HOTEL MANAGEMENT SYSTEM

BY:AMIT GUPTA

HOTEL MANAGEMENT SYSTEM

Overview:

The Hotel Management System is a comprehensive database-driven solution aimed at digitizing and automating the core functions of hotel operations. From room reservations to guest services, this system ensures smooth coordination across various departments including front desk, housekeeping, and billing.

Project Description:

This project focuses on the backend implementation of a Hotel Management System using MySQL. It involves the creation of a well-structured relational database to support essential hotel activities such as:

- Managing room availability and categories
- Handling guest information and booking history
- Recording check-ins and check-outs
- Generating invoices and payment records
- Overseeing staff roles and schedules

PROJECT AIM:

- To develop a hotel management system that automates daily hotel operations using **MySQL** as the backend database.
- To simplify and streamline the process of **room booking, check-in, and check-out** through a centralized system.
- To securely manage **customer records, reservation details, and billing information** with proper database normalization.
- To ensure accurate and real-time tracking of **room availability, room types, and service status**.
- To provide a **user-friendly interface** for hotel staff to manage operations efficiently and reduce human error.
- To enable the generation of **reports** (daily, weekly, monthly) for managerial review and financial auditing purposes.

OBJECTIVES:-

1. Set up the Hotel Management System Database:

Design and populate the database with relevant tables such as:

- **Guests** (Customer details)
- **Rooms** (Room details and types)
- **Reservations** (Booking information)
- **CheckIn / CheckOut** (Guest stay records)
- **RoomService** (Service orders linked to rooms)
- **Billing** (Invoices and payments)
- **Employees** (Hotel staff information)
- **Menu** (Restaurant menu items and pricing)

---

2. CRUD Operations:

Implement **Create, Read, Update, and Delete** operations across the system for:

- Guest registration and profile management
- Room management (add, edit, delete room types and availability)
- Booking and reservation updates
- Adding or modifying menu items and service requests
- Managing employee records
- Processing payments and updating billing details

---

3. Advanced SQL Queries:

Create and execute complex queries to:

- Generate reports on **room occupancy** by date range or room type
- Identify the **most frequently booked rooms** or highest-paying guests
- Calculate **daily/monthly revenue** from bookings and services
- Track pending **check-outs or unpaid bills**
- Summarize **room service usage per guest or per room**
- Analyze **employee performance** based on completed tasks or shifts

ER Diagram For Library Management System

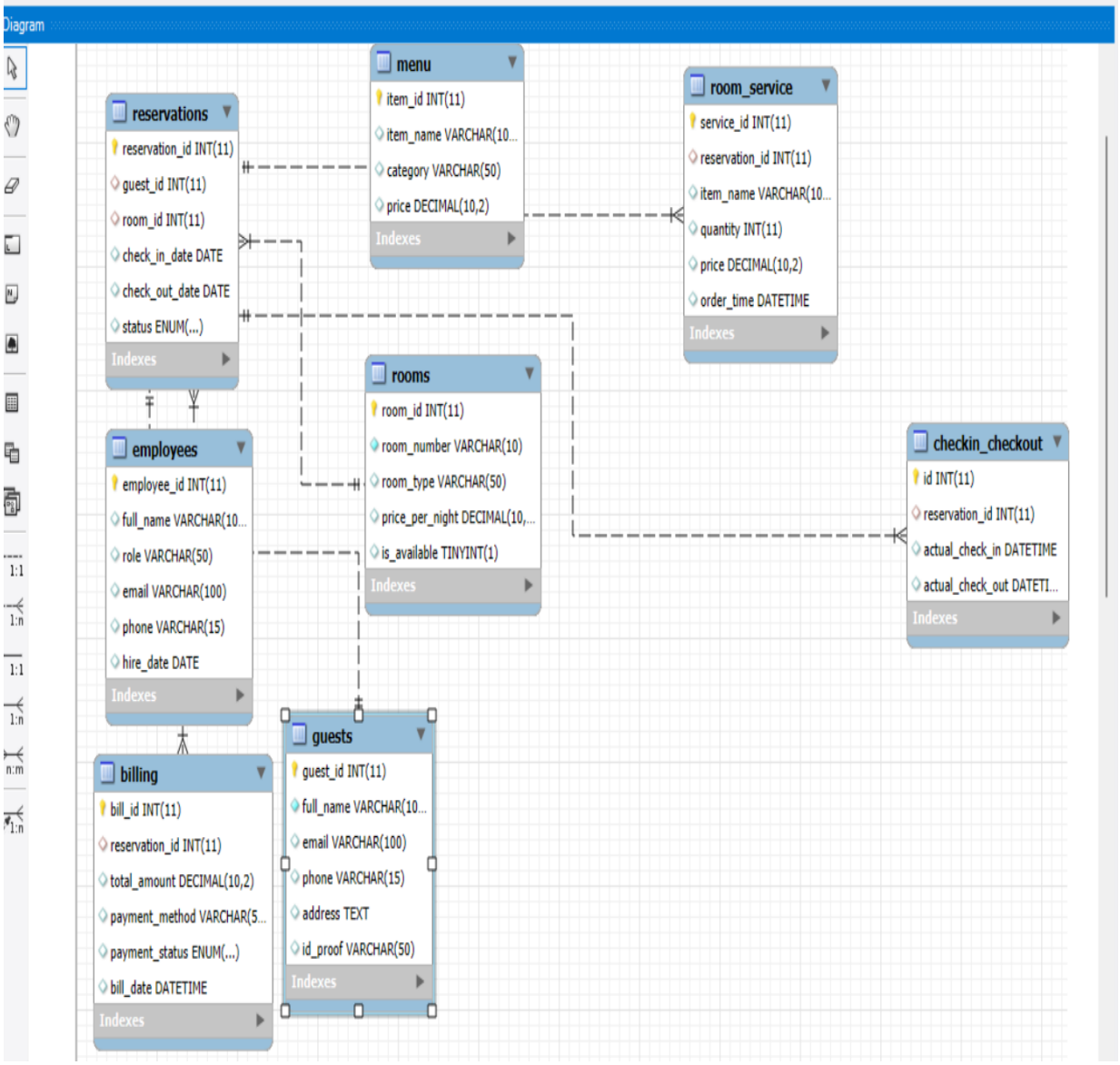

Table Description:

1.GUEST:

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

Field	Type	Null	Key	Default	Extra
guest_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(100)	NO		NULL	
email	varchar(100)	YES	UNI	NULL	
phone	varchar(15)	YES		NULL	
address	text	YES		NULL	
id_proof	varchar(50)	YES		NULL	

2.ROOMS:

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

Field	Type	Null	Key	Default	Extra
room_id	int(11)	NO	PRI	NULL	auto_increment
room_number	varchar(10)	NO	UNI	NULL	
room_type	varchar(50)	YES		NULL	
price_per_night	decimal(10,2)	YES		NULL	
is_available	tinyint(1)	YES		1	

3. reservations:

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

Field	Type	Null	Key	Default	Extra
reservation_id	int(11)	NO	PRI	NULL	auto_increment
guest_id	int(11)	YES	MUL	NULL	
room_id	int(11)	YES	MUL	NULL	
check_in_date	date	YES		NULL	
check_out_date	date	YES		NULL	
status	enum('Booked','Cancelled','Checked-In','Checke...	YES		Booked	

date

4. checkin_checkout:

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

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
reservation_id	int(11)	YES	MUL	NULL	
actual_check_in	datetime	YES		NULL	
actual_check_out	datetime	YES		NULL	

5. room_service:

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

Field	Type	Null	Key	Default	Extra
service_id	int(11)	NO	PRI	NULL	auto_increment
reservation_id	int(11)	YES	MUL	NULL	
item_name	varchar(100)	YES		NULL	
quantity	int(11)	YES		NULL	
price	decimal(10,2)	YES		NULL	
order_time	datetime	YES		current_timestamp()	

6.MENU:

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

Field	Type	Null	Key	Default	Extra
item_id	int(11)	NO	PRI	NULL	auto_increment
item_name	varchar(100)	YES		NULL	
category	varchar(50)	YES		NULL	
price	decimal(10,2)	YES		NULL	

7. BILLING:

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

Field	Type	Null	Key	Default	Extra
bill_id	int(11)	NO	PRI	NULL	auto_increment
reservation_id	int(11)	YES	MUL	NULL	
total_amount	decimal(10,2)	YES		NULL	
payment_method	varchar(50)	YES		NULL	
payment_status	enum('Paid','Unpaid')	YES		Unpaid	
bill_date	datetime	YES		current_timestamp()	

8. EMPLOYEES:

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

Field	Type	Null	Key	Default	Extra
employee_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(100)	YES		NULL	
role	varchar(50)	YES		NULL	
email	varchar(100)	YES		NULL	
phone	varchar(15)	YES		NULL	
hire_date	date	YES		NULL	

CREATING DATABASE:

CREATE DATABASE HOTEL;

USE HOTEL;

Table Creation & Insertion Commands:

1) **Create Table Guests:**

```
CREATE TABLE guests (
   guest_id INT AUTO_INCREMENT PRIMARY KEY,
   full_name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE,
   phone VARCHAR(15),
   address TEXT,
   id_proof VARCHAR(50)
);
```

inserting Values into Guests:

```
INSERT INTO guests (full_name, email, phone, address, id_proof) VALUES
('John Doe', 'john@example.com', '9876543210', 'New York', 'A12345'),
('Jane Smith', 'jane@example.com', '8765432109', 'California', 'B54321'),
('Michael Brown', 'michael@example.com', '7654321098', 'Texas', 'C12398'),
('William Wilson', 'william@example.com', '5432109876', 'Nevada', 'E09213'),
('Olivia Miller', 'olivia@example.com', '4321098765', 'Arizona', 'F87219'),
('Liam Taylor', 'liam@example.com', '3210987654', 'Oregon', 'G61278'),
('Sophia Anderson', 'sophia@example.com', '2109876543', 'Georgia', 'H43256');
```

Select * from guests;

OUTPUT:

guest_id	full_name	email	phone	address	id_proof
1	John Doe	john@example.com	9876543210	New York	A12345
2	Jane Smith	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256

2) Create Table Guests:

```
CREATE TABLE rooms (
    room_id INT AUTO_INCREMENT PRIMARY KEY,
    room_number VARCHAR(10) UNIQUE NOT NULL,
    room_type VARCHAR(50),
    price_per_night DECIMAL(10,2),
    is_available BOOLEAN DEFAULT TRUE
);
```

inserting Values into Guests:

INSERT INTO rooms (room_number, room_type, price_per_night, is_available) VALUES

('101', 'Single', 1000.00, TRUE),

('102', 'Double', 1500.00, TRUE),

('103', 'Suite', 2500.00, TRUE),

('104', 'Deluxe', 2000.00, TRUE),

('105', 'Single', 1000.00, TRUE),

('106', 'Double', 1500.00, TRUE),

('107', 'Suite', 2500.00, TRUE),

('108', 'Deluxe', 2000.00, TRUE);

Select * from rooms;

OUTPUT:

Result Grid | Filter Rows: | Edit: | Export

room_id	room_number	room_type	price_per_night	is_available
1	101	Single	1000.00	1
2	102	Double	1500.00	1
3	103	Suite	2500.00	1
4	104	Deluxe	2000.00	1
5	105	Single	1000.00	1
6	106	Double	1500.00	1
7	107	Suite	2500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

3) <u>Create Table Reservations:</u>

```
CREATE TABLE reservations (
   reservation_id INT AUTO_INCREMENT PRIMARY KEY,
   guest_id INT,
   room_id INT,
   check_in_date DATE,
   check_out_date DATE,
   status ENUM('Booked', 'Cancelled', 'Checked-In', 'Checked-Out') DEFAULT 'Booked',
   FOREIGN KEY (guest_id) REFERENCES guests(guest_id),
   FOREIGN KEY (room_id) REFERENCES rooms(room_id)
);
```

inserting Values into Guests:

```
INSERT INTO reservations (guest_id, room_id, check_in_date, check_out_date, status)
VALUES
(1, 101, '2025-07-01', '2025-07-05', 'Checked-In'),
(2, 102, '2025-07-02', '2025-07-06', 'Booked'),
(3, 103, '2025-07-01', '2025-07-03', 'Checked-Out'),
(4, 104, '2025-07-03', '2025-07-07', 'Cancelled'),
(5, 105, '2025-07-02', '2025-07-04', 'Booked'),
(6, 106, '2025-07-01', '2025-07-02', 'Checked-In'),
(7, 107, '2025-07-04', '2025-07-08', 'Booked'),
(8, 108, '2025-07-05', '2025-07-09', 'Booked');
```

Select * from reservations;

<u>OUTPUT:</u>

Result Grid | Filter Rows: | Edit: | Export/Import:

reservation_id	guest_id	room_id	check_in_date	check_out_date	status
9	1	1	2025-07-01	2025-07-05	Checked-In
10	2	2	2025-07-02	2025-07-06	Booked
11	3	3	2025-07-01	2025-07-03	Checked-Out
12	4	4	2025-07-03	2025-07-07	Cancelled
13	5	5	2025-07-02	2025-07-04	Booked
14	6	6	2025-07-01	2025-07-02	Checked-In
15	7	7	2025-07-04	2025-07-08	Booked
16	8	8	2025-07-05	2025-07-09	Booked
NULL	NULL	NULL	NULL	NULL	NULL

2025-07-03

4) **<u>Create Table checkin_checkout:</u>**

```
CREATE TABLE checkin_checkout (
   id INT AUTO_INCREMENT PRIMARY KEY,
   reservation_id INT,
   actual_check_in DATETIME,
   actual_check_out DATETIME,
   FOREIGN KEY (reservation_id) REFERENCES reservations(reservation_id)
);
```

inserting Values into checkin_checkout:

INSERT INTO checkin_checkout (reservation_id, actual_check_in, actual_check_out) VALUES

(9, '2025-07-01 14:00:00', NULL),

(11, '2025-07-01 15:00:00', '2025-07-03 11:00:00'),

(14, '2025-07-01 13:00:00', NULL),

(10, NULL, NULL),

(12, NULL, NULL),

(13, NULL, NULL),

(15, NULL, NULL),

(16, NULL, NULL);

Select * from checkin_checkout;

<u>OUTPUT:</u>

Result Grid | Filter Rows: | Edit: | Export/Imp

id	reservation_id	actual_check_in	actual_check_out
17	9	2025-07-01 14:00:00	NULL
18	11	2025-07-01 15:00:00	2025-07-03 11:00:00
19	14	2025-07-01 13:00:00	NULL
20	10	NULL	NULL
21	12	NULL	NULL
22	13	NULL	NULL
23	15	NULL	NULL
24	16	NULL	NULL
NULL	NULL	NULL	NULL

5) **Create Table Room_Services:**

```
CREATE TABLE room_service (
    service_id INT AUTO_INCREMENT PRIMARY KEY,
    reservation_id INT,
    item_name VARCHAR(100),
    quantity INT,
    price DECIMAL(10,2),
    order_time DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (reservation_id) REFERENCES reservations(reservation_id)
);
```

inserting Values into Room_Services :

```
INSERT INTO room_service (reservation_id, item_name, quantity, price) VALUES

(9, 'Tea', 2, 100),

(9, 'Sandwich', 1, 150),

(11, 'Coffee', 1, 80),

(11, 'Dinner Combo', 2, 500),

(14, 'Breakfast', 1, 250),

(14, 'Mineral Water', 3, 90),

(9, 'Fruit Platter', 1, 300),

(11, 'Cold Drink', 2, 120);
```

OUTPUT:

Result Grid | Filter Rows: | Edit: | Export/Import:

service_id	reservation_id	item_name	quantity	price	order_time
9	9	Tea	2	100.00	2025-07-02 09:58:18
10	9	Sandwich	1	150.00	2025-07-02 09:58:18
11	11	Coffee	1	80.00	2025-07-02 09:58:18
12	11	Dinner Combo	2	500.00	2025-07-02 09:58:18
13	14	Breakfast	1	250.00	2025-07-02 09:58:18
14	14	Mineral Water	3	90.00	2025-07-02 09:58:18
15	9	Fruit Platter	1	300.00	2025-07-02 09:58:18
16	11	Cold Drink	2	120.00	2025-07-02 09:58:18
NULL	NULL	NULL	NULL	NULL	NULL

6) <u>Create Table Menu:</u>

```
CREATE TABLE menu (
   item_id INT AUTO_INCREMENT PRIMARY KEY,
   item_name VARCHAR(100),
   category VARCHAR(50),
   price DECIMAL(10,2)
);
```

inserting Values into <u>Menu</u> :

inserting Values into <u>Menu</u> :

```
INSERT INTO menu (item_name, category, price) VALUES
('Tea', 'Beverage', 50),
('Coffee', 'Beverage', 80),
('Sandwich', 'Snack', 150),
('Dinner Combo', 'Meal', 250),
('Breakfast', 'Meal', 200),
('Fruit Platter', 'Snack', 300),
('Cold Drink', 'Beverage', 60),
('Mineral Water', 'Drink', 30);
```

<u>OUTPUT:</u>

Result Grid | Filter Rows: | Edit:

item_id	item_name	category	price
1	Tea	Beverage	50.00
2	Coffee	Beverage	80.00
3	Sandwich	Snack	150.00
4	Dinner Combo	Meal	250.00
5	Breakfast	Meal	200.00
6	Fruit Platter	Snack	300.00
7	Cold Drink	Beverage	60.00
8	Mineral Water	Drink	30.00
NULL	NULL	NULL	NULL

7) **Create Table Billing:**

```
CREATE TABLE billing (
    bill_id INT AUTO_INCREMENT PRIMARY KEY,
    reservation_id INT,
    total_amount DECIMAL(10,2),
    payment_method VARCHAR(50),
    payment_status ENUM('Paid', 'Unpaid') DEFAULT 'Unpaid',
    bill_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (reservation_id) REFERENCES reservations(reservation_id)
);
```

inserting Values into Billing:

INSERT INTO billing (reservation_id, total_amount, payment_method, payment_status) VALUES

(9, 3200, 'Credit Card', 'Unpaid'),

(11, 5400, 'Cash', 'Paid'),

(14, 1250, 'UPI', 'Unpaid'),

(10, 6000, 'Credit Card', 'Unpaid'),

(12, 0, 'N/A', 'Unpaid'),

(13, 3000, 'Cash', 'Unpaid'),

(15, 7200, 'UPI', 'Unpaid'),

(16, 8000, 'Debit Card', 'Unpaid');

OUTPUT:

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

bill_id	reservation_id	total_amount	payment_method	payment_status	bill_date
9	9	3200.00	Credit Card	Unpaid	2025-07-02 10:04:18
10	11	5400.00	Cash	Paid	2025-07-02 10:04:18
11	14	1250.00	UPI	Unpaid	2025-07-02 10:04:18
12	10	6000.00	Credit Card	Unpaid	2025-07-02 10:04:18
13	12	0.00	N/A	Unpaid	2025-07-02 10:04:18
14	13	3000.00	Cash	Unpaid	2025-07-02 10:04:18
15	15	7200.00	UPI	Unpaid	2025-07-02 10:04:18
16	16	8000.00	Debit Card	Unpaid	2025-07-02 10:04:18
NULL	NULL	NULL	NULL	NULL	NULL

8) **<u>Create Table employees:</u>**

```
CREATE TABLE employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(100),
    role VARCHAR(50),
    email VARCHAR(100),
    phone VARCHAR(15),
    hire_date DATE
);
```

inserting Values into <u>Billing</u>:

```
INSERT INTO employees (full_name, role, email, phone, hire_date) VALUES
('Raj Sharma', 'Manager', 'raj@hotel.com', '9990001111', '2022-01-10'),
('Priya Verma', 'Receptionist', 'priya@hotel.com', '8881112222', '2023-02-15'),
('Amit Kumar', 'Housekeeping', 'amit@hotel.com', '7772223333', '2021-03-20'),
('Anjali Rao', 'Chef', 'anjali@hotel.com', '6663334444', '2020-04-25'),
('Suresh Patel', 'Security', 'suresh@hotel.com', '5554445555', '2022-05-30'),
('Neha Gupta', 'Front Desk', 'neha@hotel.com', '4445556666', '2023-06-10'),
('Deepak Yadav', 'Waiter', 'deepak@hotel.com', '3336667777', '2021-07-12'),
('Swati Mehta', 'Cashier', 'swati@hotel.com', '2227778888', '2023-08-18');
```

<u>OUTPUT:</u>

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

employee_id	full_name	role	email	phone	hire_date
1	Raj Sharma	Manage Manager	@hotel.com	9990001111	2022-01-10
2	Priya Verma	Receptionist	priya@hotel.com	8881112222	2023-02-15
3	Amit Kumar	Housekeeping	amit@hotel.com	7772223333	2021-03-20
4	Anjali Rao	Chef	anjali@hotel.com	6663334444	2020-04-25
5	Suresh Patel	Security	suresh@hotel.com	5554445555	2022-05-30
6	Neha Gupta	Front Desk	neha@hotel.com	4445556666	2023-06-10
7	Deepak Yadav	Waiter	deepak@hotel.com	3336667777	2021-07-12
8	Swati Mehta	Cashier	swati@hotel.com	2227778888	2023-08-18
NULL	NULL	NULL	NULL	NULL	NULL

BASIC QUESTIONS

1. How do you insert a new guest into the system?

INSERT INTO guests (full_name, email, phone, address, id_proof)

VALUES ('Amit Verma', 'amit.verma@gmail.com', '9876543210', 'Delhi, India', 'AADHAR1234');

select * from guests;

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

guest_id	full_name	email	phone	address	id_proof
1	John Doe	john@example.com	9876543210	New York	A12345
2	Jane Smith	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
9	Amit Verma	amit.verma@gmail.com	9876543210	Delhi, India	AADHAR1234
NULL	NULL	NULL	NULL	NULL	NULL

Q2. How do you update the phone number of a guest?

UPDATE guests

SET phone = '9999999999'

WHERE guest_id = 1;

select * from guests;

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

guest_id	full_name	email	phone	address	id_proof
1	John Doe	john@example.com	9999999999	New York	A12345
2	Jane Smith	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
9	Amit Verma	amit.verma@gmail.com	9876543210	Delhi, India	AADHAR1234
NULL	NULL	NULL	NULL	NULL	NULL

Q3. How do you add a new column to store nationality of guests?

ALTER TABLE guests

ADD nationality VARCHAR(50);

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

guest_id	full_name	email	phone	address	id_proof	nationality
1	John Doe	john@example.com	9999999999	New York	A12345	NULL
2	Jane Smith	jane@example.com	8765432109	California	B54321	NULL
3	Michael Brown	michael@example.com	7654321098	Texas	C12398	NULL
4	Emily Davis	emily@example.com	6543210987	Florida	D65874	NULL
5	William Wilson	william@example.com	5432109876	Nevada	E09213	NULL
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219	NULL
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278	NULL
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256	NULL
9	Amit Verma	amit.verma@gmail.com	9876543210	Delhi, India	AADHAR1234	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Q4. How do you delete a column nationality of guests ?

alter table guests

drop column nationality;

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

guest_id	full_name	email	phone	address	id_proof
1	John Doe	john@example.com	9999999999	New York	A12345
2	Jane Smith	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432 5432109876	ada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
9	Amit Verma	amit.verma@gmail.com	9876543210	Delhi, India	AADHAR1234
NULL	NULL	NULL	NULL	NULL	NULL

Q5. Update room availability to false:

UPDATE rooms SET is_available = FALSE

WHERE room_id = 1;

Result Grid | Filter Rows: | Edit: | Export/

room_id	room_number	room_type	price_per_night	is_available
1	101	Single	1000.00	0
2	102	Double	1500.00	1
3	103	Suite	2500.00	1
4	104	Deluxe	2000.00	1
5	105	Single	1000.00	1
6	106	Double	1500.00	1
7	107	Suite	2500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

Q6. Promote employee to Manager:

```
UPDATE employees
SET role = 'Manager'
WHERE employee_id = 1;
select * from employees;
```

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

employee_id	full_name	role	email	phone	hire_date
1	Raj Sharma	Manager	raj@hotel.com	9990001111	2022-01-10
2	Priya Verma	Receptionist	priya@hotel.com	8881112222	2023-02-15
3	Amit Kumar	Housekeeping	amit@hotel.com	7772223333	2021-03-20
4	Anjali Rao	Chef	anjali@hotel.com	6663334444	2020-04-25
5	Suresh Patel	Security	suresh@hotel.com	5554445555	2022-05-30
6	Neha Gupta	Front Desk	neha@hotel.com	4445556666	2023-06-10
7	Deepak Yadav	Waiter	deepak@hotel.com	3336667777	2021-07-12
8	Swati Mehta	Cashier	swati@hotel.com	2227778888	2023-08-18
NULL	NULL	NULL	NULL	NULL	NULL

Q7. How can we see all tables of specific Database.

```
show tables ;
```

Result Grid | Filter Rows: | E

Tables_in_hotel
billing
checkin_checkout
employees
guests
menu
reservations
room_service
rooms

Q7. How can we delete all data from table but table should be exists.

```
Truncate table wifi;
```

Result Grid | Filter Rows: | Edit: | E

id	ssid	password	signal_strength	is_secure
NULL	NULL	NULL	NULL	NULL

Q8. How to see description about tables of Database.

Desc employees;

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

Field	Type	Null	Key	Default	Extra
employee_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(100)	YES		NULL	
role	varchar(50)	YES		NULL	
email	varchar(100)	YES		NULL	
phone	varchar(15)	YES		NULL	
hire_date	date	YES		NULL	

Q8. How can we delete a particular table from Database.

Drop table wifi;

Show tables;

Result Grid | Filter Rows:

Tables_in_hotel
billing
checkin_checkout
employees
guests
menu
reservations
room_service
rooms

Q9. How can we delete a particular row from Table

Delete from guests where guest_id=9;

Select * from guests

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

guest_id	full_name	email	phone	address	id_proof
1	John Doe	john@example.com	9999999999	New York	A12345
2	Jane John Doe	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
NULL	NULL	NULL	NULL	NULL	NULL

Q10. List all guests ordered by their full name alphabetically.

```
SELECT * FROM guests
ORDER BY full_name ASC;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
2	Jane Smith	jane@example.com	8765432109	California	B54321
1	John Doe	john@example.com	9999999999	New York	A12345
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
5	William Wilson	william@example.com	5432109876	Nevada	E09213
NULL	NULL	NULL	NULL	NULL	NULL

Q11. Show all rooms sorted by price from highest to lowest.

```
SELECT * FROM rooms
ORDER BY price_per_night DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Impo

room_id	room_number	room_type	price_per_night	is_available
3	103	Suite	2500.00	1
7	107	Suite	2500.00	1
4	104	Deluxe	2000.00	1
8	108	Deluxe	2000.00	1
2	102	Double	1500.00	1
6	106	Double	1500.00	1
1	101	Single	1000.00	0
5	105	Single	1000.00	1
NULL	NULL	NULL	NULL	NULL

Q12. Display distinct room types available in the hotel.

```
SELECT DISTINCT room_type FROM employees;
```

Result Grid | Filter Rows:

role
Manager
Receptionist
Housekeeping
Chef
Security
Front Desk
Waiter
Cashier

Q13. Display distinct room types available in the hotel.

SELECT DISTINCT room_type FROM rooms;

Result Grid

room_type
Single
Double
Suite
Deluxe

BASIC QUESTIONS ON AND ,OR,NOT:

Q1. List all available rooms of type 'Deluxe' that cost less than 4000.

```
SELECT * FROM rooms
WHERE room_type = 'Deluxe' AND price_per_night < 4000;
```

Result Grid | Filter Rows: | Edit: | Exp

room_id	room_number	room_type	price_per_night	is_available
4	104	Deluxe	2000.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

Q2. Find all employees who are either 'Manager' or 'Receptionist'.

```
SELECT * FROM employees
WHERE role = 'Manager' OR role = 'Receptionist';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	full_name	role	email	phone	hire_date
1	Raj Sharma	Manager	raj@hotel.com	9990001111	2022-01-10
2	Priya Verma	Receptionist	priya@hotel.com	8881112222	2023-02-15
NULL	NULL	NULL	NULL	NULL	NULL

Q3. Show guests who are not from 'Delhi'.

```
SELECT * FROM guests
WHERE NOT address = 'Delhi';
```

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

guest_id	full_name	email	phone	address	id_proof
2	Jane Smith	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
NULL	NULL	NULL	NULL	NULL	NULL

Aggregate Functions (COUNT, SUM, AVG, MAX, MIN):

Q1. Count how many guests are registered.

SELECT COUNT(*) AS total_guests FROM guests;

Result Grid

total_guests
8

Q2. Find the average room price.

SELECT AVG(price_per_night) AS average_price FROM rooms;

Result Grid

average_price
1750.000000

Q3. Get the maximum total amount from all billing records.

SELECT MAX(total_amount) AS max_bill FROM billing;

Result Grid

max_bill
8000.00

Q4. Find the total quantity of items ordered from room service.

SELECT SUM(quantity) AS total_items FROM room_service;

Result Grid

total_items
13

BASIC QUESTIONS IN, BETEWEEN, GROUP BY, HAVING:

Q1. Find guests whose ID is in the list (1, 2, 5, 7).

```
SELECT * FROM guests
WHERE guest_id IN (1, 2, 5, 7);
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
1	John Doe	john@example.com	9999999999	New York	A12345
2	Jane Smith	jane@example.com	8765432109	California	B54321
5	William Wilson	william@example.com	5432109876	Nevada	E09213
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
NULL	NULL	NULL	NULL	NULL	NULL

Q2. Find rooms of type either 'Deluxe', 'Suite', or 'Standard'.

```
SELECT * FROM rooms
WHERE room_type IN ('Deluxe', 'Suite', 'Standard');
```

room_id	room_number	room_type	price_per_night	is_available
3	103	Suite	2500.00	1
4	104	Deluxe	2000.00	1
7	107	Suite	2500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

Q3. Show rooms priced between 2000 and 4000.

```
SELECT * FROM rooms
WHERE price_per_night BETWEEN 2000 AND 4000;
```

Result Grid | Filter Rows: | Edit:

room_id	room_number	room_type	price_per_night	is_available
3	103	Suite	2500.00	1
4	104	Deluxe	2000.00	1
7	107	Suite	2500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

Q4. List guests with IDs between 3 and 8.

```
SELECT * FROM guests
WHERE guest_id BETWEEN 3 AND 8;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	william@example.com		F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
NULL	NULL	NULL	NULL	NULL	NULL

Q5. Group room services by reservation and show total price per reservation.

```
SELECT reservation_id, SUM(price) AS total_price
FROM room_service
GROUP BY reservation_id;
```

Result Grid | Filter Rows:

reservation_id	total_price
9	550.00
11	700.00
14	340.00

Q6. Group employees by role and count how many employees per role.

```
SELECT role, COUNT(*) AS num_employees
FROM employees
GROUP BY role;
```

Result Grid | Filter Rows:

role	num_employees
Cashier	1
Chef	1
Front Desk	1
Housekeeping	1
Manager	1
Receptionist	1
Security	1
Waiter	1

Q7. Display room service records per reservation where total price exceeds 500.

```
SELECT reservation_id, SUM(price) AS total_service_cost
FROM room_service
GROUP BY reservation_id
HAVING total_service_cost > 500;
```

Result Grid | Filter Rows:

reservation_id	total_service_cost
9	550.00
11	700.00

Window Function:

Q1. Assign a row number to each guest based on their registration order (by guest_id).

```
SELECT guest_id, full_name, ROW_NUMBER()
OVER (ORDER BY guest_id) AS row_num
FROM guests;
```

guest_id	full_name	row_num
1	John Doe	1
2	Jane Smith	2
3	Michael Brown	3
4	Emily Davis	4
5	William Wilson	5
6	Olivia Miller	6
7	Liam Taylor	7
8	Sophia Anderson	8

Q2. Rank rooms based on price (highest to lowest).

```
SELECT room_id, room_number, price_per_night,
RANK() OVER (ORDER BY price_per_night DESC) AS price_rank
FROM rooms;
```

Result Grid | Filter Rows: | Export:

room_id	room_number	price_per_night	price_rank
3	103	2500.00	1
7	107	2500.00	1
4	104	2000.00	3
8	108	2000.00	3
2	102	1500.00	5
6	106	1500.00	5
1	101	1000.00	7
5	105	1000.00	7

Q3. Rank rooms based on price (highest to lowest) using dense_rank.

```
SELECT room_id, room_number, price_per_night,
DENSE_ RANK() OVER (ORDER BY price_per_night DESC) AS price_rank
FROM rooms;
```

Result Grid | Filter Rows: | Export:

room_id	room_number	price_per_night	price_rank
3	103	2500.00	1
7	107	2500.00	1
4	104	2000.00	2
8	108	2000.00	2
2	102	1500.00	3
6	106	1500.00	3
1	101	1000.00	4
5	105	1000.00	4

Q5. Show current and previous room price (based on room_id).

```
SELECT room_id, room_number, price_per_night,
LAG(price_per_night) OVER (ORDER BY room_id) AS previous_price
FROM rooms;
```

Result Grid | Filter Rows: | Export:

room_id	room_number	price_per_night	previous_price
1	101	1000.00	NULL
2	102	1500.00	1000.00
3	103	2500.00	1500.00
4	104	2000.00	2500.00
5	105	1000.00	2000.00
6	106	1500.00	1000.00
7	107	2500.00	1500.00
8	108	2000.00	2500.00

Q6. Show each menu item with the price of the next item in the list.

```
SELECT item_id, item_name, price,
LEAD(price) OVER (ORDER BY item_id) AS next_price
FROM menu;
```

item_id	item_name	price	next_price
1	Tea	50.00	80.00
2	Coffee	80.00	150.00
3	Sandwich	150.00	250.00
4	Dinner Combo	250.00	200.00
5	Breakfast	200.00	300.00
6	Fruit Platter	300.00	60.00
7	Cold Drink	60.00	30.00
8	Mineral Water	30.00	NULL

Q7. Show each billing record along with the running total of all bills.

```
SELECT bill_id, reservation_id, total_amount,
SUM(total_amount) OVER (ORDER BY bill_date) AS running_total
FROM billing;
```

bill_id	reservation_id	total_amount	running_total
14	13	3000.00	34050.00
9	9	3200.00	34050.00
15	15	7200.00	34050.00
10	11	5400.00	34050.00
16	16	8000.00	34050.00
11	14	1250.00	34050.00
12	10	6000.00	34050.00
13	12	0.00	34050.00

WILDCARDS IN SQL:

Q1. Find guests whose names start with 'A'.

```
SELECT * FROM guests
WHERE full_name LIKE 'A%';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
10	AMIT GUPTA	AMIT@example.com	9876543210	New York	A12345
NULL	NULL	NULL	NULL	NULL	NULL

Q2. List employees whose names end with 'a'.

```
SELECT * FROM employees
WHERE full_name LIKE '%a';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

employee_id	full_name	role	email	phone	hire_date
1	Raj Sharma	Manager	raj@hotel.com	9990001111	2022-01-10
2	Priya Verma	Receptionist	priya@hotel.com	8881112222	2023-02-15
6	Neha Gupta	Front Desk	neha@hotel.com	4445556666	2023-06-10
8	Swati Mehta	Cashier	swati@hotel.com	2227778888	2023-08-18
9	AMIT GUPTA	Manager	raj@hotel.com	9990001111	2022-01-10
NULL	NULL	NULL	NULL	NULL	NULL

Q3. Find guests with 'am' anywhere in their names.

```
SELECT * FROM guests
WHERE full_name LIKE '%am%';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
5	William Wilson	william@example.com	5432109876	Nevada	E09213
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
10	AMIT GUPTA	AMIT@example.com	9876543210	New York	A12345
NULL	NULL	NULL	NULL	NULL	NULL

Q4. Find rooms with a room number that contains a '0'.

```
SELECT * FROM rooms
WHERE room_number LIKE '%0%';
```

room_id	room_number	room_type	price_per_night	is_available
1	101	Single	1000.00	0
2	102	Double	1500.00	1
3	103	Suite	2500.00	1
4	104	Deluxe	2000.00	1
5	105	Single	1000.00	1
6	106	Double	1500.00	1
7	107	Suite	2500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

Q5.Find menu items that start with 'T'and are exactly 3 letters long.

```
SELECT * FROM menu
WHERE item_name LIKE 'T__';
```

item_id	item_name	category	price
1	Tea	Beverage	50.00
NULL	NULL	NULL	NULL

Q6.Find guests whose names do *not* start with 'J'.

```
SELECT * FROM guests
WHERE full_name NOT LIKE 'R%';
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	olivia@example.com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
10	AMIT GUPTA	AMIT@example.com	9876543210	New York	A12345
NULL	NULL	NULL	NULL	NULL	NULL

SUB-QUERIES:

Q1. Show guests whose guest_id is not equal 1.

```
SELECT * FROM guests
WHERE guest_id NOT IN (
SELECT guest_id FROM reservations where guest_id=1
);
```

Result Grid | Filter Rows: | Edit: | Export/Import:

guest_id	full_name	email	phone	address	id_proof
2	Jane Smith	jane@example.com	8765432109	California	B54321
3	Michael Brown	michael@example.com	7654321098	Texas	C12398
4	Emily Davis	emily@example.com	6543210987	Florida	D65874
5	William Wilson	william@example.com	5432109876	Nevada	E09213
6	Olivia Miller	Olivia Miller .com	4321098765	Arizona	F87219
7	Liam Taylor	liam@example.com	3210987654	Oregon	G61278
8	Sophia Anderson	sophia@example.com	2109876543	Georgia	H43256
NULL	NULL	NULL	NULL	NULL	NULL

Q2. Get menu items costlier than the average price.

```
SELECT * FROM menu
WHERE price > (
   SELECT AVG(price) FROM menu
);
```

Result Grid | Filter Rows: | Edit:

item_id	item_name	category	price
3	Sandwich	Snack	150.00
4	Dinner Combo	Meal	250.00
5	Breakfast	Meal	200.00
6	Fruit Platter	Snack	300.00
NULL	NULL	NULL	NULL

Q3. List rooms cheaper than all rooms with price above ₹2000.

```
SELECT * FROM rooms
WHERE price_per_night < ALL (
  SELECT price_per_night FROM rooms
  WHERE price_per_night > 2000
);
```

Result Grid | Filter Rows: | Edit: | Export/

room_id	room_number	room_type	price_per_night	is_available
1	101	Single	1000.00	0
2	102	Double	1500.00	1
4	104	Deluxe	2000.00	1
5	105	Single	1000.00	1
6	106	Double	1500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

Q4.Get rooms priced higher than any menu item.

```
SELECT * FROM rooms
WHERE price_per_night > ANY (
  SELECT price FROM menu
);
```

Result Grid | Filter Rows: | Edit: | Expo

room_id	room_number	room_type	price_per_night	is_available
1	101	Single	1000.00	0
2	102	Double	1500.00	1
3	103	Suite	2500.00	1
4	104	Deluxe	2000.00	1
5	105	Single	1000.00	1
6	106	Double	1500.00	1
7	107	Suite	2500.00	1
8	108	Deluxe	2000.00	1
NULL	NULL	NULL	NULL	NULL

JOINS:

Q1: Show guest id, guest names along with the reservation status for guests who have made reservations using inner join.

```
SELECT g.guest_id, g.full_name, r.status

FROM guests g

INNER JOIN reservations r ON g.guest_id = r.guest_id;
```

guest_id	full_name	status
1	John Doe	Checked-In
2	Jane Smith	Booked
3	Michael Brown	Checked-Out
4	Emily Davis	Cancelled
5	William Wilson	Booked
6	Olivia Miller	Checked-In
7	Liam Taylor	Booked
8	Sophia Anderson	Booked

Q2: List all guests, along with their reservation status using right joins.

```
SELECT g.guest_id, g.full_name, r.status

FROM guests g

left JOIN reservations r ON g.guest_id = r.guest_id;
```

Result Grid | Filter Rows: | Exp

guest_id	full_name	status
1	John Doe	Checked-In
2	Jane Smith	Booked
3	Michael Brown	Checked-Out
4	Emily Davis	Cancelled
5	William Wilson	Booked
6	Olivia Miller	Checked-In
7	Liam Taylor	Booked
8	Sophia Anderson	Booked
10	AMIT GUPTA	NULL

Q3.List all room service orders along with item names and prices from the menu using right join.

```
SELECT m.item_name, m.price, rs.quantity, rs.order_time

FROM menu m

RIGHT JOIN room_service rs ON m.item_name = rs.item_name;
```

item_name	price	quantity	order_time
Tea	50.00	2	2025-07-02 09:58:18
Coffee	80.00	1	2025-07-02 09:58:18
Sandwich	150.00	1	2025-07-02 09:58:18
Dinner Combo	250.00	2	2025-07-02 09:58:18
Breakfast	200.00	1	2025-07-02 09:58:18
Fruit Platter	300.00	1	2025-07-02 09:58:18
Cold Drink	60.00	2	2025-07-02 09:58:18
Mineral Water	30.00	3	2025-07-02 09:58:18

Q4. List all names from both guests and employees (without duplicates)."

```
SELECT full_name FROM guests

UNION

SELECT full_name FROM employees;
```

full_name
John Doe
Jane Smith
Michael Brown
Emily Davis
William Wilson
Olivia Miller
Liam Taylor
Sophia Anderson
AMIT GUPTA
Raj Sharma
Priya Verma
Amit Kumar
Anjali Rao

Q5."List all names from both guests and employees (with duplicates)."

```
SELECT full_name FROM guests

UNION ALL

SELECT full_name FROM employees order by full_name;
```

Result Grid | Filter

full_name
AMIT GUPTA
AMIT GUPTA
Amit Kumar
Anjali Rao
Deepak Yadav
Emily Davis
Jane Smith
John Doe
Liam Taylor
Michael Brown
Neha Gupta
Olivia Miller
Priya Verma
Raj Sharma

Result 64

CONCLUSION:

The Hotel Management System Database project has successfully designed and implemented an efficient and scalable database solution for managing hotel operations. This system effectively handles key aspects such as guest management, reservations, check-ins and check-outs, room services, billing, and employee records.

The project has met its objectives by ensuring improved data accuracy, streamlined processes, and greater accessibility to critical hotel data. With a user-friendly design and strong backend support, the system enhances operational efficiency and supports better decision-making.

Through this project, essential SQL skills were demonstrated — including database design, table creation, data manipulation, and the execution of advanced queries. This foundation enables future expansion and integration of more hotel services, laying the groundwork for a complete, data-driven hotel management solution.