**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering, Mumbai**

A.Y. 2023 - 24

**Course: Database Management Systems**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program | MBA Tech Data Science | |
| Semester | 4 | |
| Name of the Project: | Hostel Room Allocation System | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| J3 | S061 | Sumit Patil |
| J3 | S062 | Swayam V. |
|  |  |  |
| Date of Submission: | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| S061 | Sumit Patil | Equal |
| S062 | Swayam V. | Equal |

**Github link of your project:** [sumitpatil95/DBMS-Major-Project: DBMS Major Project on Hostel Room Allocation System (github.com)](https://github.com/sumitpatil95/DBMS-Major-Project)

**Note:**

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004\_DBMSProject)
3. Submit all relevant files of your work ( Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

**Rubrics for the Project evaluation:**

|  |  |
| --- | --- |
| First phase of evaluation:  Innovative Ideas (5 Marks)  Design and Partial implementation (5 Marks) | 10 marks |
| Final phase of evaluation  Implementation, presentation and viva, Self-Learning and Learning Beyond classroom | 10 marks |

**Project Report**

**Hostel Room Allocation System**

**by**

**Sumit Patil, Roll number: S061**

**Swayam V., Roll number: S062**

**Course: DBMS**

**AY: 2023-24**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline | 4 |
| **2** | Components of Database Design | 4-6 |
| **3** | Entity Relationship Diagram | 7 |
| **4** | Relational Model | 7-8 |
| **5** | Normalization | 8-11 |
| **6** | SQL Queries | 11-24 |
| **7** | Learning from the Project | 25 |
| **8** | Project Demonstration | 24 |
| **9** | Self-learning beyond classroom | 24-25 |
| **10** | Learning from the project | 25 |
| **8** | Challenges faced | 25 |
| **9** | Conclusion | 26 |

**I. Storyline**

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document is narrative and graphical documentation of the software design for the project including ER Diagrams, Conceptual Schema, and other supporting requirement information. The purpose of this is to deal with Hostel Management System in an easy and an efficient manner. And create strong and secrete database that allows for any connection in a secret way to prevent any outside or inside attacks. Hostel Management System is designed for Hostel (like schools, Universities).

**II. Components of Database Design**

So,

This section of the document explains the entities used in the project, their attributes and how they will work together. Basically, this is intended to make the design more easy and understandable for everyone. Entities

1. Hostel

2. Administrator

3. Student

4. Room

5. Visitor

6. Fees

**1. Hostel**

An Institution has many hostels and each hostel is represented using this ‘Hostel’ entity. Hostel model takes part in the following relationships.

1. Administrator manages Hostel​.

2. Hostel​ has Students.

3. Hostel​ has Rooms.

Attributes

Name Data Type Type

Hostel\_ID integer Primary Key attribute

Hostel\_name string Non\_key attribute

No\_of\_rooms integer Non\_key attribute

no\_of\_students integer Non\_key attribute

**2. Administrator**

Every hostel has an administrator and is represented using the ‘administrator’

entity. Administrator entity takes part in following relationships.

1. Administrator​ manages Hostel.

Attributes

Name DataType Type

ID integer Primary Key attribute

Fname string Non\_key attribute

Lname string Non\_key attribute

Mob\_No string Non\_key attribute

Hostel\_id integer Foreign Key attribute

**3. Student**

Every hostel has students and they are represented by the ‘student’ entity.

Student entity participates in the following relationships.

1. Hostel has Students​.

2. Student ​has visitor.

3. Students​ stay at room

Attributes

Name Data Type Type

Student\_ID integer Primary Key attribute

Fname string Non\_key attribute

Lname string Non\_key attribute

Mob\_No string Non\_key attribute

Dept string Non\_key attribute

Year\_of\_study integer Non\_key attribute

Hostel\_id integer Foreign Key attribute

Room\_id integer Foreign Key attribute

**4. Room**

Every Hostel has rooms and they are represented using ‘room’ entity. Room

entity participates in the following relationships.

1. Hostel has Rooms​.

2. Student stays at room​.

3. Room​ has Furniture.

Attributes

Name Data Type Type

Hostel\_ID integer Foreign Key attribute

Room\_ID integer Partial Key attribute

**5. Visitors**

Every student has visitors and they are represented using ‘Visitor’ entity. Visitor

entity participates in the following relationships.

1. Student has visitors​.

Attributes

Name Data Type Type

Visitor\_ID integer Primary Key attribute

In time Date-time field Non\_key attribute

Out time Date-time field Non\_key attribute

Date Date-time field Non\_key attribute

Name string Non\_key attribute

Student\_id integer Foreign Key attribute

**6. Fees**

Every Student need to pay particular amount of fees accordingly.

1. Student has Fees

Attributes

Name Data Type Type

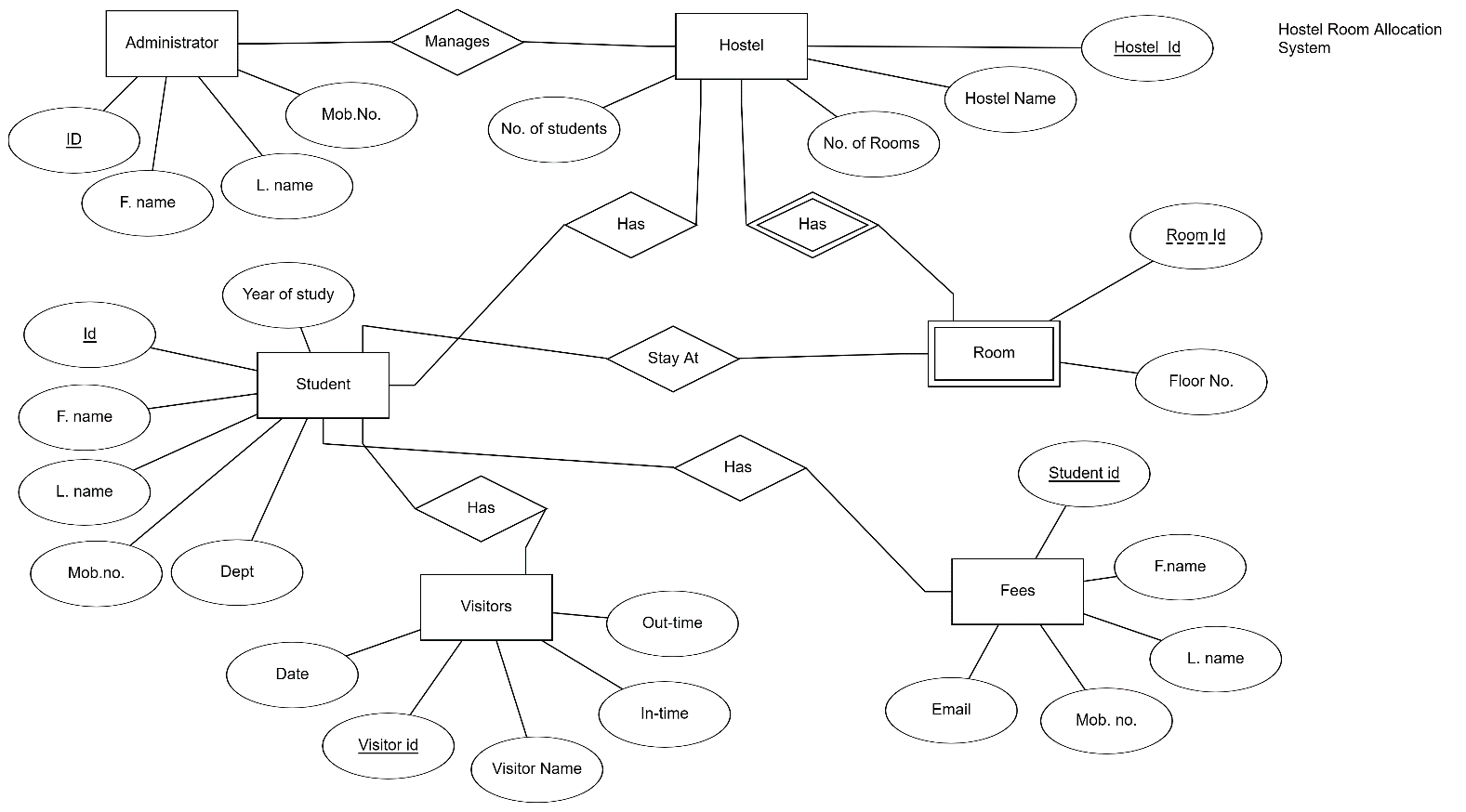
Student Id integer Primary Key attribute

First name string Non\_key attribute

Last name string Non\_key attribute

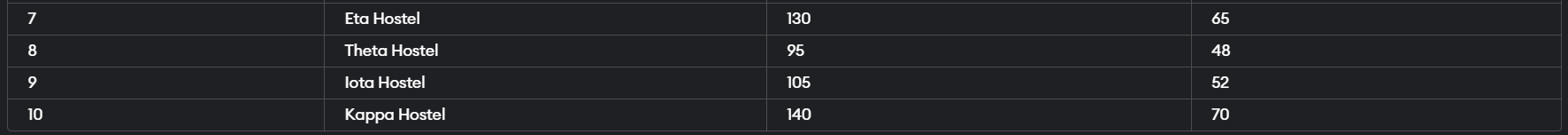
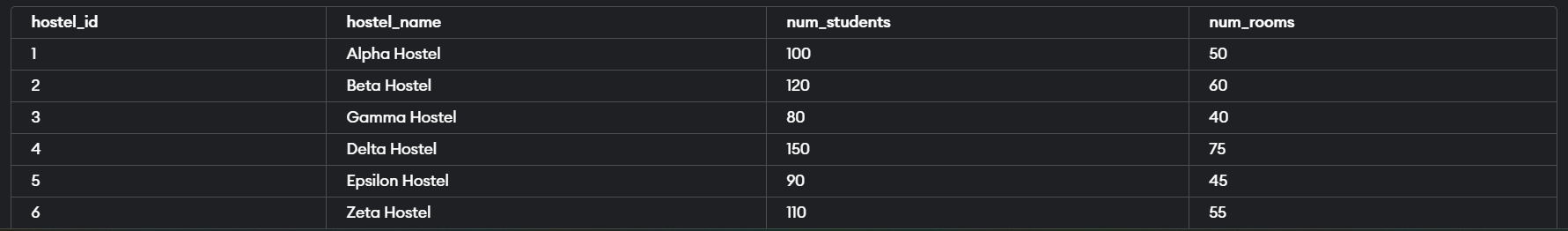
Mobile no. string Non\_key attribute

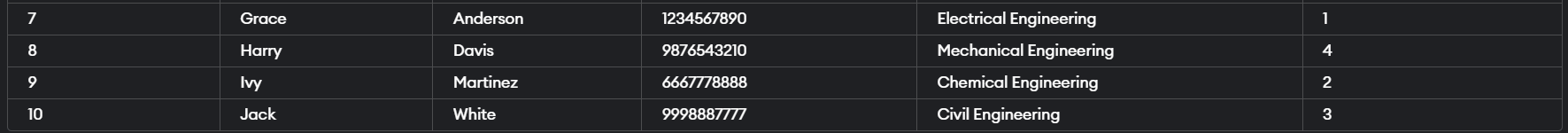
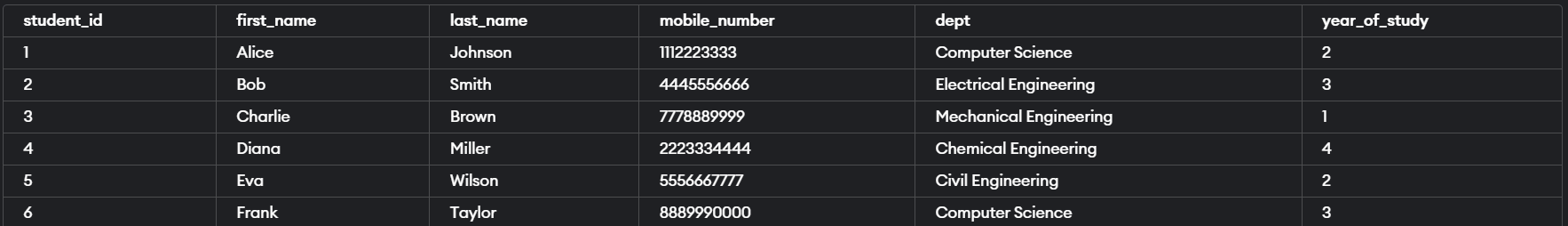
**III. Entity Relationship Diagram**

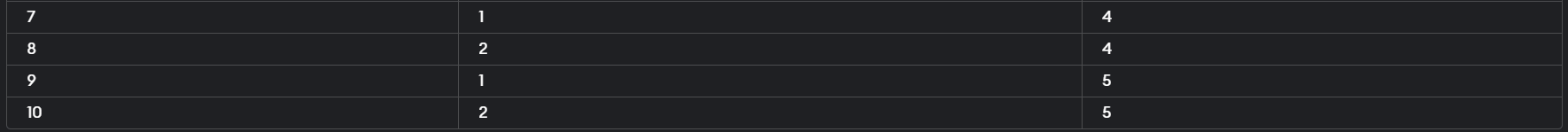
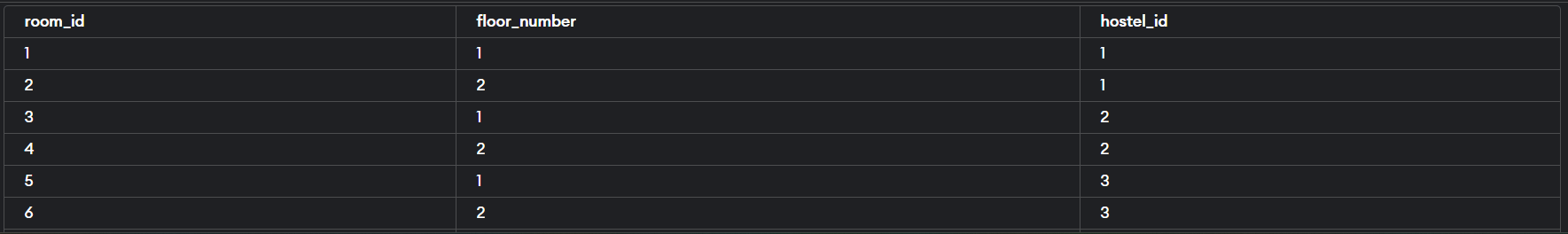


**IV. Relational Model**

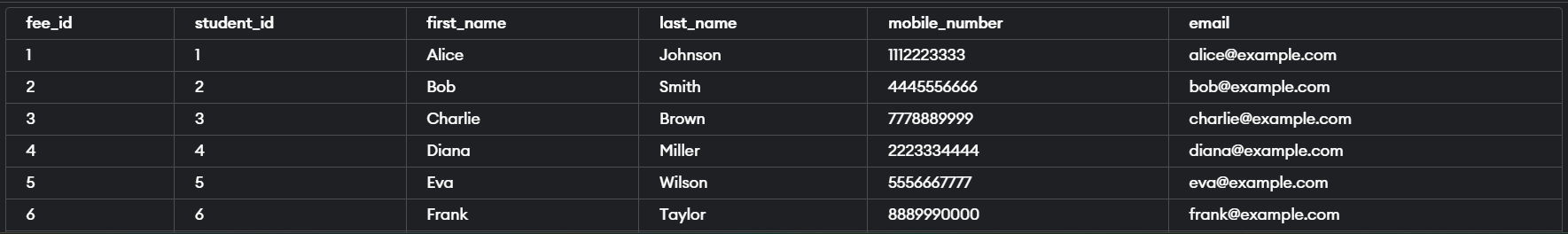












**V. Normalization**

Till 3NFof the project

Step 1: Analyze Dependencies

Before normalization, let's analyze the functional dependencies in the provided tables:

Administrator:

admin\_id -> first\_name, last\_name, mobile\_number

Hostel:

hostel\_id -> hostel\_name, num\_students, num\_rooms

Student:

student\_id -> first\_name, last\_name, mobile\_number, dept, year\_of\_study

Room:

room\_id -> floor\_number, hostel\_id

Visitors:

visitor\_id -> date, visitor\_name, in\_time, out\_time

Fees:

fee\_id -> student\_id, first\_name, last\_name, mobile\_number, email

Step 2: Normalize Tables

We will normalize the tables to third normal form (3NF), which means:

Eliminate repeating groups by putting each in a separate table.

Create separate tables for sets of values that apply to multiple records.

Remove dependencies on non-key attributes.

Based on the dependencies, here's how we can normalize the tables:

Table 1: Administrator

Attributes: admin\_id (PK), first\_name, last\_name, mobile\_number

This table appears to be in 3NF, as there are no repeating groups, and each attribute is dependent on the primary key.

Table 2: Hostel

Attributes: hostel\_id (PK), hostel\_name, num\_students, num\_rooms

This table is also in 3NF, as each attribute is dependent on the primary key.

Table 3: Student

Attributes: student\_id (PK), first\_name, last\_name, mobile\_number, dept, year\_of\_study

This table is in 3NF as well.

Table 4: Room

Attributes: room\_id (PK), floor\_number, hostel\_id (FK)

This table is in 3NF.

Table 5: Visitors

Attributes: visitor\_id (PK), date, visitor\_name, in\_time, out\_time

This table is in 3NF.

Table 6: Fees

Attributes: fee\_id (PK), student\_id (FK), email

A separate table should be created for student details to avoid redundancy:

Table 7: Student\_Details

Attributes: student\_id (PK), first\_name, last\_name, mobile\_number, dept, year\_of\_study

Updated Tables:

Table 1: Administrator

Attributes: admin\_id (PK), first\_name, last\_name, mobile\_number

Table 2: Hostel

Attributes: hostel\_id (PK), hostel\_name, num\_students, num\_rooms

Table 3: Student

Attributes: student\_id (PK), first\_name, last\_name, mobile\_number, dept, year\_of\_study

Table 4: Room

Attributes: room\_id (PK), floor\_number, hostel\_id (FK)

Table 5: Visitors

Attributes: visitor\_id (PK), date, visitor\_name, in\_time, out\_time

Table 6: Fees

Attributes: fee\_id (PK), student\_id (FK), email

Table 7: Student\_Details

Attributes: student\_id (PK), first\_name, last\_name, mobile\_number, dept, year\_of\_study

These tables are now normalized up to 3NF.

BCNF of the project

To perform Boyce-Codd Normal Form (BCNF) normalization on the tables, we need to ensure that for every functional dependency

→

X→Y, where

X is a superkey, the determinant

X is a candidate key. If any non-trivial functional dependencies violate this condition, we decompose the table accordingly.

Let's analyze each table and normalize it to BCNF:

Administrator Table

There are no non-trivial functional dependencies in this table, and the primary key

admin\_id

admin\_id uniquely identifies each tuple. Therefore, the table is already in BCNF.

Hostel Table

There are no non-trivial functional dependencies in this table, and the primary key

hostel\_id

hostel\_id uniquely identifies each tuple. Therefore, the table is already in BCNF.

Student Table

No non-trivial functional dependencies are present in this table. The primary key

student\_id

student\_id uniquely identifies each tuple. Hence, the table is in BCNF.

Room Table

The only functional dependency present is

hostel\_id

→

floor\_number

hostel\_id→floor\_number, where

hostel\_id

hostel\_id is not a superkey.

To normalize to BCNF, we decompose the table:

As room\_id and floor\_number (Room\_details) & room\_id and hostel\_id (Room1)

Visitors Table

There are no non-trivial functional dependencies in this table, and the primary key

visitor\_id

visitor\_id uniquely identifies each tuple. Therefore, the table is already in BCNF.

Fees Table

The only functional dependency present is

student\_id

→

first\_name

,

last\_name

,

mobile\_number

,

email

student\_id→first\_name,last\_name,mobile\_number,email, where

student\_id

student\_id is a superkey. Therefore, the table is already in BCNF.

After normalization, we have the following tables:

Administrator (BCNF)

Hostel (BCNF)

Student (BCNF)

Room (Decomposed into Room and Room\_Details)

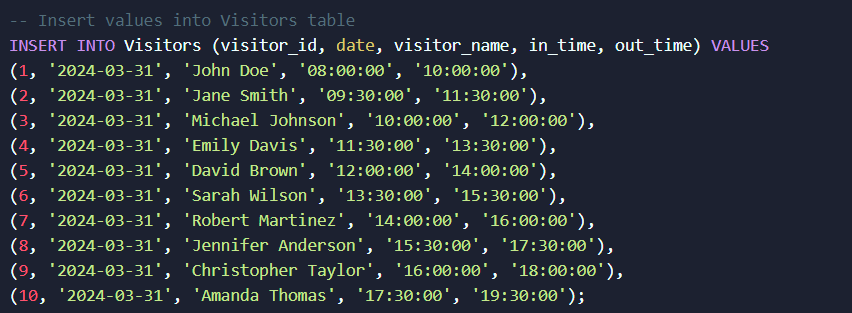
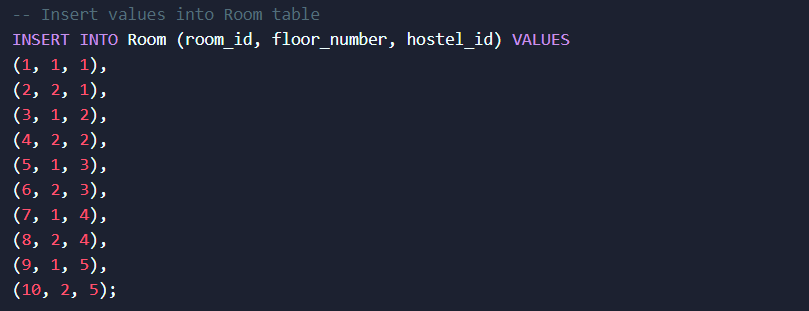
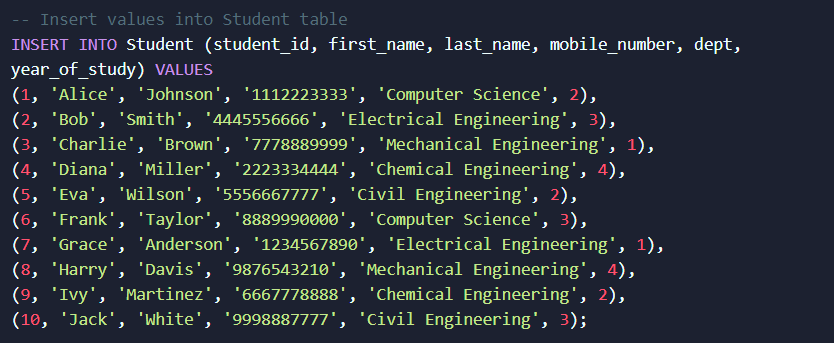
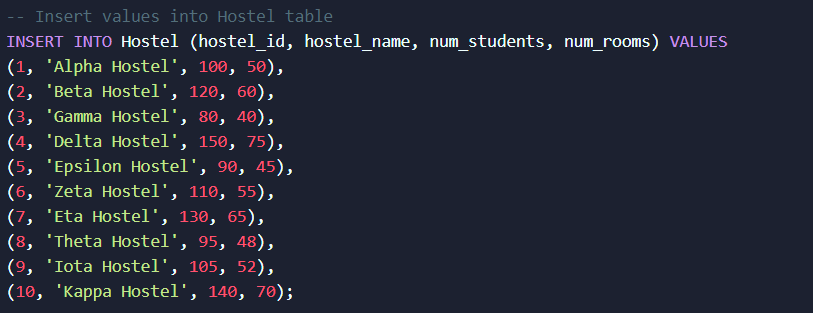
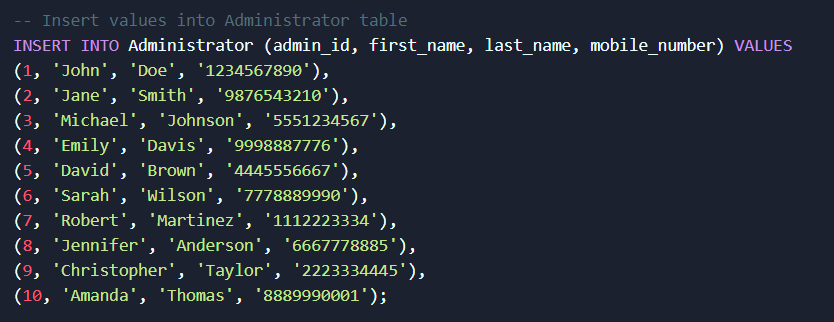
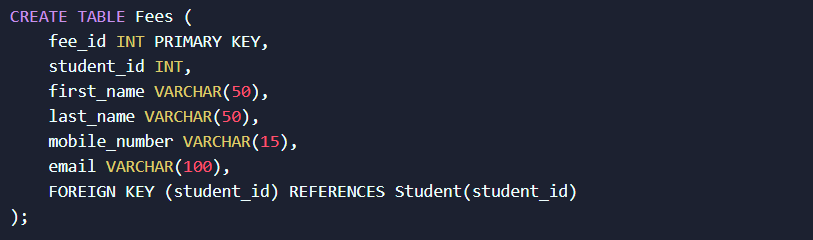
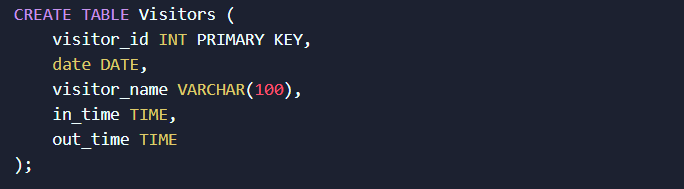
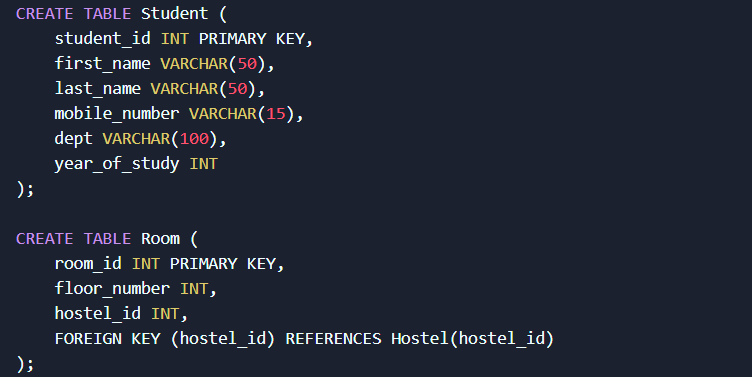
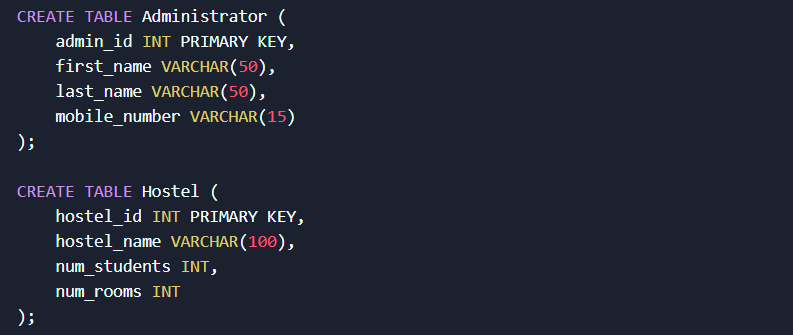
Visitors (BCNF)

Fees (BCNF)

Each table now satisfies the requirements of Boyce-Codd Normal Form.

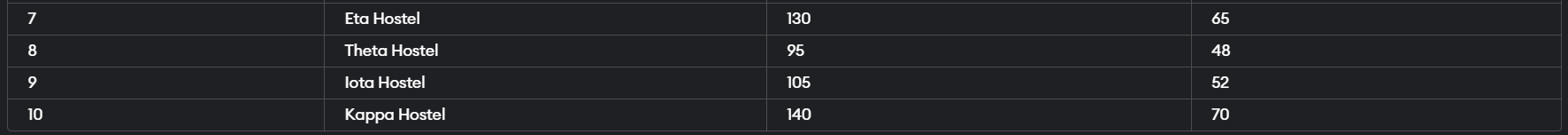
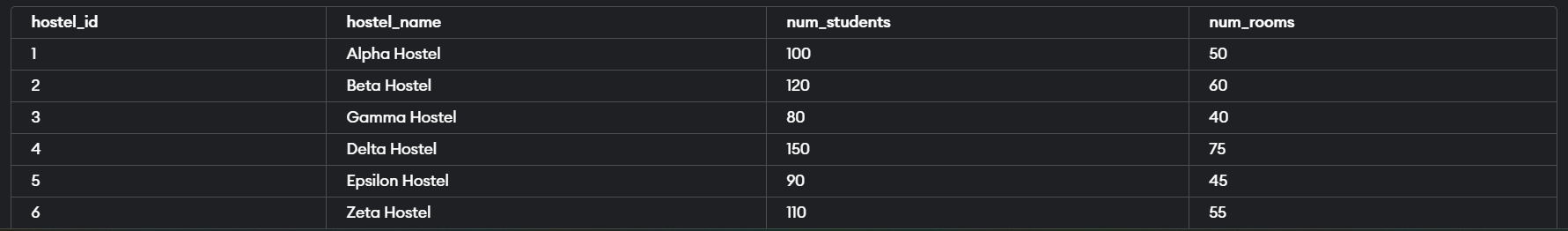
**VI. SQL Queries**

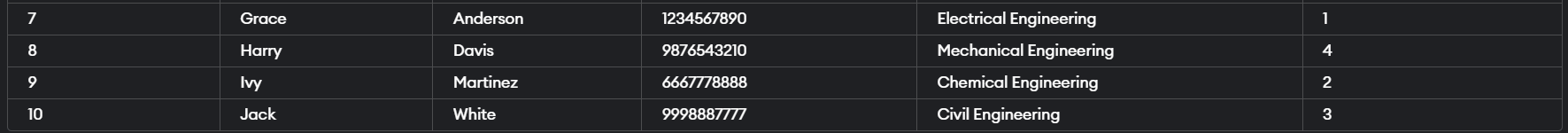
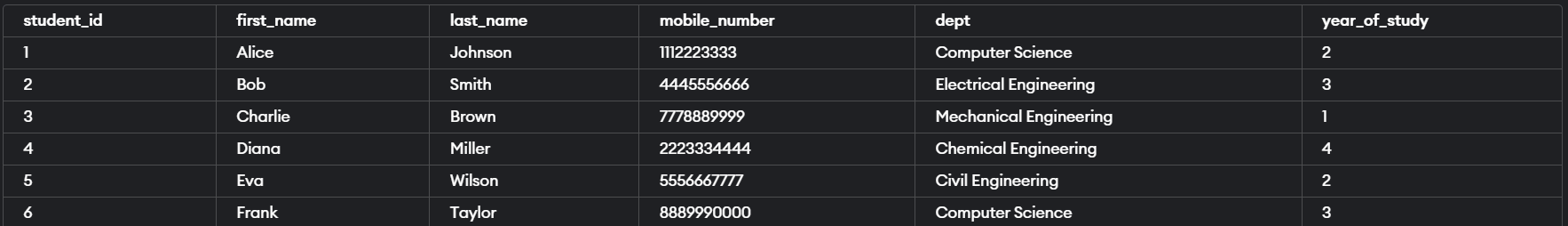
1. Creating tables and inserting values and output

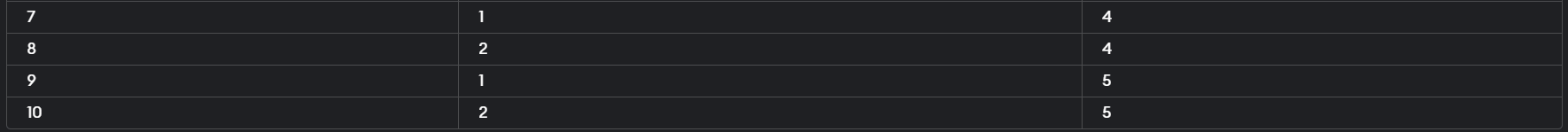
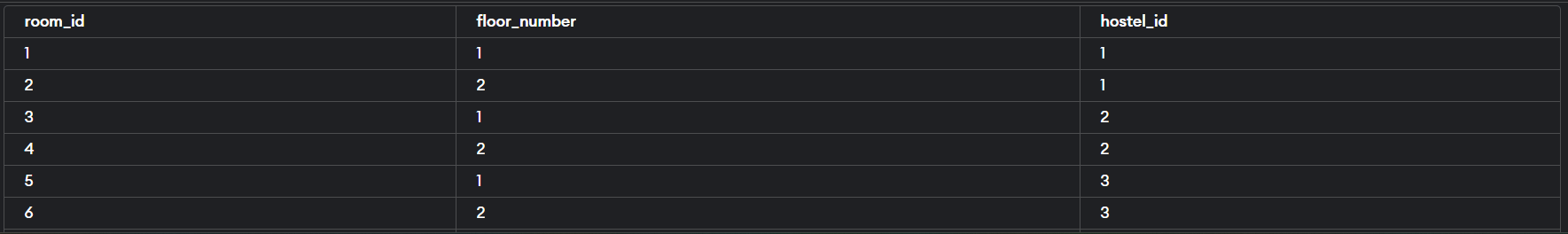


OUTPUT:







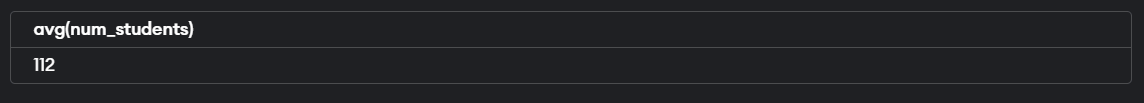




1. Finding the average number of student throughout the hostel



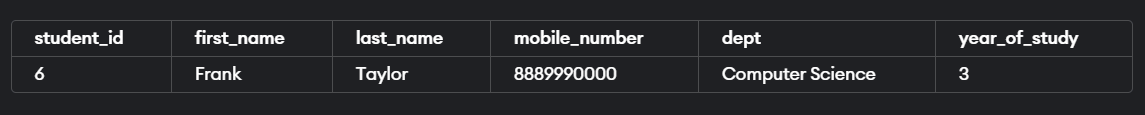
OUTPUT:



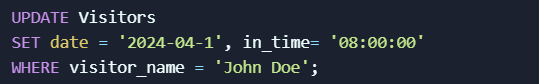
2.Finding the information of student ID 6



OUTPUT:



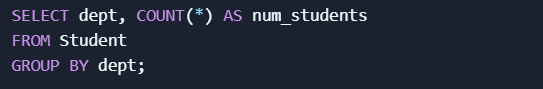
3.Updating the visitor, date and time



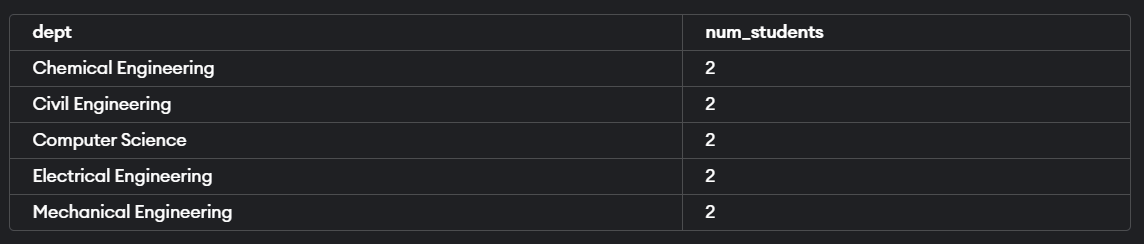
OUTPUT:



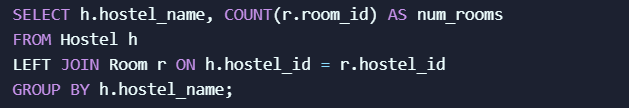
4. count no. of students in each dept



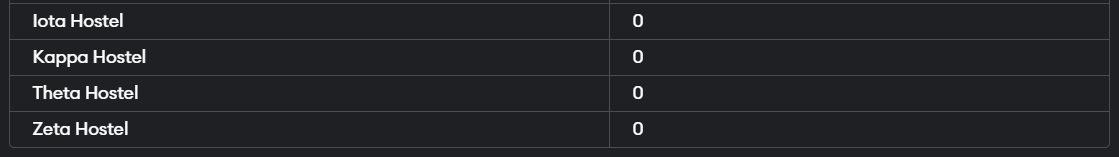
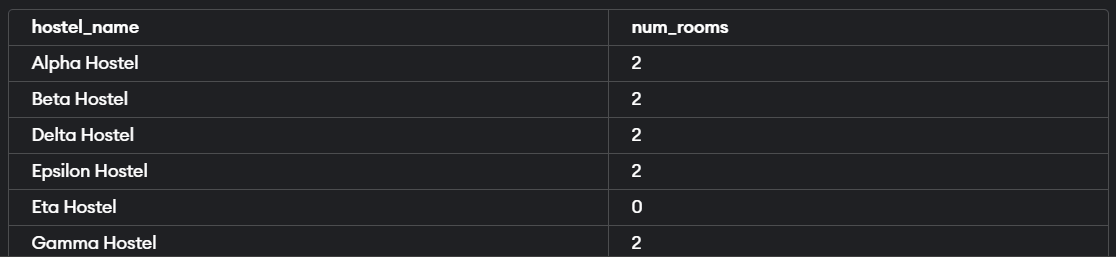
OUTPUT:



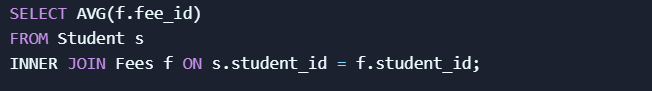
5. left join applied on hostel and room tables to see which hostels have room



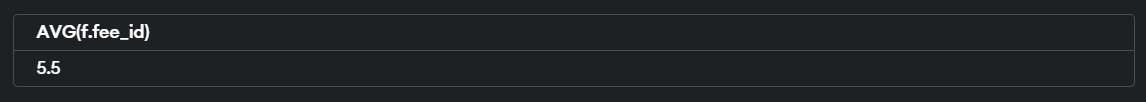
OUTPUT:



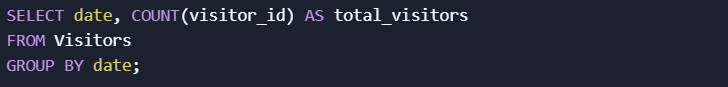
6. inner join between student and fees to get avg fee paid by student



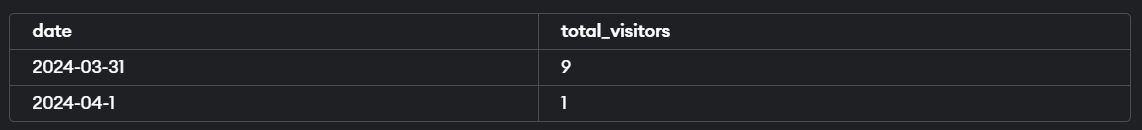
OUTPUT:



7. total no. of visitors for each date



OUTPUT:



8. Deleting admin’s ID of 7



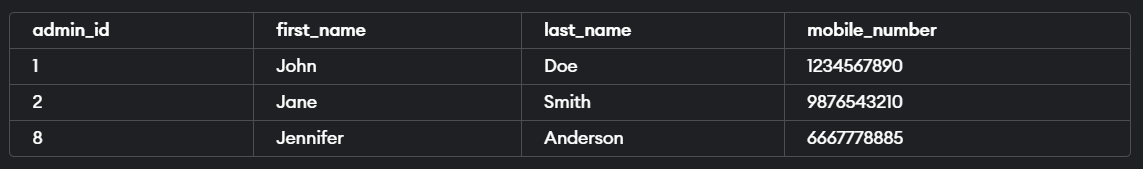
OUTPUT:



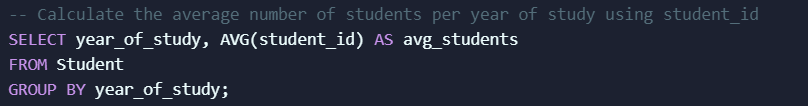
9. Getting administrator whose first name starts with “J”



OUTPUT:

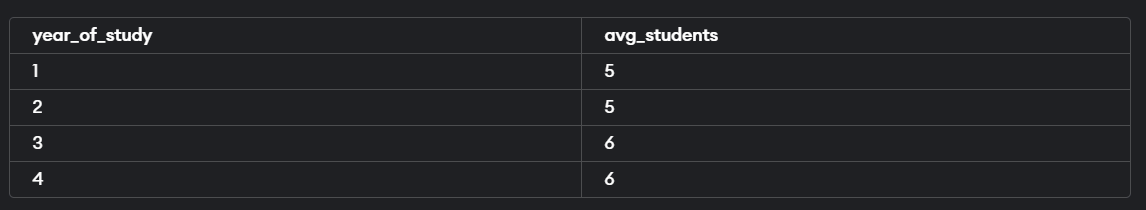


10.

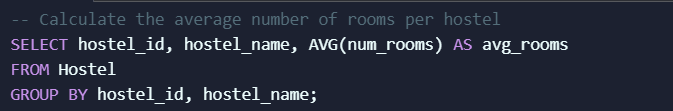


10.

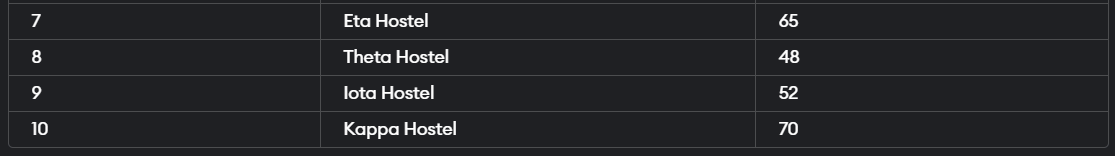
OUTPUT:



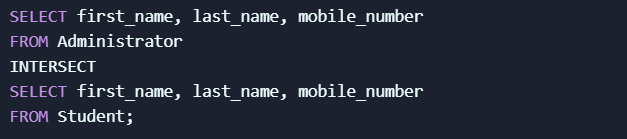
11.



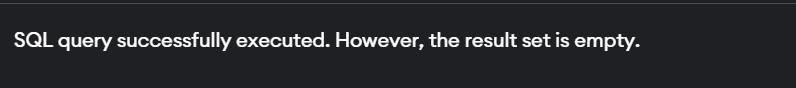
OUTPUT:



12. query returning administrators as students by finding common records between administrator and student tables



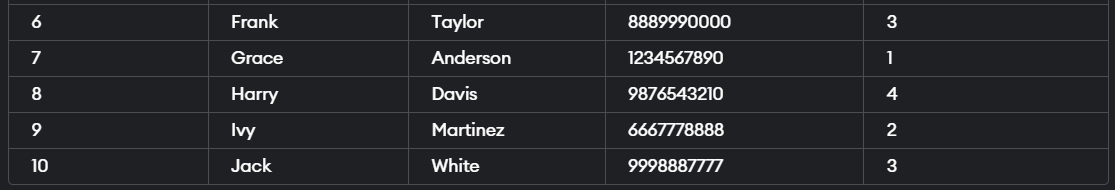
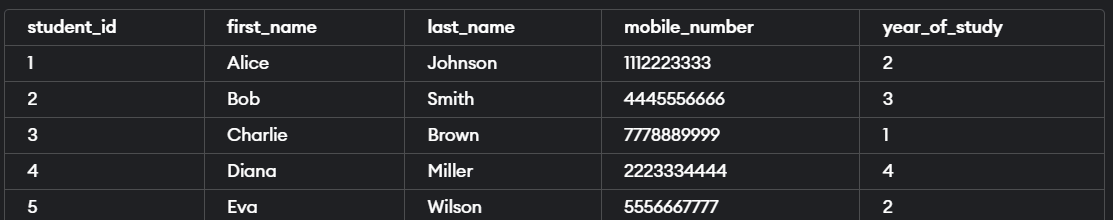
OUTPUT:



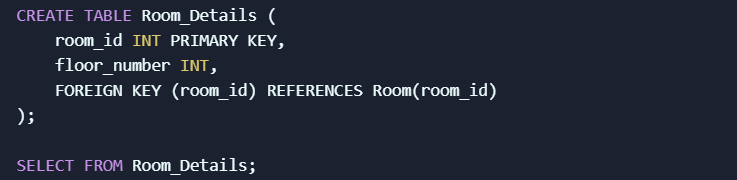
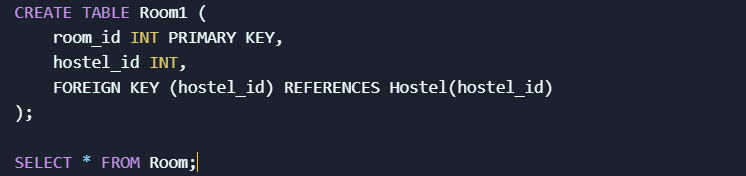
13. Deleting column of department in student table



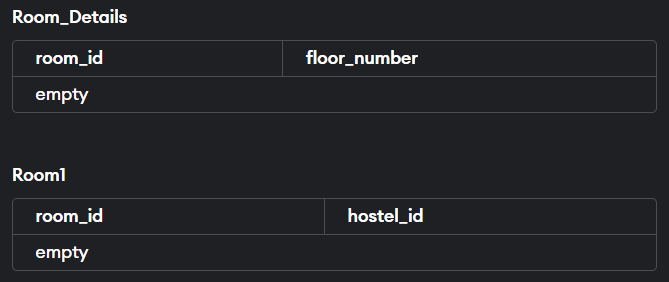
OUTPUT:



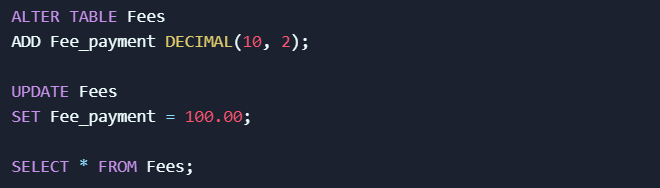
14. Decompose the Table Room as:

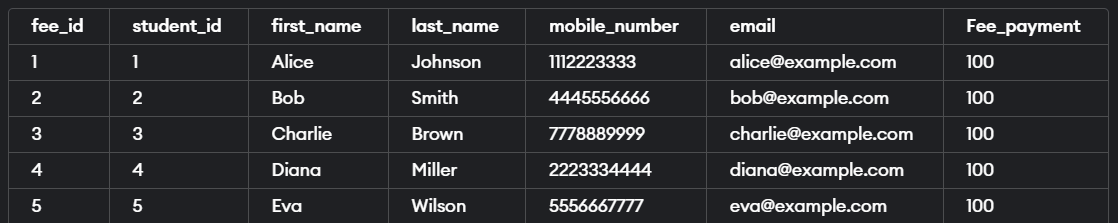
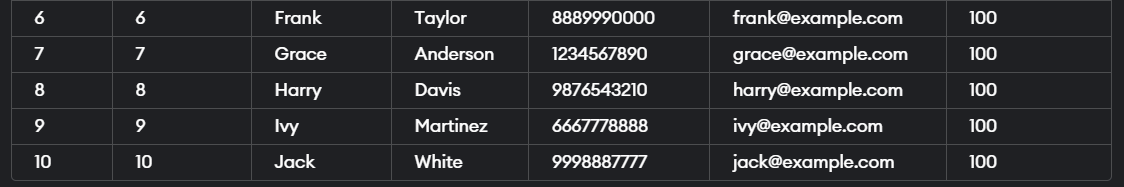
OUTPUT:



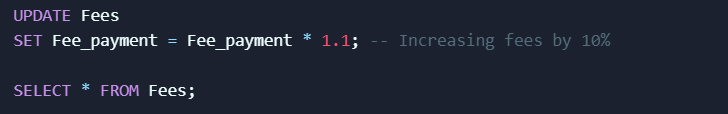
15. if fees is increased by 10% then:



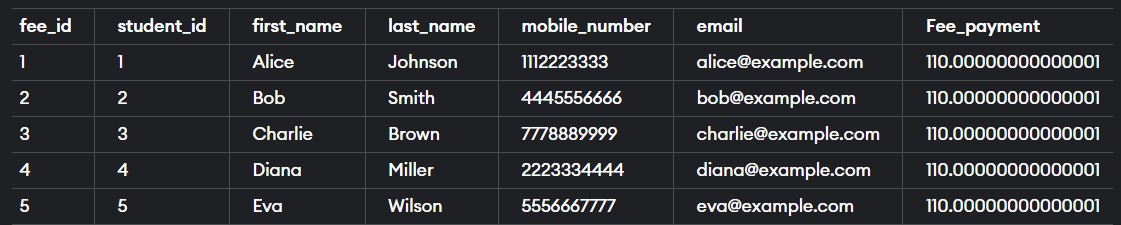
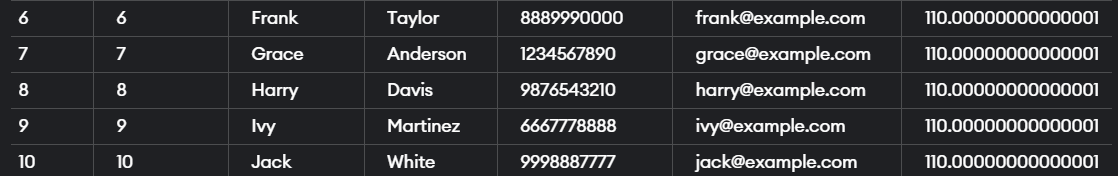
OUTPUT:

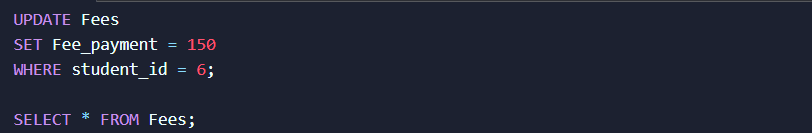
INPUT FOR MORE 10%:



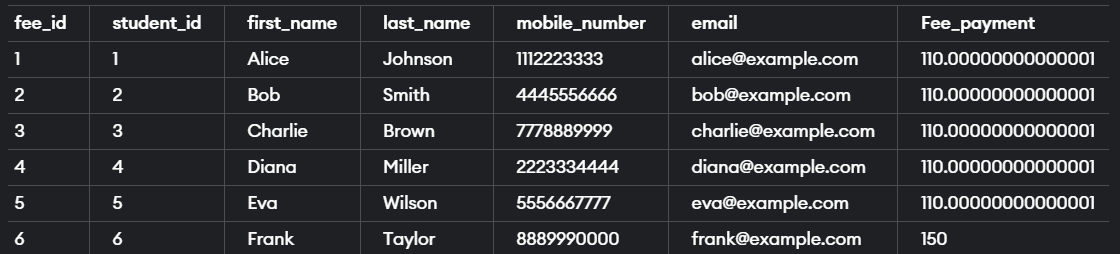
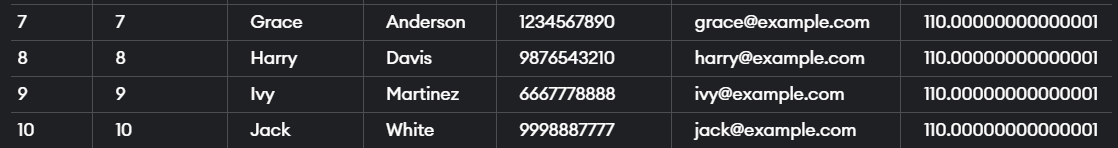
FINAL OUTPUT:

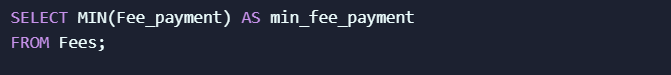
16. student on fee\_id = 6 has not paid 40 rupees, update the query on fees table:



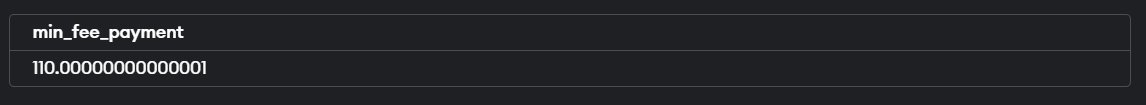
OUTPUT:

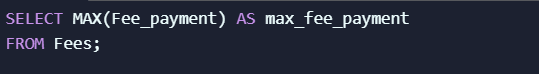
17. Find minimum fees:



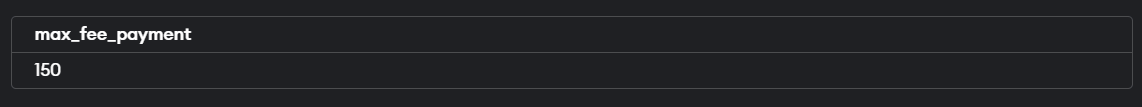
OUTPUT:



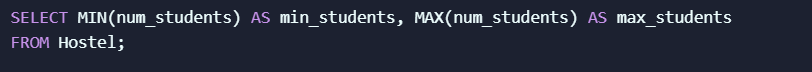
18. Find maximum fees:



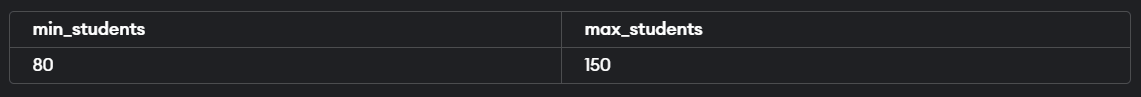
OUTPUT:



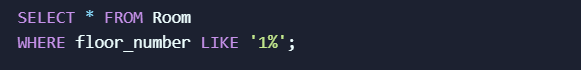
19. Find maximum number and minimum number of students in hostels:



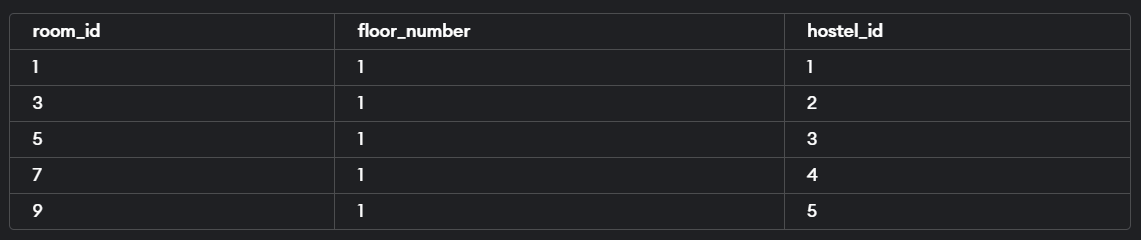
OUTPUT:



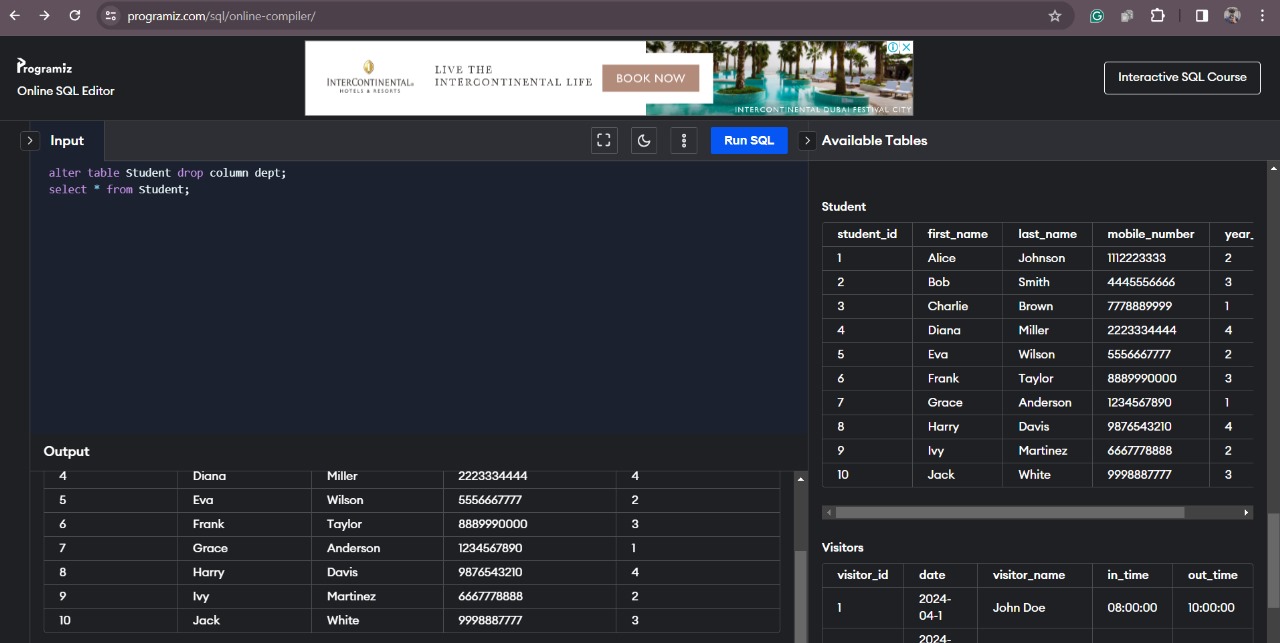
20. which hostel\_id contains floor number 1:



OUTPUT:



**VI. Project demonstration**



* **Software used is Programiz editor for sql**
* **It is an open source relational database management system(RDBMS).**
* **It helps to design, develop and manage MySQL databases.**
* **It uses structured query language to manage and manipulate data stored in tables.**

**VII. Self -Learning beyond classroom**

So, in this beyond the class I learned about Advanced Query Optimization Techniques which teaches us about learning that covers basic query optimization strategies, individuals can delve deeper into advanced optimization techniques such as indexing strategies, query execution plans, and performance tuning for specific DBMS engines like MySQL, PostgreSQL, Oracle, or SQL Server

And about the Cloud Database Services which talks about the cloud-based DBMS platforms (e.g., Amazon RDS, Google Cloud SQL, Azure SQL Database) can be valuable, including understanding cloud-specific features, scalability options, data backup and recovery strategies, and cost optimization techniques.

And also, about to show a ER Diagram in a proper manner.

**VIII. Learning from the Project**

When working on a project like a hostel room allocation system in the field of database management systems (DBMS), there are several key learnings which includes.

About data Modeling in this I have learned how to designing the database schema for the hostel room allocation system involves learning about entity-relationship modeling (ER modeling), identifying entities such as students, rooms, allocations, and relationships between them (e.g., a student can be allocated to a room). Understanding concepts like cardinality, normalization, and denormalization is crucial for creating an efficient and scalable database structure.

About what are the database design principles I this when I was applying database design principles such as choosing appropriate data types, defining primary keys, foreign keys, and constraints, and optimizing the database schema for performance and data integrity are essential skills. I Learned about indexing strategies, partitioning, and data organization techniques can also improve the efficiency of queries and data retrieval operations.

While doing the code part I learned about performance optimization which monitor’s the database performance metrics, identifying bottlenecks, optimizing SQL queries and database configurations, caching strategies, and scalability planning are important aspects of ensuring the hostel room allocation system operates efficiently and can handle increasing data loads. Which I learned about this while doing this project.

**IX. Challenges Faced**

While doing this project I have faced some challenges which are complex data relationships while managing such complex relationships between entities such as students, rooms, allocations, and hostel facilities can be challenging. Designing an efficient database schema that properly represents these relationships while ensuring data integrity and normalization requires careful planning.

In terms of error handling I had to make sure that ensuring the data quality and reliability by implementing robust data validation rules, error handling mechanisms, exception logging, and rollback procedures in case of data entry errors or system failures is critical for a reliable hostel room allocation system.

While doing the coding part some codes were not running properly. And the use of 10 tuples made the code's execution more complex.

**X. Conclusion**

In conclusion, throughout the project, the project allowed me for deep exploration and application of database design principles, query optimization techniques, transaction management strategies, and security measures. It also provided valuable insights into user interface integration, data validation, error handling, and scalability considerations.

From this project I learned about how transaction management skills with dealing with concurrency issues, implementing transaction control mechanisms, and ensuring ACID properties highlighted the importance of effective transaction management for maintaining data consistency and reliability.

Overall, the project provided valuable hands-on experience and practical skills in database management, software development, and system integration, highlighting the importance of technical expertise, problem-solving abilities, and attention to detail in developing robust and reliable database-driven applications.