANCEL... 2 row(s) returned
- Action Output: 9 21:38:14 SELECT G.GuestID, G.FirstName, G.LastName FROM GUEST G WHERE NOT EXISTS (SELECT * FROM R... 5 row(s) returned

Nested Query with Division Operation

The screenshot shows the MySQL Workbench interface with a SQL editor window displaying the following code:

```
549 -- 14).
550 • SELECT G.GuestID, G.FirstName, G.LastName
551   FROM GUEST G
552   WHERE EXISTS (
553     SELECT *
554       FROM RESERVATION R
555     NATURAL JOIN PAYMENT P
556     WHERE G.GuestID = R.GuestID
557   );
```

The SQL editor has tabs for "SQL File 8*", "SQL File 10*", and "SQL File 13*" with the current tab being "HotelManagementSystem". Below the editor is a "Result Grid" showing the results of the query:

GuestID	FirstName	LastName
GT001	Sasanka	Kothalawala
GT002	Annie	Fernando
GT003	Sandeepa	Muthukumar
GT004	Nadha	Jabeer
GT005	Nalfa	Naleef
GT006	Gagari	Vedamulla
GT007	Sadun	Chamara
GT008	Minoch	Ahamed
GT009	Umesh	Umesh

On the left, the Navigator pane shows the schema structure, including tables like cancellation, contactnum, invoice, payment, reservation, room, roomtype, staff, and views. The "reservation" table is selected. The bottom status bar shows the weather as 29°C Partly cloudy and the system time as 9:30 PM on 4/8/2024.

Chapter 5 – Tuning

Chapter 5 describes how indexing strategies are used to improve the efficiency of complex queries. For every complex query, the following steps are applied during query tuning in order to make the process easier to understand:

1. Removing Existing External Indexes: Start by getting rid of any old indexes that weren't needed for query optimization.
2. Displaying Table Indexes Before Index Creation (Using SHOW INDEX Command): Evaluate the current state of indexes on the relevant tables before making any modifications.
3. Determining the Number of Accessed Rows before Index Creation (Using EXPLAIN Command)
4. Creating a Suitable Index: Based on the analysis, establish an appropriate index to optimize the query.
5. Displaying Table Indexes after Index Creation (Using SHOW INDEX Command).
6. Determining the Number of Accessed Rows after Index Creation (Using EXPLAIN Command)

A comparison of the number of rows accessed in the query execution plan prior to and during the development of the appropriate index serves as an example of the query tuning procedure. After adding the index, if there is a decrease in the number of accessed rows, the query has been successfully optimized.

1) Tuning Nested Complex Query

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with tables like cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff, and views.
- SQL Editor:** Contains the query:

```
767
768 -- Before Tuning
769 • explain SELECT G.GuestID, G.FirstName, G.LastName
770   FROM GUEST G
771   WHERE EXISTS (
772     SELECT *
773       FROM RESERVATION R
774     NATURAL JOIN PAYMENT P
775     WHERE G.GuestID = R.GuestID
776   );
```
- Result Grid:** Displays the execution plan for the EXPLAIN command. It shows three rows in the table:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	G	ALL	PRIMARY,`idx_guest_guestid`	idx_guest_guestid	9	100.00		1	100.00	Using index
1	SIMPLE	R	ALL	ref	idx_reservation_guestid,`idx_reservation_guestid4`	idx_reservation_guestid4	42	hoteldatabase1.G.GuestID	1	100.00	Using index
1	SIMPLE	P	ALL	ref	FK_PAYMENT_GUEST	FK_PAYMENT_GUEST	42	hoteldatabase1.G.GuestID	1	100.00	Using index; FirstMatch
- Output:** Shows the results of the EXPLAIN command, indicating 2 row(s) returned for the first query and 3 row(s) returned for the second.

Creating Index (Payment)

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with tables like cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff, and views.
- SQL Editor:** Contains the query:

```
773
774   FROM RESERVATION R
775   NATURAL JOIN PAYMENT P
776   WHERE G.GuestID = R.GuestID
777
778 -- Create indexes
779 • CREATE INDEX idx_reservation_guestid ON RESERVATION (GuestID);
780 • show indexes from RESERVATION;
781 • CREATE INDEX idx_payment_guestid ON PAYMENT (GuestID);
782 • show indexes from PAYMENT;
```
- Result Grid:** Displays the execution plan for the EXPLAIN command. It shows three rows in the table:

Table	Seq_in_index	Key_name	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
payment	0	PRIMARY	1	PaymentID	A	2	YES	NO	BTREE	YES	NO	NO	
payment	1	FK_PAYMENT_GUEST	1	GuestID	A	2	YES	NO	BTREE	YES	NO	NO	
payment	1	idx_paymentid	1	PaymentID	A	9	YES	NO	BTREE	YES	NO	NO	
- Output:** Shows the results of the EXPLAIN command, indicating 3 row(s) returned for both queries.

Creating Index (Reservation)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, payment
- Code Editor:** SQL File 8*
- SQL:**

```
773     FROM RESERVATION R
774     NATURAL JOIN PAYMENT P
775     WHERE G.GuestID = R.GuestID
776   );
777
778 -- Create indexes
779 • CREATE INDEX idx_reservation_guestid ON RESERVATION (GuestID);
780 • show indexes from RESERVATION;
781 • CREATE INDEX idx_payment_guestid ON PAYMENT (GuestID);
782 • show indexes from PAYMENT;
```
- Result Grid:** Shows the creation of three indexes:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
reservation	1	FK_RESERVATION_ROOM	1	RoomNumber	A	2			YES	BTREE			YES	MAX
reservation	1	idx_reservation_guestid	1	GuestID	A	8			YES	BTREE			YES	MAX
reservation	1	idx_reservation_guestid4	1	GuestID	A	8			YES	BTREE			YES	MAX
- Action Output:**
 - 85 22:28:15 explain SELECT G.GuestID, G.FirstName, G.LastName FROM GUEST G WHERE EXISTS (SELECT * FROM... 3 row(s) returned
 - 86 22:28:36 show indexes from RESERVATION 3 row(s) returned
- System Bar:** Shows the date and time as 4/8/2024 at 10:28 PM.

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

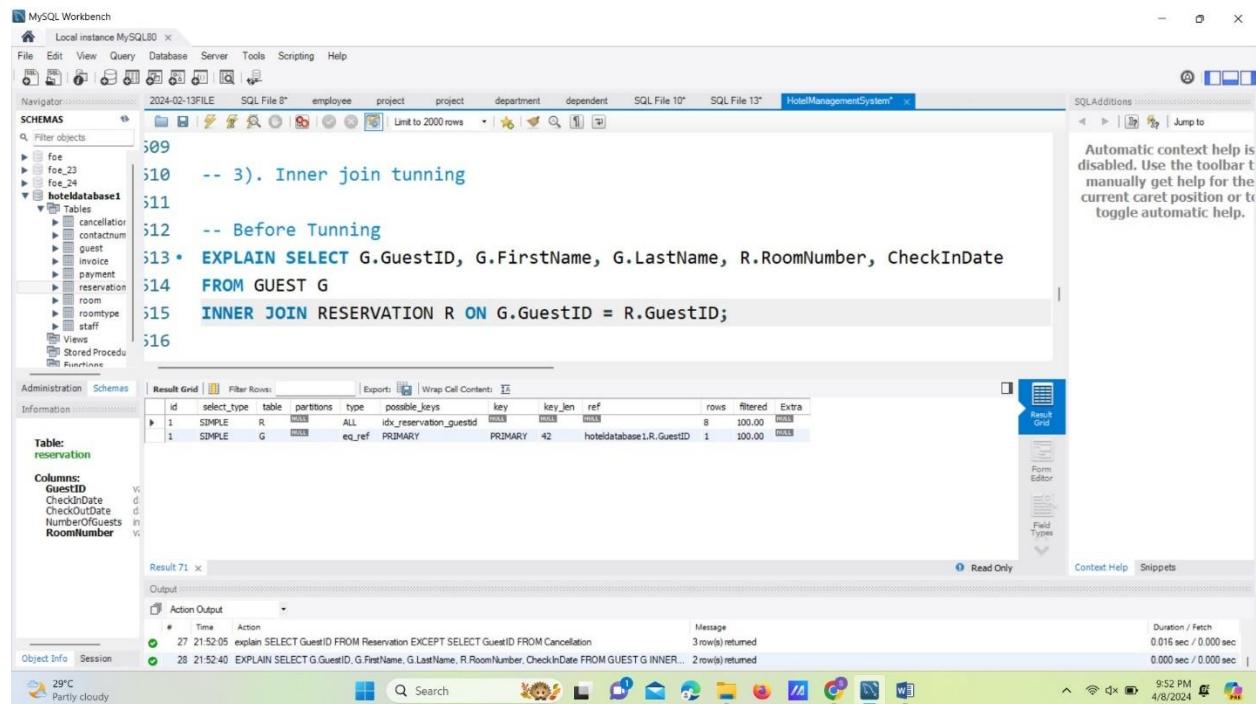
- Schemas:** hoteldatabase1
- Tables:** reservation, payment
- Code Editor:** SQL File 8*
- SQL:**

```
782 • show indexes from PAYMENT;
783
784 -- After Tuning
785 • explain SELECT G.GuestID, G.FirstName, G.LastName
    FROM GUEST G
    WHERE EXISTS (
        SELECT *
        FROM RESERVATION R
        NATURAL JOIN PAYMENT P
        WHERE G.GuestID = R.GuestID
    );
```
- Result Grid:** Shows the execution plan for the tuned query:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	G		ALL	PRIMARY,`idx_guest_guestid`,`idx_guest_id`	idx_guest_guestid	42		9	100.00	
1	SIMPLE	R		ref	idx_reservation_guestid,`idx_reservation_guestid4`	idx_reservation_guestid4	42	hoteldatabase1.G.GuestID	1	100.00	Using index
1	SIMPLE	P		ref	FK_PAYMENT_GUEST	FK_PAYMENT_GUEST	42	hoteldatabase1.G.GuestID	1	100.00	Using index; FirstMatch(
- Action Output:**
 - 87 22:29:17 show indexes from PAYMENT 3 row(s) returned
 - 88 22:30:14 explain SELECT G.GuestID, G.FirstName, G.LastName FROM GUEST G WHERE EXISTS (SELECT * FROM... 3 row(s) returned
- System Bar:** Shows the date and time as 4/8/2024 at 10:30 PM.

2) Tuning Inner Join Complex Query

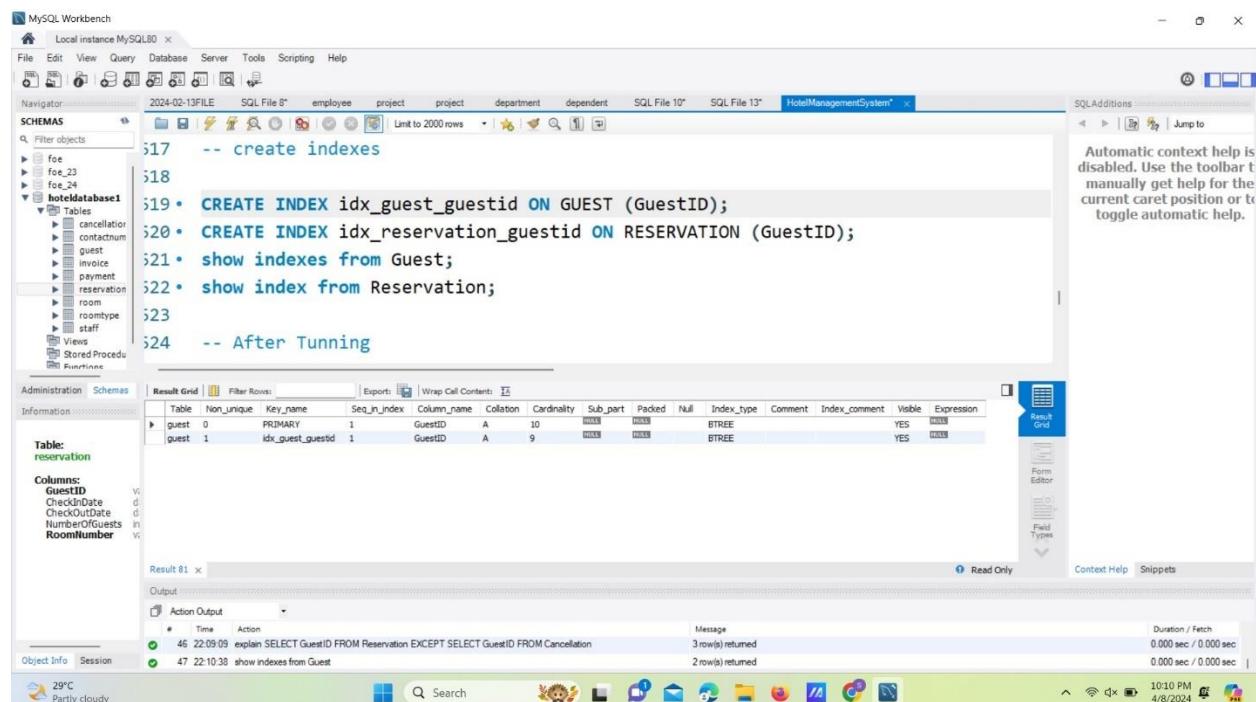
Before Tuning



```
509
510 -- 3). Inner join tuning
511
512 -- Before Tuning
513 • EXPLAIN SELECT G.GuestID, G.FirstName, G.LastName, R.RoomNumber, CheckInDate
514   FROM GUEST G
515     INNER JOIN RESERVATION R ON G.GuestID = R.GuestID;
516
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	R	NULL	ALL	idx_reservation_guestid	NULL	NULL	NULL	8	100.00	NULL
1	SIMPLE	G	NULL	eq_ref	PRIMARY	PRIMARY	42	hoteldatabase1.R.GuestID	1	100.00	NULL

Creating Index (Guest)



```
517 -- create indexes
518
519 • CREATE INDEX idx_guest_guestid ON GUEST (GuestID);
520 • CREATE INDEX idx_reservation_guestid ON RESERVATION (GuestID);
521 • show indexes from Guest;
522 • show index from Reservation;
523
524 -- After Tuning
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
guest	0	PRIMARY	1	GuestID	A	10	NULL	NULL	NULL	BTREE		YES	NULL	
guest	1	idx_guest_guestid	1	GuestID	A	9	NULL	NULL	NULL	BTREE		YES	NULL	

Creating Index (Reservation)

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (foe, foe_23, foe_24, hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff).
- SQL Editor:** Contains the following SQL code:

```
596 select GuestID from cancellation;
597
598 -- Creating Indexes
599 • CREATE INDEX idx_reservation_guestid ON Reservation (GuestID);
600 • CREATE INDEX idx_cancellation_guestid ON Cancellation (GuestID);
601 • show indexes from Reservation;
602 • show index from CANCELLATION;
603
```
- Result Grid:** Shows the results of the SHOW INDEXES query for the Reservation table.

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
reservation	1	FK_RESERVATION_ROOM	1	RoomNumber	A	2			YES	BTREE			YES	VISIBLE
reservation	1	idx_reservation_guestid	1	GuestID	A	8			YES	BTREE			YES	VISIBLE
reservation	1	idx_reservation_guestid4	1	GuestID	A	8			YES	BTREE			YES	VISIBLE
- Output:** Shows EXPLAIN and SHOW INDEXES queries with their respective durations and row counts.
- System Bar:** Shows the date (4/8/2024), time (10:08 PM), and weather (29°C, Partly cloudy).

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (foe, foe_23, foe_24, hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff).
- SQL Editor:** Contains the following SQL code:

```
523
524 -- After Tuning
525 • EXPLAIN
526 SELECT G.GuestID, G.FirstName, G.LastName, R.RoomNumber, CheckInDate
527 FROM GUEST G
528 INNER JOIN RESERVATION R ON G.GuestID = R.GuestID;
```
- Result Grid:** Shows the EXPLAIN output for the query.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	R	NULL	ALL	idx_reservation_guestid, idx_reservation_guestid4	PRIMARY	NULL	NULL	8	100.00	NULL
1	SIMPLE	G	NULL	eq_ref	PRIMARY, idx_guest_guestid	PRIMARY	42	hoteldatabase1.R.GuestID	1	100.00	NULL
- Output:** Shows EXPLAIN and SHOW INDEXES queries with their respective durations and row counts.
- System Bar:** Shows the date (4/8/2024), time (10:12 PM), and weather (29°C, Partly cloudy).

3) Tuning Union Complex Query

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the original complex query:

```
573 -- Before Tuning
574 • explain SELECT PaymentID FROM PAYMENT
UNION
576   SELECT CancellationID FROM CANCELLATION;
577
578 -- Creating Indexes
579 • CREATE INDEX idx_paymentid ON PAYMENT (PaymentID);
580 • CREATE INDEX idx_cancellationid ON CANCELLATION (CancellationID);
581 • show indexes from PAYMENT;
582 • show index from CANCELLATION;
```
- Result Grid:** Displays the execution plan for the EXPLAIN command, showing three rows: PRIMARY, UNION, and UNION RESULT.
- Action Output:** Shows two log entries related to the creation of indexes.

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the query after creating indexes:

```
576   SELECT CancellationID FROM CANCELLATION;
577
578 -- Creating Indexes
579 • CREATE INDEX idx_paymentid ON PAYMENT (PaymentID);
580 • CREATE INDEX idx_cancellationid ON CANCELLATION (CancellationID);
581 • show indexes from PAYMENT;
582 • show index from CANCELLATION;
```
- Result Grid:** Displays the execution plan for the EXPLAIN command, showing four rows: cancellation (PRIMARY), cancellation (FK_CANCELLATION_GUEST), cancellation (FK_PAYMENT_PAYMENT), and cancellation (idx_cancellationid).
- Action Output:** Shows two log entries related to the creation of indexes.

Creating Index (Payment)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** reservation, payment, cancellation, guest, invoice, room, roomtype, staff.
- Code Editor:** SQL File 8* (HotelManagementSystem*)
The code is:

```
576  SELECT CancellationID FROM CANCELLATION;
577
578  -- Creating Indexes
579 • CREATE INDEX idx_paymentid ON PAYMENT (PaymentID);
580 • CREATE INDEX idx_cancellationid ON CANCELLATION (CancellationID);
581 • show indexes from PAYMENT;
582 • show index from CANCELLATION;
583
```
- Result Grid:** Shows the results of the 'show index' command for the PAYMENT table.

Table	Non_Unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
payment	0	PRIMARY	1	PaymentID	A	2				BTREE		YES	NO	
payment	1	FK_PAYMENT_GUEST	1	GuestID	A	2				BTREE		YES	NO	
payment	1	idx_paymentid	1	PaymentID	A	9				BTREE		YES	NO	
- Action Output:** Shows two log entries:
 - 19 21:47:33 show index from CANCELLATION Message 4 row(s) returned 0.000 sec / 0.000 sec
 - 19 21:47:41 show indexes from PAYMENT Message 3 row(s) returned 0.000 sec / 0.000 sec
- System Bar:** Shows the date and time as 4/8/2024 9:47 PM.

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** reservation, payment, cancellation, guest, invoice, room, roomtype, staff.
- Code Editor:** SQL File 10* (HotelManagementSystem*)
The code is:

```
582 • show index from CANCELLATION;
583
584  -- After Tuning
585 • explain SELECT PaymentID FROM PAYMENT
UNION
SELECT CancellationID FROM CANCELLATION;
```
- Result Grid:** Shows the results of the 'explain' command for the combined query.

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	PRIMARY	PAYMENT	NULL	index	NULL	FK_PAYMENT_GUEST	42	NULL	9	100.00	Using index
2	UNION	CANCELLATION	NULL	index	NULL	idx_cancellationid	22	NULL	6	100.00	Using index
3	UNION RESULT	<union1,2>	NULL	ALL	NULL	NULL	NULL	NULL	NULL	NULL	Using temporary
- Action Output:** Shows two log entries:
 - 20 21:48:09 show index from CANCELLATION Message 4 row(s) returned 0.016 sec / 0.000 sec
 - 21 21:48:42 explain SELECT PaymentID FROM PAYMENT UNION SELECT CancellationID FROM CANCELLATION Message 3 row(s) returned 0.016 sec / 0.000 sec
- System Bar:** Shows the date and time as 4/8/2024 9:48 PM.

4) Tuning Left Outer Join

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, guest, cancellation, room, roomtype, staff, contactnum, invoice, payment.
- SQL Editor:** Contains the following code:

```
532
533 -- Before Tuning
534 • CREATE VIEW GuestAndCancellation AS
535   SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
536   FROM GUEST G
537   LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID;
538
539 • explain select * from GuestAndCancellation ;
```
- Result Grid:** Shows the execution of the EXPLAIN command. The output is:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	NULL	ALL	NULL	NULL	NULL	NULL	9	100.00	NULL
1	SIMPLE	c	NULL	ref	idx_cancellation_guestid	idx_cancellation_guestid	42	hoteldatabase1.g.GuestID	1	100.00	NULL
- Message Area:** Shows two log entries:
 - CREATE VIEW GuestAndCancellation AS SELECT G.GuestID, G.FirstName, G.LastName, C.Reason FROM GUEST G LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID; Error Code: 1050. Table 'GuestAndCancellation' already exists
 - explain select * from GuestAndCancellation 2 row(s) returned
- System Bar:** Shows the date and time as 4/8/2024 10:12 PM.

Creating Index (Guest)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, guest, cancellation, room, roomtype, staff, contactnum, invoice, payment.
- SQL Editor:** Contains the following code:

```
538 • explain select * from GuestAndCancellation ;
539
540 -- Index creation for GUEST table
541 • CREATE INDEX idx_guest_id ON GUEST (GuestID);
542 -- Index creation for CANCELLATION table
543
544 • CREATE INDEX idx_cancellation_guest_id ON CANCELLATION (GuestID);
545 • show indexes from Guest;
```
- Result Grid:** Shows the execution of the SHOW INDEXES command for the Guest table. The output is:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
guest	0	PRIMARY	1	GuestID	A	10	NULL	BTREE	YES	BTREE			YES	BTREE
guest	1	idx_guest_guestid	1	GuestID	A	9	NULL	BTREE	YES	BTREE			YES	BTREE
guest	1	idx_guest_id	1	GuestID	A	9	NULL	BTREE	YES	BTREE			YES	BTREE
- Message Area:** Shows two log entries:
 - CREATE INDEX idx_guest_id ON GUEST (GuestID) 0 row(s) affected, 1 warning(s): 1831 Duplicate index 'idx_guest_id' defined on the table 'hoteldatabase1.guest'. This ... Duration / Fetch 0.109 sec
 - show indexes from Guest 3 row(s) returned 0.016 sec / 0.000 sec
- System Bar:** Shows the date and time as 4/8/2024 10:13 PM.

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
539 • explain select * from GuestAndCancellation ;
540
541 -- Index creation for GUEST table
542 • CREATE INDEX idx_guest_id ON GUEST (GuestID);
543 -- Index creation for CANCELLATION table
544 • CREATE INDEX idx_cancellation_guest_id ON CANCELLATION (GuestID);
545 • show indexes from Guest;
```
- Result Grid:** Shows the results of the EXPLAIN command for the first query, displaying index usage information.
- Output Panel:** Shows the execution log with two entries: "show indexes from Guest" and "explain select * from GuestAndCancellation".
- System Bar:** Includes the date (2024-02-13), time (10:13 PM), and system status (4/8/2024).

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
547
548 -- After Tuning
549 -- Recreate the view after optimization
550 • CREATE VIEW GuestAndCancellation1 AS
551 SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
      FROM GUEST G
      LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID;
554 • explain select * from GuestAndCancellation1 ;
```
- Result Grid:** Shows the results of the EXPLAIN command for the view creation, displaying the execution plan.
- Output Panel:** Shows the execution log with two entries: "CREATE VIEW GuestAndCancellation1 AS" and "explain select * from GuestAndCancellation1".
- System Bar:** Includes the date (2024-02-13), time (10:14 PM), and system status (4/8/2024).

5) Tuning Difference

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Schemas (hoteldatabase1), Tables (cancellation, guest, reservation, room, staff).
- SQL Editor:** SQL File 8* (containing the script below). A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Result Grid:** Shows the execution of the EXPLAIN command for the query: `explain select GuestID from reservation EXCEPT select GuestID from cancellation;`. The output includes columns like id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, and Extra.
- Output:** Action Output shows two log entries: 21 21:49:42 and 22 21:49:57.
- System Bar:** Shows the date and time (4/8/2024, 9:50 PM), system icons, and a search bar.

```
;91 -- 2). Set Difference Tuning
;92
;93 -- Before Tuning
;94 explain select GuestID from reservation
;95 EXCEPT
;96 select GuestID from cancellation;
;97
;98 -- Creating Indexes
```

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Schemas (hoteldatabase1), Tables (cancellation, guest, reservation, room, staff).
- SQL Editor:** SQL File 8* (containing the script below). A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Result Grid:** Shows the execution of the SHOW INDEXES command for the Reservation and CANCELLATION tables. The output includes columns like Table, Non_unique, Key_name, Seq_in_index, Column_name, Collation, Cardinality, Sub_part, Packed, Null, Index_type, Comment, Index_comment, Visible, and Ex.
- Output:** Action Output shows two log entries: 25 21:50:35 and 26 21:51:36.
- System Bar:** Shows the date and time (4/8/2024, 9:51 PM), system icons, and a search bar.

```
;97
;98 -- Creating Indexes
;99 CREATE INDEX idx_reservation_guestid ON Reservation (GuestID);
;100 CREATE INDEX idx_cancellation_guestid ON CANCELLATION (GuestID);
;01 show indexes from Reservation;
;02 show index from CANCELLATION;
;03
;04 -- After Tuning
```

Creating Index (Reservation)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, cancellation
- SQL Editor:** Contains the following SQL code:

```
i97
i98  -- Creating Indexes
i99 • CREATE INDEX idx_reservation_guestid ON Reservation (GuestID);
i00 • CREATE INDEX idx_cancellation_guestid ON Cancellation (GuestID);
i01 • show indexes from Reservation;
i02 • show index from CANCELLATION;
i03
i04  -- After Tuning
```
- Result Grid:** Shows the results of the SHOW INDEXES command for the reservation table.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
reservation	1	FK_RESERVATION_ROOM	1	RoomNumber	A	2			YES	BTREE			YES	VISIBLE
reservation	1	idx_reservation_guestid	1	GuestID	A	8			YES	BTREE			YES	VISIBLE
- Action Output:** Displays the execution log:
 - 24 21:50:31 CREATE INDEX idx_cancellation_guestid ON Cancellation (GuestID) 0 rows affected Records: 0 Duplicates: 0 Warnings: 0 Duration / Fetch: 0.125 sec.
 - 25 21:50:35 show indexes from Reservation 2 rows returned Duration / Fetch: 0.015 sec / 0.000 sec
- Object Info:** Session
- System Bar:** Shows the date and time (4/8/2024 9:50 PM), system status (Near record), and system icons.

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation, cancellation
- SQL Editor:** Contains the following SQL code:

```
i00 • CREATE INDEX idx_cancellation_guestid ON Cancellation (GuestID);
i01 • show indexes from Reservation;
i02 • show index from CANCELLATION;
i03
i04  -- After Tuning
i05 • explain SELECT GuestID FROM Reservation
i06 □ EXCEPT
i07  SELECT GuestID FROM Cancellation;
```
- Result Grid:** Shows the results of the EXPLAIN command for the combined query.

ID	Select_Type	Table	Partitions	Type	Possible_keys	Key	Key_len	Ref	Rows	Filtered	Extra
1	PRIMARY	Reservation		index		idx_reservation_guestid	42	NULL	8	100.00	Using index
2	EXCEPT	Cancellation		index		idx_cancellation_guestid	42	NULL	6	100.00	Using index
3	EXCEPT RESULT	<except1,2>		ALL			NULL	NULL	NULL	NULL	Using temporary
- Action Output:** Displays the execution log:
 - 26 21:51:36 show index from CANCELLATION 4 rows returned Duration / Fetch: 0.016 sec / 0.000 sec
 - 27 21:52:05 explain SELECT GuestID FROM Reservation EXCEPT SELECT GuestID FROM Cancellation 3 rows returned Duration / Fetch: 0.016 sec / 0.000 sec
- Object Info:** Session
- System Bar:** Shows the date and time (4/8/2024 9:52 PM), system status (Partly cloudy), and system icons.

6) Tuning Left Outer Join

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `hoteldatabase1` containing tables like `cancellation`, `contactnum`, `guest`, `invoice`, `payment`, `reservation`, `room`, `roomtype`, `staff`, and `Guest`.
- SQL Editor:** Contains the following SQL code:

```
532
533 -- Before Tuning
534 • CREATE VIEW GuestAndCancellation AS
535   SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
536   FROM GUEST G
537   LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID;
538
539 • explain select * from GuestAndCancellation ;
```
- Result Grid:** Displays the execution plan for the EXPLAIN command. The output shows a single query with one row in the table `Guest` and one row in the table `CANCELLATION`.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	NULL	ALL	NULL	NULL	NULL	NULL	9	100.00	NULL
1	SIMPLE	c	NULL	ref	idx_cancellation_guestid	idx_cancellation_guestid	42	hoteldatabase1.g.GuestID	1	100.00	NULL
- Object Info:** Shows the `reservation` table with columns: `GuestID`, `CheckInDate`, `CheckOutDate`, `NumberOfGuests`, and `RoomNumber`.
- Action Output:** Shows the creation of the view and the EXPLAIN command.

Action	Time	Message	Duration / Fetch
CREATE VIEW GuestAndCancellation AS SELECT G.GuestID, G.FirstName, G.LastName, C.Reason FROM GUEST G LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID;	49 22:12:43	Error Code: 1050. Table 'GuestAndCancellation' already exists	0.000 sec
explain select * from GuestAndCancellation	50 22:12:48	2 row(s) returned	0.000 sec / 0.000 sec

Creating Index (Guest)

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema `hoteldatabase1` containing tables like `cancellation`, `contactnum`, `guest`, `invoice`, `payment`, `reservation`, `room`, `roomtype`, `staff`, and `Guest`.
- SQL Editor:** Contains the following SQL code:

```
538
539 • explain select * from GuestAndCancellation ;
540
541 -- Index creation for GUEST table
542 • CREATE INDEX idx_guest_id ON GUEST (GuestID);
543 -- Index creation for CANCELLATION table
544 • CREATE INDEX idx_cancellation_guest_id ON CANCELLATION (GuestID);
545 • show indexes from Guest;
```
- Result Grid:** Displays the execution plan for the EXPLAIN command. The output shows a single query with one row in the table `Guest` and one row in the table `CANCELLATION`.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
guest	0	PRIMARY	1	GuestID	A	10	NULL	NULL	NULL	BTREE	YES	NULL	YES	NULL
guest	1	idx_guest_guestid	1	GuestID	A	9	NULL	NULL	NULL	BTREE	YES	NULL	YES	NULL
guest	1	idx_guest_id	1	GuestID	A	9	NULL	NULL	NULL	BTREE	YES	NULL	YES	NULL
- Object Info:** Shows the `reservation` table with columns: `GuestID`, `CheckInDate`, `CheckOutDate`, `NumberOfGuests`, and `RoomNumber`.
- Action Output:** Shows the creation of the index and the SHOW INDEXES command.

Action	Time	Message	Duration / Fetch
CREATE INDEX idx_guest_id ON GUEST (GuestID)	51 22:13:19	0 row(s) affected, 1 warning(s): 1831 Duplicate index 'idx_guest_id' defined on the table 'hoteldatabase1.guest'. This ...	0.109 sec
show indexes from Guest	52 22:13:24	3 row(s) returned	0.016 sec / 0.000 sec

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff), Views, Stored Procedures, Functions.
- SQL Editor:** SQL File 8*, SQL File 10*, SQL File 13*, HotelManagementSystem*. The current query is:

```
339 • explain select * from GuestAndCancellation ;
340
341 -- Index creation for GUEST table
342 • CREATE INDEX idx_guest_id ON GUEST (GuestID);
343 -- Index creation for CANCELLATION table
344 • CREATE INDEX idx_cancellation_guest_id ON CANCELLATION (GuestID);
345 • show indexes from Guest;
```

- Result Grid:** Shows the results of the EXPLAIN command for the SELECT statement. It includes columns: Table, Non_Unique, Key_name, Seq_in_index, Column_name, Collation, Cardinality, Sub_part, Packed, Null, Index_type, Comment, Index_comment, Visible, Expression. The results are:

Table	Non_Unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
guest	0	PRIMARY	1	GuestID	A	10		MUL		BTREE		YES	HULL	
guest	1	idx_guest_id	1	GuestID	A	9		MUL		BTREE		YES	HULL	
guest	1	idx_guest_id	1	GuestID	A	9		MUL		BTREE		YES	HULL	

- Output Panel:** Shows the execution of the SHOW INDEXES FROM Guest command, returning 3 rows.
- Object Info:** Reservation table.
- Session Info:** 29°C Partly cloudy.
- System Bar:** Search, Taskbar icons, Date/Time (10:13 PM, 4/8/2024).

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff), Views, Stored Procedures, Functions.
- SQL Editor:** SQL File 8*, SQL File 10*, SQL File 13*, HotelManagementSystem*. The current query is:

```
347
348 -- After Tuning
349 -- Recreate the view after optimization
350 • CREATE VIEW GuestAndCancellation1 AS
351   SELECT G.GuestID, G.FirstName, G.LastName, C.Reason
352   FROM GUEST G
353   LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID;
354 • explain select * from GuestAndCancellation1 ;
```

- Result Grid:** Shows the results of the EXPLAIN command for the SELECT statement. It includes columns: id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, Extra. The results are:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	ALL	ALL	NULLS	NULLS	NULLS	NULLS	9	100.00	MUL
1	SIMPLE	c	ALL	ref	idx_cancellation_guestid	idx_cancellation_guestid	42	hoteldatabase1.g.GuestID	1	100.00	MUL

- Output Panel:** Shows the creation of the view (CREATE VIEW) and the execution of the EXPLAIN command, both returning 0 rows affected.
- Object Info:** Reservation table.
- Session Info:** 29°C Partly cloudy.
- System Bar:** Search, Taskbar icons, Date/Time (10:14 PM, 4/8/2024).

7) Tuning Nested Query with Division

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Schemas (hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff).
- SQL Editor:** SQL File 8*, employee, project, project, department, dependent, SQL File 10*, SQL File 13*, HotelManagementSystem*. The code is as follows:

```
746
747 -- 8). Division Tuning
748
749 -- Before Tuning
750 • explain select GuestID, FirstName, LastName FROM GUEST
751 WHERE GuestID NOT IN (
    SELECT GuestID FROM CANCELLATION
753 );
```

Result Grid: Shows the execution of the EXPLAIN command. The output indicates a full table scan on the GUEST table (rows: 9, filtered: 100.00) and a full table scan on the CANCELLATION table (rows: 42, filtered: 100.00). The message says "Using where; Not exists; Using index".

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	GUEST	NULL	ALL	NULL	NULL	NULL	NULL	9	100.00	Using where; Not exists; Using index
1	SIMPLE	CANCELLATION	NULL	ref	idx_cancellation_guestid	idx_cancellation_guestid	42	hoteldatabase1.GUEST.GuestID	1	100.00	

Action Output: Shows two log entries related to the EXPLAIN command.

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Local instance MySQL80, Schemas (hoteldatabase1), Tables (cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff).
- SQL Editor:** SQL File 8*, employee, project, project, department, dependent, SQL File 10*, SQL File 13*, HotelManagementSystem*. The code is as follows:

```
755 -- Create index
756 • CREATE INDEX idx_cancellation_guestid ON CANCELLATION (GuestID);
757 • show indexes from cancellation;
758
759 -- After Tuning
760 • explain SELECT GuestID, FirstName, LastName FROM GUEST
761 WHERE GuestID NOT IN (
    SELECT GuestID FROM CANCELLATION
762 );
```

Result Grid: Shows the execution of the EXPLAIN command after creating the index. The output indicates a full table scan on the GUEST table (rows: 9, filtered: 100.00) and a full table scan on the CANCELLATION table (rows: 4, filtered: 100.00). The message says "Using where; Not exists; Using index".

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Ext
cancellation	0	PRIMARY	1	CancellationID	A	6	NULL	NULL	YES	BTREE		YES	NULL	
cancellation	1	FK_CANCELLATION_PAYMENT	1	PaymentID	A	6	NULL	NULL	YES	BTREE		YES	NULL	
cancellation	1	idx_cancellationid	1	CancellationID	A	5	NULL	NULL	YES	BTREE		YES	NULL	
cancellation	1	idx_cancellation_guestid	1	GuestID	A	5	NULL	NULL	YES	BTREE		YES	NULL	

Action Output: Shows two log entries related to the EXPLAIN command and the SHOW INDEXES command.

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the query:

```
758
759  -- After Tuning
760 • explain SELECT GuestID, FirstName, LastName FROM GUEST
761   WHERE GuestID NOT IN (
762     SELECT GuestID FROM CANCELLATION
763   );
764
765
```
- Result Grid:** Shows the execution plan for the EXPLAIN command. The output includes columns for id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, and Extra.
- Action Output:** Displays log entries for the execution of the EXPLAIN command, showing time, action, message, and duration.
- System Bar:** Shows the date (4/8/2024), time (10:22 PM), and weather (28°C, Partly cloudy).

8) Tuning Outer Union

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the query:

```
718
719  -- Before Tuning
720 • CREATE VIEW CancellationAndGuest3 AS
721   SELECT GuestID AS ID, FirstName AS Name, 'Guest' AS Type
722   FROM GUEST
723   UNION
724   SELECT GuestID AS ID, Reason AS Name, 'Cancellation' AS Type
725   FROM CANCELLATION;
726
727 • explain SELECT * FROM CancellationAndGuest3;
```
- Result Grid:** Shows the execution plan for the EXPLAIN command. The output includes columns for id, select_type, table, partitions, type, possible_keys, key, key_len, ref, rows, filtered, and Extra.
- Action Output:** Displays log entries for the creation of the view and the execution of the EXPLAIN command, showing time, action, message, and duration.
- System Bar:** Shows the date (4/8/2024), time (10:22 PM), and weather (Tomorrow's high Near record).

Creating Index (Guest)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** guest, cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff.
- SQL Editor:** Contains the following SQL code:

```
731 • CREATE INDEX idx_guest_guestid ON GUEST (GuestID);
732 • CREATE INDEX idx_cancellation_guestid ON CANCELLATION (GuestID);
733 • show indexes from guest;
734 • show indexes from cancellation;
735
736 -- After Tuning
737 • CREATE VIEW CancellationAndGuest1 AS
    SELECT GuestID AS ID, FirstName AS Name, 'Guest' AS Type
    FROM GUEST
```
- Result Grid:** Shows the results of the 'show indexes' command for the GUEST table.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
guest	0	PRIMARY	1	GuestID	A	10				BTREE			YES	
guest	1	idx_guest_guestid	1	GuestID	A	9				BTREE			YES	
guest	1	idx_guest_id	1	GuestID	A	9				BTREE			YES	
- Action Output:** Displays the execution of the 'explain' and 'show indexes' commands.

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** cancellation, contactnum, guest, invoice, payment, reservation, room, roomtype, staff.
- SQL Editor:** Contains the following SQL code:

```
731 • CREATE INDEX idx_guest_guestid ON GUEST (GuestID);
732 • CREATE INDEX idx_cancellation_guestid ON CANCELLATION (GuestID);
733 • show indexes from guest;
734 • show indexes from cancellation;
735
736 -- After Tuning
737 • CREATE VIEW CancellationAndGuest1 AS
```
- Result Grid:** Shows the results of the 'show indexes' command for the CANCELLATION table.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
cancellation	0	PRIMARY	1	CancellationID	A	6				BTREE			YES	
cancellation	1	FK_CANCELLATION_PAYMENT	1	PaymentID	A	6				BTREE			YES	
cancellation	1	idx_cancellationid	1	CancellationID	A	5				BTREE			YES	
cancellation	1	idx_cancellation_guestid	1	GuestID	A	5				BTREE			YES	
- Action Output:** Displays the execution of the 'show indexes' command for both the GUEST and CANCELLATION tables.

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

SQL Editor:

```
737 • CREATE VIEW CancellationAndGuest1 AS
738     SELECT GuestID AS ID, FirstName AS Name, 'Guest' AS Type
739     FROM GUEST
740     UNION
741     SELECT GuestID AS ID, Reason AS Name, 'Cancellation' AS Type
742     FROM CANCELLATION;
743
744 • explain SELECT * FROM CancellationAndGuest1;
```

Result Grid:

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	PRIMARY	<derived2>	ALL	ALL	idx_guest_id	idx_guest_id	42	hoteldatabase1.GuestID	15	100.00	Using temporary
2	DERIVED	guest	ALL	ALL	idx_guest_id	idx_guest_id	42	hoteldatabase1.GuestID	9	100.00	
3	UNION	cancellation	ALL	ALL	idx_reason	idx_reason	42	hoteldatabase1.Cancellation	6	100.00	
4	UNION RESULT	<union2,3>	ALL	ALL	idx_guest_id	idx_guest_id	42	hoteldatabase1.GuestID	Using temporary		

Action Output:

#	Time	Action	Message	Duration / Fetch
1	80 22:25:29	CREATE VIEW CancellationAndGuest1 AS SELECT GuestID AS ID, FirstName AS Name, 'Guest' AS Type FROM ...	0 row(s) affected	0.015 sec
2	81 22:25:33	explain SELECT * FROM CancellationAndGuest1	4 row(s) returned	0.000 sec / 0.000 sec

9) Tuning Full Outer Join

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

SQL Editor:

```
588 -- Before Tuning
589 • explain SELECT *
590     FROM GUEST G
591     LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID
592     UNION ALL
593
594     SELECT *
595     FROM GUEST G
596     RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID
597     WHERE G.GuestID IS NULL;
598
```

Result Grid:

ID	Select_Type	Table	Partitions	Type	Possible_Keys	Key	Key_Len	Ref	Rows	Filtered	Extra
1	PRIMARY	G	ALL	ALL	idx_guest_id	idx_guest_id	42	hoteldatabase1.GuestID	9	100.00	
1	PRIMARY	C	ALL	ref	idx_cancellation_guestid	idx_cancellation_guestid	42	hoteldatabase1.GuestID	1	100.00	
2	UNION	C	ALL	ALL	idx_reason	idx_reason	42	hoteldatabase1.Cancellation	6	100.00	
2	UNION	G	ALL	PRIMARY,ref	idx_guest_id, idx_guest_id	idx_guest_id, idx_guest_id	42	hoteldatabase1.GuestID	9	11.11	Using where; Not exists; Using temporary

Action Output:

#	Time	Action	Message	Duration / Fetch
1	68 22:18:20	explain SELECT * FROM RoomAndReservation2	2 row(s) returned	0.000 sec / 0.000 sec
2	69 22:19:39	explain SELECT * FROM GUEST G LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID UNION ALL SELECT * FROM GUEST G RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID WHERE G.GuestID IS NULL;	4 row(s) returned	0.000 sec / 0.000 sec

Creating Index (Guest)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation
- SQL Editor:** Contains the following SQL code:

```
594 SELECT *
595 FROM GUEST G
596 RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID
597 WHERE G.GuestID IS NULL;
598
599
700 • CREATE INDEX idx_guest_guestid ON GUEST (GuestID);
701 • CREATE INDEX idx_cancellation_guestid ON CANCELLATION (GuestID);
702 • show indexes from guest;
703 • show index from cancellation;
```
- Result Grid:** Shows the results of the SHOW INDEXES query for the guest table.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
guest	0	PRIMARY	1	GuestID	A	10				BTREE			YES	
guest	1	idx_guest_guestid	1	GuestID	A	9				BTREE			YES	
guest	1	idx_guest_id	1	GuestID	A	9				BTREE			YES	
- Output:** Shows the execution of the SHOW INDEXES command with a duration of 0.016 sec / 0.000 sec.

Creating Index (Cancellation)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** hoteldatabase1
- Tables:** reservation
- SQL Editor:** Contains the same SQL code as the previous screenshot:

```
594 SELECT *
595 FROM GUEST G
596 RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID
597 WHERE G.GuestID IS NULL;
598
599
700 • CREATE INDEX idx_guest_guestid ON GUEST (GuestID);
701 • CREATE INDEX idx_cancellation_guestid ON CANCELLATION (GuestID);
702 • show indexes from guest;
703 • show index from cancellation;
```
- Result Grid:** Shows the results of the SHOW INDEXES query for the cancellation table.

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
cancellation	0	PRIMARY	1	CancellationID	A	6				BTREE			YES	
cancellation	1	FK_CANCELLATION_PAYMENT	1	PaymentID	A	6				BTREE			YES	
cancellation	1	idx_cancellationid	1	CancellationID	A	5				BTREE			YES	
cancellation	1	idx_cancellation_guestid	1	GuestID	A	5				BTREE			YES	
- Output:** Shows the execution of the SHOW INDEXES command with a duration of 0.016 sec / 0.000 sec.

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

SQL Editor:

```
706 • explain SELECT *
707   FROM GUEST G
708   LEFT JOIN CANCELLATION C ON G.GuestID = C.GuestID
709   UNION ALL
710   -- Right Outer Join
711   SELECT *
712   FROM GUEST G
713   RIGHT JOIN CANCELLATION C ON G.GuestID = C.GuestID
714   WHERE G.GuestID IS NULL;
715
716
717    Outer Union Tuning
```

Result Grid:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	G	NULL	ALL	NULL	NULL	NULL	NULL	9	100.00	Using where; Not exists; Using temporary; Using filesort
1	PRIMARY	C	NULL	ref	idx_cancellation_guestid	idx_cancellation_guestid	42	hoteldatabase1.G.GuestID	1	100.00	Using where; Not exists; Using temporary; Using filesort
2	UNION	C	NULL	ALL	NULL	NULL	NULL	NULL	6	100.00	Using where; Not exists; Using temporary; Using filesort
2	UNION	G	NULL	ALL	PRIMARY, idx_guest_guestid, idx_guest_id	idx_guest_guestid	NULL	NULL	9	11.11	Using where; Not exists; Using temporary; Using filesort

Session Information:

- Object Info: reservation
- Session: 73 22:20:47 show index from cancellation
- System: Duration / Fetch: 0.000 sec / 0.000 sec | 10:22 PM 4/8/2024
- Environment: 28°C Partly cloudy

10) Tuning Right Outer Join

Before Tuning

The screenshot shows the MySQL Workbench interface with the following details:

SQL Editor:

```
560
561   -- Before Tuning
562 • CREATE VIEW RoomAndReservation1 AS
563   SELECT R.RoomNumber, R.Type, RES.NumberOfGuests
564   FROM ROOM R
565   RIGHT JOIN RESERVATION RES ON R.RoomNumber = RES.RoomNumber;
566
567 • explain SELECT * FROM RoomAndReservation1;
```

Result Grid:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	res	NULL	ALL	NULL	NULL	NULL	NULL	8	100.00	Using where; Using temporary; Using filesort
1	SIMPLE	r	NULL	eq_ref	PRIMARY	PRIMARY	22	hoteldatabase1.res.RoomNumber	1	100.00	Using where; Using temporary; Using filesort

Session Information:

- Object Info: reservation
- Session: 60 22:15:54 CREATE VIEW RoomAndReservation1 AS SELECT R.RoomNumber, R.Type, RES.NumberOfGuests FROM ROO... 0 row(s) affected | 61 22:15:00 explain SELECT * FROM RoomAndReservation1 2 row(s) returned
- System: Duration / Fetch: 0.015 sec / 0.000 sec / 0.000 sec | 10:16 PM 4/8/2024
- Environment: 29°C Partly cloudy

Creating Index (Room)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** reservation, room, roomtype, staff, payment, guest, contactnum, cancellator, employee, project, department, dependent.
- SQL Editor:** Contains the following SQL code:

```
569 -- Index creation for ROOM table
570 • CREATE INDEX idx_room_number ON ROOM (RoomNumber);
571 -- Index creation for RESERVATION table
572 • CREATE INDEX idx_reservation_room_number ON RESERVATION (RoomNumber);
573 • show indexes from room;
574 • show index from Reservation;
575
576 -- After Tuning
```
- Result Grid:** Shows the results of the 'show indexes from room' command.

Table	Non_Unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
room	0	PRIMARY	1	RoomNumber	A	10		HASH	HASH	BTREE			YES	HASH
room	1	idx_room_number	1	RoomNumber	A	9		HASH	HASH	BTREE			YES	HASH
- Output:** Shows the execution log with two entries:
 - 62 22:16:55 CREATE INDEX idx_room_number ON ROOM (RoomNumber)
 - 63 22:17:00 show indexes from room
- System Bar:** Includes icons for search, file operations, and system status (29°C, Partly cloudy).

Creating Index (Reservation)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Local instance MySQL80
- Tables:** reservation, room, roomtype, staff, payment, guest, contactnum, cancellator, employee, project, department, dependent.
- SQL Editor:** Contains the following SQL code:

```
569 -- Index creation for ROOM table
570 • CREATE INDEX idx_room_number ON ROOM (RoomNumber);
571 -- Index creation for RESERVATION table
572 • CREATE INDEX idx_reservation_room_number ON RESERVATION (RoomNumber);
573 • show indexes from room;
574 • show index from Reservation;
575
576 -- After Tuning
```
- Result Grid:** Shows the results of the 'show indexes from reservation' command.

Table	Non_unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
reservation	1	FK_RESERVATION_ROOM	1	RoomNumber	A	2		HASH	HASH	YES	BTREE		YES	HASH
reservation	1	idx_reservation_guestid	1	GuestID	A	8		HASH	HASH	BTREE			YES	HASH
reservation	1	idx_reservation_guestid4	1	GuestID	A	8		HASH	HASH	BTREE			YES	HASH
- Output:** Shows the execution log with two entries:
 - 63 22:17:24 show indexes from room
 - 64 22:17:24 show index from Reservation
- System Bar:** Includes icons for search, file operations, and system status (29°C, Partly cloudy).

After Tuning

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:

```
576 -- After Tuning
577 -- Recreate the view after optimization
578 • CREATE VIEW RoomAndReservation2 AS
579   SELECT R.RoomNumber, R.Type, RES.NumberOfGuests
580   FROM ROOM R
581   RIGHT JOIN RESERVATION RES ON R.RoomNumber = RES.RoomNumber;
582
583 • explain SELECT * FROM RoomAndReservation2;
```
- Result Grid:** Shows the execution plan for the EXPLAIN command. The output is:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	res	NULL	ALL	NULL	NULL	NULL	NULL	8	100.00	NULL
1	SIMPLE	r	NULL	eq_ref	PRIMARY,IDX_ROOM_NUMBER	PRIMARY	22	hoteldatabase1.res.RoomNumber	1	100.00	NULL
- Information Panel:** Shows the definition of the 'reservation' table:

```
Table: reservation
Columns:
  id          int(11)    NOT NULL
  CheckInDate date      NOT NULL
  CheckOutDate date      NOT NULL
  NumberofGuests int      NOT NULL
  RoomNumber  int      NOT NULL
```
- Action Output:** Displays log entries for the creation of the view and the EXPLAIN command.

Time	Action	Message	Duration / Fetch
67 22:18:16	CREATE VIEW RoomAndReservation2 AS SELECT R.RoomNumber, R.Type, RES.NumberOfGuests FROM ROO...	0 row(s) affected	0.015 sec
68 22:18:20	explain SELECT * FROM RoomAndReservation2	2 row(s) returned	0.000 sec / 0.000 sec
- System Status:** Shows the weather as 29°C Partly cloudy.