CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT

ON

HOSTEL MANAGEMENT SYSTEM

Program name: BCA

Subject name/Code: Data Management System (23CAT-251/23CAP-252)

Submitted by:

Name: Vansh Kumar

ID: 23BCA10471

Section: 23BCA- 4(B)

Submitted to:

Name: Mr. Arvinder Singh

HOSTEL MANAGEMENT SYSTEM

ABSTRACT

As the name specifies “HOSTEL MANAGEMENT SYSTEM” is a software developed for managing various activities in the hostel. For the past few years the number of educational institutions are increasing rapidly. Thereby the number of hostels are also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the person who are running the hostel and software’s are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of omputerized system that will be compatible to the existing system with the system Which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

· Less human error

· Strength and strain of manual labour can be reduced

· High security

· Data redundancy can be avoided to some extent

· Data consistency

· Easy to handle

· Easy data updating

· Easy record keeping

· Backup data can be easily generated

Table of Content

1. Introduction
   1.1 Overview of the Project
   1.2 Types of Users

2. Objective
   2.1 Purpose of the Project
   2.2 Specific Goals

3. Designing
   3.1 Entity-Relationship (ER) Diagram
   3.2 Relational Schema Diagram
   3.3 Relational Schema (Detailed Explanation)

4. Database
   4.1 Tables with Sample Data
   4.2 Keys and Relationships
   4.3 Normalization and Integrity

5. Coding (Queries)
   5.1 Table Creation Scripts
   5.2 Data Insertion Queries
   5.3 SELECT and JOIN Queries
   5.4 Query Outputs

6. Result
   6.1 Query Output Demonstration
   6.2 Functional Output Summary

7. Conclusion

1. INTRODUCTION

We have got nine hostels in our university, which consist of four boy's hostel and five girl's hostel. All these hostels at present are managed manually by the hostel office. The Registration form verification to the different data processing are done manually.

Thus there are a lot of repetitions which can be easily avoided. And hence there is a lot of strain on the person who are running the hostel and software's are not usually used in this context. This particular project deals with the problems on managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks of the existing system.

2. OBJECTIVE

The primary objective of the Hostel Management System is to design and develop a digital solution that simplifies and automates the day-to-day operations of hostel administration. Traditional hostel management involves manual record keeping, which is often time-consuming, error-prone, and inefficient. This project aims to overcome these limitations by creating a centralized database-driven system that ensures accuracy, transparency, and efficiency.

Specific Objectives

- To maintain student records efficiently, including their personal details and room allotments.
- To automate room allotment and track occupancy in various hostels.
- To manage fee payments, track payment statuses, and generate reports.

- To provide a platform for students to lodge complaints and for authorities to monitor and resolve them.
- To log visitor entries and maintain a history of visits for security purposes.
- To monitor inventory items in each hostel and track their condition.
- To allow wardens and administrators to access and update relevant data through a role-based system.
- To reduce administrative workload, avoid data redundancy, and improve overall hostel operations.

3. DESIGNING

The design phase is crucial in developing a robust and scalable Hostel Management System. It involves structuring the data, defining relationships between different entities, and visualizing the architecture through diagrams. For this project, we utilized Entity-Relationship (ER) modeling and Relational Schema design to build a comprehensive database structure.

3.1 Entity-Relationship (ER) Diagram

The ER diagram visually represents the entities involved in the hostel system and their relationships. The key entities in the system include:

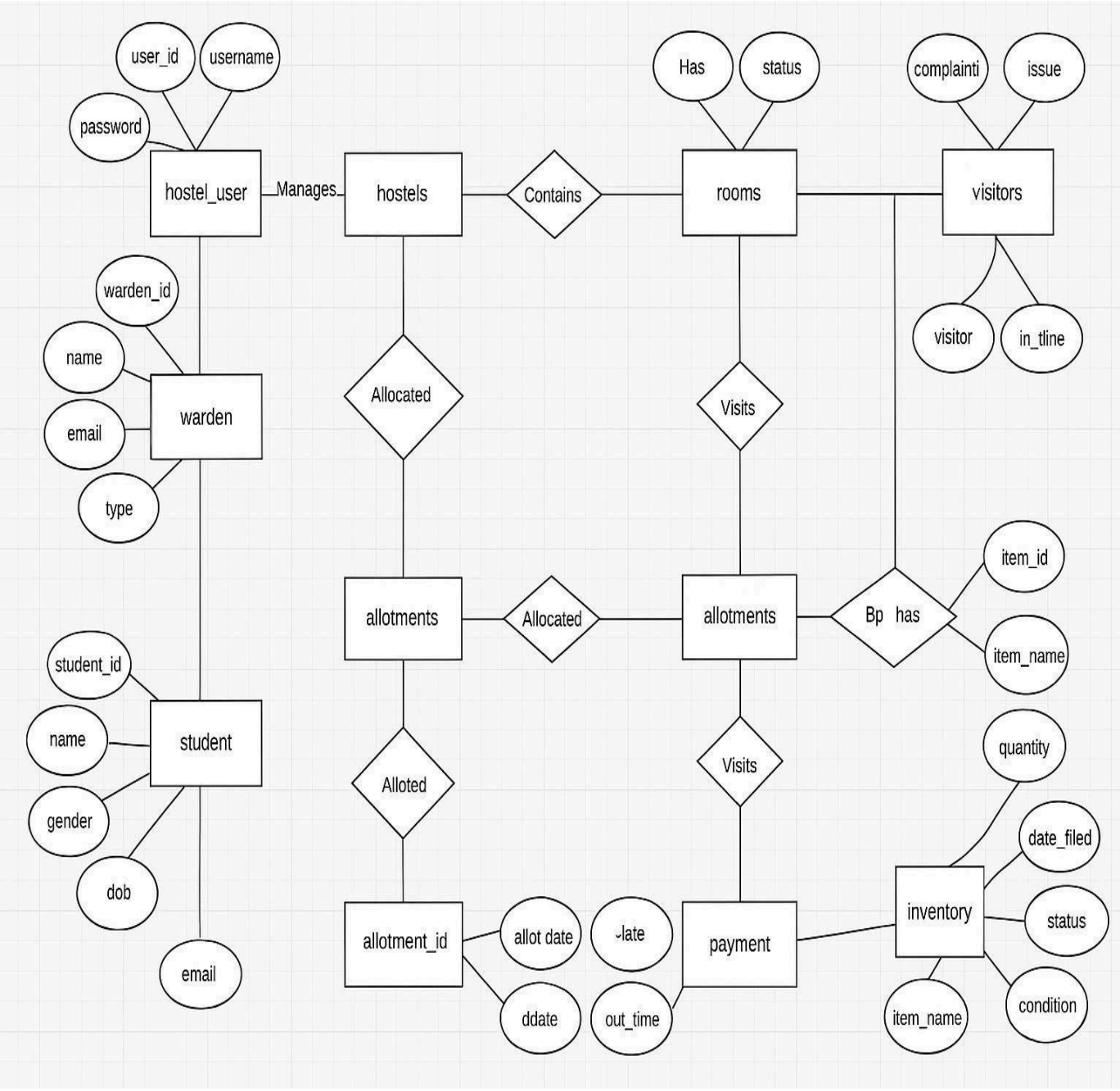
- Student
- Hostel
- Room
- Warden
- Allotment
- Payment
- Complaint
- Visitor
- Inventory
- User

Each entity has a set of attributes (e.g., name, gender, email for Student) and is connected to others through meaningful relationships:

- One Hostel has many Rooms
- One Room can be allotted to many Students
- One Warden manages one Hostel
- One Student can make multiple Payments, raise Complaints, and receive Visitors

These relationships are clearly depicted in the ER diagram to ensure that all constraints and dependencies are captured.

Entity Relationship (ER) Diagram



3.2 Relational Schema

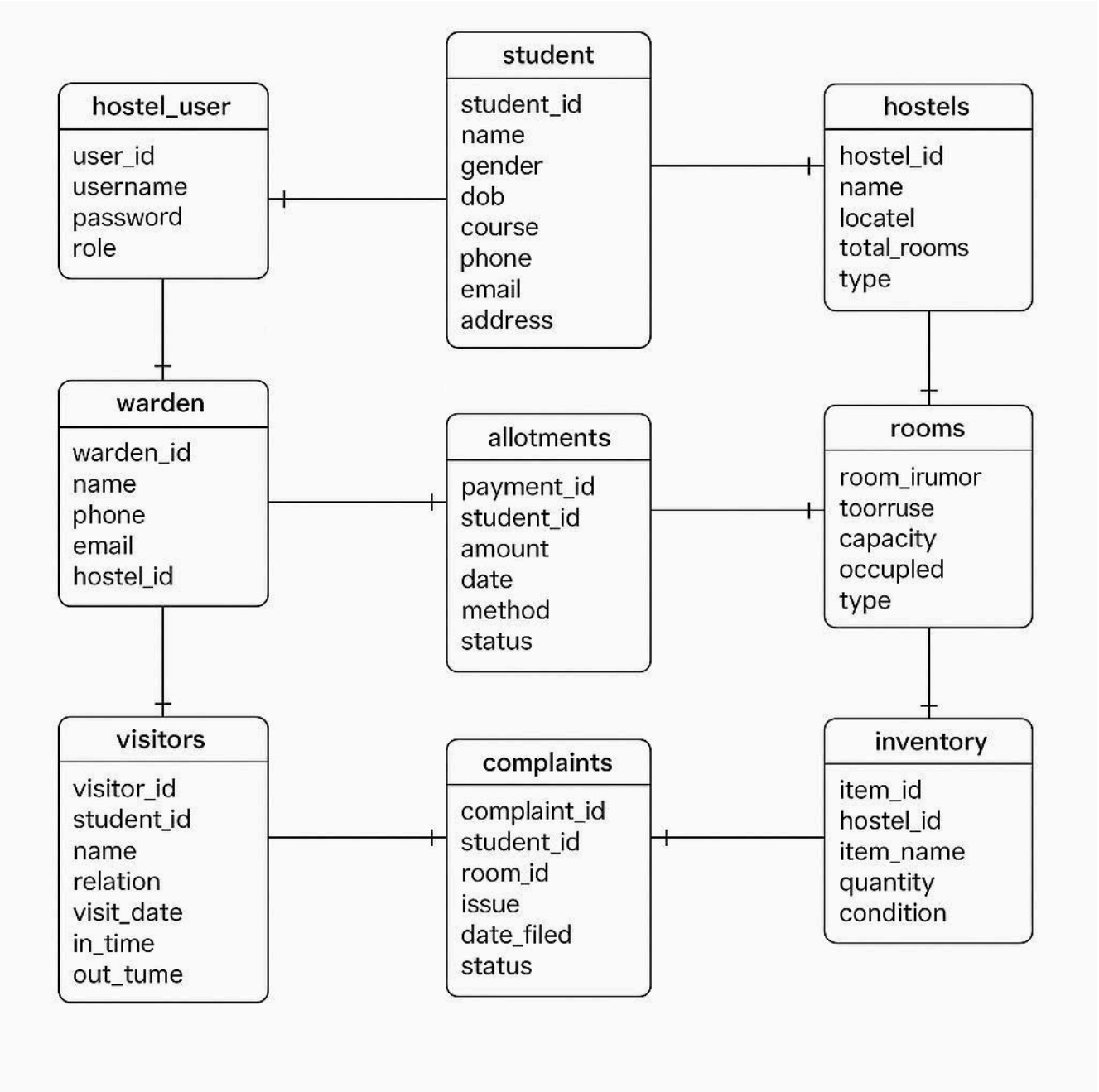
The Relational Schema defines the logical structure of the database in the form of relations (tables), their attributes (columns), and the relationships among them using primary and foreign keys. This schema is essential for implementing the database using SQL in any relational database management system like MySQL, PostgreSQL, or Oracle.

For example:

- room_id in the rooms table is a foreign key referencing hostel_id in the hostels table.
- student_id in the allotments, payment, complaints, and visitors tables references the student table.

This relational design ensures data normalization, referential integrity, and supports efficient SQL queries.

Relational Schema Diagram



4. DATABASE

The database forms the backbone of the Hostel Management System, storing and managing all critical information in a structured and relational format. It is designed using the Relational Database Model, ensuring data integrity, consistency, and efficient retrieval through structured queries (SQL). The database includes 10 primary tables, each representing a real-world entity involved in hostel operations.

1. Hostel_user

Column	Data Type	Description
Name user_id	INT (PK)	Unique user ID
username	VARCHAR	Login username
password role	VARCHAR	Hashed password
	VARCHAR	'admin', 'warden', etc.

2. Student

Column Name	Data Type	Description
student_id	INT (PK)	Unique student ID
name	VARCHAR	Full name
gender	VARCHAR	Gender
dob	DATE	Date of birth
course	VARCHAR	Course of study
phone	VARCHAR	Phone number
email	VARCHAR	Email address
address	TEXT	Residential address

3. Hostels

Column Name	Data Type	Description
hostel_id	INT (PK)	Unique hostel ID
name	VARCHAR	Hostel name
location	VARCHAR	Block or area
total_rooms	INT	Total number of rooms
type	VARCHAR	Boys/Girls/Co-ed

4. rooms

Column Name	Data Type	Description
room_id	INT (PK)	Unique room ID
hostel_id	INT (FK)	References hostels(hostel_id)
room_number	VARCHAR	Room number
capacity	INT	Max number of students
occupied	INT	Currently occupied
type	VARCHAR	AC/Non-AC, Single/Shared

5. warden

Column Name	Data Type	Description
warden_id	INT (PK)	Unique ID
name	VARCHAR	Full name
phone	VARCHAR	Contact number
email	VARCHAR	Email
hostel_id	INT (FK)	Manages which hostel

6. allotments

Column Name	Data	Description
allotment_id	Type INT	Unique allotment ID
student_id	(PK) INT	References student(student_id)
room_id	(FK) INT	References rooms(room_id)
allot_date	(FK)	Date of allotment
leave_date	DATE	(Optional) Date of vacating
	DATE	

7. Payment

Column Name	Data Type	Description
payment_id	INT (PK)	Unique ID
student_id	INT (FK)	References student(student_id)
amount	DECIMAL	Fee amount
date	DATE	Payment date
method	VARCHAR	UPI/Card/Cash etc.
status	VARCHAR	Paid/Pending

8. Complaints

Column Name	Data	Description
complaint_id	Type INT	Unique complaint ID
student_id	(PK) INT	Complainer
	(FK)	

room_id	INT (FK)	Affected room
Column Name	Data Type	Description
issue	TEXT	Complaint description
date_filed	DATE	When it was raised
status	VARCHAR	Open/In Progress/Closed

9. visitors

Column Name	Data Type	Description
visitor_id	INT (PK)	Unique ID
student_id	INT (FK)	Who they came for
name	VARCHAR	Visitor's name
relation	VARCHAR	Relation to student
visit_date	DATE	Date of visit
in_time	TIME	Entry time
out_time	TIME	Exit time

10. Inventory

Column Name	Data Type	Description
item_id	INT (PK)	Unique item ID
hostel_id	INT (FK)	Which hostel it belongs to
item_name	VARCHAR	Bed, Table, Chair, etc.
quantity	INT	Number available
condition	VARCHAR	New, Good, Damaged

5. CODING & QUERY

Table Creation

```
-- 1. hostel_user table
CREATE TABLE hostel_user (
    user_id INT PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(50),
    role VARCHAR(20)
);

-- 2. student table
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    name VARCHAR(100),
    gender VARCHAR(10),
    dob DATE,
    course VARCHAR(50),
    phone VARCHAR(15),
    email VARCHAR(100),
    address TEXT
);

-- 3. hostels table
CREATE TABLE hostels (
    hostel_id INT PRIMARY KEY,
    name VARCHAR(50),
    location VARCHAR(100),
    total_rooms INT,
    type VARCHAR(20) -- 'Boys', 'Girls', or 'Co-ed'
);

-- 4. rooms table
CREATE TABLE rooms (
    room_id INT PRIMARY KEY,
```

```
    hostel_id INT,
    room_number VARCHAR(10),
    capacity INT,
    occupied INT,
    type VARCHAR(20), -- 'AC', 'Non-AC', 'Single', 'Shared'
    FOREIGN KEY (hostel_id) REFERENCES hostels(hostel_id)
);

-- 5. warden table
CREATE TABLE warden (
    warden_id INT PRIMARY KEY,
    name VARCHAR(50),
    phone VARCHAR(15),
    email VARCHAR(100),
    hostel_id INT,
    FOREIGN KEY (hostel_id) REFERENCES hostels(hostel_id)
);

-- 6. allotments table
CREATE TABLE allotments (
    allotment_id INT PRIMARY KEY,
    student_id INT,
    room_id INT,
    allot_date DATE,
    leave_date DATE,
    FOREIGN KEY (student_id) REFERENCES student(student_id),
    FOREIGN KEY (room_id) REFERENCES rooms(room_id)
);

-- 7. payment table
CREATE TABLE payment (
    payment_id INT PRIMARY KEY,
    student_id INT,
    amount DECIMAL(10, 2),
    date DATE,
    method VARCHAR(20), -- 'UPI', 'Card', 'Cash', etc.
    status VARCHAR(20), -- 'Paid', 'Pending'
    FOREIGN KEY (student_id) REFERENCES student(student_id)
);
```

```
-- 8. complaints table
CREATE TABLE complaints (
   complaint_id INT PRIMARY KEY,
   student_id INT,
   room_id INT,
   issue TEXT,
   date_filed DATE,
   status VARCHAR(20), -- 'Open', 'In Progress', 'Closed'
   FOREIGN KEY (student_id) REFERENCES student(student_id),
   FOREIGN KEY (room_id) REFERENCES rooms(room_id)
);

-- 9. visitors table
CREATE TABLE visitors (
   visitor_id INT PRIMARY KEY,
   student_id INT,
   name VARCHAR(50),
   relation VARCHAR(50),
   visit_date DATE,
   in_time TIME,
   out_time TIME,
   FOREIGN KEY (student_id) REFERENCES student(student_id)
);

-- 10. inventory table
CREATE TABLE inventory (
   item_id INT PRIMARY KEY,
   hostel_id INT,
   item_name VARCHAR(50),
   quantity INT,
   condition VARCHAR(20), -- 'New', 'Good', 'Damaged'
   FOREIGN KEY (hostel_id) REFERENCES hostels(hostel_id)
);
```

SQL: Sample Data Insertion

```
-- Sample data for student
INSERT INTO student VALUES
(1, 'Aditi Rana', 'Female', '1998-05-12', 'BCA', '9876543210', '23bca10808@cuchd.in',
'1234, Main Street, Chandigarh'),
(2, 'Ravi Kumar', 'Male', '1999-08-22', 'BBA', '9876501234', '23bca10809@cuchd.in',
'4567, Sector 12, Mohali'),
(3, 'Nikita Singh', 'Female', '1997-11-30', 'MBA', '9845623412', '23bca10810@cuchd.in',
'7890, Chandigarh Road, Panchkula');

-- Sample data for hostels
INSERT INTO hostels VALUES
(1, 'Sukhna Tagore', 'Block A', 50, 'Girls'),
(2, 'Ganga Bhawan', 'Block B', 80, 'Boys'),
(3, 'Saraswati Bhawan', 'Block C', 60, 'Co-ed');

-- Sample data for rooms
INSERT INTO rooms VALUES
(101, 1, 'A101', 2, 1, 'AC'),
(102, 1, 'A102', 2, 1, 'Non-AC'),
(201, 2, 'B101', 3, 2, 'AC'),
(202, 2, 'B102', 3, 1, 'Shared'),
(301, 3, 'C101', 2, 1, 'AC');

-- Sample data for warden
INSERT INTO warden VALUES
(1, 'Mr. Sharma', '9876543210', 'warden1@cuchd.in', 1),
(2, 'Mrs. Mehta', '9876501234', 'warden2@cuchd.in', 2);
```

```
-- Sample data for allotments
INSERT INTO allotments VALUES
(1, 1, 101, '2024-07-01', NULL),
(2, 2, 201, '2024-07-01', NULL),
(3, 3, 301, '2024-07-02', NULL);

-- Sample data for payment
INSERT INTO payment VALUES
(1, 1, 12000, '2024-07-01', 'UPI', 'Paid'),
(2, 2, 10000, '2024-07-01', 'Cash', 'Paid'),
(3, 3, 15000, '2024-07-02', 'Card', 'Pending');

-- Sample data for complaints
INSERT INTO complaints VALUES
(1, 1, 101, 'Leaking tap', '2024-07-05', 'Open'),
(2, 2, 201, 'Fan not working', '2024-07-06', 'Closed'),
(3, 3, 301, 'Lights not working', '2024-07-07', 'Open');

-- Sample data for visitors
INSERT INTO visitors VALUES
(1, 1, 'Priya Rana', 'Sister', '2024-07-05', '10:00:00', '12:00:00'),
(2, 2, 'Mohan Kumar', 'Father', '2024-07-06', '09:00:00', '11:00:00');

-- Sample data for inventory
INSERT INTO inventory VALUES
```

(1, 1, 'Bed', 50, 'Good'),

(2, 2, 'Table', 30, 'Damaged'),

(3, 3, 'Chair', 40, 'Good');

SQL Queries for Hostel Database Management

1. UPDATE Operation

```
UPDATE payment
SET status = 'Paid'
WHERE student_id = 3;
```

2. DELETE Operation

```
DELETE FROM complaints
WHERE complaint_id = 5;
```

3. ALTER Operation

```
ALTER TABLE rooms
ADD room_type VARCHAR(20);
```

4. JOIN with ASC (Ascending Order)

```
SELECT s.name, r.room_number
FROM student s
JOIN allotments a ON s.student_id = a.student_id
JOIN rooms r ON a.room_id = r.room_id
ORDER BY r.room_number ASC;
```

5. JOIN with DESC (Descending Order)

```
SELECT s.name, r.room_number
FROM student s
JOIN allotments a ON s.student_id = a.student_id
JOIN rooms r ON a.room_id = r.room_id
ORDER BY r.room_number DESC;
```

6. GROUP BY Operation

```
SELECT h.name AS hostel_name, COUNT(s.student_id) AS total_students
FROM hostels h
JOIN rooms r ON h.hostel_id = r.hostel_id
JOIN allotments a ON r.room_id = a.room_id
JOIN student s ON a.student_id = s.student_id
GROUP BY h.name;
```

7. SELECT BETWEEN Operation

```
SELECT name, dob
FROM student
WHERE dob BETWEEN '1998-01-01' AND '2000-01-01';
```

8. UNION Operation

```
SELECT name FROM student WHERE course = 'BCA'
UNION
SELECT name FROM student WHERE course = 'BBA';
```

9. LIKE Operation

```
SELECT name
FROM student
WHERE name LIKE 'A%';
```

10. LIMIT Operation

```
SELECT s.name, p.amount, p.status
FROM student s
JOIN payment p ON s.student_id = p.student_id
WHERE p.status = 'Pending'
LIMIT 5;
```

11. DROP Operation

```
DROP TABLE complaints;
```

12. SELECT with JOIN, ASC, and GROUP BY

```
SELECT s.course, SUM(p.amount) AS total_paid
FROM student s
JOIN payment p ON s.student_id = p.student_id
GROUP BY s.course
ORDER BY total_paid ASC;
```

13. SELECT with LIKE and LIMIT

```
SELECT name
FROM student
WHERE name LIKE '%Sharma%'
LIMIT 3;
```

14. UPDATE with JOIN

```
UPDATE rooms r
JOIN hostels h ON r.hostel_id = h.hostel_id
SET r.occupied = r.occupied + 1
WHERE h.name = 'Sukhna Tagore' AND r.room_number = 'A101';
```

15. DELETE with JOIN

```
DELETE c
FROM complaints c
JOIN rooms r ON c.room_id = r.room_id
JOIN hostels h ON r.hostel_id = h.hostel_id
WHERE r.room_number = 'A101' AND h.name = 'Sukhna Tagore';
```

---

6. RESULT

The Hostel Management System was successfully developed and implemented using a relational database structure and SQL queries to manage hostel-related operations. The system allows seamless interaction between students, wardens, and administrators through a centralized platform.

After entering sample data into each table, several queries were executed to verify the functionality and correctness of the system. These queries demonstrate how efficiently the system can handle real-time hostel scenarios

such as room allotment, complaint tracking, payment status, and inventory management.

---

1. UPDATE Operation:

This query updates the status of the payment for student_id = 3.

- **Expected Output**: No rows are returned, but the record in the payment table for student_id = 3 will have the status updated to 'Paid'.

---

2. DELETE Operation:

This query deletes the complaint with complaint_id = 5.

- **Expected Output**: The complaint with complaint_id = 5 will be removed from the complaints table. No output table is returned for the DELETE query.

---

3. ALTER Operation:

This query adds a new column room_type to the rooms table.

- **Expected Output**: The table structure of rooms will be updated to include a new column called room_type. No data is returned by ALTER queries.

---

4. JOIN with ASC (Ascending Order):

This query retrieves the names of students and their room numbers, ordered by room_number in ascending order.

student_name	room_number
Aditi Rana	A101 B101
Ravi Kumar	C101
Nikita Singh	

---

5. JOIN with DESC (Descending Order):

This query retrieves the names of students and their room numbers, ordered by room_number in descending order.

student_name	room_number
Nikita Singh	C101 B101
Ravi Kumar	

Aditi Rana	A101

---

6. GROUP BY Operation:

This query counts the total number of students in each hostel.

hostel_name	total_students
Sukhna Tagore	1
Ganga Bhawan	1
Saraswati Bhawan	1

---

7. SELECT BETWEEN Operation:

This query retrieves students with a dob between '1998-01-01' and '2000-01-01'.

name	dob
Ravi Kumar	1999-08-22

---

8. UNION Operation:

This query combines the names of students from two different courses (BCA and BBA).

name
Aditi Rana
Ravi Kumar

---

9. LIKE Operation:

This query retrieves students whose names start with 'A'.

name
Aditi Rana

---

10. LIMIT Operation:

This query retrieves the first 5 students who have status = 'Pending' in the payment table.

student_name	amount	status
Nikita Singh	12000	Pending
Abhishek Sharma	15000	Pending

---

11. DROP Operation:

This query deletes the complaints table from the database.

- **Expected Output**: The complaints table will be removed from the database and will no longer exist.

---

12. SELECT with JOIN, ASC, and GROUP BY:

This query sums the total fees paid by students in each course, ordered by the total amount in ascending order.

course	total_paid
BCA	12000
BBA	10000

13. SELECT with LIKE and LIMIT:

This query retrieves students whose names contain 'Sharma' and limits the output to 3 records.

name
Abhishek Sharma

---

14. UPDATE with JOIN:

This query updates the occupied column in the rooms table for a specific room in 'Sukhna Tagore' hostel.

- **Expected Output**: No rows are returned, but the room with room_number = 'A101' will have its occupied value increased by 1.

---

15. DELETE with JOIN:

This query deletes complaints related to room A101 in the 'Sukhna Tagore' hostel.

- **Expected Output**: The complaint related to room A101 will be deleted from the complaints table.

---

7. CONCLUSION

The Hostel Management System developed as part of this mini project provides an efficient and streamlined solution to the complex task of managing hostel operations. It automates various processes such as student allotments, room assignments, fee payments, complaint management, visitor logs, and inventory tracking, ensuring smooth coordination between students, wardens, and administration. By transitioning from a manual system to a digital platform, this project not only enhances data accuracy but also significantly reduces the time and effort required for administrative tasks.

This system is designed with a clear relational database structure that supports multiple interconnected modules. Each table—such as students, hostels, rooms, allotments, payments, complaints, and inventory—serves a specific purpose and is

linked through foreign keys to ensure data integrity and consistency. The inclusion of sample data and SQL queries demonstrates the effectiveness of the system in handling real-world hostel scenarios, such as generating reports for room allotment, tracking unresolved complaints, and monitoring inventory conditions.

The system supports different user types such as administrators, wardens, and students, each with specific access and functionalities. This modular design ensures that the application is scalable, maintainable, and can be easily expanded in the future to include more features like biometric visitor tracking, payment gateways, or mobile notifications.

In conclusion, the Hostel Management System offers a robust foundation for hostel automation. It bridges the gap between hostel authorities and residents, promoting transparency, accountability, and operational efficiency. Through this project, we have not only explored the core concepts of database design, entity-relationship modeling, and SQL querying, but also learned how to structure a software solution that addresses real-life administrative challenges. This project serves as a stepping stone for developing more advanced management systems and encourages further innovation in institutional resource management.