



CASE STUDY

Hotel Management System

Under the guidance of
Ms.V.Asha(Assi.Professor)

TEAM : 2

22WH1A6602 - V. Harshitha

22WH1A6610 - L. Sharanya

22WH1A6626 - R. Ashritha

22WH1A6631 - B. Rishita

22WH1A6632 - B. Padma sri

22WH1A6654 - R. Geetika Sri

22WH1A6655 - B. Anusha

22WH1A6659 - B. Anitha

22WH1A6662 - B. Sri Vaishnavi

23WH5A6602 - P. Kavya

INDEX

S.NO.	TOPIC	PAGE NO.
1	Introduction	3
2	Requirement Analysis	4
3	ER-Diagram	7
4	Relational model	10
5	Table creation	13
6	Testing and results	17

INTRODUCTION

A Hotel Management System (HMS) in Database Management System is a comprehensive application designed to manage various aspects of hotel operations. Databases in Hotel Management System is used to store all the hotel information it provides security to the data.

Entity Relation Diagrams (ER) are used to represent the Idea of the HMS . It contains Entities in HMS room, reservation, customer, payment , Hotel, service are entities represents as rectangular shape. Attributes represents ellipse shape in HMS attributes are the subparts of the Entity Ex: Room contains Room no, type, price, status are attributes. Relations in ER diagram used to connect one entity with other entity with rhombus symbol.

In Reservation Management to provides the real -time Availability it Track and manage room availability, ensuring up-to-date information for online and in-person bookings. It stores detailed information about each reservation, including customer details, room type, duration of stay, and special requests.

As well as in Customer Relationship management Maintains detailed records of guests, including personal preferences, previous stays, and feedback, to provide personalized services. Likewise there are some tables created in HMS they are payment, Service, Room etc.

This tables are created with different attributes, Tuples, Entities. Ex: Room management contains room no, type, price, status these are attributes of Room. To find the relations between the tables we use primary keys, Foreign Keys, Candidate Keys and Relationships they are one to one, one to many, many to one, many to many.

We can retrieve the data by queries . We have return some queries to work on the tables.

REQUIREMENT ANALYSIS

FUNTIONAL REQUIREMENTS

User Management

- **Registration:** Users can create accounts with personal details.
- **Login:** Users can log in using email/username and password.
- **Profile Management:** Users can update personal details, view booking history, and manage preferences.

Room Search and Booking

- **Room Availability:** Users can search for available rooms based on dates, room type, and number of occupants.
- **Room Details:** Users can view detailed descriptions, photos, and amenities of available rooms.
- **Pricing:** Display room rates, including taxes, fees, and any applicable discounts.
- **Booking:** Users can book rooms by selecting dates, room type, and entering guest information.
- **Payment:** Integration with payment gateways for secure online payments.

Service Booking

- **Service Availability:** Users can browse and select additional services (e.g., room service, spa, etc.).
- **Service Details:** Users can view descriptions, prices, and availability of services.
- **Booking Services:** Users can add services to their room reservation or book services independently.
- **Payment:** Integration with payment gateways for service payments.

Booking Management

- **View Bookings:** Users can view current, past, and upcoming bookings.
- **Modify/Cancel Bookings:** Users can modify or cancel bookings based on hotel policies.
- **Confirmation:** Users receive email/SMS confirmation for successful bookings, modifications, and cancellations.

Admin Management

- **Dashboard:** Admins can view system statistics, such as occupancy rates, booking trends, and revenue.

- **Room Management:** Admins can add, update, or remove room details and availability.
- **Service Management:** Admins can add, update, or remove services.
- **Booking Management:** Admins can view, modify, or cancel any bookings.
- **User Management:** Admins can manage user accounts, including deactivation or deletion.

Notifications and Alerts

- **Booking Confirmations:** Users receive email/SMS notifications for bookings.
- **Reminders:** Users receive reminders for upcoming bookings and services.
- **Promotions:** Users receive notifications about special offers and promotions.

Review and Rating

- **Review Submission:** Users can submit reviews and ratings for their stay and services.
- **Review Management:** Admins can moderate reviews (approve, reject, or respond).

NON-FUNCTIONAL REQUIREMENTS

Performance

- **Response Time:** The system should respond to user queries within 2 seconds.
- **Scalability:** The system should handle increased load during peak times without performance degradation.

Security

- **Data Protection:** Secure storage and transmission of user data, including personal and payment information.
- **Authentication:** Implement strong authentication mechanisms to prevent unauthorized access.
- **Authorization:** Ensure appropriate access control levels for users and admins.

Usability

- **User Interface:** Intuitive and user-friendly interface for both web and mobile platforms.
- **Accessibility:** The system should be accessible to users with disabilities, complying with relevant standards (e.g., WCAG).

Reliability

- **Availability:** The system should be available 99.9% of the time, with minimal downtime.
- **Backup and Recovery:** Regular data backups and a robust disaster recovery plan.

Maintainability

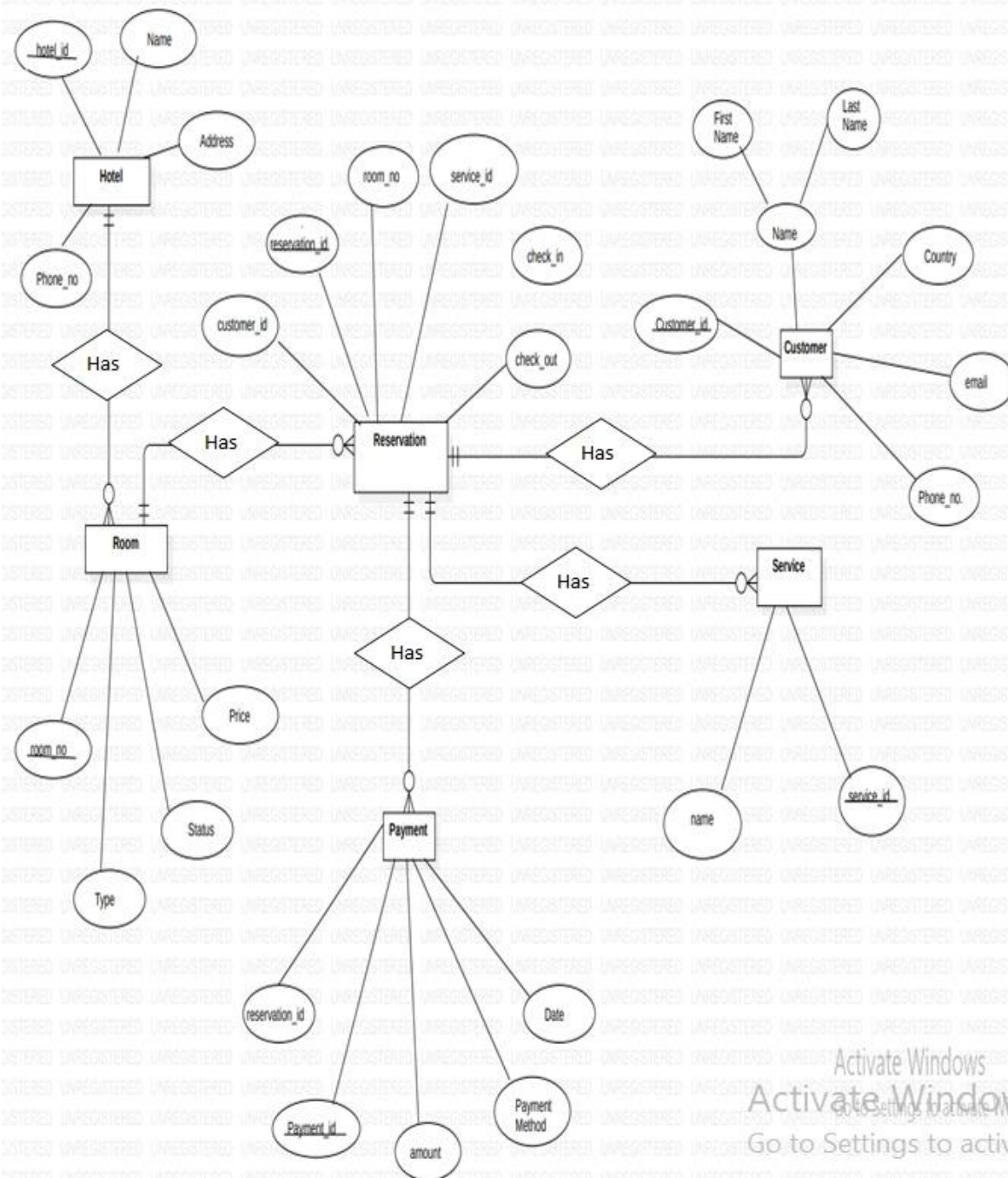
- **Documentation:** Comprehensive documentation for users and administrators.
- **Modularity:** The system should be modular to facilitate easy updates and maintenance.

Compliance

- **Legal:** Compliance with relevant laws and regulations, such as GDPR for data protection.

ER – DIAGRAM

Data Model1-ERDiagram1



ER – DIAGARM

Entities in the ER Diagram

1. Customer:

- Represents individuals or entities who book rooms and services at the hotel.
- Attributes may include: customer_id, name, phone_no, email, Country etc.

2. Rooms:

- Represents the rooms available in the hotel.
- Attributes may include: room_no., type, status (occupied or vacant), price, etc.

3. Hotel:

- Represents the hotel itself.
- Attributes may include: hotel_id, name, address, Phone_no., etc.

4. Service:

- Represents additional services offered by the hotel (e.g., room service, spa services).
- Attributes may include: service_id, name, etc.

5. Payment:

- Represents payments made by customers for reservations and services.
- Attributes may include: payment_id, reservation_id, amount, date, payment_method, status, etc.

6. Reservation:

- Represents the reservation made by a customer for a room and possibly services.
- Attributes may include: reservation_id, customer_id, room_no, check_in, check_out, etc.

Relationships in the ER Diagram

1. Customer-Reservation:

- A customer can make one or multiple reservations.
- One reservation is made by exactly one customer.
- This is a one-to-many relationship from Customer to Reservation.

2. Reservation-Rooms:

- A reservation can involve one or multiple rooms.
- A room can be part of zero or more reservations (especially for recurring bookings).
- This is a many-to-many relationship, typically implemented using a junction table in a relational database.

3. Reservation-Service:

- A reservation can include zero or multiple services.
- A service can be included in zero or more reservations.
- This is also a many-to-many relationship, managed similarly to ReservationRooms.

4. Reservation-Payment:

- A reservation involves one or more payments.
- A payment is linked to exactly one reservation.

- This is a one-to-many relationship from Reservation to Payment.

5. **Rooms-Hotel:**

- Each room belongs to one hotel.
- A hotel can have many rooms.
- This is a one-to-many relationship from Hotel to Rooms

RELATIONAL MODELS

Relational models represent the organization of data into tables (relations) consisting of rows and columns. Each table, identified by a unique name, stores records (tuples) where each record has multiple fields (attributes) defined by the schema. Relationships between tables are established using keys: primary keys uniquely identify records within a table, while foreign keys reference primary keys in related tables, ensuring referential integrity. This structure facilitates efficient data retrieval, management, and manipulation by using SQL queries to maintain consistency and integrity across the database.

- **Keys:** Unique identifiers used to establish and enforce relationships between tables. Primary keys uniquely identify each record in a table, while foreign keys reference primary keys in other tables.
- **Attributes:** Columns in a table that define the properties or characteristics of an entity. Each attribute holds a specific type of data.
- **Entities:** Distinct objects or concepts represented in a database, typically modeled as tables. Examples include customers, rooms, and reservations in a hotel management system.

In this **hotel management system**, we have designed six primary relational models (tables) to manage different aspects of hotel operations. These tables are: **Hotel**, **Reservation**, **Customer**, **Room**, **Service**, and **Payment**. Each table consists of various attributes and keys that define its structure and the relationships between them. Below is a detailed description of each table, including their attributes, keys, and the entities they represent.

1. Hotel

Description: This table stores information about the hotels in the system.

- **Attributes:**
 - **Hotel ID (Primary Key):** Unique identifier for each hotel.
 - **Name:** Name of the hotel.
 - **Address:** Location of the hotel.
 - **Phone:** Contact phone number of the hotel.

Entity: Hotel

2. Reservation

Description: This table records the reservations made by customers for rooms in the hotel.

- **Attributes:**
 - **ReservationID (Primary Key):** Unique identifier for each reservation.
 - **CustomerID (Foreign Key):** References CustomerID in the Customer table.
 - **RoomNo(Foreign Key):** References RoomID in the Room table.
 - **HotelID (Foreign Key):** References HotelID in the Hotel table.
 - **CheckIn:** Date when the customer is scheduled to check in.
 - **CheckOut:** Date when the customer is scheduled to check out.

Entity: Reservation

3. Customer

Description: This table stores information about the customers who make reservations at the hotel.

- **Attributes:**

- **CustomerID (Primary Key):** Unique identifier for each customer.
- **FirstName:** First name of the customer.
- **LastName:** Last name of the customer.
- **Phone:** Contact phone number of the customer.
- **Email:** Email address of the customer.
- **Country:** Home address of the customer.

Entity: Customer

4. Room

Description: This table contains details about the rooms available in the hotels.

- **Attributes:**

- **RoomNo (Primary Key):** Unique identifier for each room.
- **RoomType:** Type of the room (e.g., Single, Double, Suite).
- **Capacity:** Maximum number of guests that can stay in the room.
- **Price:** Cost per night for the room.
- **Status:** Current status of the room (e.g., Available, Booked, Maintenance).

Entity: Room

5. Service

Description: This table holds information about additional services offered by the hotel.

- **Attributes:**

- **ServiceID (Primary Key):** Unique identifier for each service.
- **ServiceName:** Name of the service (e.g., Room Service, Spa, Laundry).

Entity: Service

6. Payment

Description: This table records payment details for reservations and services availed by customers.

- **Attributes:**

- **PaymentID (Primary Key):** Unique identifier for each payment.
- **ReservationID (Foreign Key):** References ReservationID in the Reservation table.
- **CustomerID (Foreign Key):** References CustomerID in the Customer table.
- **PaymentDate:** Date when the payment was made.
- **Amount:** Amount paid by the customer.
- **PaymentMethod:** Method of payment (e.g., Credit Card, Debit Card, Cash, Online).

Entity: Payment

Relational Model Schema

The relational model schema for the hotel management system establishes the relationships between these tables to maintain data integrity and consistency. The relationships are defined by foreign keys that reference primary keys in other tables.

- **Hotel-Customer:** Customers can make reservations in multiple hotels, and each hotel can have multiple customers making reservations.
- **Hotel-Room:** Each hotel has multiple rooms, but each room is associated with one hotel.
- **Customer-Reservation:** Each customer can make multiple reservations, but each reservation is associated with one customer.
- **Room-Reservation:** Each room can be reserved multiple times by different customers over time, but each reservation is for one room.
- **Hotel-Service:** Each hotel can offer multiple services, but each service is associated with one hotel.
- **Reservation-Payment:** Each reservation can have multiple payments associated with it, but each payment is linked to one reservation.

Some Example of Relationships

1. **Hotel and Room:**
 - **HotelID (Primary Key) in Hotel Table**
 - **HotelID (Foreign Key) in Room Table**
2. **Customer and Reservation:**
 - **CustomerID (Primary Key) in Customer Table**
 - **CustomerID (Foreign Key) in Reservation Table**
3. **Room and Reservation:**
 - **RoomNo(Primary Key) in Room Table**
 - **RoomNo (Foreign Key) in Reservation Table**
4. **Hotel and Service:**
 - **HotelID (Primary Key) in Hotel Table**
 - **HotelID (Foreign Key) in Service Table**
5. **Reservation and Payment:**
 - **ReservationID (Primary Key) in Reservation Table**
 - **ReservationID (Foreign Key) in Payment Table**

These relational models ensure efficient data management and retrieval, enabling the hotel management system to function smoothly and provide a seamless experience for both customers and hotel staff.

DATABASE IMPLEMENTATION

Creation of Reservations table

```
CREATE TABLE hms.reservation_details(
```

```
    RESERVATION_ID int,  
    CUSTOMER_ID int,  
    ROOM_NO int,  
    SERVICE_ID int,  
    CHECK_IN varchar(30),  
    CHECK_OUT varchar(30)  
);
```

```
INSERT INTO hms.reservations_details (RESERVATION_ID, CUSTOMER_ID,  
ROOM_NO, SERVICE_ID, CHECK_IN, CHECK_OUT) VALUES  
(601,1001,101,201,"2024-06-15","2024-06-20"),  
(602,1002,102,202, "2024-06-16","2024-06-21"),  
(603, 1003, 103, 203, "2024-06-17","2024-06-22"),  
(604, 1004, 204, 204," 2024-06-18","2024-06-23"),  
(605, 1005, 105,205 ," 2024-06-19"," 2024-06-24"),  
(606,1006,106, 206, "2024-06-20","2024-06-25");
```

```
SELECT*FROM reservation_details;
```

OUTPUT:

RESERVATION_ID	CUSTOMER_ID	ROOM_NO	SERVICE_ID	CHECK_IN	CHECK_OUT
601	1001	101	201	2024-06-15	2024-06-20
602	1002	102	202	2024-06-16	2024-06-21
603	1003	103	203	2024-06-17	2024-06-22
604	1004	104	204	2024-06-18	2024-06-23
605	1005	105	205	2024-06-19	2024-06-24
606	1006	106	206	2024-06-20	2024-06-25

6 rows in set (0.00 sec)

Creation of Rooms table

```
CREATE TABLE hms.room_details (
```

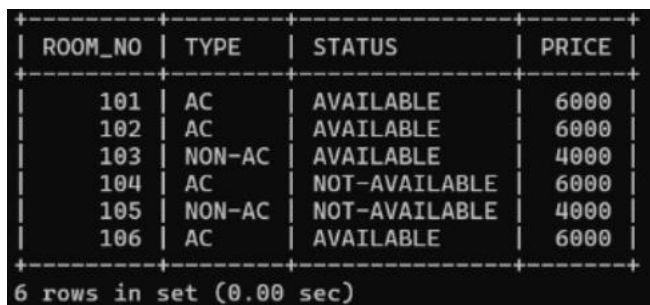
```
    ROOM_ID int,  
    TYPE varchar(38),
```

```
STATUS varchar(38),
PRICE int
);
```

```
INSERT INTO hms. room_details (ROOM_NO, TYPE, STATUS, PRICE) VALUES
(101, "AC", "AVAILABLE", 6000),
(102, "AC", "AVAILABLE", 6000),
(103, "NON-AC", "AVAILABLE", 4000),
(104, "AC", "NOT AVAILABLE 6000),
(105, "NON-AC", "NOT AVAILABLE", 4000)
(106, "AC", "AVAILABLE 6000);
```

```
SELECT*FROM room_details;
```

OUTPUT:



ROOM_NO	TYPE	STATUS	PRICE
101	AC	AVAILABLE	6000
102	AC	AVAILABLE	6000
103	NON-AC	AVAILABLE	4000
104	AC	NOT-AVAILABLE	6000
105	NON-AC	NOT-AVAILABLE	4000
106	AC	AVAILABLE	6000

6 rows in set (0.00 sec)

Creation of Services table

```
CREATE TABLE hms.service_details(

NAME varchar(38),
SERVICE ID int
);
```

```
INSERT INTO hms.service_details(NAME, SERVICE_ID) VALUES
("JOHN", 201),
("BILLY", 202),
("PETER", 203),
("JAMES",204),
("HENRY",205),
("SAM", 206);
```

```
SELECT*FROM servies_details;
```

OUTPUT:

NAME	SERVICE_ID
JOHN	201
BILLY	202
PETER	203
JAMES	204
HENRY	205
SAM	206

6 rows in set (0.00 sec)

Creation of Payment table

```
CREATE TABLE hms.payment (
  PAYMENT_ID int,
  AMOUNT int,
  PAYMENT_METHOD varchar(30),
  DATE varchar(30)
);
```

```
INSERT INTO hms.payment (PAYMENT_ID, AMOUNT, PAYMENT_METHOD, DATE)
VALUES
(1, 4000, "ONLINE", "2024-06-01"),
(2, 3500, "CASH", "2024-06-02"),
(3, 2000, "ONLINE", "2024-06-03"),
(4, 2500, "CASH", "2024-06-04"),
(5, 1000, "ONLINE", "2024-06-05"),
(6, 1500, "CASH", "2024-06-06");
```

```
SELECT * FROM payment;
```

OUTPUT:

PAYMENT_ID	AMOUNT	PAYMENT_METHOD	DATE
1	4000	ONLINE	2024-06-01
2	3500	CASH	2024-06-02
3	2000	ONLINE	2024-06-03
4	2500	CASH	2024-06-04
5	1000	ONLINE	2024-06-05
6	1500	CASH	2024-06-06

6 rows in set (0.00 sec)

Creation of Customer table

```
CREATE TABLE hms.customer details(
```

```
    CUSTOMER_ID int,  
    CUSTOMER_NAME varchar(36),  
    COUNTRY varchar(30),  
    CUSTOMER_EMAIL varchar(38),  
    PHONE_NO varchar(30))
```

```
INSERT INTO hms.customer details (CUSTOMER_ID, CUSTOMER_NAME,  
COUNTRY, CUSTOMER_EMAIL, PHONE_NO) VALUES  
(1001, "RAHUL", "INDIA", "rahul@gmail.com", "9632587413"),  
(1002, "NABI", "KOREA", "nabi@gmail.com", "8521479631"),  
(1003, "PRIYA", "INDIA", "priya@gmail.com", "7412369857"),  
(1004, "STEVE", "AMERICA", "steve@gmail.com", "9874 163219"),  
(1005, "ANANYA", "INDIA", "ananya@gmail.com", "3214569873"),  
(1006, "ERMA", "LONDON", "emma@", "6543219873");
```

```
SELECT*FROM customer_details;
```

OUTPUT:

CUSTOMER_ID	CUSTOMER_NAME	COUNTRY	CUSTOMER_EMAIL	PHONE_NO
1001	RAHUL	INDIA	rahul@gmail.com	9632587413
1002	NABI	KOREA	nabi@gmail.com	8521479631
1003	PRIYA	INDIA	priya@gmail.com	7412369857
1004	STEVE	AMERICA	steve@gmail.com	9874563219
1005	ANANYA	INDIA	ananya@gmail.com	3214569873
1006	EMMA	LONDON	emma@gmail.com	6543219873

6 rows in set (0.00 sec)

TESTING AND RESULTS

1. Write an SQL query to retrieve reservation details along with customer names

```
SELECT reservations_details.RESERVATION_ID,customer_details.CUSTOMER-  
NAME,reservations-  
details.ROOM_NO,reservations_details.CHECK_IN,reservations_details.CHECK_OUT  
FROM reservations_details INNER JOIN customer_details ON  
reservations_details.CUSTOMER_ID = customer_details.CUSTOMER_ID;
```

OUTPUT:

RESERVATION_ID	CUSTOMER_NAME	ROOM_NO	CHECK_IN	CHECK_OUT
601	RAHUL	101	2024-06-15	2024-06-20
602	NABI	102	2024-06-16	2024-06-21
603	PRIYA	103	2024-06-17	2024-06-22
604	STEVE	104	2024-06-18	2024-06-23
605	ANANYA	105	2024-06-19	2024-06-24
606	EMMA	106	2024-06-20	2024-06-25

6 rows in set (0.00 sec)

2. Write query to retrieve payments in the range 2024-06-02,2024-06-05.

```
SELECT *  
FROM payment  
WHERE DATE BETWEEN 2024-06-02 AND 2024-06-05;
```

OUTPUT:

PAYMENT_ID	AMOUNT	PAYMENT_METHOD	DATE
2	3500	CASH	2024-06-02
3	2000	ONLINE	2024-06-03
4	2500	CASH	2024-06-04

3 rows in set (0.00 sec)

3. Write query to determine the type of room that has the highest number of reservations in a hotel.

```
SELECT room_details.TYPE, COUNT(reservation_details.RESERVATION_ID) AS  
number_of_reservations
```

FROM room_details

JOIN reservations_details reservation_details ON room_details.ROOM_NO = reservation_details.ROOM_NO

GROUP BY room_details.TYPE

ORDER BY number_of_reservations DESC

LIMIT 1;

OUTPUT:

TYPE	no_of_reservations
AC	4
1 row in set (0.01 sec)	

4. Write query to retrieve the maximum SERVICE_ID.

SELECT max(SERVICE_ID) as max_service_id from hms.service_details;

OUTPUT:

```
+-----+
| max_service_id |
+-----+
|                206 |
+-----+
1 row in set (0.01 sec)
```

5. Write query to retrieve all details of rooms that are currently available in a hotel.

SELECT * FROM room_details where STATUS="AVAILABLE";

Output:

ROOM_NO	TYPE	STATUS	PRICE
101	AC	AVAILABLE	6000
102	AC	AVAILABLE	6000
103	NON-AC	AVAILABLE	4000
106	AC	AVAILABLE	6000
4 rows in set (0.00 sec)			

6. Write query to find the names and countries of customers who have made reservations but have not made any payments.

```
SELECT customer_details.CUSTOMER_NAME, customer_details.COUNTRY FROM  
customer_details LEFT JOIN reservation_details ON customer_details.CUSTOMER_ID =  
reservation_details.CUSTOMER_ID LEFT JOIN payment_details ON reservation_details.RESERVATION_ID  
= payment_details.PAYMENT_ID where payment_details.PAYMENT_ID is NULL;
```

OUTPUT:

CUSTOMER_NAME	COUNTRY
RAHUL	INDIA
NABI	KOREA
PRIYA	INDIA
STEVE	AMERICA
ANANYA	INDIA
EMMA	LONDON

6 rows in set (0.02 sec)

7. Write a query to find the customer IDs and countries of customers who have reserved rooms of a specific type, along with the type of room they reserved.

```
SELECT  
customer_details.CUSTOMER_ID, customer_details.COUNTRY, room_details.TYPE as  
roomtype FROM customer_details JOIN reservations_details ON  
customer_details.CUSTOMER_ID = reservation_details.CUSTOMER_ID JOIN ON  
reservations_details.ROOM_NO = room_details.ROOM_NO where  
room_details.TYPE="AC";
```

OUTPUT:

CUSTOMER_NAME	COUNTRY	roomtype
RAHUL	INDIA	AC
NABI	KOREA	AC
STEVE	AMERICA	AC
EMMA	LONDON	AC

4 rows in set (0.00 sec)

8. Write a query to find the reservation IDs, names, and countries of customers who have made reservations and are from India.

SELECT

reservations_details.RESERVATION_ID,customer_details.CUSTOMER_NAME,customer_details.COUNTRY from reservations_details JOIN customer_details ON reservations_details.CUSTOMER_ID = customer_details.CUSTOMER_ID where customer_details.COUNTRY = "INDIA";

OUTPUT:

RESERVATION_ID	CUSTOMER_NAME	COUNTRY
601	RAHUL	INDIA
603	PRIYA	INDIA
605	ANANYA	INDIA

3 rows in set (0.00 sec)

9. Write an SQL query to retrieve all details of the most recent payment made in a hotel.

SELECT *FROM payment order by DATE desc limit 1;

OUTPUT:

PAYMENT_ID	AMOUNT	PAYMENT_METHOD	DATE
6	1500	CASH	2024-06-06

1 row in set (0.00 sec)

10. Write an SQL query to update the status of a specific room in a hotel to "available".

UPDATE room_details SET status = "available" where ROOM_NO = 104;

OUTPUT:

ROOM_NO	TYPE	STATUS	PRICE
101	AC	AVAILABLE	6000
102	AC	AVAILABLE	6000
103	NON-AC	AVAILABLE	4000
104	AC	available	6000
105	NON-AC	NOT-AVAILABLE	4000
106	AC	AVAILABLE	6000

6 rows in set (0.00 sec)

11. Write an SQL query to delete a specific record from the service_details table.

DELETE FROM hms.service_details WHERE NAME = "JOHN";

OUTPUT:

```
+-----+-----+
| NAME | SERVICE_ID |
+-----+-----+
| BILLY |          202 |
| PETER |          203 |
| JAMES |          204 |
| HENRY |          205 |
| SAM   |          206 |
| ALICE |          207 |
+-----+-----+
6 rows in set (0.00 sec)
```

12. Write an SQL query to find the number of unique service names in the services_details table

```
SELECT COUNT(DISTINCT NAME) AS UNIQUE_NAMES FROM hms.services_details;
```

OUTPUT:

```
+-----+
| unique_names |
+-----+
|             6 |
+-----+
1 row in set (0.00 sec)
```

13. Write an SQL query to retrieve and display all service names from the service_details table of a hotel management system, sorted in alphabetical order.

```
SELECT NAME FROM hms.service_details ORDER BY NAME;
```

OUTPUT:

```
+-----+
| NAME |
+-----+
| ALICE |
| BILLY |
| HENRY |
| JAMES |
| PETER |
| SAM   |
+-----+
6 rows in set (0.00 sec)
```

14. Write an SQL query to calculate the total revenue generated each month from payments in a hotel.

```
SELECT payment.DATE as month,sum(payment.AMOUNT) AS totalrevenue FROM  
payment GROUP BY payment.DATE;
```

OUTPUT:

month	totalrevenue
2024-06-01	4000
2024-06-02	3500
2024-06-03	2000
2024-06-04	2500
2024-06-05	1000
2024-06-06	1500

6 rows in set (0.01 sec)

15. Write an SQL query to retrieve the names of customers along with their countries, check-in dates, and check-out dates who are currently booked in room number 104.

```
SELECT  
customer_details.CUSTOMER_NAME,customer_details.COUNTRY,reservations_details.C  
HECK_IN, reservations_details.CHECK_OUT FROM reservations_details JOIN  
customer_details.CUSTOMER_ID = customer_details.CUSTOMER_ID JOIN room_details  
ON reservations_details.ROOM_NO = room_details.ROOM_NO WHERE  
room_details.ROOM_NO = 104;
```

OUTPUT:

CUSTOMER_NAME	COUNTRY	CHECK_IN	CHECK_OUT
STEVE	AMERICA	2024-06-18	2024-06-23

1 row in set (0.00 sec)