A Practical Activity Report

Submitted for

DATABASE MANAGEMENT SYSTEMS (UCS310)

By

Devansh Agarwal	**102103620**
Dhruv Gupta	**102103622**
Shaurya Chichra	**102103625**
Hardik Yadav	**102103600**

Submitted to

DR. NITIGYA SAMBYAL

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,

(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

INDIA

Jan-May 2023

INDEX

INTRODUCTION

The Online Hostel Room Allotment System uses MySQL as its database management system. This project aims to provide a comprehensive and efficient platform for hostel room allotment, catering to the needs of both students and hostel management. Hostel room allotment can be a complex and time-consuming process, especially for large institutions that house hundreds of students. This project aims to simplify the process and make it easier for everyone involved.

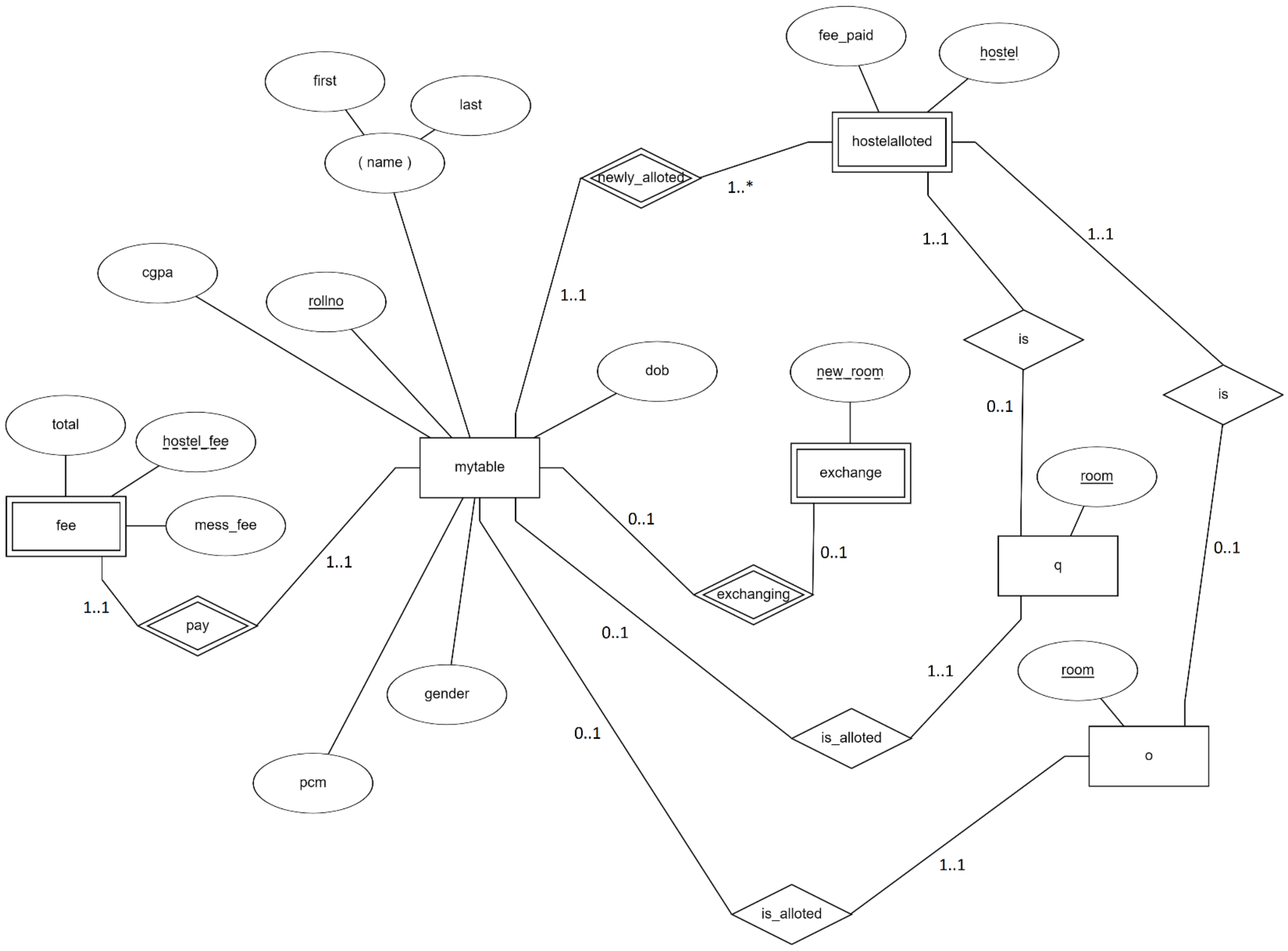
The system is designed using Python, a powerful and versatile programming language, and the Django web framework, which follows the model-view-controller (MVC) architectural pattern. The system uses MySQL, a popular and reliable relational database management system, to store and manage data related to hostel room allotment. MySQL offers high performance, scalability, and data security, making it an ideal choice for large-scale applications like hostel room allotment systems.

The system provides a user-friendly interface for students to apply for hostel

rooms, check the status of their applications, and view their allotted rooms. A basic payment page is also made where the students can look at their fees and pay the dues. Also, the option to exchange rooms is also available for students.

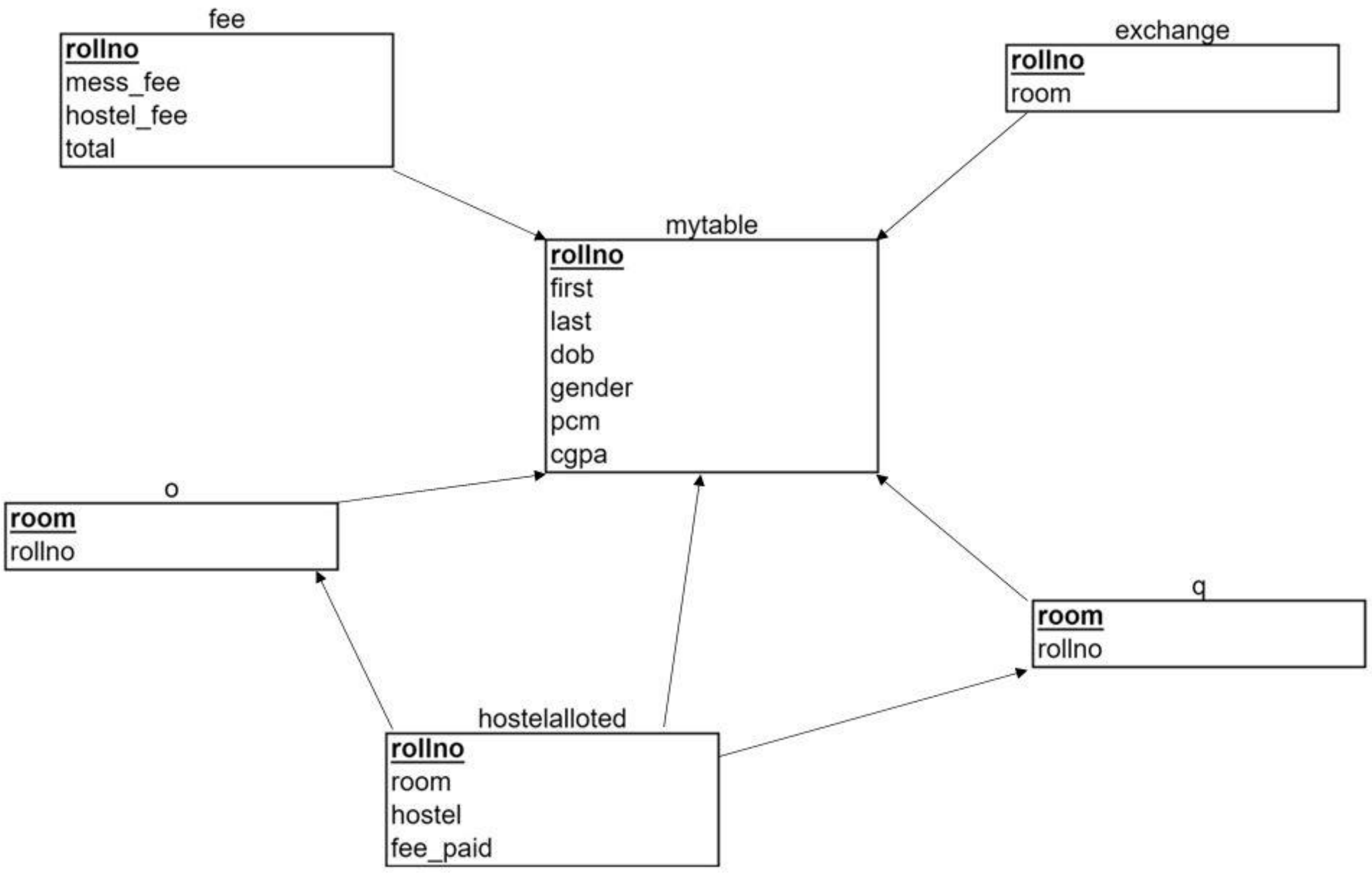
By using MySQL as the database management system, the system ensures that data is stored securely, processed efficiently, and easily accessible when required. With this project, we hope to simplify the process of hostel room allotment, minimize errors, and provide a hassle-free experience for both students and hostel management.

ER – DIAGRAM



fee_paid
hostel
first
last
(name)
newly_alloted
1..*
hostelalloted
1..1
1..1
cgpa
rollno
1..1
is
is
dob
new_room
0..1
total
hostel_fee
mytable
exchange
room
fee
mess_fee
0..1
0..1
q
0..1
1..1
exchanging
1..1
pay
0..1
1..1
room
gender
is_alloted
0..1
o
pcm
1..1
is_alloted

ER TO TABLE


fee
rollno
mess_fee
hostel_fee
total
exchange
rollno
room
mytable
rollno
first
last
dob
gender
pcm
cgpa
o
room
rollno
q
room
rollno
hostelalloted
rollno
room
hostel
fee_paid

NORMALIZATION

Normalization is the process to eliminate data redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

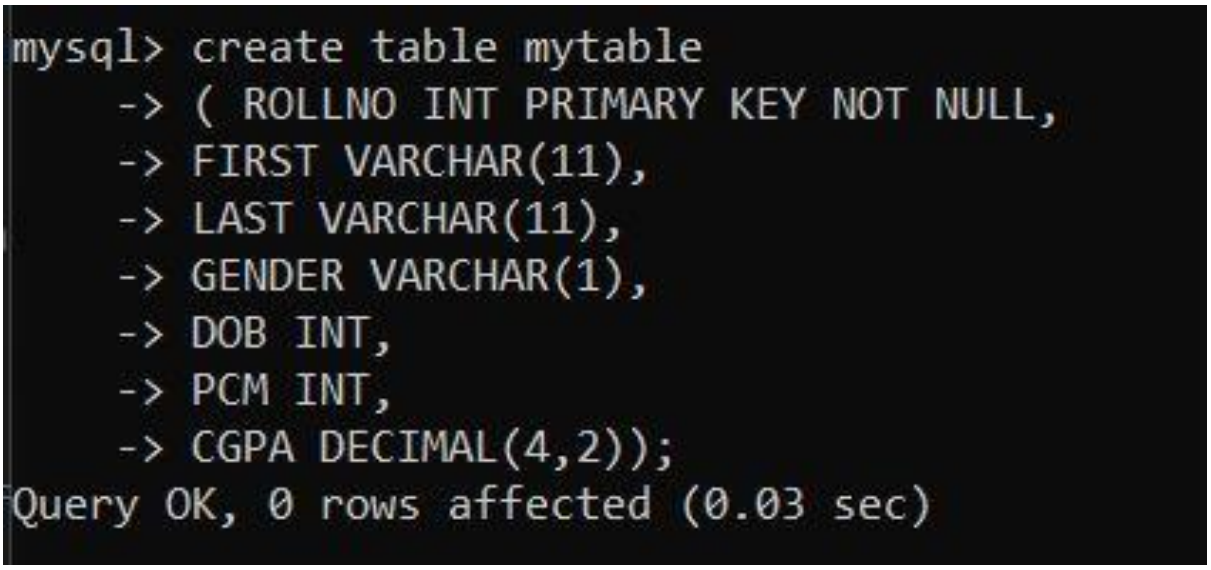
Normalization organizes the columns and tables of a database to ensure that database integrity constraints properly execute their dependencies. It is a systematic technique of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update, and Deletion anomalies.

1NF: The tables we have used had 'name' as multivalued attribute (in mytable), so we divided it into first and last name. Now, no multi-valued attribute exists and the tables are now in First Normal Form.

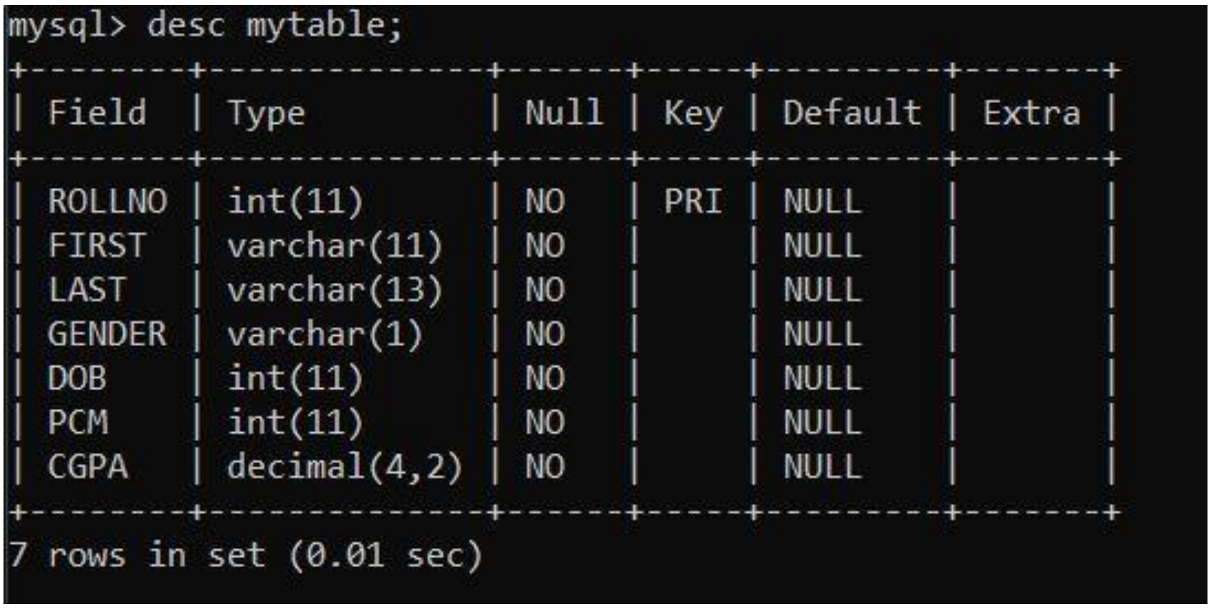
2NF: All the partial dependencies have been resolved and any partial dependency does not exist. The tables are now in Second Normal Form.

3NF: All the attributes having transitive dependencies have been shifted to different tables and now no transitive dependencies exist. The tables are now in Third Normal Form.

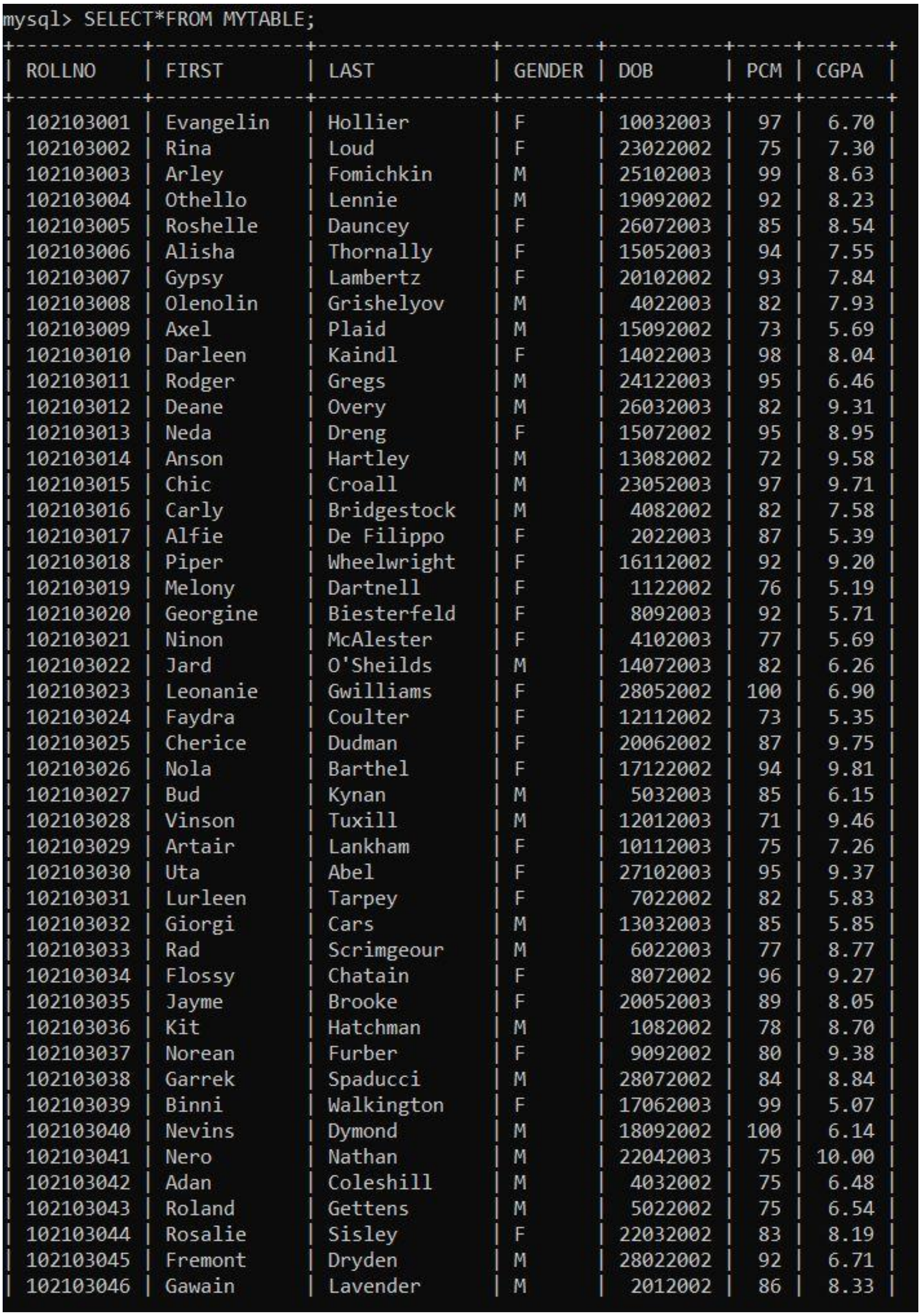
TABLE CREATION AND DESCRIPTION



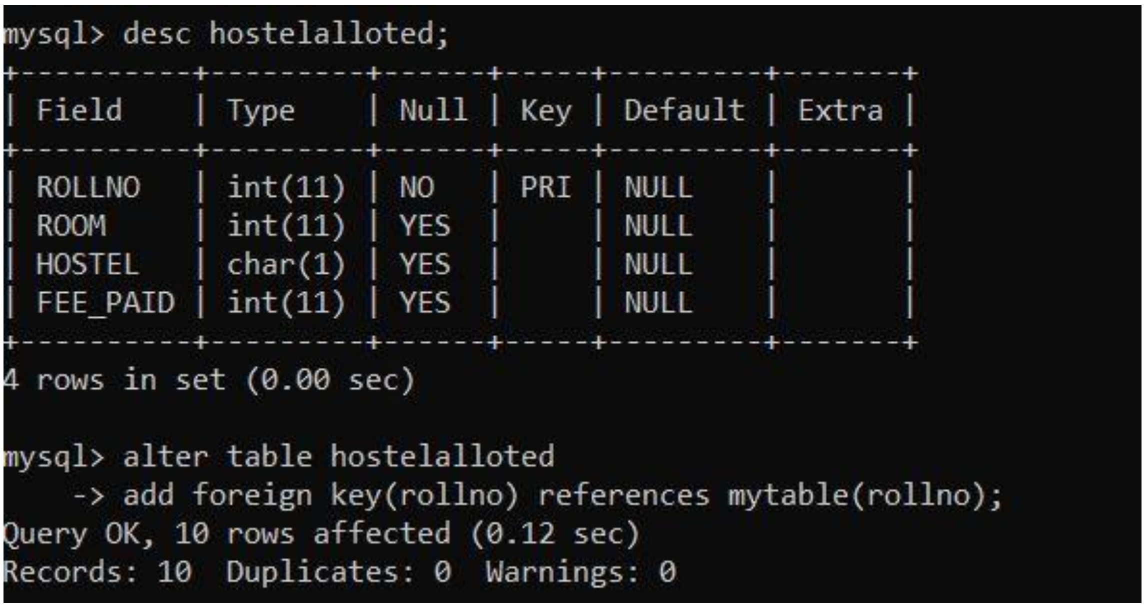
```
mysql> create table mytable
    -> ( ROLLNO INT PRIMARY KEY NOT NULL,
    -> FIRST VARCHAR(11),
    -> LAST VARCHAR(11),
    -> GENDER VARCHAR(1),
    -> DOB INT,
    -> PCM INT,
    -> CGPA DECIMAL(4,2));
Query OK, 0 rows affected (0.03 sec)
```



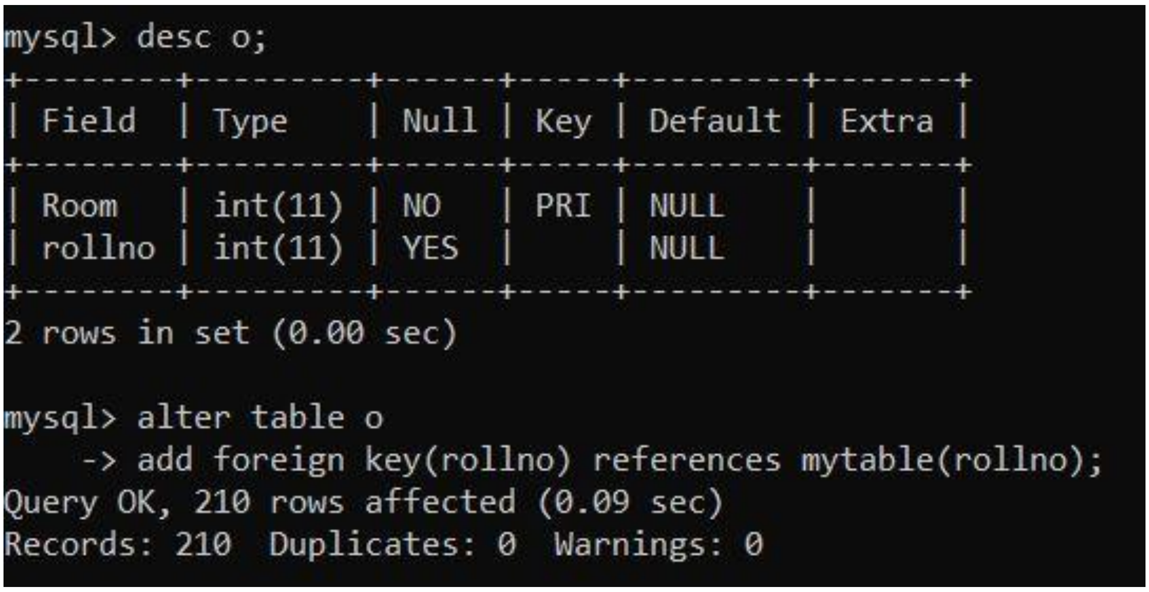
```
mysql> desc mytable;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| ROLLNO | int(11)      | NO   | PRI | NULL    |       |
| FIRST  | varchar(11)  | NO   |     | NULL    |       |
| LAST   | varchar(13)  | NO   |     | NULL    |       |
| GENDER | varchar(1)   | NO   |     | NULL    |       |
| DOB    | int(11)      | NO   |     | NULL    |       |
| PCM    | int(11)      | NO   |     | NULL    |       |
| CGPA   | decimal(4,2) | NO   |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```



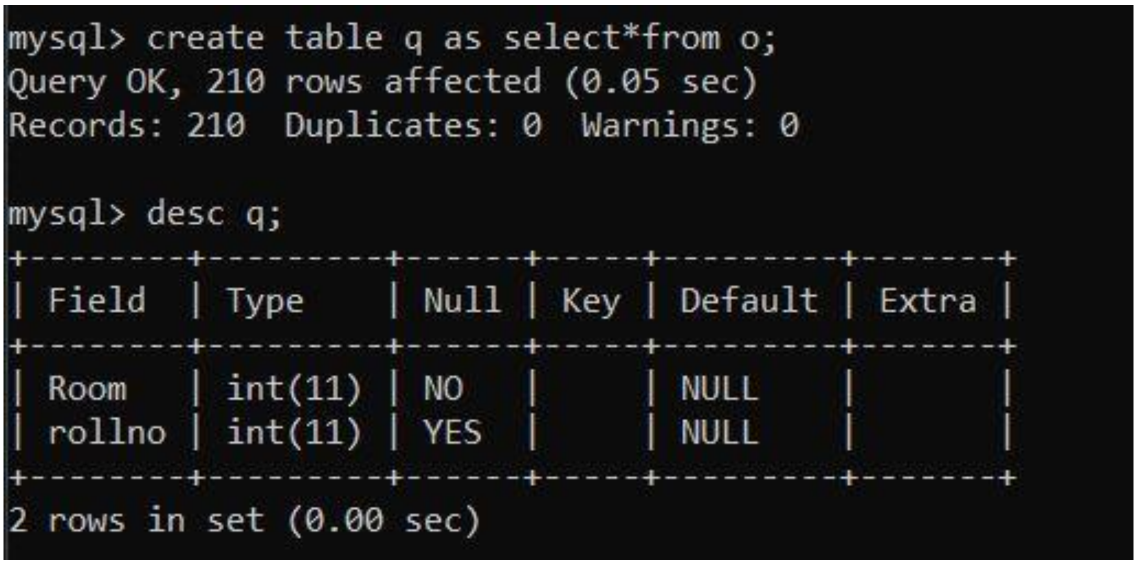
```
mysql> SELECT*FROM MYTABLE;
+-----------+-------------+---------------+--------+----------+-----+-------+
| ROLLNO    | FIRST       | LAST          | GENDER | DOB      | PCM | CGPA  |
+-----------+-------------+---------------+--------+----------+-----+-------+
| 102103001 | Evangelin   | Hollier       | F      | 10032003 |  97 |  6.70 |
| 102103002 | Rina        | Loud          | F      | 23022002 |  75 |  7.30 |
| 102103003 | Arley       | Fomichkin     | M      | 25102003 |  99 |  8.63 |
| 102103004 | Othello     | Lennie        | M      | 19092002 |  92 |  8.23 |
| 102103005 | Roshelle    | Dauncey       | F      | 26072003 |  85 |  8.54 |
| 102103006 | Alisha      | Thornally     | F      | 15052003 |  94 |  7.55 |
| 102103007 | Gypsy       | Lambertz      | F      | 20102002 |  93 |  7.84 |
| 102103008 | Olenolin    | Grishelyov    | M      |  4022003 |  82 |  7.93 |
| 102103009 | Axel        | Plaid         | M      | 15092002 |  73 |  5.69 |
| 102103010 | Darleen     | Kaindl        | F      | 14022003 |  98 |  8.04 |
| 102103011 | Rodger      | Gregs         | M      | 24122003 |  95 |  6.46 |
| 102103012 | Deane       | Overy         | M      | 26032003 |  82 |  9.31 |
| 102103013 | Neda        | Dreng         | F      | 15072002 |  95 |  8.95 |
| 102103014 | Anson       | Hartley       | M      | 13082002 |  72 |  9.58 |
| 102103015 | Chic        | Croall        | M      | 23052003 |  97 |  9.71 |
| 102103016 | Carly       | Bridgestock   | M      |  4082002 |  82 |  7.58 |
| 102103017 | Alfie       | De Filippo    | F      |  2022003 |  87 |  5.39 |
| 102103018 | Piper       | Wheelwright   | F      | 16112002 |  92 |  9.20 |
| 102103019 | Melony      | Dartnell      | F      |  1122002 |  76 |  5.19 |
| 102103020 | Georgine    | Biesterfeld   | F      |  8092003 |  92 |  5.71 |
| 102103021 | Ninon       | McAlester     | F      |  4102003 |  77 |  5.69 |
| 102103022 | Jard        | O'Sheilds     | M      | 14072003 |  82 |  6.26 |
| 102103023 | Leonanie    | Gwilliams     | F      | 28052002 | 100 |  6.90 |
| 102103024 | Faydra      | Coulter       | F      | 12112002 |  73 |  5.35 |
| 102103025 | Cherice     | Dudman        | F      | 20062002 |  87 |  9.75 |
| 102103026 | Nola        | Barthel       | F      | 17122002 |  94 |  9.81 |
| 102103027 | Bud         | Kynan         | M      |  5032003 |  85 |  6.15 |
| 102103028 | Vinson      | Tuxill        | M      | 12012003 |  71 |  9.46 |
| 102103029 | Artair      | Lankham       | F      | 10112003 |  75 |  7.26 |
| 102103030 | Uta         | Abel          | F      | 27102003 |  95 |  9.37 |
| 102103031 | Lurleen     | Tarpey        | F      |  7022002 |  82 |  5.83 |
| 102103032 | Giorgi      | Cars          | M      | 13032003 |  85 |  5.85 |
| 102103033 | Rad         | Scrimgeour    | M      |  6022003 |  77 |  8.77 |
| 102103034 | Flossy      | Chatain       | F      |  8072002 |  96 |  9.27 |
| 102103035 | Jayme       | Brooke        | F      | 20052003 |  89 |  8.05 |
| 102103036 | Kit         | Hatchman      | M      |  1082002 |  78 |  8.70 |
| 102103037 | Norean      | Furber        | F      |  9092002 |  80 |  9.38 |
| 102103038 | Garrek      | Spaducci      | M      | 28072002 |  84 |  8.84 |
| 102103039 | Binni       | Walkington    | F      | 17062003 |  99 |  5.07 |
| 102103040 | Nevins      | Dymond        | M      | 18092002 | 100 |  6.14 |
| 102103041 | Nero        | Nathan        | M      | 22042003 |  75 | 10.00 |
| 102103042 | Adan        | Coleshill     | M      |  4032002 |  75 |  6.48 |
| 102103043 | Roland      | Gettens       | M      |  5022002 |  75 |  6.54 |
| 102103044 | Rosalie     | Sisley        | F      | 22032002 |  83 |  8.19 |
| 102103045 | Fremont     | Dryden        | M      | 28022002 |  92 |  6.71 |
| 102103046 | Gawain      | Lavender      | M      |  2012002 |  86 |  8.33 |
```



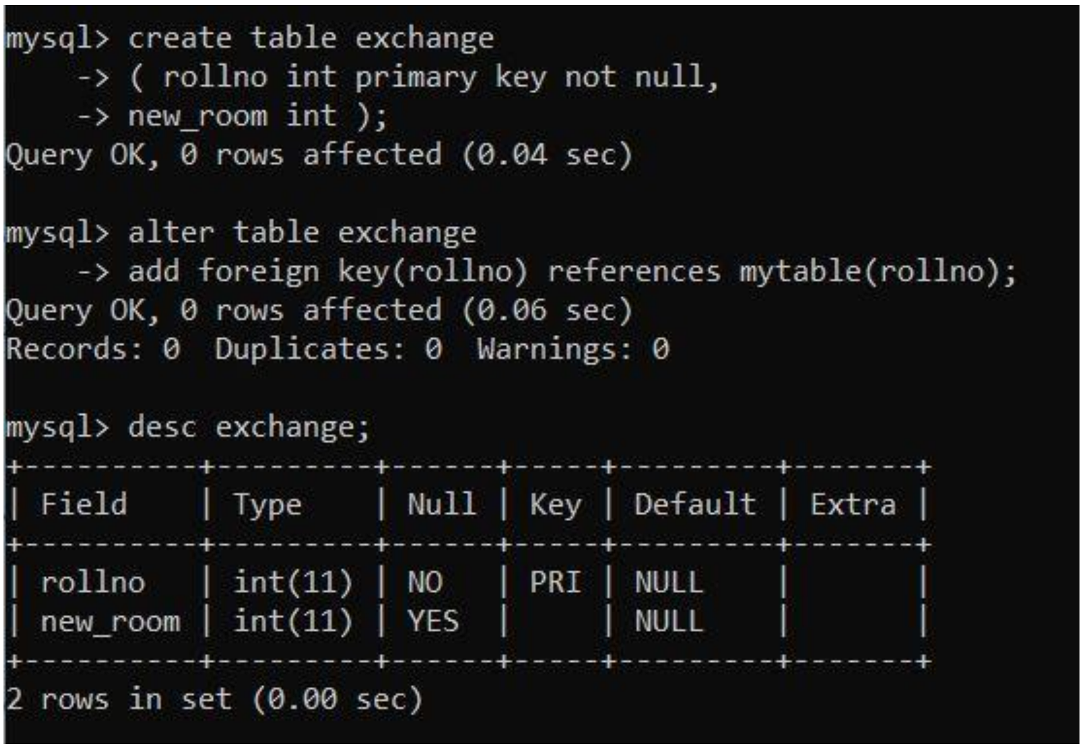
```
mysql> desc hostelalloted;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| ROLLNO   | int(11) | NO   | PRI | NULL    |       |
| ROOM     | int(11) | YES  |     | NULL    |       |
| HOSTEL   | char(1) | YES  |     | NULL    |       |
| FEE_PAID | int(11) | YES  |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> alter table hostelalloted
    -> add foreign key(rollno) references mytable(rollno);
Query OK, 10 rows affected (0.12 sec)
Records: 10  Duplicates: 0  Warnings: 0
```



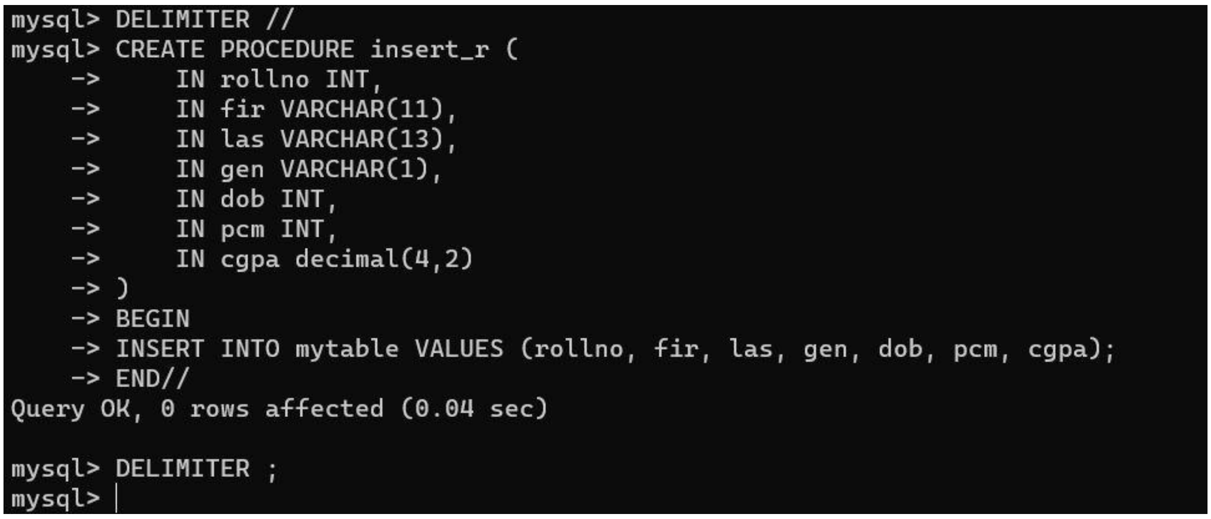
```
mysql> desc o;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| Room   | int(11) | NO   | PRI | NULL    |       |
| rollno | int(11) | YES  |     | NULL    |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> alter table o
    -> add foreign key(rollno) references mytable(rollno);
Query OK, 210 rows affected (0.09 sec)
Records: 210  Duplicates: 0  Warnings: 0
```



```
mysql> create table q as select*from o;
Query OK, 210 rows affected (0.05 sec)
Records: 210  Duplicates: 0  Warnings: 0

mysql> desc q;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| Room   | int(11) | NO   |     | NULL    |       |
| rollno | int(11) | YES  |     | NULL    |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



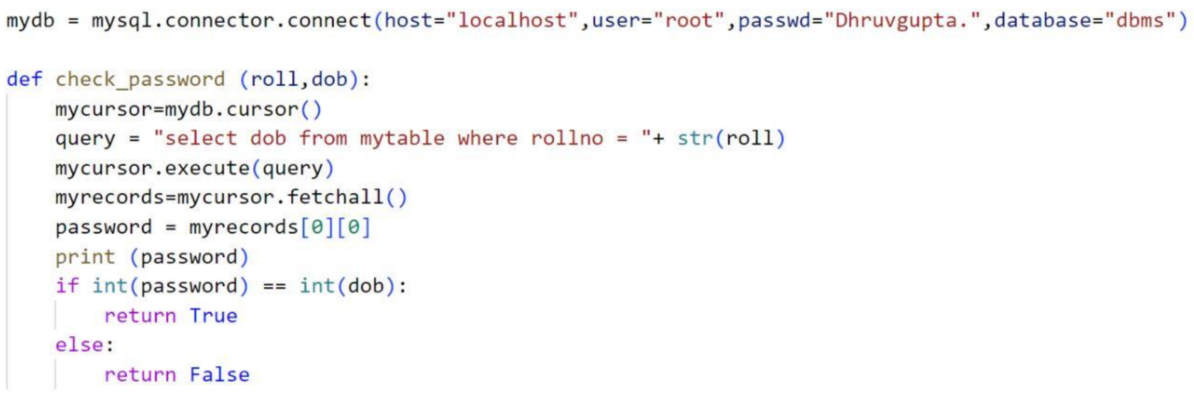
```
mysql> create table exchange
    -> ( rollno int primary key not null,
    -> new_room int );
Query OK, 0 rows affected (0.04 sec)

mysql> alter table exchange
    -> add foreign key(rollno) references mytable(rollno);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc exchange;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| rollno   | int(11) | NO   | PRI | NULL    |       |
| new_room | int(11) | YES  |     | NULL    |       |
+----------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



```
mysql> DELIMITER //
mysql> CREATE PROCEDURE insert_r (
    ->     IN rollno INT,
    ->     IN fir VARCHAR(11),
    ->     IN las VARCHAR(13),
    ->     IN gen VARCHAR(1),
    ->     IN dob INT,
    ->     IN pcm INT,
    ->     IN cgpa decimal(4,2)
    -> )
    -> BEGIN
    -> INSERT INTO mytable VALUES (rollno, fir, las, gen, dob, pcm, cgpa);
    -> END//
Query OK, 0 rows affected (0.04 sec)

mysql> DELIMITER ;
mysql>
```

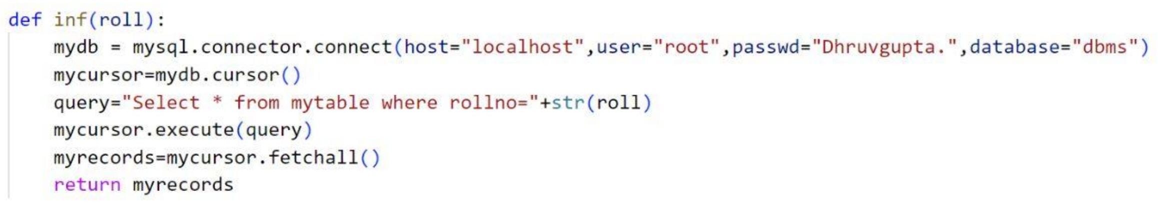
QUERIES USING MYSQL CONNECTOR IN PYTHON

Authentication:



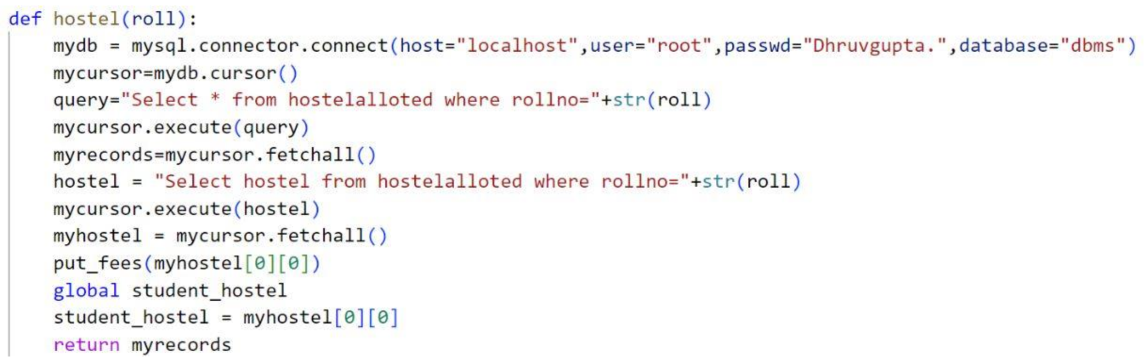
```
mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")

def check_password (roll,dob):
    mycursor=mydb.cursor()
    query = "select dob from mytable where rollno = "+ str(roll)
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    password = myrecords[0][0]
    print (password)
    if int(password) == int(dob):
        return True
    else:
        return False
```

Fetching Student Details:



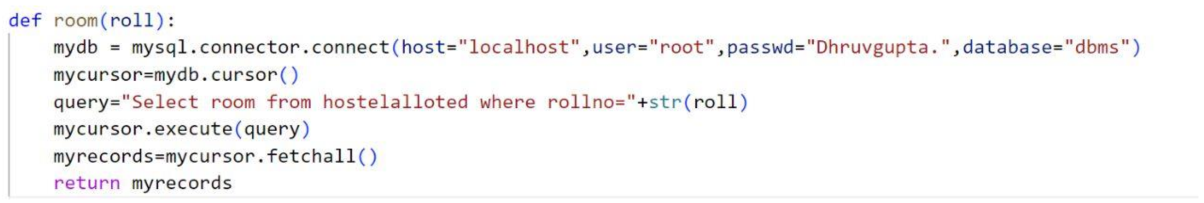
```
def inf(roll):
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="Select * from mytable where rollno="+str(roll)
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    return myrecords
```

Fetching Student’s Hostel Status:



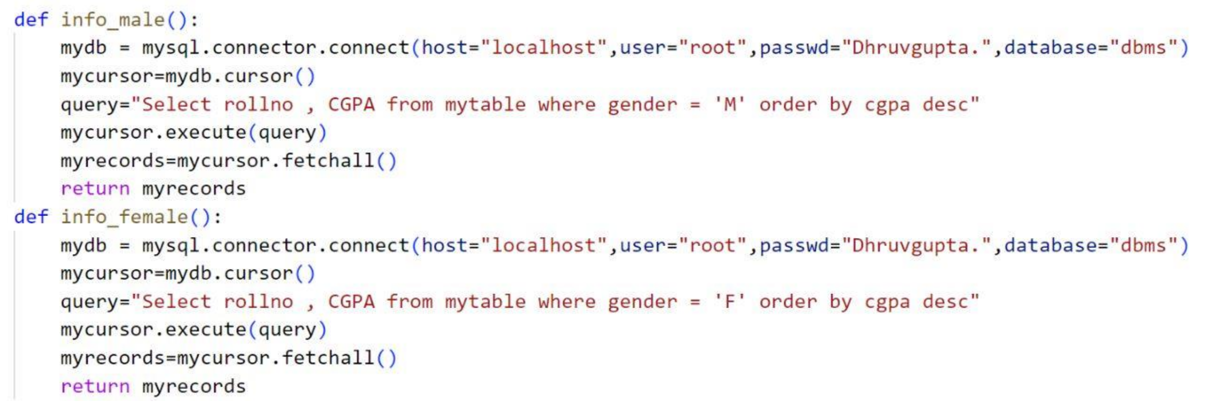
```
def hostel(roll):
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="Select * from hostelalloted where rollno="+str(roll)
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    hostel = "Select hostel from hostelalloted where rollno="+str(roll)
    mycursor.execute(hostel)
    myhostel = mycursor.fetchall()
    put_fees(myhostel[0][0])
    global student_hostel
    student_hostel = myhostel[0][0]
    return myrecords
```

Fetching Student’s Room Status:

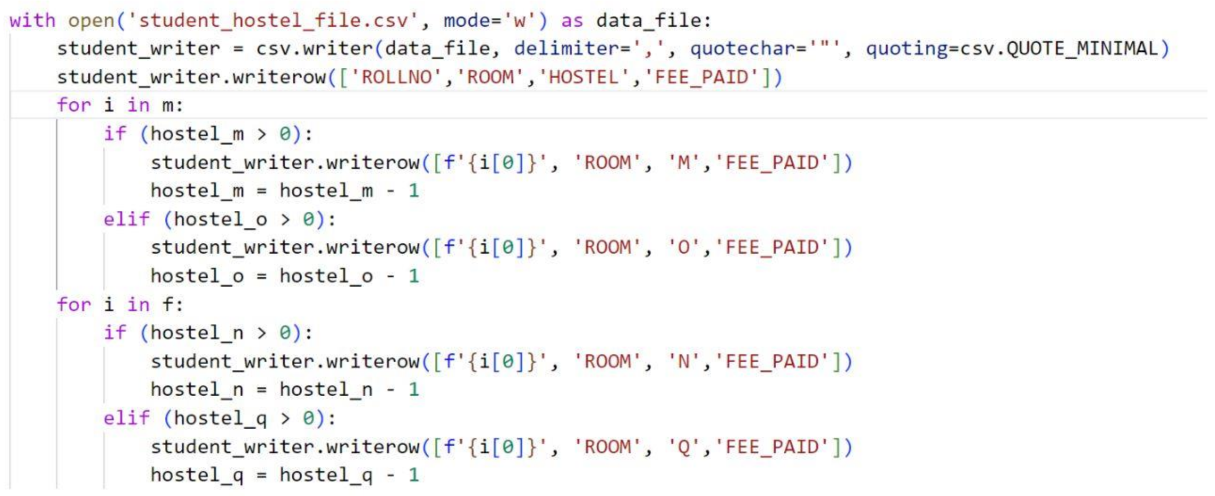


```
def room(roll):
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="Select room from hostelalloted where rollno="+str(roll)
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    return myrecords
```

Allotting Hostel to Students on the basis of Gender and CGPA:

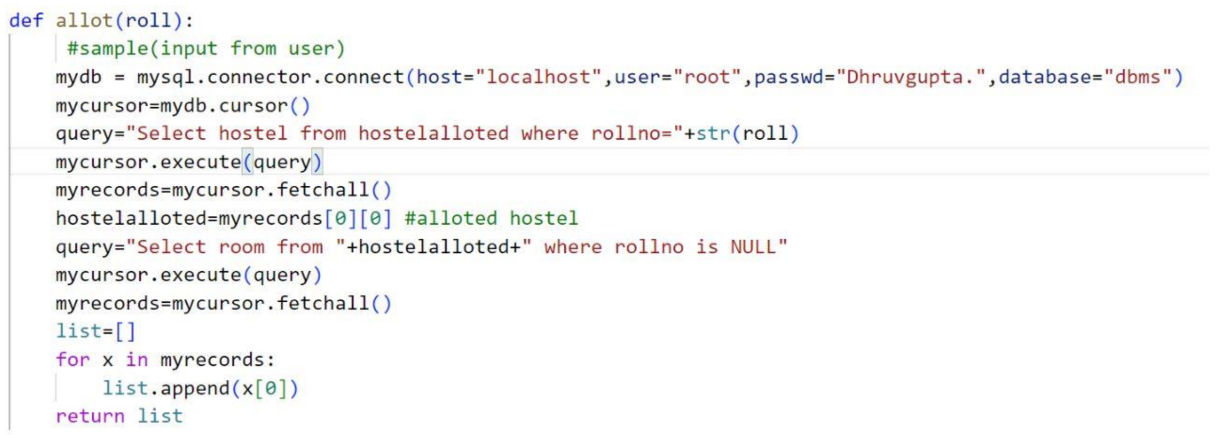


```python
def info_male():
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="Select rollno , CGPA from mytable where gender = 'M' order by cgpa desc"
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    return myrecords
def info_female():
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="Select rollno , CGPA from mytable where gender = 'F' order by cgpa desc"
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    return myrecords
```



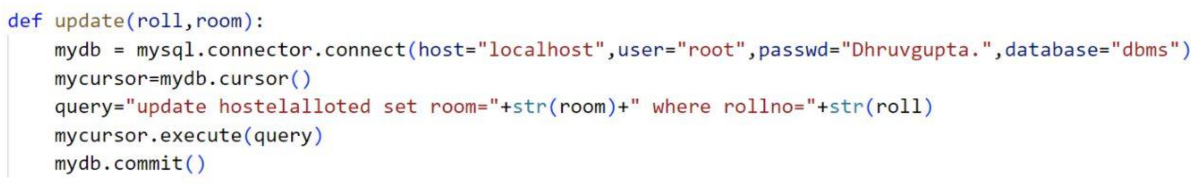
```python
with open('student_hostel_file.csv', mode='w') as data_file:
    student_writer = csv.writer(data_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
    student_writer.writerow(['ROLLNO','ROOM','HOSTEL','FEE_PAID'])
    for i in m:
        if (hostel_m > 0):
            student_writer.writerow([f'{i[0]}', 'ROOM', 'M','FEE_PAID'])
            hostel_m = hostel_m - 1
        elif (hostel_o > 0):
            student_writer.writerow([f'{i[0]}', 'ROOM', 'O','FEE_PAID'])
            hostel_o = hostel_o - 1
    for i in f:
        if (hostel_n > 0):
            student_writer.writerow([f'{i[0]}', 'ROOM', 'N','FEE_PAID'])
            hostel_n = hostel_n - 1
        elif (hostel_q > 0):
            student_writer.writerow([f'{i[0]}', 'ROOM', 'Q','FEE_PAID'])
            hostel_q = hostel_q - 1
```

Showing only Empty rooms to student:



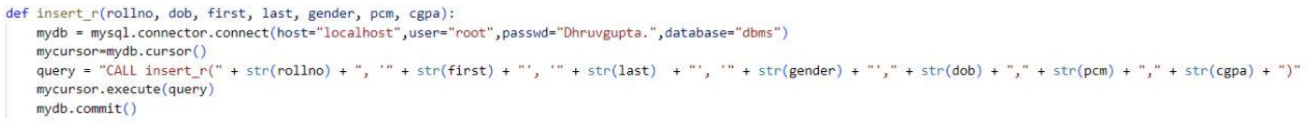
```python
def allot(roll):
     #sample(input from user)
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="Select hostel from hostelalloted where rollno="+str(roll)
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    hostelalloted=myrecords[0][0] #alloted hostel
    query="Select room from "+hostelalloted+" where rollno is NULL"
    mycursor.execute(query)
    myrecords=mycursor.fetchall()
    list=[]
    for x in myrecords:
        list.append(x[0])
    return list
```

Updating the selected room in Database:



```
def update(roll,room):
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query="update hostelalloted set room="+str(room)+" where rollno="+str(roll)
    mycursor.execute(query)
    mydb.commit()
```

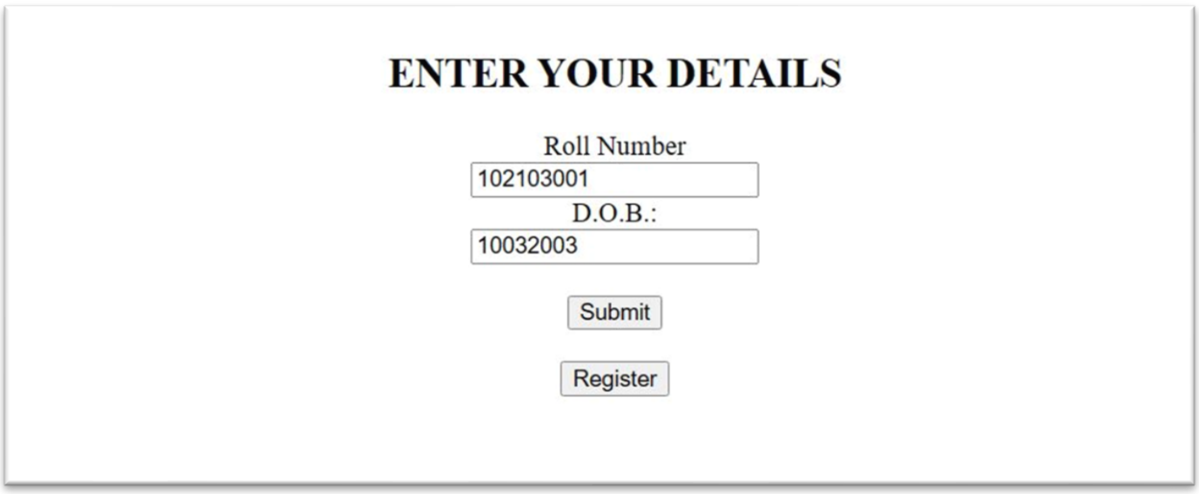
Calling Procedure to Add new Student:



```
def insert_r(rollno, dob, first, last, gender, pcm, cgpa):
    mydb = mysql.connector.connect(host="localhost",user="root",passwd="Dhruvgupta.",database="dbms")
    mycursor=mydb.cursor()
    query = "CALL insert_r(" + str(rollno) + ", '" + str(first) + "', '" + str(last)  + "', '" + str(gender) + "'," + str(dob) + "," + str(pcm) + "," + str(cgpa) + ")"
    mycursor.execute(query)
    mydb.commit()
```

FRONT-END AND WORKING

Home Page:



ENTER YOUR DETAILS

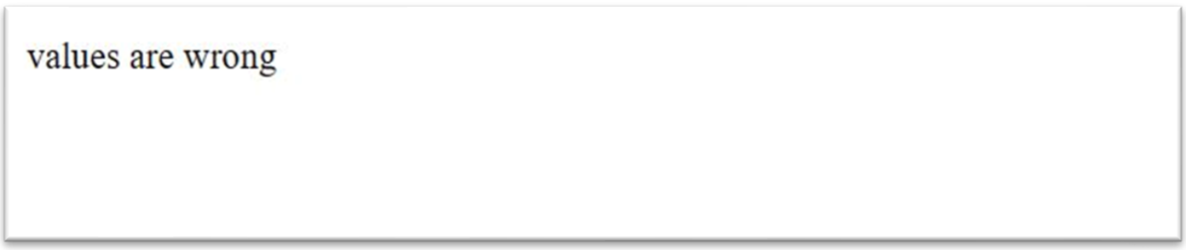
Roll Number

102103001

D.O.B.:

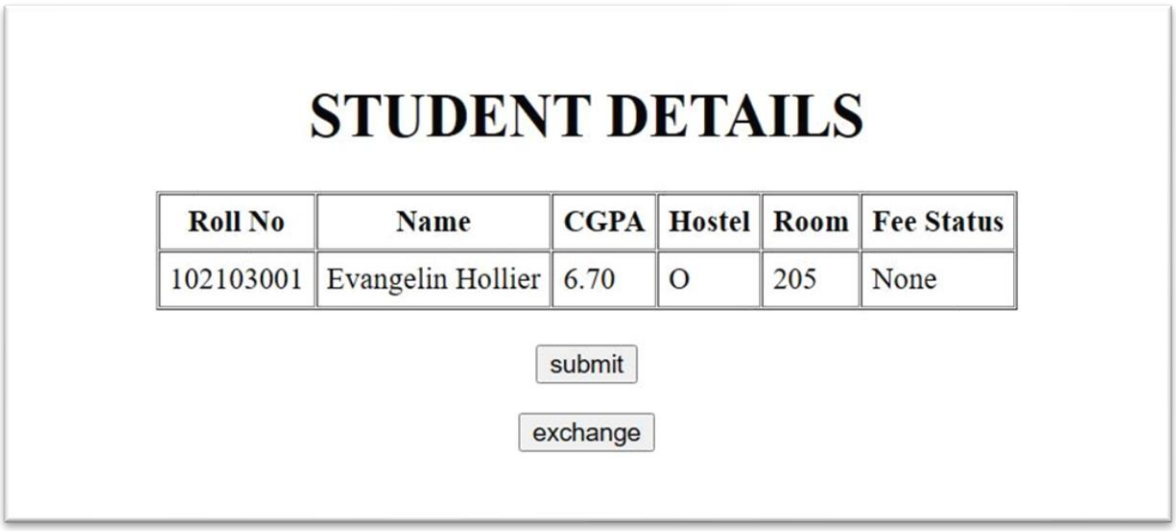
10032003

If Wrong Details Entered:



values are wrong

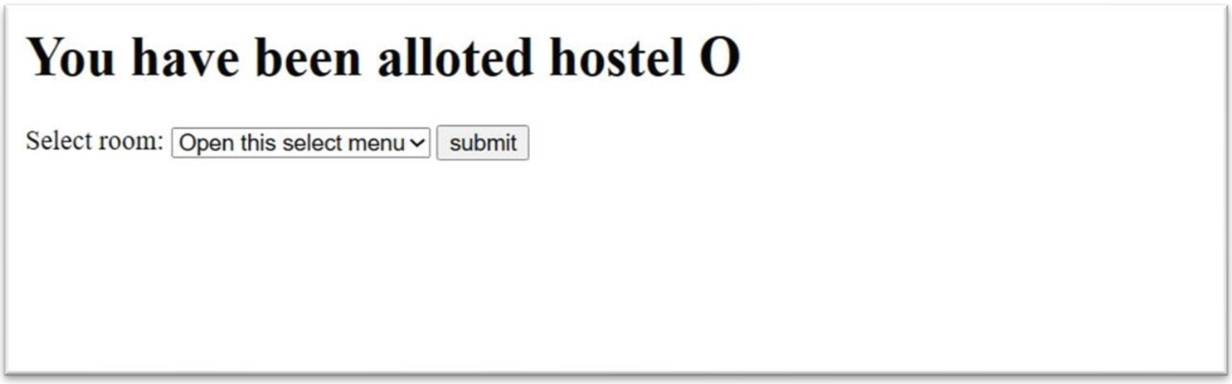
Info Page:



STUDENT DETAILS

Roll No	Name	CGPA	Hostel	Room	Fee Status
102103001	Evangelin Hollier	6.70	O	205	None

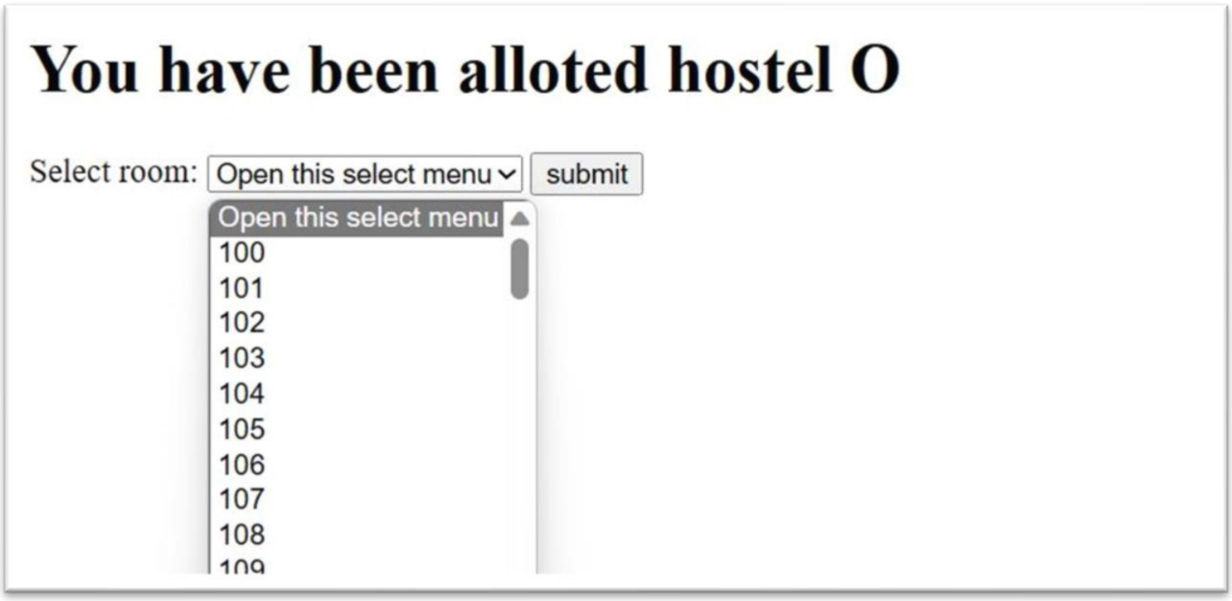
Room Selection Page:



You have been alloted hostel O

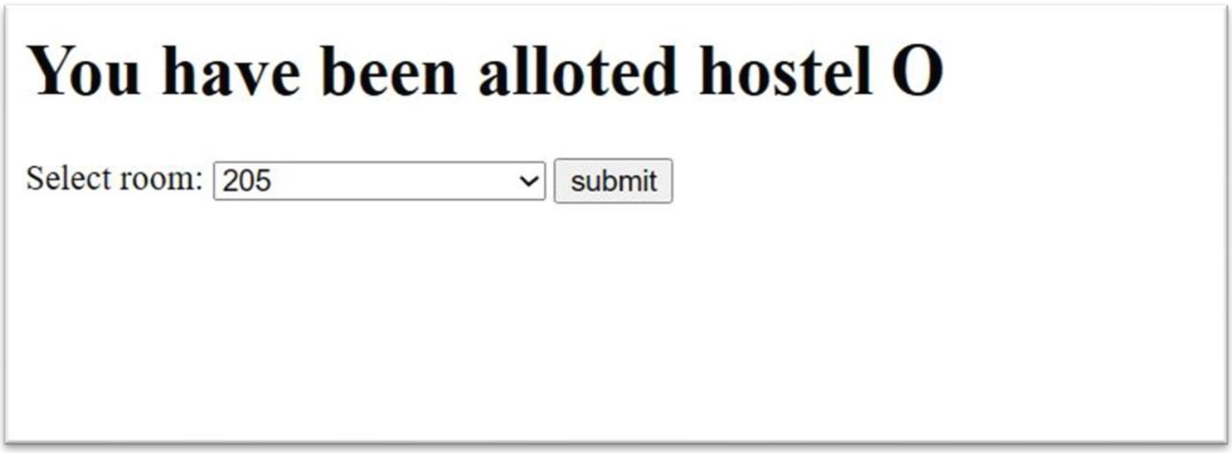
Select room: Open this select menu submit

List of available Rooms:



You have been alloted hostel O

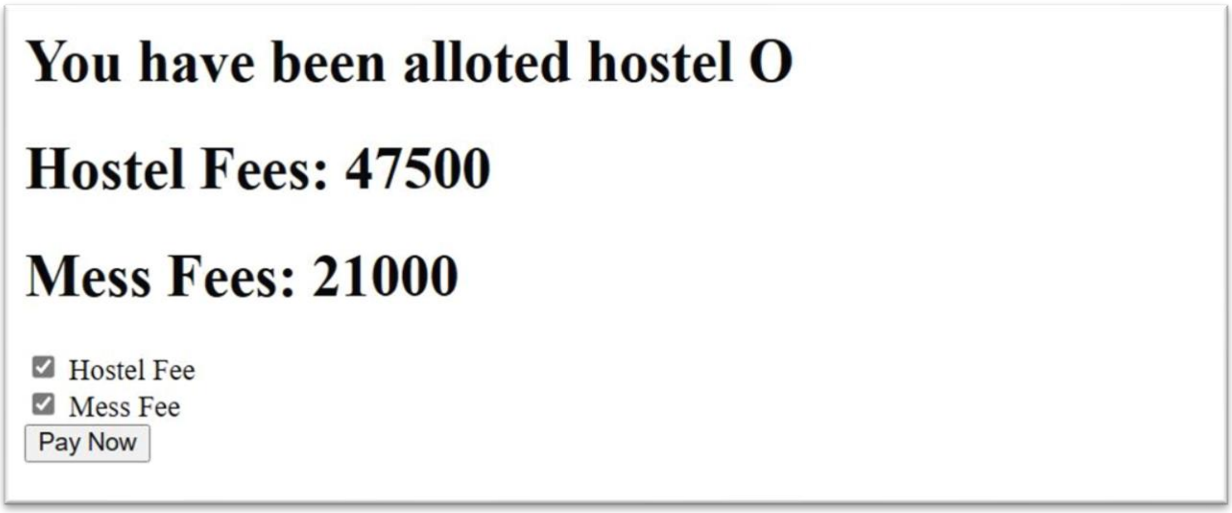
Select room: Open this select menu submit



You have been alloted hostel O

Select room: 205 submit

Fee Payment Page:



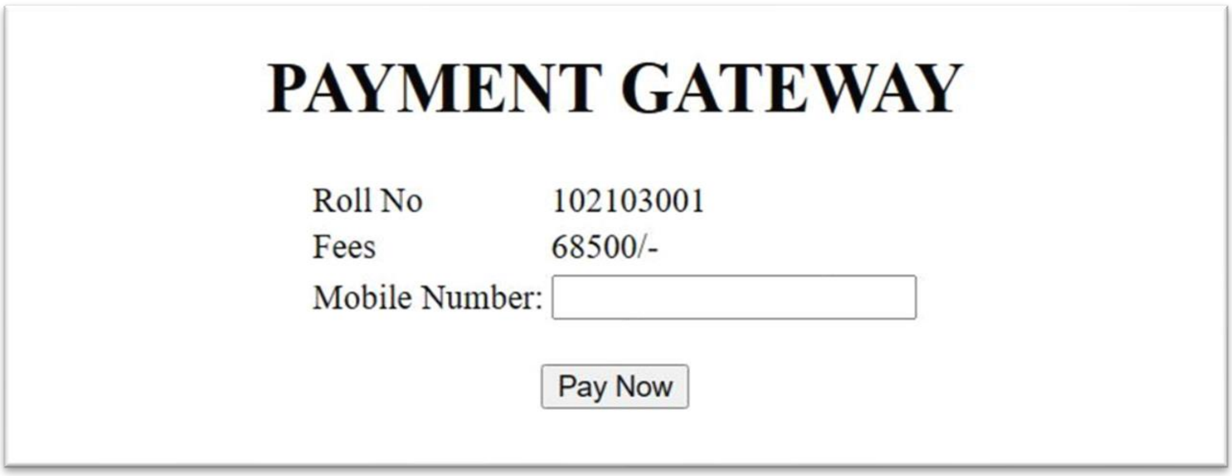
You have been alloted hostel O

Hostel Fees: 47500

Mess Fees: 21000

- [✓] Hostel Fee
- [✓] Mess Fee

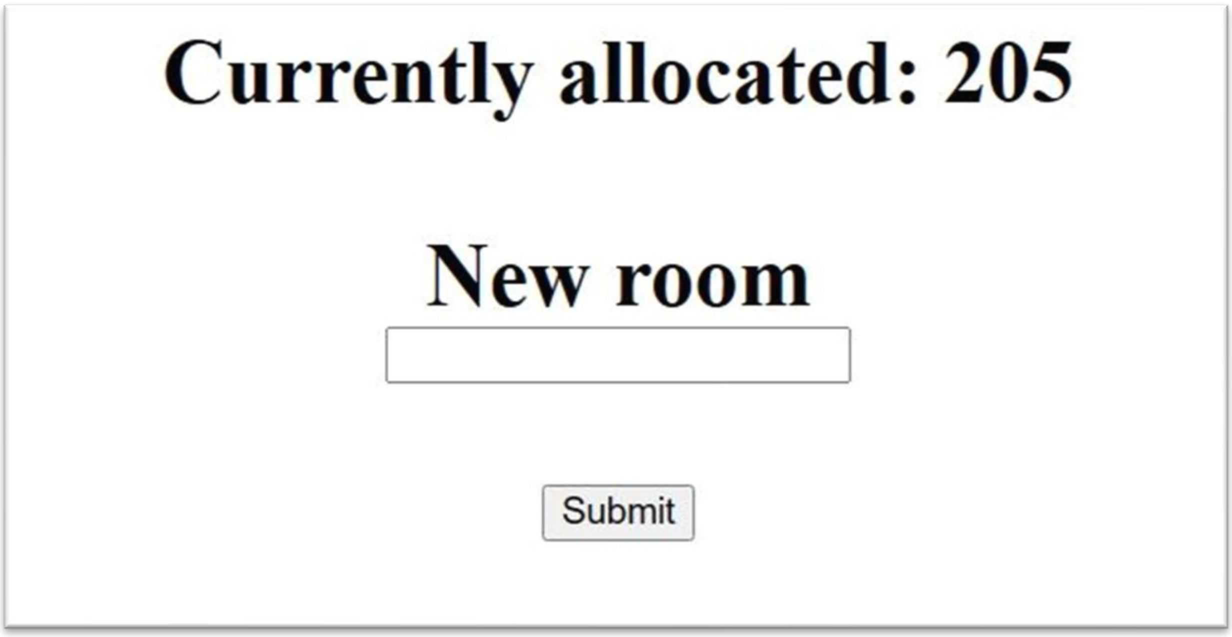
Payment Gateway Page:



PAYMENT GATEWAY

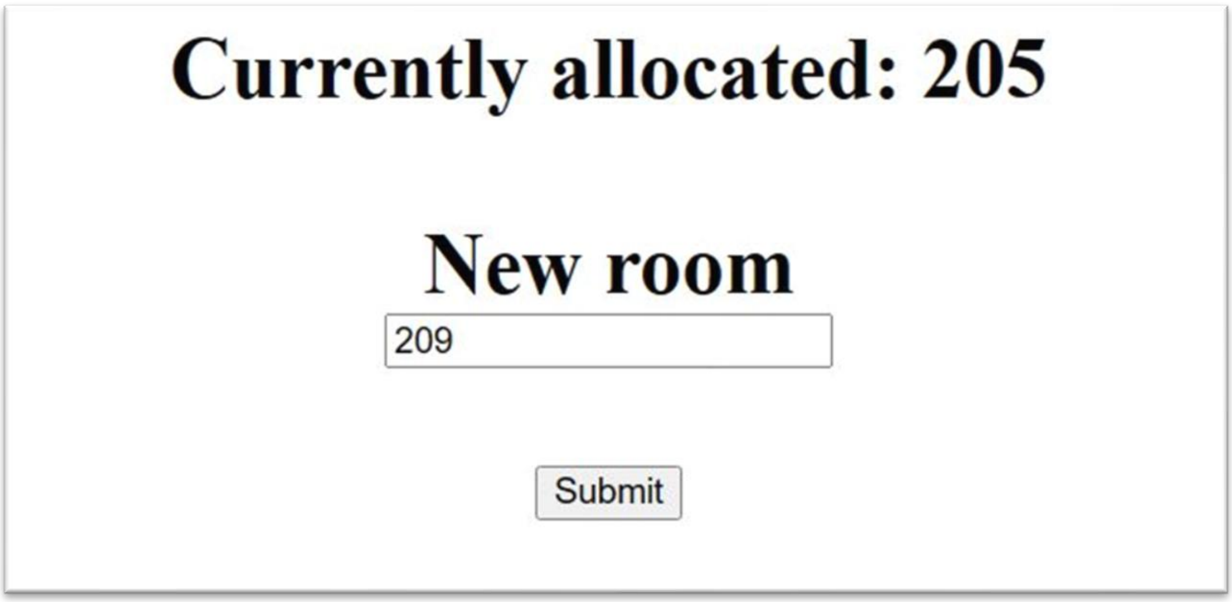
Roll No	102103001
Fees	68500/-
Mobile Number:	

Room Exchange Page:


Currently allocated: 205

New room

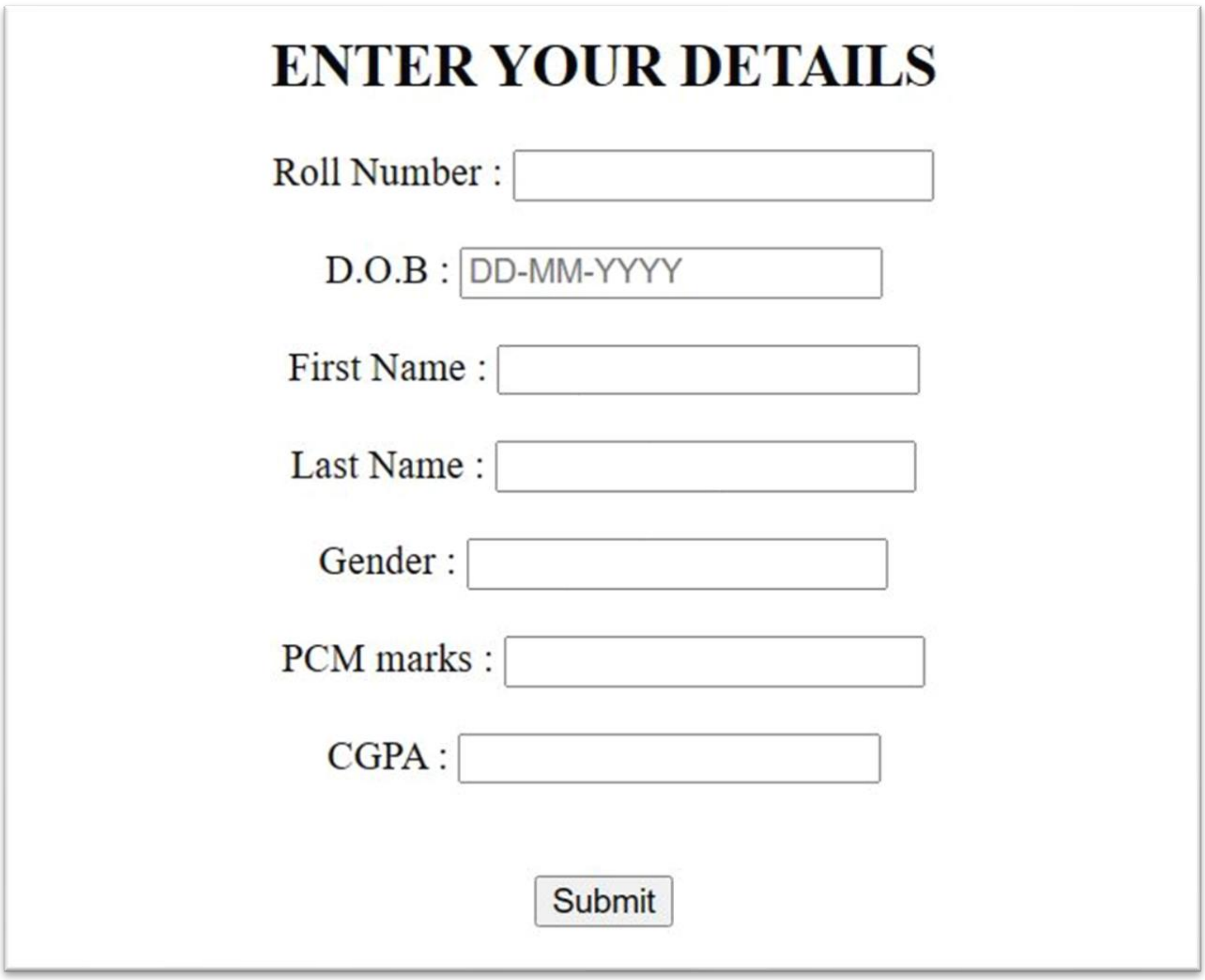
Selecting new Room:


Currently allocated: 205

New room

209

New Student Registration:



ENTER YOUR DETAILS

Roll Number :	
D.O.B :	DD-MM-YYYY
First Name :	
Last Name :	
Gender :	
PCM marks :	
CGPA :	

CONCLUSION

In conclusion, the online hostel room allotment system using MySQL as the database management system provides an efficient and user-friendly solution for managing hostel room allotment. The system allows for easy and quick registration, booking, and cancellation of hostel rooms. The MySQL database provides a secure and reliable platform for storing and retrieving data related to room allotment, student information, and payment details. The system is designed to streamline the process of room allotment, ensuring that students get the rooms they need in a timely and organized manner. Overall, the use of MySQL as the database management system has greatly enhanced the functionality and effectiveness of the online hostel room allotment system, making it a valuable tool for managing hostel operations.