HOTEL BOOKING MANAGEMENT SYSTEM

SQL Project

ABSTRACT

A software solution designed to streamline and automate the management of hotel operations, focusing on the booking process

Presented by,
Aswathi Bhojan(NF1001601)
Inderjeet Kaur(NF1000809)
Aarmi Patel(NF1002881)

Submitted To,
Professor Sundus Shanef
University Of Niagara Falls, Canada

INTRODUCTION

In today's data-driven world, the ability to design, implement, and manage robust database systems is an essential skill for effectively handling complex data structures. This project focuses on leveraging SQL databases to develop a comprehensive database management system for a Hotel Booking Management System, simulating a real-world business environment. The database aims to efficiently manage various operational aspects of a hotel, including guest management, room assignments, booking records, payment processing, employee administration, service tracking, customer feedback, and maintenance scheduling.

By employing a structured and relational approach, this project will demonstrate proficiency in the use of Data Definition Language (DDL), Data Manipulation Language (DML), and Data Query Language (DQL) statements. The database is designed to meet the requirements of a scalable system that ensures data consistency, reduces redundancy, and facilitates data-driven decision-making.

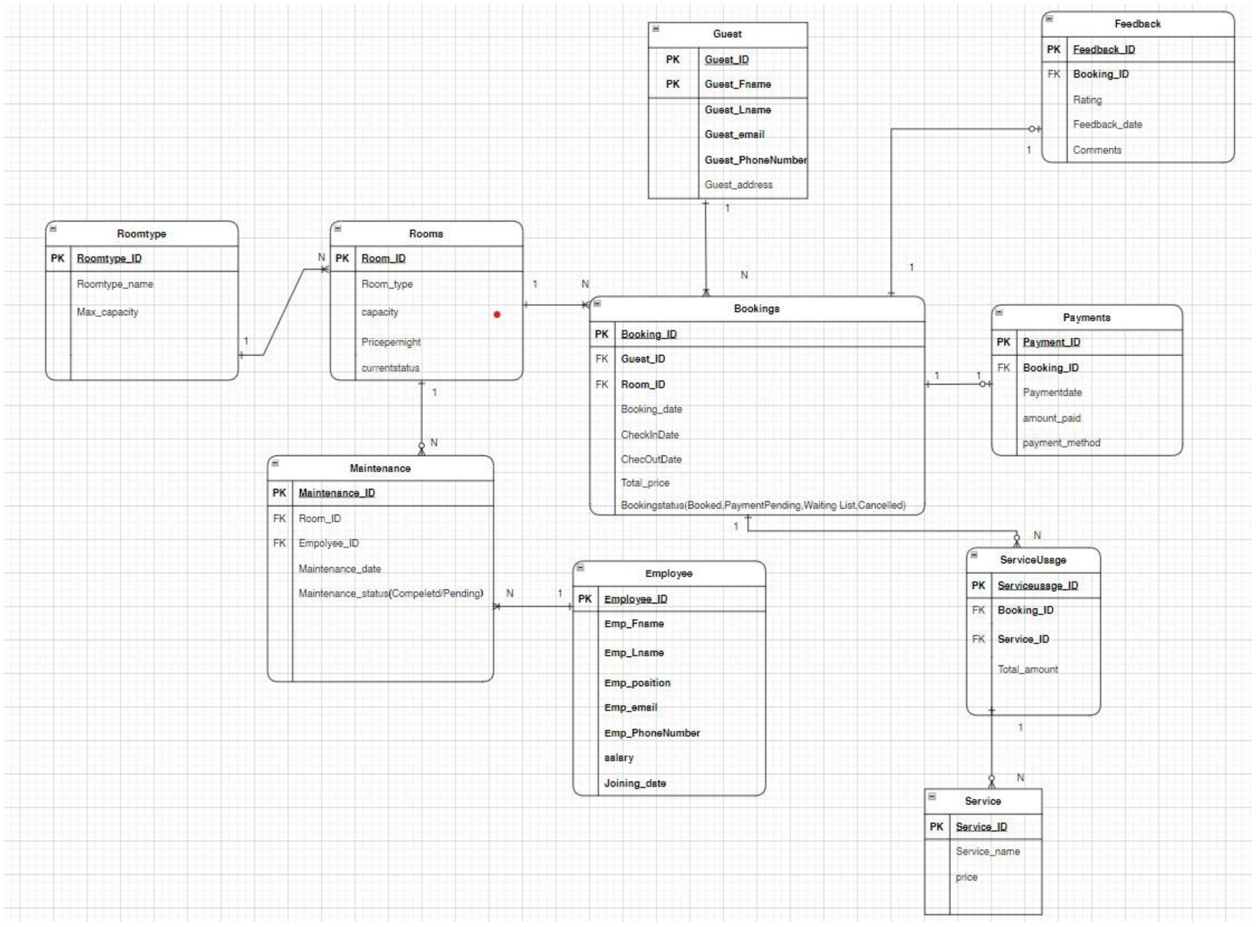
PURPOSE

The primary purpose of this project is to apply theoretical and practical knowledge of SQL databases in designing a complex and interrelated database system that mirrors the functionality of a hotel booking system. The goals of the project include:

1. **Database Design and Schema Development**: Create a schema with at least 10 interrelated tables, representing key entities such as guests, rooms, bookings, payments, services, employees, and maintenance activities. The schema will enforce data integrity through primary and foreign key constraints, as well as other relational database principles.
2. **Data Definition and Manipulation**: Utilize DDL statements to define the database structure and relationships. Populate the tables with realistic and dynamic data using DML statements, simulating real-world operational scenarios.
3. **Data Query and Analysis**: Leverage DQL statements to extract and analyze data, demonstrating the capability to retrieve meaningful insights. These queries will support tasks such as revenue tracking, room occupancy analysis, employee performance evaluation, and customer satisfaction assessment.
4. **Machine Learning Dataset Preparation**: Design the database in a way that supports the extraction of clean and structured datasets suitable for machine learning projects. This involves creating queries to select relevant features for predictive modeling and analysis.
5. **Practical Application**: Provide a foundation for understanding how database systems underpin real-world applications and prepare datasets for advanced analytics, making the system versatile for both operational management and strategic decision-making

Database Design Document: LodgingHub(Database)

1. ER Diagram



2. Tables and Columns

2.1 Guest Table

Column Name	Data Type
Guest_ID	INT, PRIMARY KEY, UNIQUE, NOT NULL
Guest_Fname	VARCHAR (50), NOT NULL
Guest_Lname	VARCHAR (50), NOT NULL
Guest_email	VARCHAR (100), NOT NULL
Guest_PhoneNumber	VARCHAR (15), NOT NULL
Guest_address	VARCHAR(200)

2.2 Rooms Table

Column Name	Data Type

Room_ID	INT, PRIMARY KEY, UNIQUE, NOT NULL
Room_type	VARCHAR(50)
Capacity	INT
Pricepernight	DECIMAL(10,2)
Current_status	BOOLEAN, DEFAULT TRUE

2.3 RoomType Table

Column Name	Data Type
Roomtype_ID	INT, PRIMARY KEY
Roomtype_name	VARCHAR(50)
Max_capacity	INT

2.4 Bookings Table

Column Name	Data Type
Booking_ID	INT, PRIMARY KEY, UNIQUE NOT NULL
Guest_ID	INT, NOT NULL
Room_ID	INT, NOT NULL
CheckInDate	DATE, NOT NULL
CheckOutDate	DATE, NOT NULL
Total_price	DECIMAL (10,2)
Bookingstatus	ENUM ('Booked','Payment Pending','Waiting List','Cancelled')

2.5 Payments Table

Column Name	Data Type
Payment_ID	INT, PRIMARY KEY, UNIQUE, NOT NULL
Booking_ID	INT, NOT NULL
Paymentdate	DATE
Amount_paid	DECIMAL (10,2)
Payment_method	VARCHAR (20)

2.6 Employee Table

Column Name	Data Type
Employee_ID	INT, PRIMARY KEY, UNIQUE, NOT NULL
Emp_Fname	VARCHAR (50), NOT NULL
Emp_Lname	VARCHAR (50), NOT NULL
Emp_position	VARCHAR (50), NOT NULL
Emp_email	VARCHAR (100), NOT NULL
Emp_PhoneNumber	VARCHAR (15), NOT NULL
Salary	DECIMAL (10,2), NOT NULL
Joining_date	DATE, NOT NULL

2.7 Service Table

Column Name	Data Type
Service_ID	INT, PRIMARY KEY, UNIQUE, NOT NULL
Service_name	VARCHAR (50)
Price	DECIMAL (10,2)

2.8 ServiceUsage Table

Column Name	Data Type
Serviceusage_ID	INT, PRIMARY KEY, UNIQUE
Service_ID	INT, NOT NULL
Booking_ID	INT, NOT NULL
Total_amount	DECIMAL (10,2)

2.9 Feedback Table

Column Name	Data Type
Feedback_ID	INT, PRIMARY KEY
Booking_ID	INT, NOT NULL
Rating	INT
Feedback_date	DATE
Comments	VARCHAR (300)

2.10 Maintenance Table

Column Name	Data Type
Maintenance_ID	INT, PRIMARY KEY, UNIQUE
Room_ID	INT
Employee_ID	INT
Maintenance_date	DATE
Maintenance_status	ENUM('Completed','Pending')

3. Primary Key and Foreign Key

The following primary and foreign keys are used in the database schema:

1. Guest: Guest_ID (Primary Key)
2. Rooms: Room_ID (Primary Key)
3. RoomType: Roomtype_ID (Primary Key)
4. Bookings: Booking_ID (Primary Key), Guest_ID and Room_ID (Foreign Keys)
5. Payments: Payment_ID (Primary Key), Booking_ID (Foreign Key)
6. Employee: Employee_ID (Primary Key)
7. Service: Service_ID (Primary Key)
8. ServiceUsage: Serviceusage_ID (Primary Key), Service_ID and Booking_ID (Foreign Keys)
9. Feedback: Feedback_ID (Primary Key), Booking_ID (Foreign Key)
10. Maintenance: Maintenance_ID (Primary Key), Room_ID and Employee_ID (Foreign Keys)

4. Relationships Between Tables

1. Guest – Bookings (One-to-Many (Mandatory))
2. Rooms – Bookings (One-to-Many (Mandatory))
3. Bookings – Payments (One-to-One (Optional))
4. Bookings – Feedback (One-to-One (Optional))
5. Bookings – ServiceUsage (One-to-Many (Optional))
6. Rooms -Maintenance (One-to-Many (Optional))
7. Employee -Maintenance (One-to-Many (Mandatory))
8. Service – ServiceUsage (One-to-Many (Optional))
9. Rooms – RoomType (Many-To-One(Mandatory))

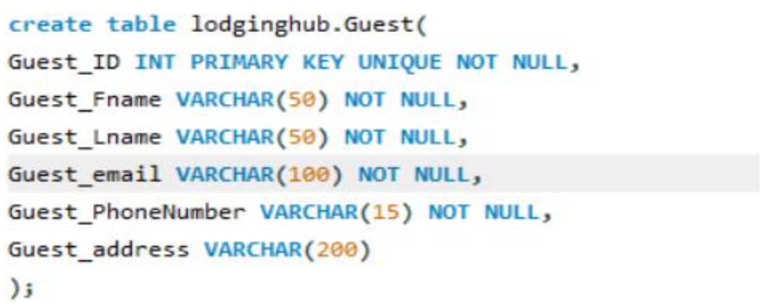
DATA DEFINITION LANGUAGE(DDL)

DDL which involves definition statements such as CREATE, DROP, and ALTER. In this project Schema and Table creation are done using the query CREATE. Below are the query used to :

Creating database

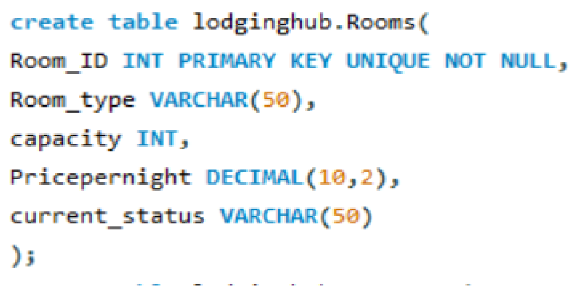
create database lodginghub;

Create Table Guest



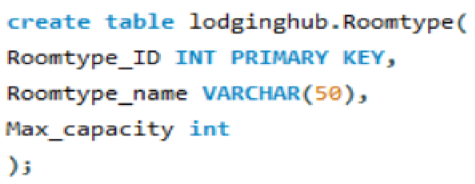
```
create table lodginghub.Guest(
Guest_ID INT PRIMARY KEY UNIQUE NOT NULL,
Guest_Fname VARCHAR(50) NOT NULL,
Guest_Lname VARCHAR(50) NOT NULL,
Guest_email VARCHAR(100) NOT NULL,
Guest_PhoneNumber VARCHAR(15) NOT NULL,
Guest_address VARCHAR(200)
);
```

Create Table Rooms



```
create table lodginghub.Rooms(
Room_ID INT PRIMARY KEY UNIQUE NOT NULL,
Room_type VARCHAR(50),
capacity INT,
Pricepernight DECIMAL(10,2),
current_status VARCHAR(50)
);
```

Create Table Roomtype



```
create table lodginghub.Roomtype(
Roomtype_ID INT PRIMARY KEY,
Roomtype_name VARCHAR(50),
Max_capacity int
);
```

Create Table Bookings

```sql
create table lodginghub.Bookings(
Booking_ID INT PRIMARY KEY UNIQUE NOT NULL,
Guest_ID INT NOT NULL,
Room_ID INT NOT NULL,
Booking_date date not null,
CheckInDate date not null,
CheckOutDate date not null,
Total_price decimal(10,2),
Bookingstatus enum('Booked','Payment Pending','Waiting List','Cancelled'),
FOREIGN KEY (Guest_ID) REFERENCES lodginghub.Guest(Guest_ID),
FOREIGN KEY (Room_ID) REFERENCES lodginghub.Rooms(Room_ID)
);
```

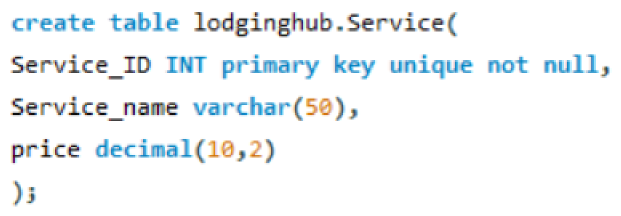
Create Table Payments

```sql
create table lodginghub.Payments(
Payment_ID INT primary key unique not null,
Booking_ID INT not null,
Paymentdate DATE,
amount_paid decimal(10,2),
payment_method varchar(20),
foreign key(Booking_ID) references lodginghub.Bookings(Booking_ID)
);
```

Create Table Employee

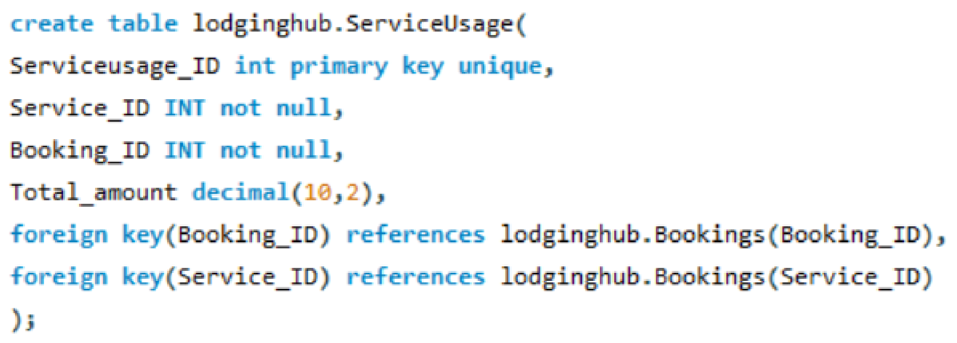
```sql
create table lodginghub.Employee(
Employee_ID INT primary key unique not null,
Emp_Fname VARCHAR(50) NOT NULL,
Emp_Lname VARCHAR(50) NOT NULL,
Emp_position VARCHAR(50) NOT NULL,
Emp_email varchar(100) not null,
Emp_PhoneNumber VARCHAR(15) NOT NULL,
salary decimal(10,2) not null,
Joining_date date not null
);
```

Create Table Service



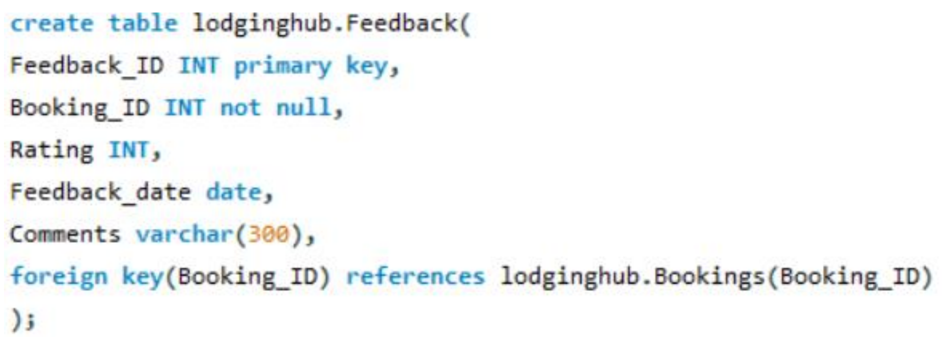
```
create table lodginghub.Service(
Service_ID INT primary key unique not null,
Service_name varchar(50),
price decimal(10,2)
);
```

Create Table Serviceusage



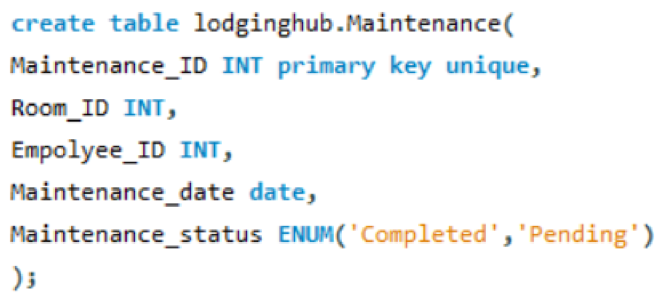
```
create table lodginghub.ServiceUsage(
Serviceusage_ID int primary key unique,
Service_ID INT not null,
Booking_ID INT not null,
Total_amount decimal(10,2),
foreign key(Booking_ID) references lodginghub.Bookings(Booking_ID),
foreign key(Service_ID) references lodginghub.Bookings(Service_ID)
);
```

Crete Table Feedback



```
create table lodginghub.Feedback(
Feedback_ID INT primary key,
Booking_ID INT not null,
Rating INT,
Feedback_date date,
Comments varchar(300),
foreign key(Booking_ID) references lodginghub.Bookings(Booking_ID)
);
```

Create Table Maintenance



```
create table lodginghub.Maintenance(
Maintenance_ID INT primary key unique,
Room_ID INT,
Empolyee_ID INT,
Maintenance_date date,
Maintenance_status ENUM('Completed','Pending')
);
```

DATA RETRIEVAL STATEMENTS(DQL)

DQL is a subset of SQL (Structured Query Language) used for retrieving and querying data from a database. It focuses on fetching the required data based on specific criteria without modifying the database's structure or the stored data itself. Unlike DML (Data Manipulation Language), it does not modify or update data. It allows the use of filters, conditions, and aggregation to get precise results.

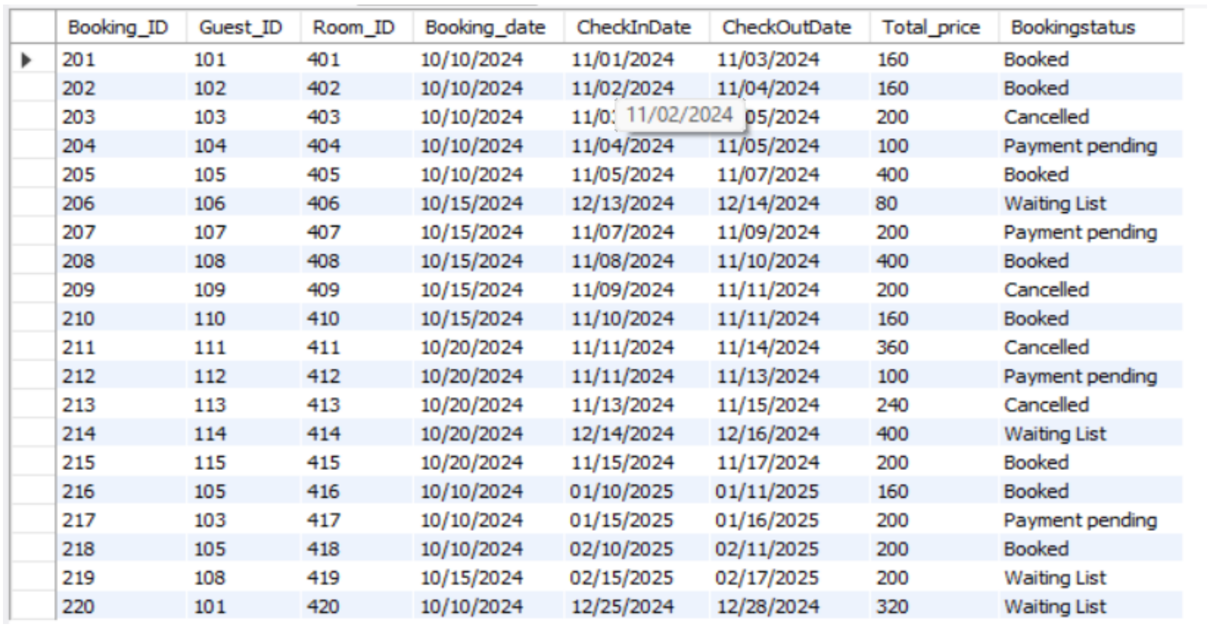
The main command in DQL is:

SELECT

The SELECT statement is used to retrieve data from one or more tables

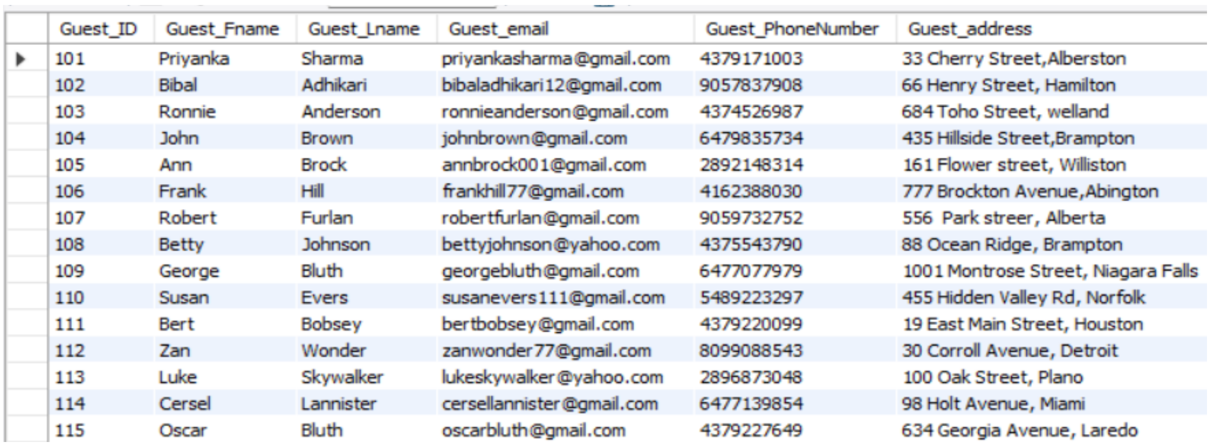
Below is the output of retrieval of table involved in this project using SELECT

select * from lodginghub.bookings;



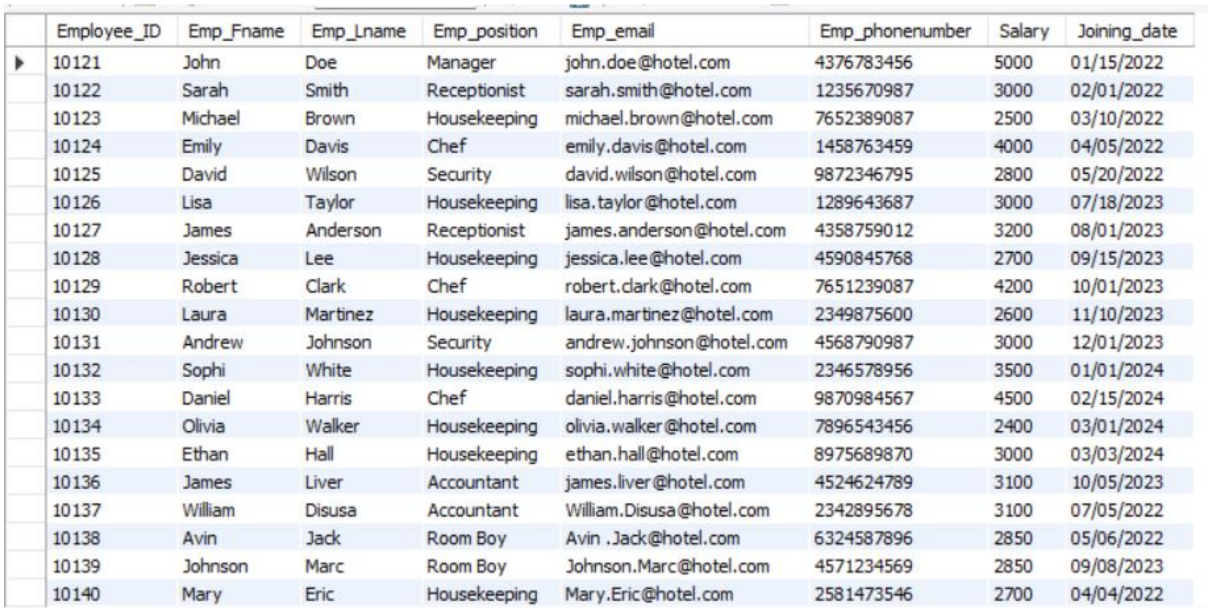
Booking_ID	Guest_ID	Room_ID	Booking_date	CheckInDate	CheckOutDate	Total_price	Bookingstatus
201	101	401	10/10/2024	11/01/2024	11/03/2024	160	Booked
202	102	402	10/10/2024	11/02/2024	11/04/2024	160	Booked
203	103	403	10/10/2024	11/0 11/02/2024	05/2024	200	Cancelled
204	104	404	10/10/2024	11/04/2024	11/05/2024	100	Payment pending
205	105	405	10/10/2024	11/05/2024	11/07/2024	400	Booked
206	106	406	10/15/2024	12/13/2024	12/14/2024	80	Waiting List
207	107	407	10/15/2024	11/07/2024	11/09/2024	200	Payment pending
208	108	408	10/15/2024	11/08/2024	11/10/2024	400	Booked
209	109	409	10/15/2024	11/09/2024	11/11/2024	200	Cancelled
210	110	410	10/15/2024	11/10/2024	11/11/2024	160	Booked
211	111	411	10/20/2024	11/11/2024	11/14/2024	360	Cancelled
212	112	412	10/20/2024	11/11/2024	11/13/2024	100	Payment pending
213	113	413	10/20/2024	11/13/2024	11/15/2024	240	Cancelled
214	114	414	10/20/2024	12/14/2024	12/16/2024	400	Waiting List
215	115	415	10/20/2024	11/15/2024	11/17/2024	200	Booked
216	105	416	10/10/2024	01/10/2025	01/11/2025	160	Booked
217	103	417	10/10/2024	01/15/2025	01/16/2025	200	Payment pending
218	105	418	10/10/2024	02/10/2025	02/11/2025	200	Booked
219	108	419	10/15/2024	02/15/2025	02/17/2025	200	Waiting List
220	101	420	10/10/2024	12/25/2024	12/28/2024	320	Waiting List

select * from lodginghub.guest;



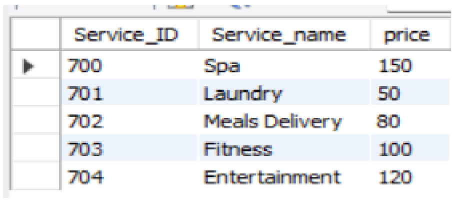
Guest_ID	Guest_Fname	Guest_Lname	Guest_email	Guest_PhoneNumber	Guest_address
101	Priyanka	Sharma	priyankasharma@gmail.com	4379171003	33 Cherry Street,Alberston
102	Bibal	Adhikari	bibaladhikari12@gmail.com	9057837908	66 Henry Street, Hamilton
103	Ronnie	Anderson	ronnieanderson@gmail.com	4374526987	684 Toho Street, welland
104	John	Brown	johnbrown@gmail.com	6479835734	435 Hillside Street,Brampton
105	Ann	Brock	annbrock001@gmail.com	2892148314	161 Flower street, Williston
106	Frank	Hill	frankhill77@gmail.com	4162388030	777 Brockton Avenue,Abington
107	Robert	Furlan	robertfurlan@gmail.com	9059732752	556 Park streer, Alberta
108	Betty	Johnson	bettyjohnson@yahoo.com	4375543790	88 Ocean Ridge, Brampton
109	George	Bluth	georgebluth@gmail.com	6477077979	1001 Montrose Street, Niagara Falls
110	Susan	Evers	susanevers111@gmail.com	5489223297	455 Hidden Valley Rd, Norfolk
111	Bert	Bobsey	bertbobsey@gmail.com	4379220099	19 East Main Street, Houston
112	Zan	Wonder	zanwonder77@gmail.com	8099088543	30 Corroll Avenue, Detroit
113	Luke	Skywalker	lukeskywalker@yahoo.com	2896873048	100 Oak Street, Plano
114	Cersel	Lannister	cersellannister@gmail.com	6477139854	98 Holt Avenue, Miami
115	Oscar	Bluth	oscarbluth@gmail.com	4379227649	634 Georgia Avenue, Laredo

select * from lodginghub.employee;



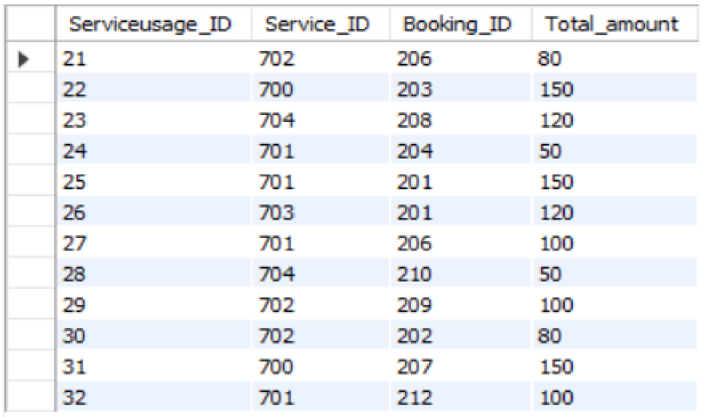
Employee_ID	Emp_Fname	Emp_Lname	Emp_position	Emp_email	Emp_phonenumber	Salary	Joining_date
10121	John	Doe	Manager	john.doe@hotel.com	4376783456	5000	01/15/2022
10122	Sarah	Smith	Receptionist	sarah.smith@hotel.com	1235670987	3000	02/01/2022
10123	Michael	Brown	Housekeeping	michael.brown@hotel.com	7652389087	2500	03/10/2022
10124	Emily	Davis	Chef	emily.davis@hotel.com	1458763459	4000	04/05/2022
10125	David	Wilson	Security	david.wilson@hotel.com	9872346795	2800	05/20/2022
10126	Lisa	Taylor	Housekeeping	lisa.taylor@hotel.com	1289643687	3000	07/18/2023
10127	James	Anderson	Receptionist	james.anderson@hotel.com	4358759012	3200	08/01/2023
10128	Jessica	Lee	Housekeeping	jessica.lee@hotel.com	4590845768	2700	09/15/2023
10129	Robert	Clark	Chef	robert.clark@hotel.com	7651239087	4200	10/01/2023
10130	Laura	Martinez	Housekeeping	laura.martinez@hotel.com	2349875600	2600	11/10/2023
10131	Andrew	Johnson	Security	andrew.johnson@hotel.com	4568790987	3000	12/01/2023
10132	Sophi	White	Housekeeping	sophi.white@hotel.com	2346578956	3500	01/01/2024
10133	Daniel	Harris	Chef	daniel.harris@hotel.com	9870984567	4500	02/15/2024
10134	Olivia	Walker	Housekeeping	olivia.walker@hotel.com	7896543456	2400	03/01/2024
10135	Ethan	Hall	Housekeeping	ethan.hall@hotel.com	8975689870	3000	03/03/2024
10136	James	Liver	Accountant	james.liver@hotel.com	4524624789	3100	10/05/2023
10137	William	Disusa	Accountant	William.Disusa@hotel.com	2342895678	3100	07/05/2022
10138	Avin	Jack	Room Boy	Avin .Jack@hotel.com	6324587896	2850	05/06/2022
10139	Johnson	Marc	Room Boy	Johnson.Marc@hotel.com	4571234569	2850	09/08/2023
10140	Mary	Eric	Housekeeping	Mary.Eric@hotel.com	2581473546	2700	04/04/2022

select * from lodginghub.service;



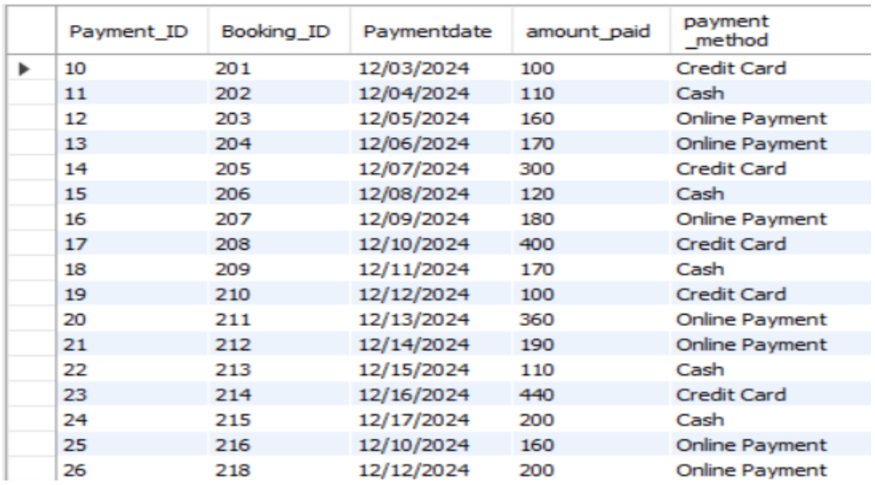
Service_ID	Service_name	price
700	Spa	150
701	Laundry	50
702	Meals Delivery	80
703	Fitness	100
704	Entertainment	120

select * from lodginghub.serviceusage;



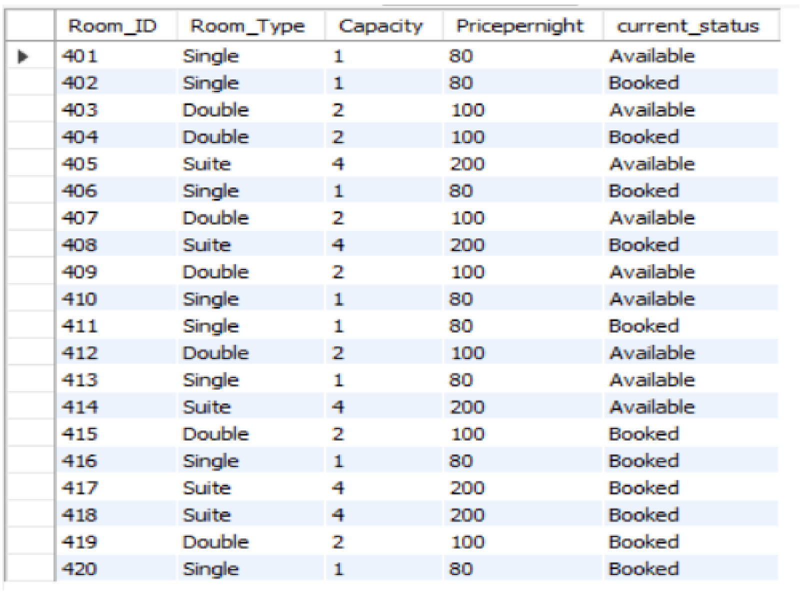
Serviceusage_ID	Service_ID	Booking_ID	Total_amount
21	702	206	80
22	700	203	150
23	704	208	120
24	701	204	50
25	701	201	150
26	703	201	120
27	701	206	100
28	704	210	50
29	702	209	100
30	702	202	80
31	700	207	150
32	701	212	100

select * from lodginghub.payments;



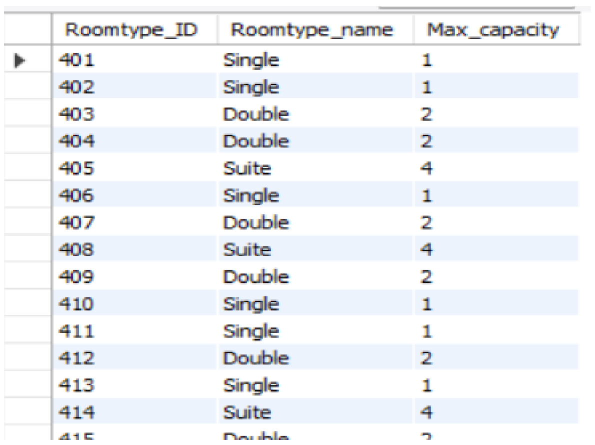
Payment_ID	Booking_ID	Paymentdate	amount_paid	payment _method
10	201	12/03/2024	100	Credit Card
11	202	12/04/2024	110	Cash
12	203	12/05/2024	160	Online Payment
13	204	12/06/2024	170	Online Payment
14	205	12/07/2024	300	Credit Card
15	206	12/08/2024	120	Cash
16	207	12/09/2024	180	Online Payment
17	208	12/10/2024	400	Credit Card
18	209	12/11/2024	170	Cash
19	210	12/12/2024	100	Credit Card
20	211	12/13/2024	360	Online Payment
21	212	12/14/2024	190	Online Payment
22	213	12/15/2024	110	Cash
23	214	12/16/2024	440	Credit Card
24	215	12/17/2024	200	Cash
25	216	12/10/2024	160	Online Payment
26	218	12/12/2024	200	Online Payment

select * from lodginghub.rooms;



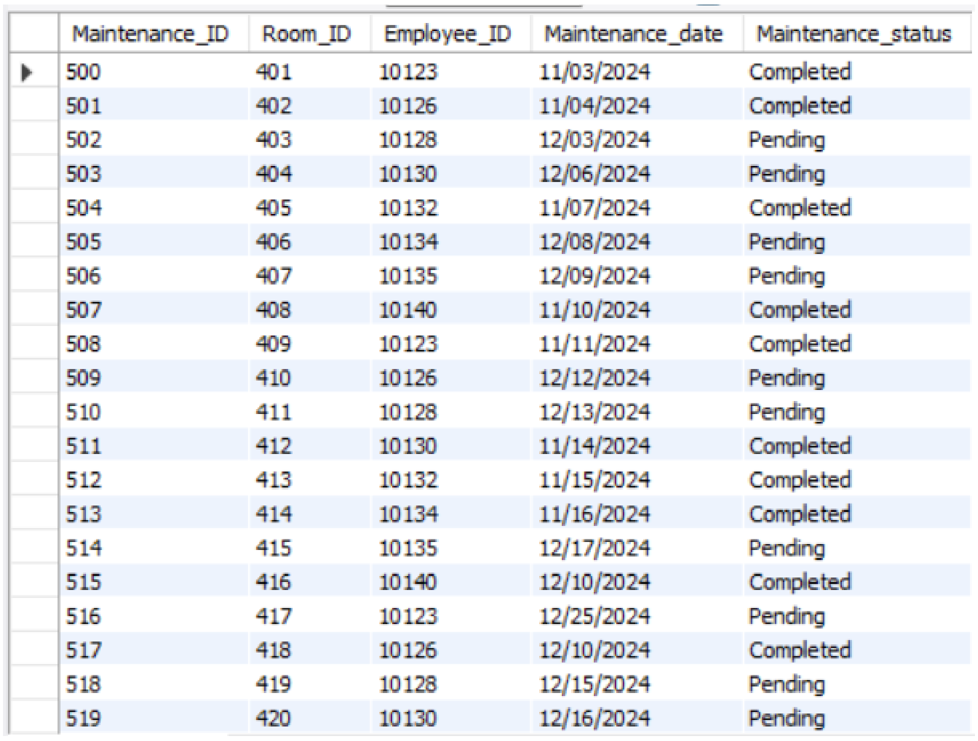
Room_ID	Room_Type	Capacity	Pricepernight	current_status
401	Single	1	80	Available
402	Single	1	80	Booked
403	Double	2	100	Available
404	Double	2	100	Booked
405	Suite	4	200	Available
406	Single	1	80	Booked
407	Double	2	100	Available
408	Suite	4	200	Booked
409	Double	2	100	Available
410	Single	1	80	Available
411	Single	1	80	Booked
412	Double	2	100	Available
413	Single	1	80	Available
414	Suite	4	200	Available
415	Double	2	100	Booked
416	Single	1	80	Booked
417	Suite	4	200	Booked
418	Suite	4	200	Booked
419	Double	2	100	Booked
420	Single	1	80	Booked

select * from lodginghub.roomtype;



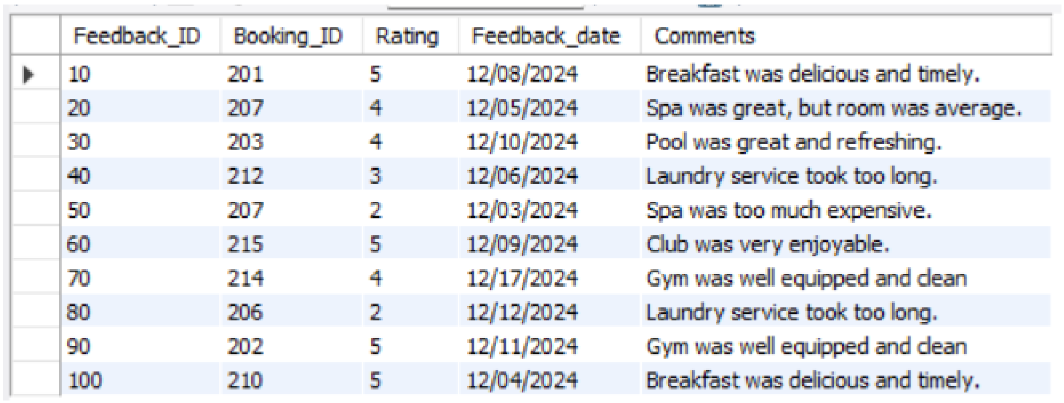
Roomtype_ID	Roomtype_name	Max_capacity
401	Single	1
402	Single	1
403	Double	2
404	Double	2
405	Suite	4
406	Single	1
407	Double	2
408	Suite	4
409	Double	2
410	Single	1
411	Single	1
412	Double	2
413	Single	1
414	Suite	4
415	Double	2

select * from lodginghub.maintenance;



Maintenance_ID	Room_ID	Employee_ID	Maintenance_date	Maintenance_status
500	401	10123	11/03/2024	Completed
501	402	10126	11/04/2024	Completed
502	403	10128	12/03/2024	Pending
503	404	10130	12/06/2024	Pending
504	405	10132	11/07/2024	Completed
505	406	10134	12/08/2024	Pending
506	407	10135	12/09/2024	Pending
507	408	10140	11/10/2024	Completed
508	409	10123	11/11/2024	Completed
509	410	10126	12/12/2024	Pending
510	411	10128	12/13/2024	Pending
511	412	10130	11/14/2024	Completed
512	413	10132	11/15/2024	Completed
513	414	10134	11/16/2024	Completed
514	415	10135	12/17/2024	Pending
515	416	10140	12/10/2024	Completed
516	417	10123	12/25/2024	Pending
517	418	10126	12/10/2024	Completed
518	419	10128	12/15/2024	Pending
519	420	10130	12/16/2024	Pending

select * from lodginghub.feedback;

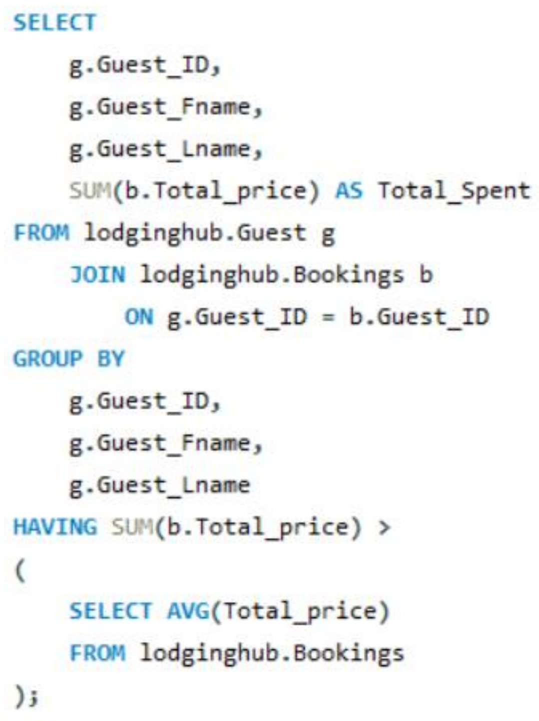


Feedback_ID	Booking_ID	Rating	Feedback_date	Comments
10	201	5	12/08/2024	Breakfast was delicious and timely.
20	207	4	12/05/2024	Spa was great, but room was average.
30	203	4	12/10/2024	Pool was great and refreshing.
40	212	3	12/06/2024	Laundry service took too long.
50	207	2	12/03/2024	Spa was too much expensive.
60	215	5	12/09/2024	Club was very enjoyable.
70	214	4	12/17/2024	Gym was well equipped and clean
80	206	2	12/12/2024	Laundry service took too long.
90	202	5	12/11/2024	Gym was well equipped and clean
100	210	5	12/04/2024	Breakfast was delicious and timely.

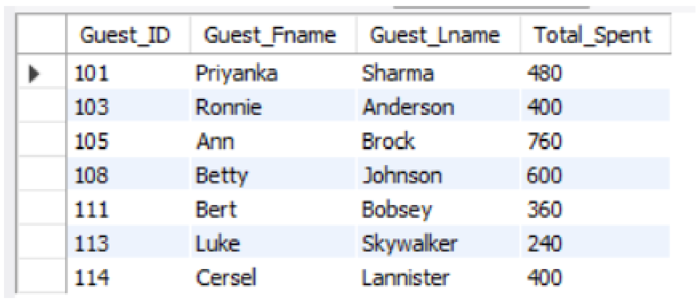
SQL CODES ON DATASET

Below codes explain different operations performed on the dataset based on certain condition using Aggregate functions, Window functions and Common Table Expressions(CTEs)

Query 1: Identify guests who have spent more than the average total booking amount across all bookings.

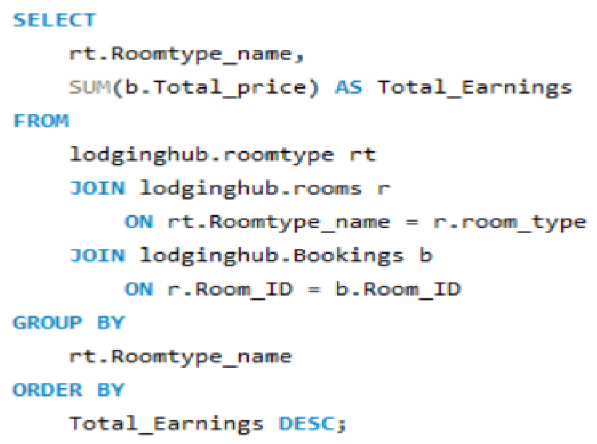


```
SELECT
    g.Guest_ID,
    g.Guest_Fname,
    g.Guest_Lname,
    SUM(b.Total_price) AS Total_Spent
FROM lodginghub.Guest g
    JOIN lodginghub.Bookings b
        ON g.Guest_ID = b.Guest_ID
GROUP BY
    g.Guest_ID,
    g.Guest_Fname,
    g.Guest_Lname
HAVING SUM(b.Total_price) >
(
    SELECT AVG(Total_price)
    FROM lodginghub.Bookings
);
```

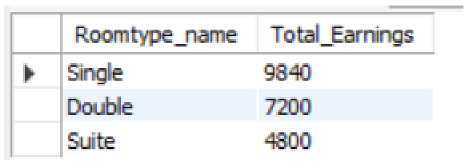


	Guest_ID	Guest_Fname	Guest_Lname	Total_Spent
▶	101	Priyanka	Sharma	480
	103	Ronnie	Anderson	400
	105	Ann	Brock	760
	108	Betty	Johnson	600
	111	Bert	Bobsey	360
	113	Luke	Skywalker	240
	114	Cersel	Lannister	400

Query 2: Calculate total earnings for each room type and sort them in descending order of earnings.

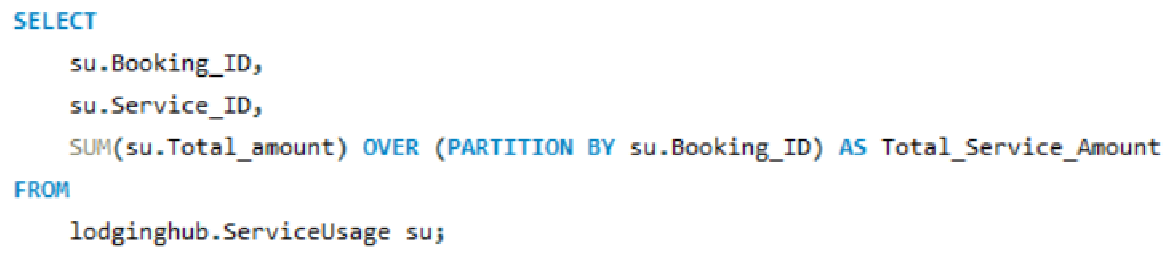


```
SELECT
    rt.Roomtype_name,
    SUM(b.Total_price) AS Total_Earnings
FROM
    lodginghub.roomtype rt
    JOIN lodginghub.rooms r
        ON rt.Roomtype_name = r.room_type
    JOIN lodginghub.Bookings b
        ON r.Room_ID = b.Room_ID
GROUP BY
    rt.Roomtype_name
ORDER BY
    Total_Earnings DESC;
```

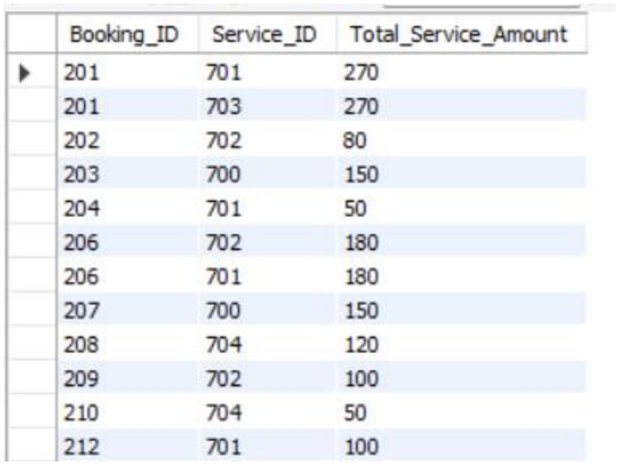


Roomtype_name	Total_Earnings
Single	9840
Double	7200
Suite	4800

Query 3: Compute the total amount spent on services for each booking, using PARTITION BY.



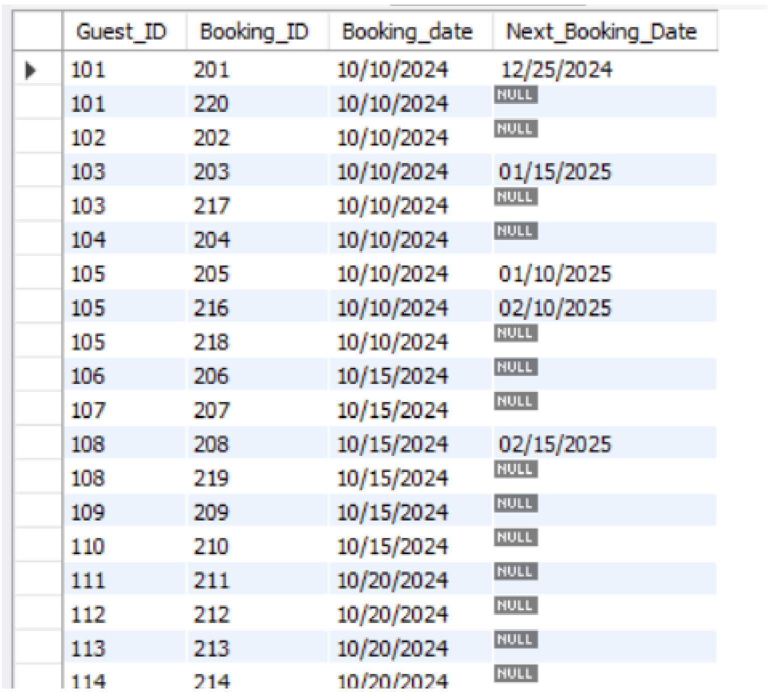
```
SELECT
    su.Booking_ID,
    su.Service_ID,
    SUM(su.Total_amount) OVER (PARTITION BY su.Booking_ID) AS Total_Service_Amount
FROM
    lodginghub.ServiceUsage su;
```



Booking_ID	Service_ID	Total_Service_Amount
201	701	270
201	703	270
202	702	80
203	700	150
204	701	50
206	702	180
206	701	180
207	700	150
208	704	120
209	702	100
210	704	50
212	701	100

Query 4: For each guest, find the next booking date if they have multiple bookings

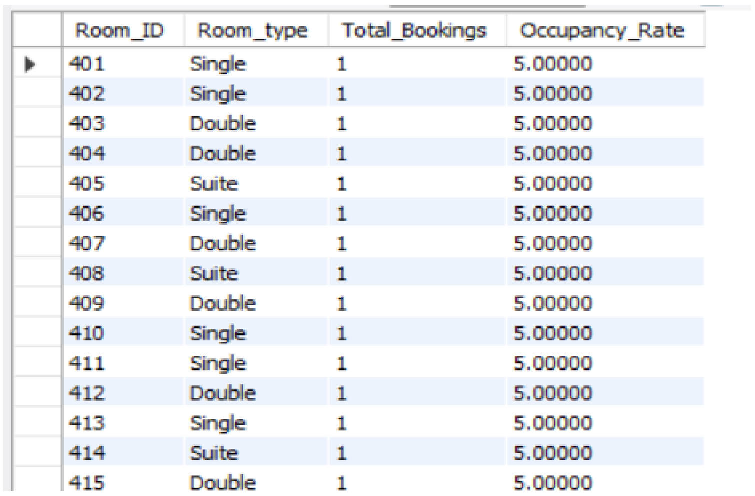
```
SELECT
    Guest_ID,
    Booking_ID,
    Booking_date,
    lead(checkindate) OVER (PARTITION BY Guest_ID ORDER BY Booking_date) AS Next_Booking_Date
FROM
    lodginghub.Bookings;
```



Guest_ID	Booking_ID	Booking_date	Next_Booking_Date
101	201	10/10/2024	12/25/2024
101	220	10/10/2024	NULL
102	202	10/10/2024	NULL
103	203	10/10/2024	01/15/2025
103	217	10/10/2024	NULL
104	204	10/10/2024	NULL
105	205	10/10/2024	01/10/2025
105	216	10/10/2024	02/10/2025
105	218	10/10/2024	NULL
106	206	10/15/2024	NULL
107	207	10/15/2024	NULL
108	208	10/15/2024	02/15/2025
108	219	10/15/2024	NULL
109	209	10/15/2024	NULL
110	210	10/15/2024	NULL
111	211	10/20/2024	NULL
112	212	10/20/2024	NULL
113	213	10/20/2024	NULL
114	214	10/20/2024	NULL

Query 5: Calculate the occupancy rate for each room as a percentage of total bookings

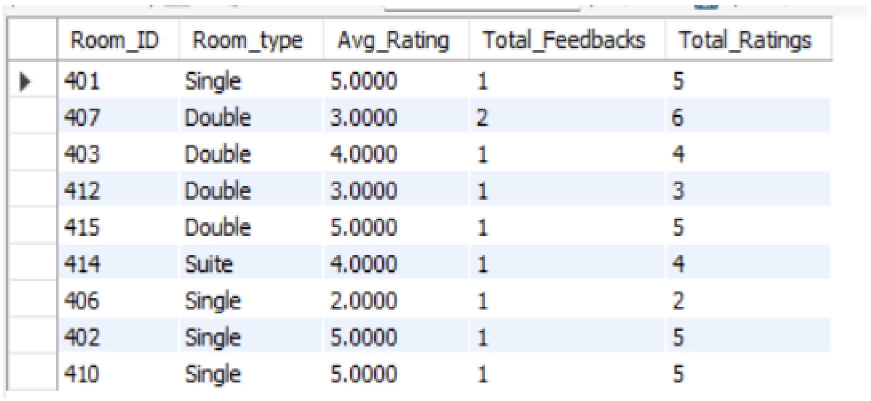
```
SELECT
    r.Room_ID,
    r.Room_type,
    COUNT(b.Booking_ID) AS Total_Bookings,
    COUNT(b.Booking_ID) * 100.0 / COUNT(*) OVER () AS Occupancy_Rate
FROM
    lodginghub.Rooms r
    LEFT JOIN lodginghub.Bookings b
        ON r.Room_ID = b.Room_ID
GROUP BY
    r.Room_ID,
    r.Room_type;
```



Room_ID	Room_type	Total_Bookings	Occupancy_Rate
401	Single	1	5.00000
402	Single	1	5.00000
403	Double	1	5.00000
404	Double	1	5.00000
405	Suite	1	5.00000
406	Single	1	5.00000
407	Double	1	5.00000
408	Suite	1	5.00000
409	Double	1	5.00000
410	Single	1	5.00000
411	Single	1	5.00000
412	Double	1	5.00000
413	Single	1	5.00000
414	Suite	1	5.00000
415	Double	1	5.00000

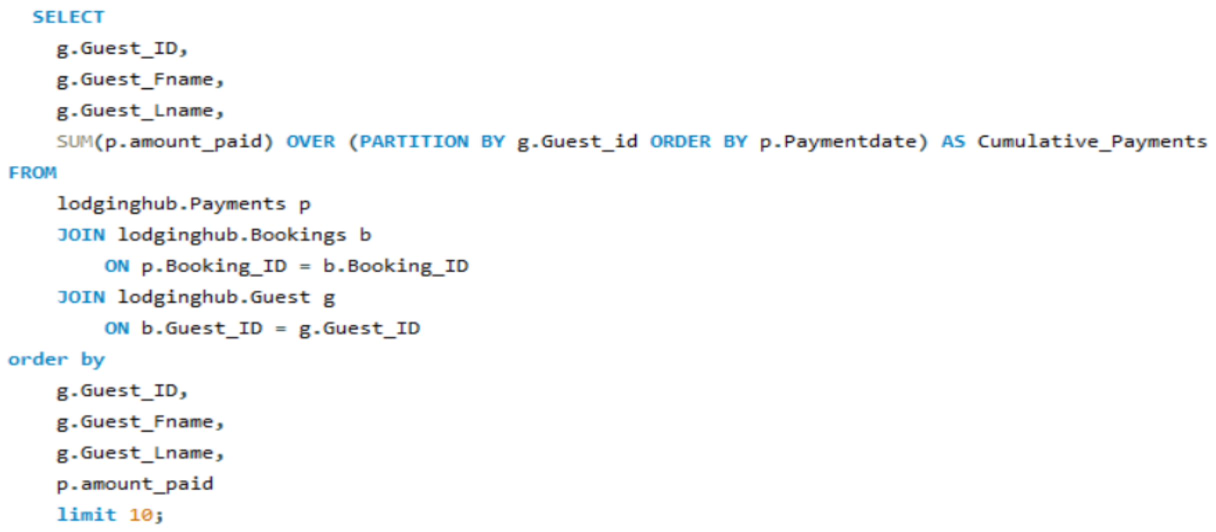
Query 6: Compute the average feedback rating, total feedbacks, and total ratings received for each room

```
SELECT
    r.Room_ID,
    r.Room_type,
    AVG(f.Rating) AS Avg_Rating,
    COUNT(f.Feedback_ID) AS Total_Feedbacks,
    SUM(f.Rating) AS Total_Ratings
FROM
    lodginghub.Rooms r
    LEFT JOIN lodginghub.Bookings b
        ON r.Room_ID = b.Room_ID
    RIGHT JOIN lodginghub.Feedback f
        ON b.Booking_ID = f.Booking_ID
GROUP BY
    r.Room_ID,
    r.Room_type;
```

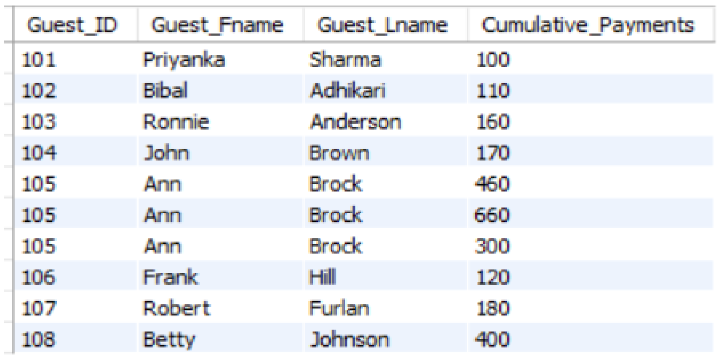


Room_ID	Room_type	Avg_Rating	Total_Feedbacks	Total_Ratings
401	Single	5.0000	1	5
407	Double	3.0000	2	6
403	Double	4.0000	1	4
412	Double	3.0000	1	3
415	Double	5.0000	1	5
414	Suite	4.0000	1	4
406	Single	2.0000	1	2
402	Single	5.0000	1	5
410	Single	5.0000	1	5

Query 7: Show cumulative payments made by each guest over time with limit 10

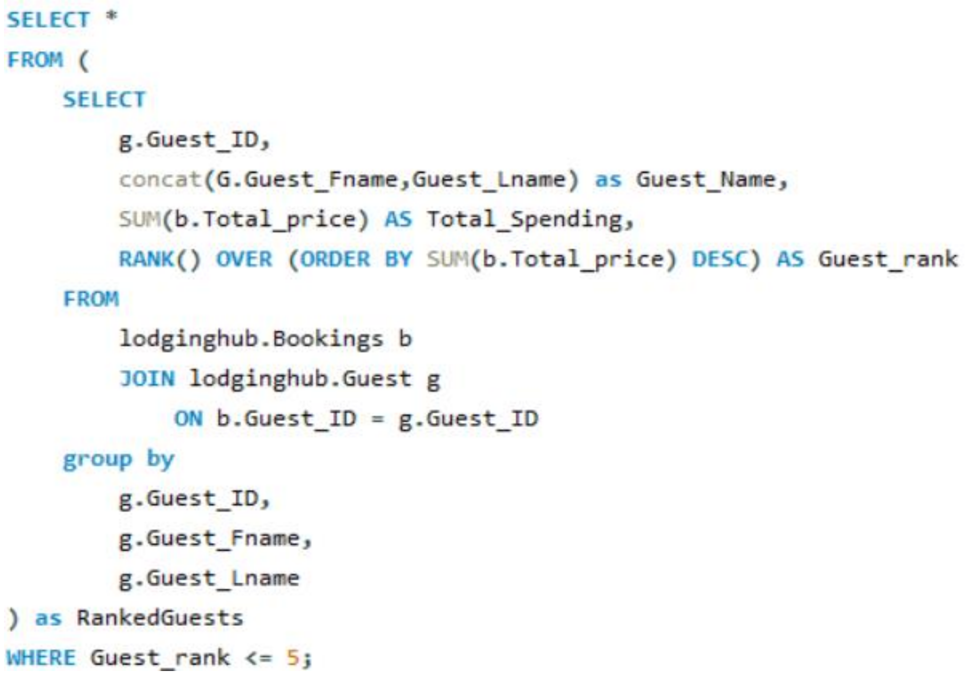


```
    SELECT
      g.Guest_ID,
      g.Guest_Fname,
      g.Guest_Lname,
      SUM(p.amount_paid) OVER (PARTITION BY g.Guest_id ORDER BY p.Paymentdate) AS Cumulative_Payments
  FROM
      lodginghub.Payments p
      JOIN lodginghub.Bookings b
          ON p.Booking_ID = b.Booking_ID
      JOIN lodginghub.Guest g
          ON b.Guest_ID = g.Guest_ID
  order by
      g.Guest_ID,
      g.Guest_Fname,
      g.Guest_Lname,
      p.amount_paid
      limit 10;
```

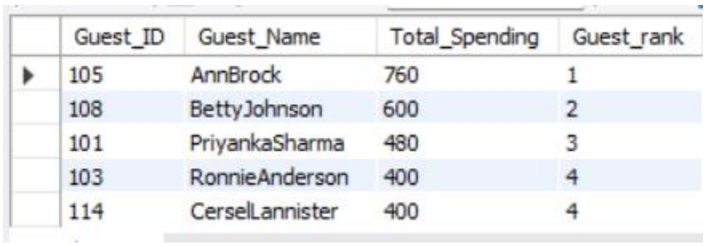


Guest_ID	Guest_Fname	Guest_Lname	Cumulative_Payments
101	Priyanka	Sharma	100
102	Bibal	Adhikari	110
103	Ronnie	Anderson	160
104	John	Brown	170
105	Ann	Brock	460
105	Ann	Brock	660
105	Ann	Brock	300
106	Frank	Hill	120
107	Robert	Furlan	180
108	Betty	Johnson	400

Query 8:Rank guests by their total spending and show the top 5



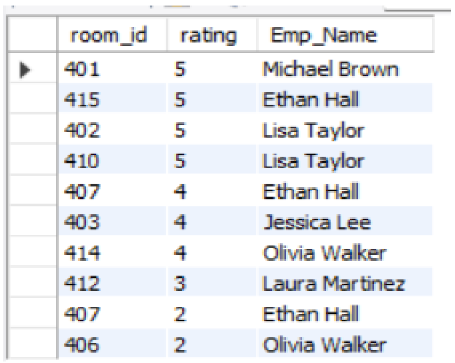
```
SELECT *
FROM (
    SELECT
        g.Guest_ID,
        concat(G.Guest_Fname,Guest_Lname) as Guest_Name,
        SUM(b.Total_price) AS Total_Spending,
        RANK() OVER (ORDER BY SUM(b.Total_price) DESC) AS Guest_rank
    FROM
        lodginghub.Bookings b
        JOIN lodginghub.Guest g
            ON b.Guest_ID = g.Guest_ID
    group by
        g.Guest_ID,
        g.Guest_Fname,
        g.Guest_Lname
) as RankedGuests
WHERE Guest_rank <= 5;
```



	Guest_ID	Guest_Name	Total_Spending	Guest_rank
▶	105	AnnBrock	760	1
	108	BettyJohnson	600	2
	101	PriyankaSharma	480	3
	103	RonnieAnderson	400	4
	114	CerselLannister	400	4

Query 9: Sort rooms based on the rating and employee who maintained it

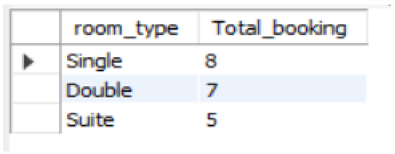
```
select
    b.room_id,
    f.rating,
    concat(e.emp_fname," ",e.emp_lname) as Emp_Name
from
    lodginghub.bookings b
    right join lodginghub.feedback f
    on b.Booking_ID=f.Booking_ID
    left join lodginghub.maintenance m
    on b.Room_ID=m.Room_ID
    left join lodginghub.employee e
    on m.Employee_ID=e.Employee_ID
order by f.rating desc;
```



	room_id	rating	Emp_Name
▶	401	5	Michael Brown
	415	5	Ethan Hall
	402	5	Lisa Taylor
	410	5	Lisa Taylor
	407	4	Ethan Hall
	403	4	Jessica Lee
	414	4	Olivia Walker
	412	3	Laura Martinez
	407	2	Ethan Hall
	406	2	Olivia Walker

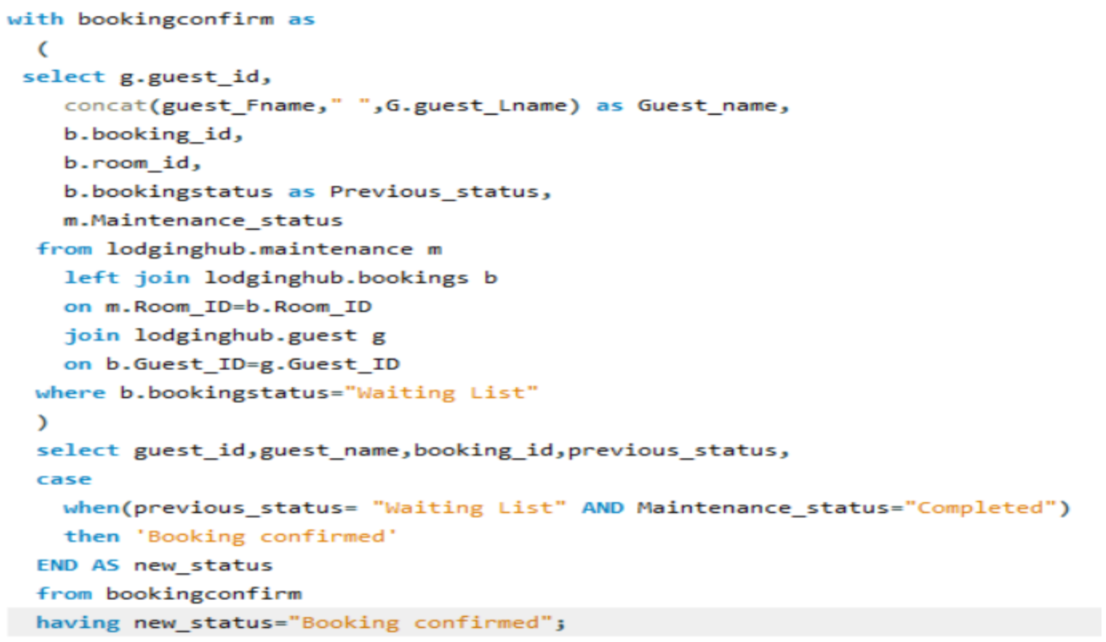
Query 10: Bookings trends of Customer based on room type

```
select
r.room_type,
count(b.booking_id) as Total_booking
from
    lodginghub.bookings b
JOIN lodginghub.rooms r
ON b.Room_ID=r.Room_ID
group by
    r.Room_Type;
```

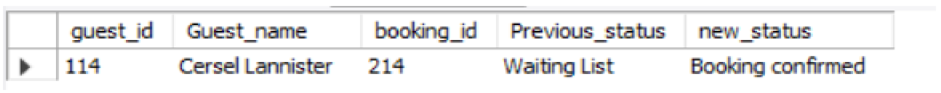


	room_type	Total_booking
▶	Single	8
	Double	7
	Suite	5

Query 11: Assigning maintenance completed room to guest whose booking is in waiting list

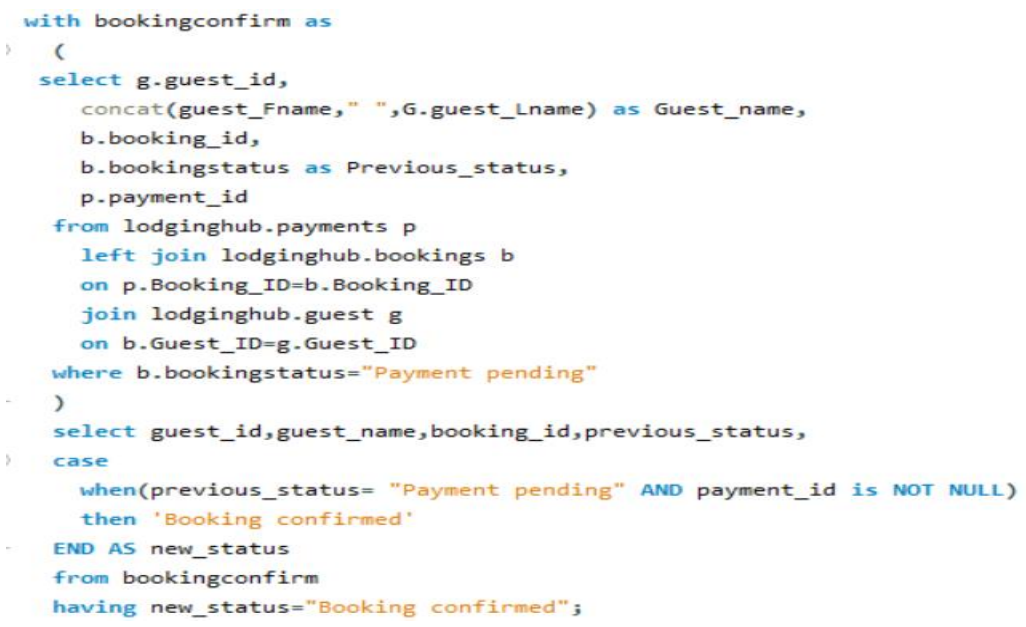


```
with bookingconfirm as
  (
 select g.guest_id,
     concat(guest_Fname," ",G.guest_Lname) as Guest_name,
     b.booking_id,
     b.room_id,
     b.bookingstatus as Previous_status,
     m.Maintenance_status
  from lodginghub.maintenance m
    left join lodginghub.bookings b
    on m.Room_ID=b.Room_ID
    join lodginghub.guest g
    on b.Guest_ID=g.Guest_ID
  where b.bookingstatus="Waiting List"
  )
  select guest_id,guest_name,booking_id,previous_status,
  case
    when(previous_status= "Waiting List" AND Maintenance_status="Completed")
    then 'Booking confirmed'
  END AS new_status
  from bookingconfirm
  having new_status="Booking confirmed";
```

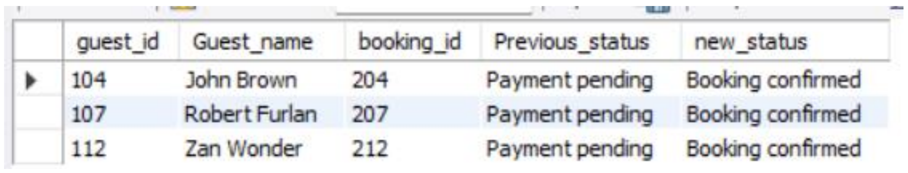


	guest_id	Guest_name	booking_id	Previous_status	new_status
▶	114	Cersel Lannister	214	Waiting List	Booking confirmed

Query 12: Assigning payment pending booking to confirmed when payments are made



```
with bookingconfirm as
  (
 select g.guest_id,
     concat(guest_Fname," ",G.guest_Lname) as Guest_name,
     b.booking_id,
     b.bookingstatus as Previous_status,
     p.payment_id
  from lodginghub.payments p
    left join lodginghub.bookings b
    on p.Booking_ID=b.Booking_ID
    join lodginghub.guest g
    on b.Guest_ID=g.Guest_ID
  where b.bookingstatus="Payment pending"
  )
  select guest_id,guest_name,booking_id,previous_status,
  case
    when(previous_status= "Payment pending" AND payment_id is NOT NULL)
    then 'Booking confirmed'
  END AS new_status
  from bookingconfirm
  having new_status="Booking confirmed";
```



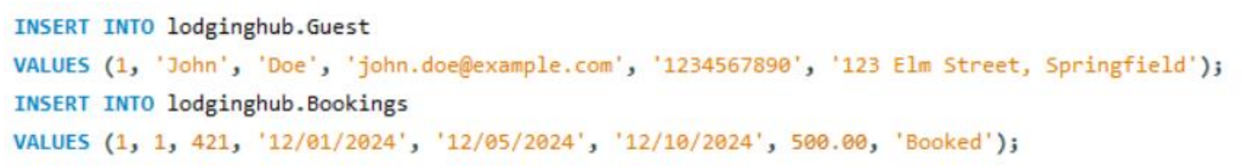
guest_id	Guest_name	booking_id	Previous_status	new_status
104	John Brown	204	Payment pending	Booking confirmed
107	Robert Furlan	207	Payment pending	Booking confirmed
112	Zan Wonder	212	Payment pending	Booking confirmed

DATA MANIPULATION LANGUAGE(DML)

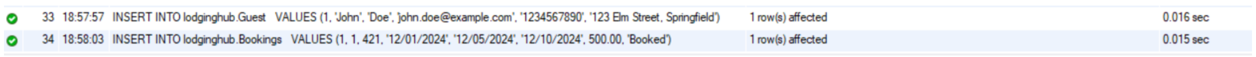
DML is used to manipulate data stored in a database. DML commands allow users to perform tasks such as inserting, updating and deleting.

Below are SQL queries performed on the schema to update the existing table;

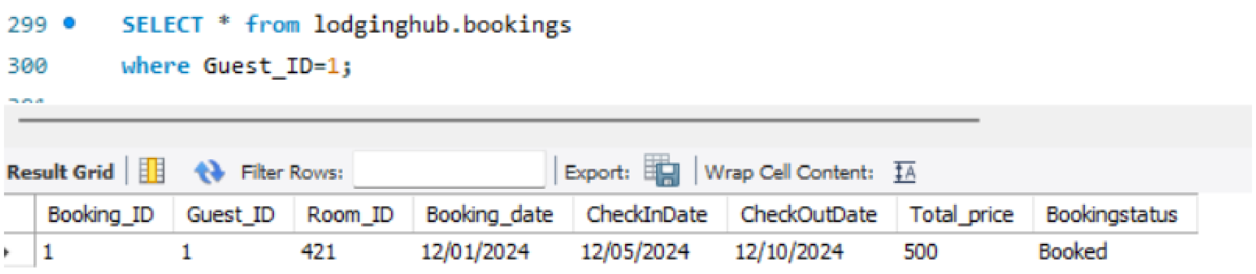
INSERT



```sql
INSERT INTO lodginghub.Guest
VALUES (1, 'John', 'Doe', 'john.doe@example.com', '1234567890', '123 Elm Street, Springfield');
INSERT INTO lodginghub.Bookings
VALUES (1, 1, 421, '12/01/2024', '12/05/2024', '12/10/2024', 500.00, 'Booked');
```



```
33  18:57:57  INSERT INTO lodginghub.Guest   VALUES (1, 'John', 'Doe', 'john.doe@example.com', '1234567890', '123 Elm Street, Springfield')   1 row(s) affected   0.016 sec
34  18:58:03  INSERT INTO lodginghub.Bookings   VALUES (1, 1, 421, '12/01/2024', '12/05/2024', '12/10/2024', 500.00, 'Booked')   1 row(s) affected   0.015 sec
```

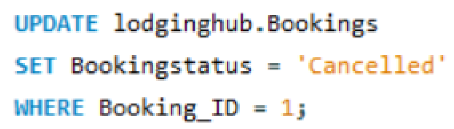


```sql
SELECT * from lodginghub.bookings
where Guest_ID=1;
```

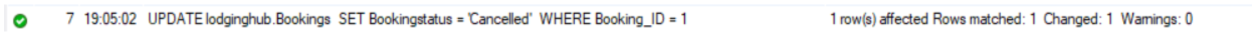
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Booking_ID	Guest_ID	Room_ID	Booking_date	CheckInDate	CheckOutDate	Total_price	Bookingstatus
1	1	421	12/01/2024	12/05/2024	12/10/2024	500	Booked

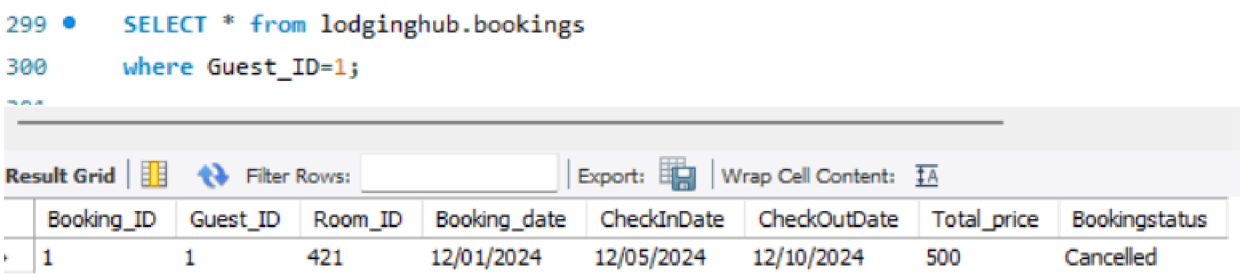
UPDATE



```sql
UPDATE lodginghub.Bookings
SET Bookingstatus = 'Cancelled'
WHERE Booking_ID = 1;
```



```
7  19:05:02  UPDATE lodginghub.Bookings  SET Bookingstatus = 'Cancelled'  WHERE Booking_ID = 1   1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0
```

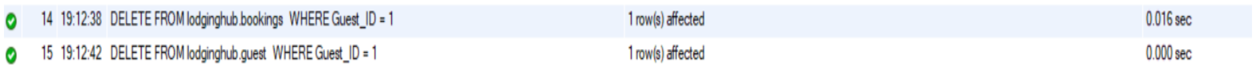


```
SELECT * from lodginghub.bookings
where Guest_ID=1;
```

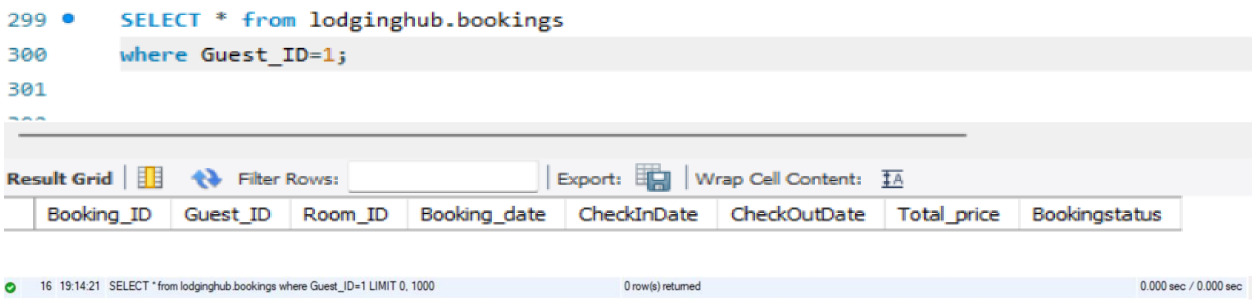
Booking_ID	Guest_ID	Room_ID	Booking_date	CheckInDate	CheckOutDate	Total_price	Bookingstatus
1	1	421	12/01/2024	12/05/2024	12/10/2024	500	Cancelled

DELETE

```
DELETE FROM lodginghub.bookings
WHERE Guest_ID = 1;
DELETE FROM lodginghub.guest
WHERE Guest_ID = 1;
```



```
14  19:12:38  DELETE FROM lodginghub.bookings  WHERE Guest_ID = 1    1 row(s) affected    0.016 sec
15  19:12:42  DELETE FROM lodginghub.guest  WHERE Guest_ID = 1       1 row(s) affected    0.000 sec
```



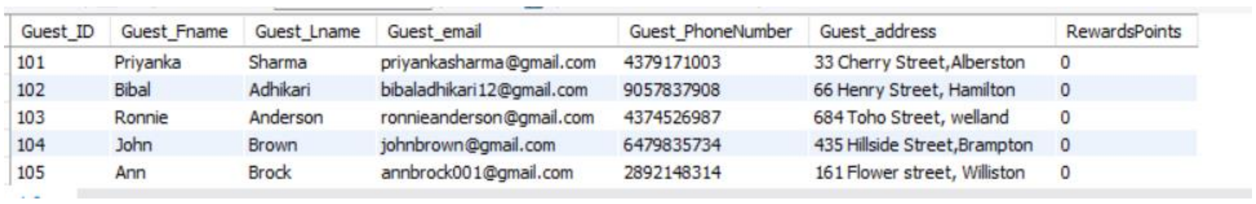
```
SELECT * from lodginghub.bookings
where Guest_ID=1;
```

Booking_ID	Guest_ID	Room_ID	Booking_date	CheckInDate	CheckOutDate	Total_price	Bookingstatus

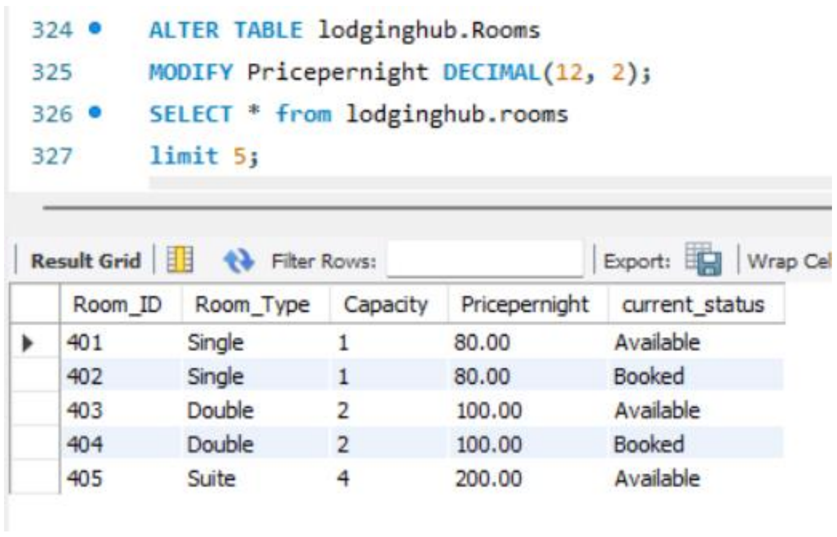
```
16  19:14:21  SELECT * from lodginghub.bookings where Guest_ID=1 LIMIT 0, 1000    0 row(s) returned    0.000 sec / 0.000 sec
```

ALTER

```
ALTER TABLE lodginghub.Guest
ADD RewardsPoints INT DEFAULT 0;
SELECT * from lodginghub.guest
limit 5;
```



Guest_ID	Guest_Fname	Guest_Lname	Guest_email	Guest_PhoneNumber	Guest_address	RewardsPoints
101	Priyanka	Sharma	priyankasharma@gmail.com	4379171003	33 Cherry Street,Alberston	0
102	Bibal	Adhikari	bibaladhikari12@gmail.com	9057837908	66 Henry Street, Hamilton	0
103	Ronnie	Anderson	ronnieanderson@gmail.com	4374526987	684 Toho Street, welland	0
104	John	Brown	johnbrown@gmail.com	6479835734	435 Hillside Street,Brampton	0
105	Ann	Brock	annbrock001@gmail.com	2892148314	161 Flower street, Williston	0



```
ALTER TABLE lodginghub.Rooms
MODIFY Pricepernight DECIMAL(12, 2);
SELECT * from lodginghub.rooms
limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cel

Room_ID	Room_Type	Capacity	Pricepernight	current_status
401	Single	1	80.00	Available
402	Single	1	80.00	Booked
403	Double	2	100.00	Available
404	Double	2	100.00	Booked
405	Suite	4	200.00	Available

CONCLUSION

The Hotel Booking Management System project represents a significant step toward streamlining hotel operations, improving customer satisfaction, and enhancing overall efficiency. The project successfully demonstrates how technology can be leveraged to simplify complex processes and deliver a seamless experience for both hotel staff and customers

Key Achievements:

1. **Data Design and Implementation**
   - ➢ Designed a robust relational database schema with normalized tables such as Guests, Rooms, Bookings, and Payments to minimize redundancy and maintain data integrity.
   - ➢ Implemented appropriate constraints, such as primary keys, foreign keys, and unique constraints, to ensure referential integrity and data consistency

2. **Data Management**
   - ➢ Utilized SQL queries to insert, update, delete, and retrieve data, showcasing the use of Data Manipulation Language (DML) effectively.

3. **Data Analytics**
   - ➢ Developed advanced queries for generating insightful reports, such as monthly revenue, occupancy rates, and guest booking trends.
   - ➢ Showcased the use of SQL aggregate functions (SUM, COUNT, AVG) and grouping techniques (GROUP BY) to extract actionable insights.

4. **Error Handling and Validation**
    - Incorporated error-checking mechanisms, such as verifying date ranges (Check_Out > Check_In) and ensuring rooms are not double-booked
    - Prevented potential data anomalies using SQL constraints and transaction isolation levels.
5. **Real Time Applications**
    - Real-time tracking of room availability and bookings.
    - Features such as booking modifications and cancellations improved customer convenience

REFERENCES

1. **W3Schools SQL Tutorial** https://www.w3schools.com/sql/
2. Learning SQL, Alan Beaulieu
3. SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL (Coding & Programming - QuickStart Guides), Paperback