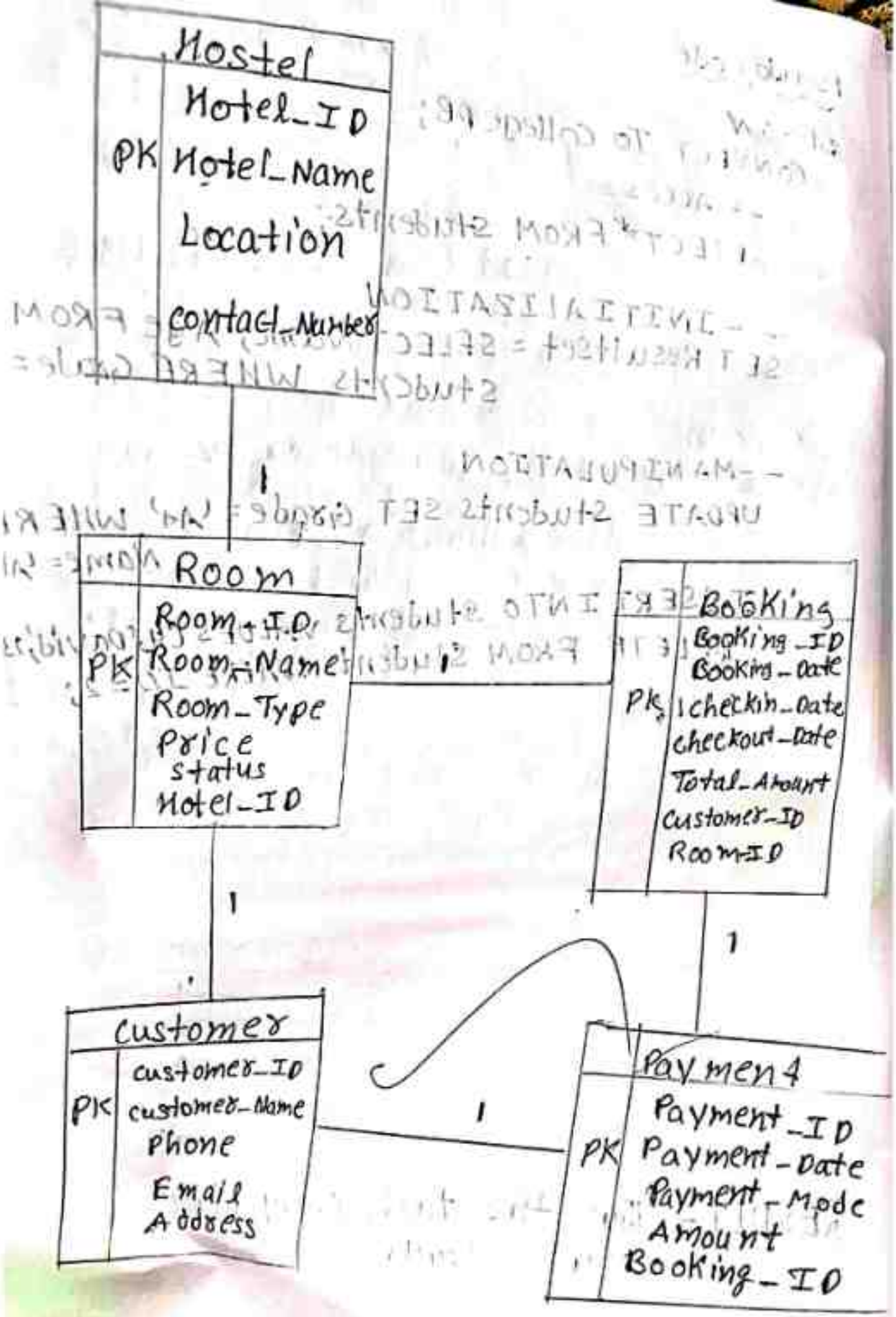
Hostel
Hotel_ID
PK Hotel_Name
Location
Contact_Number
1
Room
Room_ID
PK Room_Name
Room_Type
Price
status
Hotel_ID
1
Booking
Booking_ID
Booking_Date
PK checkin_Date
checkout_Date
Total_Amount
Customer_ID
Room_ID
1
1
Customer
customer_ID
PK customer_Name
Phone
Email
Address
1
Payment
Payment_ID
PK Payment_Date
Payment_Mode
Amount
Booking_ID

Q30: Draw E-R diagram for hotels, rooms, bookings, and customers.

Entites and Attributes:

1. Hotel
   - hotel_id(PK)
   - hotel-name
   - location
   - rating
   - contact_no

2. Room
   - room_id(PK)
   - hotel_id(FK)
   - room_type(Single/Double/Deluxe)
   - price_per_night
   - status(Available/Booked)

3) Customer
   - customer_id(PK)
   - name
   - email
   - Phone
   - address

4) Booking
   - booking_id(PK)
   - customer_id(PK)
   - room_id(FK)
   - check_in_date
   - check_out_date
   - total_amount
   - payment_status

5) Relationships:

- Hotel → Room = 1:M one hotel has many rooms
- Customer → Booking = 1:M one customer can have many booking
- Room → Booking = 1:1 or M:1 (A room can have many bookings over time, but one active booking at a time)

Q31) Normalize relations up to BCNF and Justify each step.

A) Booking(hotel_id, hostel_name, location, room_id, room_type, price, customer_id, customer_name, customer_email, check_in, check_out, total_amount)

1NF (First Normal Form)

→ No repeating groups.
All attributes contain atomic values

2NF (Second Normal Form)

→ Remove partial dependency (non-key attributes depending only on part of the composite key).
Decompose into smaller tables:

- Hotel(hotel_id, hotel_name, location)
- Room(room_id, hotel_id, room_type, price)
- Customer(customer_id, customer_name, customer_email)
- Booking(booking_id, room_id, customer_id, check_in, check_out, total_amount)

3NF (Third Normal Form)

Remove transtive dependencies (non-key depending on another non-key).
No such dependencies now

BCNF (Boyce-Codd Normal Form)

→ Every determent must be a candidate key each table's determinant(like hotel_id, room_id, etc) is a candidate key.

Q32. Write SQL queries for room availability, customer, check-in/out, and billing.

a) Room Availability

```
SELECT room_id, room_type, price_per_night
FROM Room
WHERE Status = 'Available';
```

b) Customer check-in (update status)

```
UPDATE Room
SET status = 'Booked'
WHERE room_id = 101;
```

c) Customer check-out (update status)

```
UPDATE Room
SET status = 'Booked' 'Available'
WHERE room_id = 101;
```

d) Billing

```
SELECT b. booking_id, c.name AS
Customer_name,
    r.room_type, r.price_per_night,
    DATEDIFF(b.check_out_date,
  b.check_in_date) AS nights,
      (r.price_per_night *
DATEDIFF (b.check_out_date,
 b.check_in_date) As nights,
      (r.price_per_night *
 DATEDIFF(b.check_out_date, b.check_in_date)
AS total_amount
FROM Booking b
JOIN Customer c ON b.customer_id =
 c.Customer_id
JOIN Room r ON b.room_id =>
```

Discuss how rollback and commit ensure reliability in booking management.

Explanation:

- COMMIT:

Confirms all changes made by a transaction are saved permanently in the database.

Example: when a booking is successfully completed, the system COMMITs the transaction - so the booking and payment data are stored permanently.

- ROLLBACK:

Reverts changes made during a transaction if any error occurs

Example: IF payment fails midway, the system ROLLBACKs - meaning the booking and room status changes are undone, keeping data consistent

Importance in Hotel Management:

- Prevents double booking of rooms.
- Maintains data integrity during multiple simultaneous operations
- Ensures atomicity of transactions - either all steps succeed or none do

Q34. Implement CRUD operations using MongoDB for guest management
Assume collection name = Customers

1. Create (Insert)

```
db.Customers.insertOne({
Customer_id : 1
name : "Karthik"
email : "Karthik@gmail.com",
phone : "9876543210",
address : "Chennai"
});
```

2. Read (Find)

```
db.Customers.find({name: "Karthik"});
```

3. Update

```
db.Customers.updateOne(
  {customer_id:1},
  {$set : {phone: "9998887776"}}
);
```

4. Delete

```
db.Customers.deleteOne({customer_id:1});
```

Bonus - View all guests:

```
db.Customers.find().prettY();
```

VEL TECH - CSE	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (4)	
TOTAL (15)	15
SIGN WITH DATE	