Database Design Guide

This guide will help the student to create a database on the Hotel Rooms Management System. It will help to manage the below functionalities.

- Hotel
- Customer
- Employee
- Rooms
- Payment
- Booking
- Review
-

We will use MySQL as the DBMS to create the database and its related operations.

1. Introduction to MySQL

MySQL is an open-source relational database management system (RDBMS) that uses structured query language (SQL) to manage and manipulate data in a database. It is widely used for various applications, from small web applications to large enterprise systems.

MySQL's key features include:

- Scalability: Capable of handling large amounts of data and concurrent connections.
- Flexibility: Supports various data types and storage engines.
- Performance: Optimized for speed and efficiency.
- Reliability: Known for its stability and robustness.

2. Installation of MySQL

MySQL can be installed on various operating systems, including Windows, macOS, and Linux. Here are the general steps to install MySQL:

Windows:

- Download the MySQL installer from the official website. https://dev.mysql.com/downloads/installer/
- Run the installer and follow the on-screen instructions.
- Choose the installation type (Typical, Complete, or Custom). Recommended Custom.
- Set a root password for the MySQL server.

3. E-R Diagram (ERD)

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that shows the entities, attributes, relationships between entities, and cardinality. ERDs are commonly used in database design to help developers and stakeholders understand the structure and relationships within a database.

Identify Entities

- Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data.
- Each entity should correspond to a table in your database.

Define Attributes

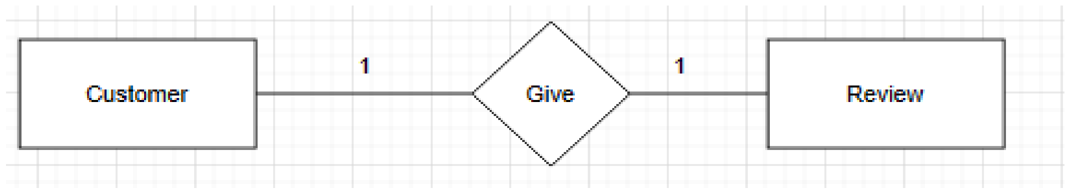
- For each entity, list the attributes (properties or fields) that describe it.
- These attributes will become columns in the corresponding database table.

Identify Relationships

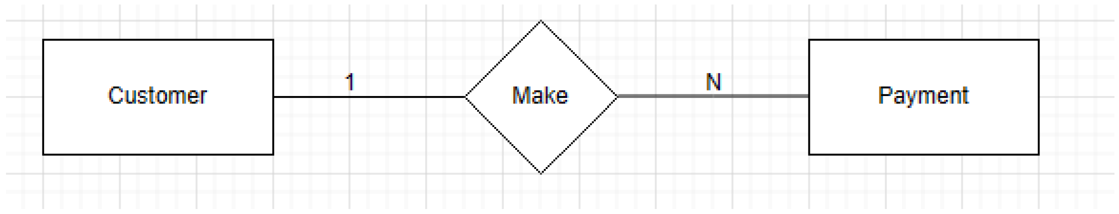
- Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
- Represent these relationships using lines connecting the entities.

Let's see a few examples of relationships:

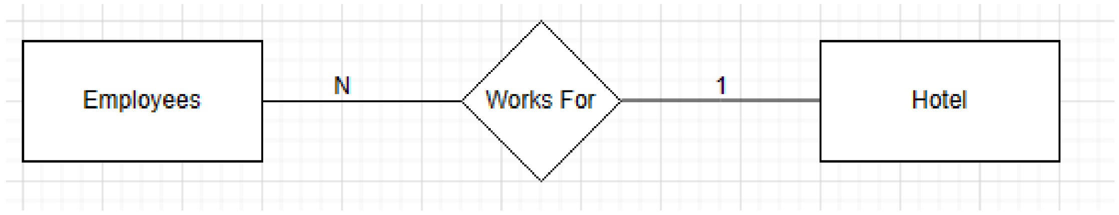
One to One



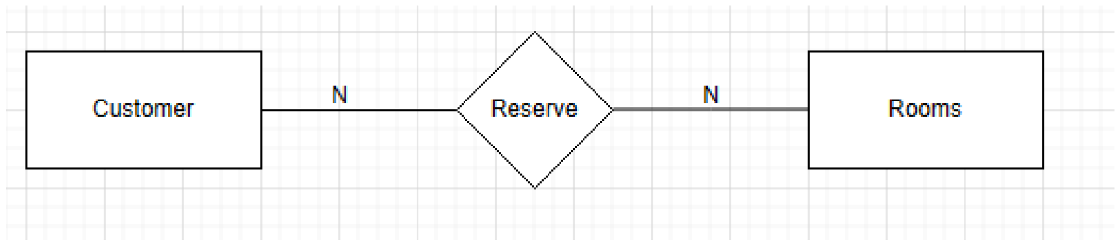
One to Many



Many to One



Many to Many



Cardinality Notation
Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

- Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship.
- Cardinality describes how many instances of one entity are related to how many instances of another entity.
- Common notations include:
    - One (1)
    - Zero or one (0..1)
    - Many (N)
    - Zero or many (0..N)

Optional:
Add Attributes and Constraints

- Include additional information in your ERD, such as primary keys, foreign keys, and constraints (e.g., unique constraints).

Create the Diagram

- Use specialized diagramming software or tools (e.g., Lucidchart, draw.io, or even pen and paper) to create your ERD.

Refine and Review:

- Review your ERD with stakeholders and team members to ensure it accurately represents the data model and relationships. Make any necessary refinements.

Let's identify the entities of the Hotel Rooms Management System:

- Hotel
- Customer
- Employees
- Rooms
- Payment
- Bookings
- Review

*** Now let’s identify the attributes and relationships of each entity for the Student Management System.

<u>**Hotel**</u>

- **Attributes:**
  HotelId(Primary Key)
  HotelName
  HotelAddress
  HotelEmailId
  HotelPhoneNo
- **Relationships:**
  One **Hotel** has Many **Custome**r (**One-to-Many**)
  One **Hotel** owns Many **Rooms** (**One-to-Many**)

<u>**Customer**</u>

- **Attributes:**
  custID (Primary Key)
  custName
  custPhoneNo
  custAddress
  custEmaild

- **Relationships:**
  Many **Customer** Check in or check out Many **Rooms** (**Many-to-Many**)
  One **Customer** Check in or check out One **Rooms** (**One-to-One**)

<u>**Rooms**</u>

- **Attributes:**
  roomId(Primary Key)
  roomNo
  roomType
  roomPrice
  status
  hoteltId(Foreign Key)
- **Relationships:**
  One **Customer** has Many **Rooms** (**One-to-Many**)
  Rooms Type:Classic,Deluxe

<u>**Booking**</u>

- **Attributes**:
  BookingId (Primary Key)
  checkInDate
  checkOutDate
  totalAmount
  custId(Foreign Key)
  roomId(Foreign Key)

- **Relationships:**
  Many **Customer** has many **Booking (Many-to-Many)**
  One **Customer** has many **Booking (One-to-Many)**

Employee

- **Attributes**:
  empID (Primary Key)
  empName
  empPhoneNo
  empAddress
  salary
- **Relationships:**
  Many **Employee** works for One **Hotel (Many-to-one)**

Payment

- **Attributes**:
  PaymentId(Primary Key)
  PaymentDate
  PaymentMethodS
  PaymentAmount
  BookingId(Foreign key)

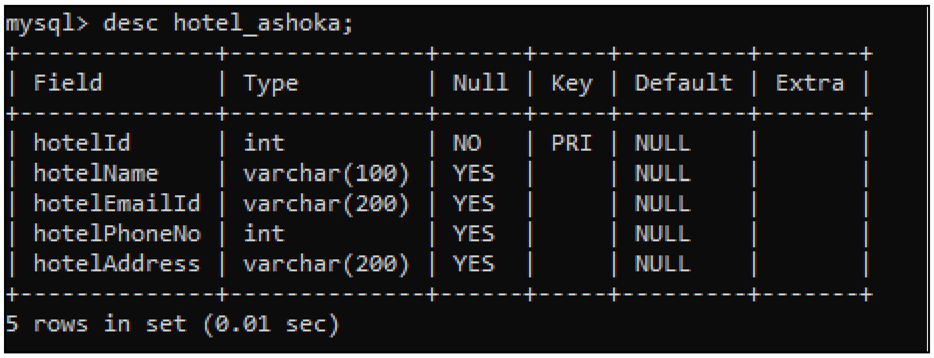
- **Relationships:**
  One **Customer** can make multiple **payment(One-to-Many)**

Review

- **Attributes**:
  ReviewID (Primary Key)
  Rating
  ReviewDate
  Comment
  bookingId(Foreign key)
- **Relationships:**
  One **Customer** can give Many **Review(One-to-Many)**

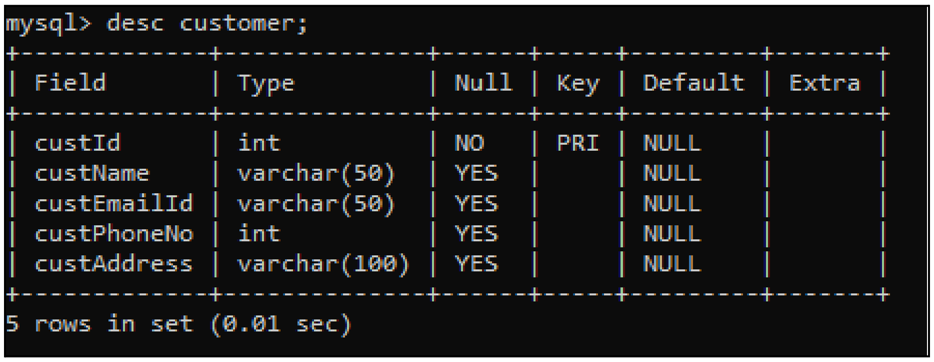
Table Structure

1. Hotel



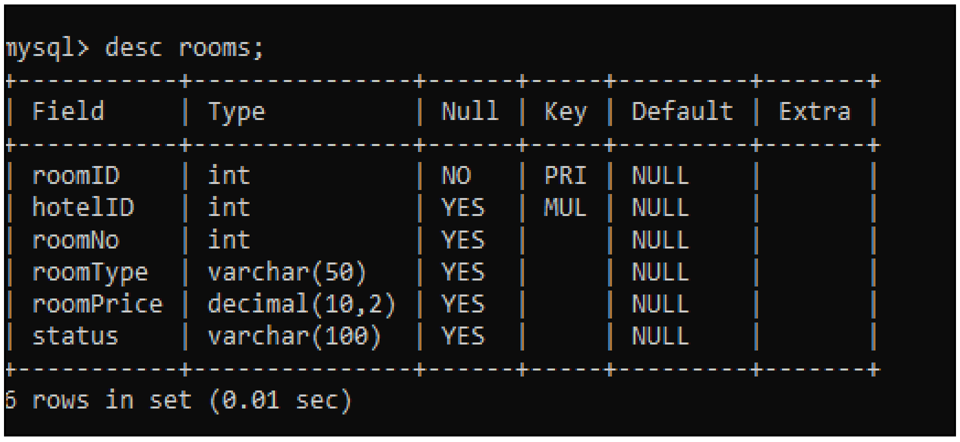
```
mysql> desc hotel_ashoka;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| hotelId      | int          | NO   | PRI | NULL    |       |
| hotelName    | varchar(100) | YES  |     | NULL    |       |
| hotelEmailId | varchar(200) | YES  |     | NULL    |       |
| hotelPhoneNo | int          | YES  |     | NULL    |       |
| hotelAddress | varchar(200) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

2. Customer



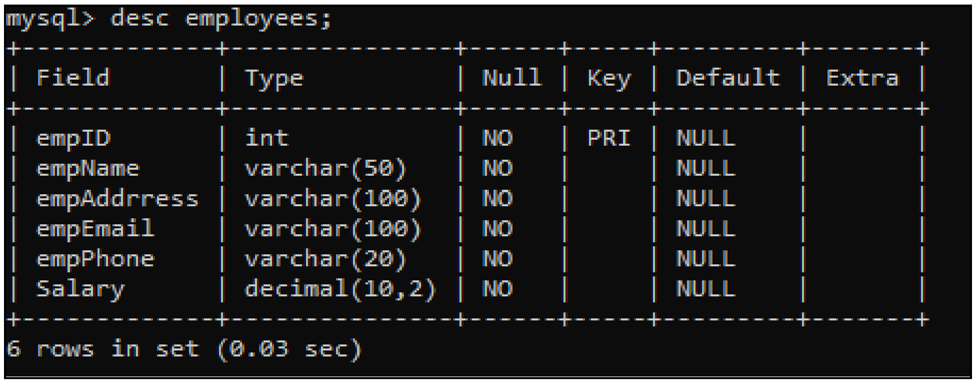
```
mysql> desc customer;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| custId      | int          | NO   | PRI | NULL    |       |
| custName    | varchar(50)  | YES  |     | NULL    |       |
| custEmailId | varchar(50)  | YES  |     | NULL    |       |
| custPhoneNo | int          | YES  |     | NULL    |       |
| custAddress | varchar(100) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

3. Rooms



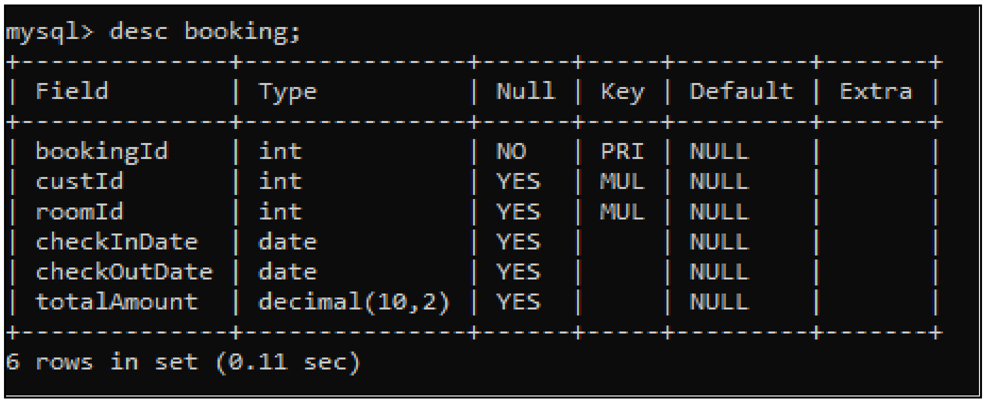
```
mysql> desc rooms;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| roomID    | int           | NO   | PRI | NULL    |       |
| hotelID   | int           | YES  | MUL | NULL    |       |
| roomNo    | int           | YES  |     | NULL    |       |
| roomType  | varchar(50)   | YES  |     | NULL    |       |
| roomPrice | decimal(10,2) | YES  |     | NULL    |       |
| status    | varchar(100)  | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

4. Employees:



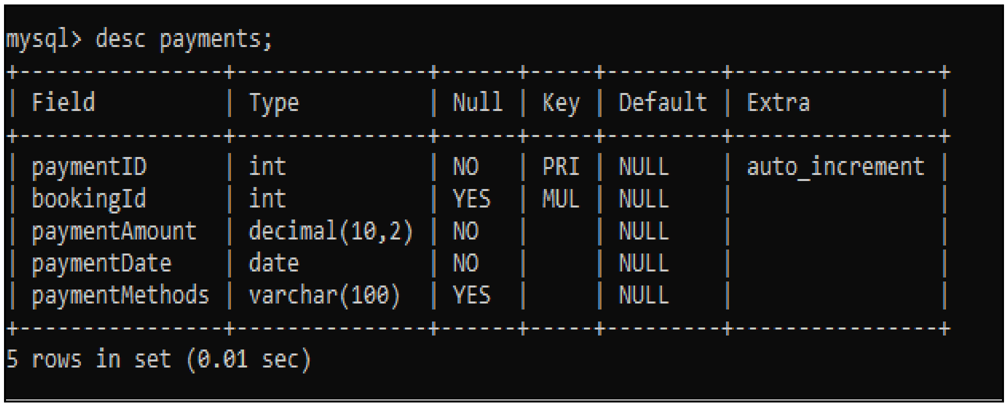
```
mysql> desc employees;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| empID       | int           | NO   | PRI | NULL    |       |
| empName     | varchar(50)   | NO   |     | NULL    |       |
| empAddrress | varchar(100)  | NO   |     | NULL    |       |
| empEmail    | varchar(100)  | NO   |     | NULL    |       |
| empPhone    | varchar(20)   | NO   |     | NULL    |       |
| Salary      | decimal(10,2) | NO   |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

5.Booking:



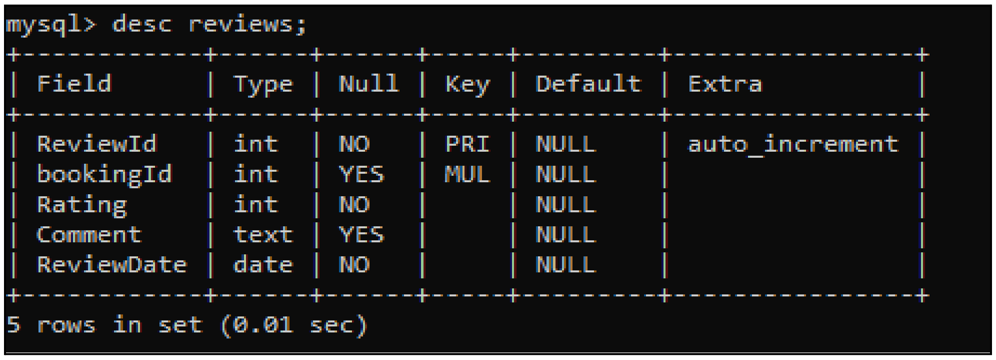
```
mysql> desc booking;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| bookingId    | int           | NO   | PRI | NULL    |       |
| custId       | int           | YES  | MUL | NULL    |       |
| roomId       | int           | YES  | MUL | NULL    |       |
| checkInDate  | date          | YES  |     | NULL    |       |
| checkOutDate | date          | YES  |     | NULL    |       |
| totalAmount  | decimal(10,2) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
6 rows in set (0.11 sec)
```

6.Payments:



```
mysql> desc payments;
+----------------+---------------+------+-----+---------+----------------+
| Field          | Type          | Null | Key | Default | Extra          |
+----------------+---------------+------+-----+---------+----------------+
| paymentID      | int           | NO   | PRI | NULL    | auto_increment |
| bookingId      | int           | YES  | MUL | NULL    |                |
| paymentAmount  | decimal(10,2) | NO   |     | NULL    |                |
| paymentDate    | date          | NO   |     | NULL    |                |
| paymentMethods | varchar(100)  | YES  |     | NULL    |                |
+----------------+---------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

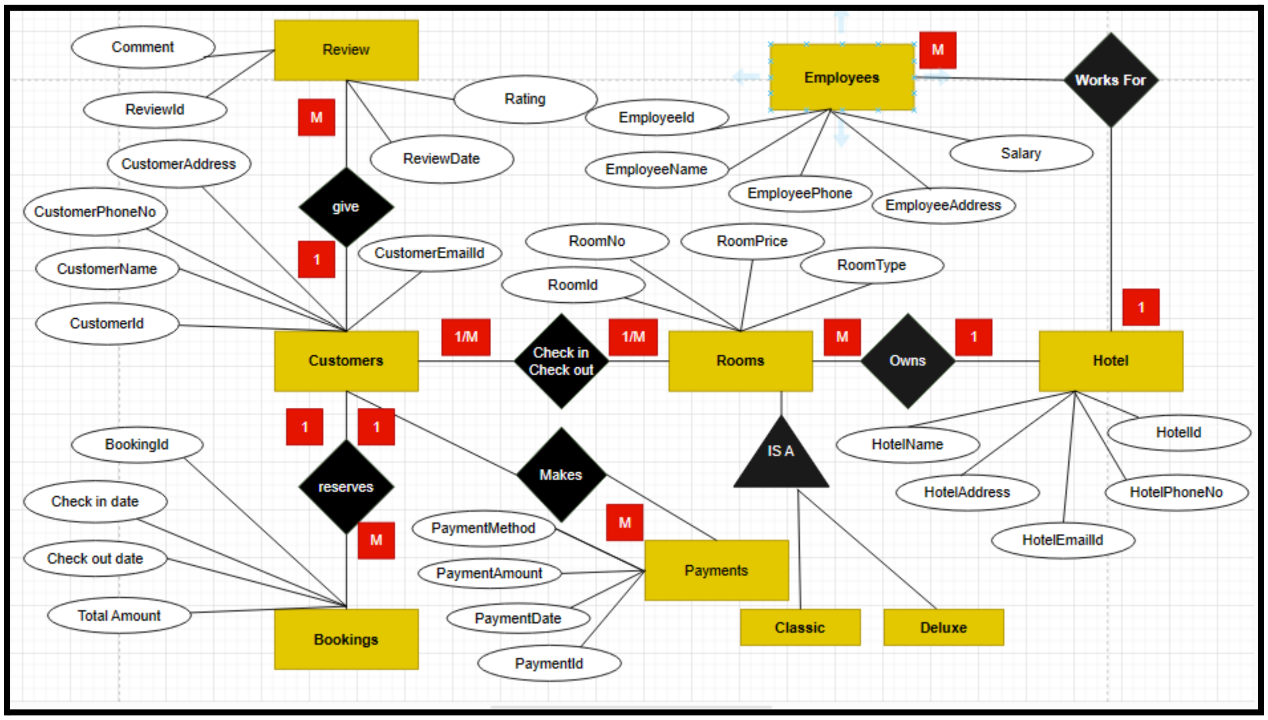
7.Reviews:



```
mysql> desc reviews;
+------------+------+------+-----+---------+----------------+
| Field      | Type | Null | Key | Default | Extra          |
+------------+------+------+-----+---------+----------------+
| ReviewId   | int  | NO   | PRI | NULL    | auto_increment |
| bookingId  | int  | YES  | MUL | NULL    |                |
| Rating     | int  | NO   |     | NULL    |                |
| Comment    | text | YES  |     | NULL    |                |
| ReviewDate | date | NO   |     | NULL    |                |
+------------+------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Now, let’s create the ER diagram to visually represent the entities and relationships.

ERD Diagram



In this ERD:

- Guest can book and each room can have multiple students, creating a many-to-many relationship.
- The Enrollment entity serves as a bridge table between Student and Course entities to represent this relationship.
- Multiple courses can be taught by one Instructor (many-to-one relationship).
- Each Instructor can teach multiple courses (one-to-many relationship).
- A student can give multiple feedbacks

- Student may have scores of multiple courses

4. Creating a Database
Using MySQL server, create a new database for your student management system. You can do this with SQL commands or through the graphical interface.

```
CREATE DATABASE HOTELS;
```

5. Using a Database
Before performing any operations on a database, you need to select it using the USE statement:

```
USE HOTELS;
```

6. Creating the tables for each entity

```
create database hotels;
use hotels;
create table Hotel_ashoka(
hotelId INT primary KEY,
hotelName VARCHAR(100),
hotelEmailId varchar(200),
hotelPhoneNo int(10),
hotelAddress varchar(200)
);
CREATE TABLE Rooms (
   roomId INT PRIMARY KEY,
   hotelId INT,
   roomNo INT,
   roomType VARCHAR(50),
   roomPrice DECIMAL(10, 2),
   status VARCHAR(100);
   FOREIGN KEY (hotelId) REFERENCES Hotel_ashoka(hotelId)
);
CREATE TABLE Customer (
   custId INT PRIMARY KEY,
   custName VARCHAR(50),
   custEmailId VARCHAR(50),
   custPhoneNo varchar(20),
   custAddress VARCHAR(100)
);
CREATE TABLE Booking (
   bookingId INT PRIMARY KEY,
   custId INT,
   roomId INT,
   checkInDate DATE,
   checkOutDate DATE,
   totalAmount DECIMAL(10, 2),
   FOREIGN KEY (custId) REFERENCES Customer(custId),
   FOREIGN KEY (roomId) REFERENCES Rooms(roomId)
);
CREATE TABLE Payments (
   paymentID INT PRIMARY KEY,
   bookingId INT,
   paymentAmount DECIMAL(10, 2) NOT NULL,
   paymentDate DATE NOT NULL,
   paymentMethods varchar(100),
   FOREIGN KEY (bookingId) REFERENCES Booking(bookingId)
);
CREATE TABLE Employees (
   empID INT PRIMARY KEY,
   empName VARCHAR(50) NOT NULL,
   empAddrress VARCHAR(100) NOT NULL,
   empEmail VARCHAR(100) NOT NULL,
   empPhone VARCHAR(20) NOT NULL,
   Salary DECIMAL(10, 2) NOT NULL
);
CREATE TABLE Reviews (
   ReviewId INT PRIMARY KEY,
   bookingId INT,
   Rating INT NOT NULL,
   Comment TEXT,
   ReviewDate DATE NOT NULL,
   FOREIGN KEY (bookingId) REFERENCES Booking(bookingId)
);
```

Auto-increment

```
ALTER TABLE Reviews modify ReviewId int auto_increment;
ALTER TABLE booking bookingId int auto_increment;
```

7. Insert records

Add data to your tables to work with. This step helps you test your database.

-- Insert hotels:

```
INSERT INTO hotel_Ashoka value(
664464,'Ashoka_hotel','ashokdhadi@gmail.com',986789764,'Mankhurd'
);
```

-- Insert rooms

```
 INSERT INTO rooms (roomID,hotelID,roomNo,roomType,roomPrice) VALUES
    (10023,664464,101,'classis',2500.00),
    (10024,664464,102,'classis',2500.00),
    (10025,664464,103,'classis',2500.00),
     (10026,664464,104,'deluxe',4000.00),
     (10027,664464,105,'classis',2500.00),
      (10028,664464,201,'deluxe',4000.00),
      (10029,664464,202,'classis',2500.00),
       (10030,664464,203,'deluxe',4000.00);
```

-- Enroll customer

```
INSERT INTO customer ( custId,custName,custEmailId,custPhoneNo,custAddress)
VALUES
        (40010,'priti singh','pritisingh89@gmail.com',897654345,'alibagh'),
        (40011,'vidhi kumar','vidhikumar43@gmail.com',789867543,'panvel'),
        (40012,'sara jadahv','sarajadhv@gmail.com',675645348,'kharghar'),
        (40013,'sagar yadhv','sagarjadhv45@gmail.com',986547868,'pune'),
        (40014,'suresh kamble','sureshkamble@gmail.com',778896503,'raigad'),
        (40015,'yogesh more','yogeshkamble@gmail.com',908796583,'pen'),
        (40016,'mayur patil','mayurpatil@gmail.com',976896430,'sagli');
```

-- Insert employees

```
INSERT INTO employees
(empID,empName,empAddrress,empEmail,empPhone,Salary) VALUES

(5001,'ram kadam','vadala','ramkadam@gmail.com',642308976,20000),

(5002,'sham lal','ghatkoper','shamlal@gmail.com',897896573,25000),

(5003,'kishor patil','nerul','kishorpatil@gmail.com',787965478,25000),

(5004,'anish jadhv','kurla','anishjadhv@gmail.com',897860087,20000),

(5005,'sahil zage','chembur','sahilzage@gmail.com',677689055,20000),

(5006,'vinayak sharma','juinagar','vinayaksharma12@gmail.com',998768964,25000);
```

-- Insert booking

```
INSERT INTO  booking

(bookingId,custId,roomId,checkInDate,checkOutDate,totalAmount)VALUES

(6001,40010,10023,'2023-12-01','2023-12-02',4000),

(6002,40011,10024,'2023-11-29','2023-11-30',4000),

(6003,40012,10025,'2023-11-26','2023-11-28',8000),
```

(6004,40013,10026,'2023-11-29','2023-12-01',10000);

-- Insert payments

INSERT INTO payments
(paymentID,bookingId,paymentAmount,paymentDate,paymentMethods)VALUES

(7001,6001,4000,'2023-12-01','cash'),

(7002,6002,4000,'2023-11-29','online'),

(7003,6003,8000,'2023-11-26','cash'),

(7004,6004,10000,'2023-11-29','cash');

-- Insert booking

INSERT INTO reviews(ReviewId,bookingId,Rating,Comment,ReviewDate)VALUES

(8001,6001,5,'Great experience!','2023-12-02'),

(8002,6002,5,'Excellent hotel and services!','2023-11-30'),

(8003,6003,4,'Great experience!','2023-11-28'),

(8004,6004,5,'Good experience!','2023-12-01')

8. Select records
Write SQL queries to retrieve and manage data.

For example:

Retrieve all student:

Select * FROM customer;

Retrieve a student's enrolled courses:

select * from booking where custId=40010;

select * from payments where bookingId=6001;

*Now try similar Select queries with other tables

9. Update records

Write SQL statements to update record(s) when needed. For example:

Update a customer phoneNo:

UPDATE customer

update customer set custPhoneNo=797654345 where custId=40010;
update customer set custEmaiulI='mayurpatil@gmail.com' where custId=40016;

10. Delete records

Write SQL statements to delete record(s) when needed.

DELETE FROM customer where custName='suresh kamble'';

PN: Ideally no data should be deleted from any tables. You can use an additional column to set the status of that record to ‘Active/Inactive’, etc. Or you can use an Archive table to move the unnecessary records out of the main table.