DATABASE MANAGEMENT SYSTEMS

PROJECT REPORT

HOSTEL MANAGEMENT

Contributes: R. Amshu Naik (B20CS046)
Vedasamhitha Challapalli (B20CS078)
Khobragade Atul Yashwant (B20CS027)

Abstract

Following is a detailed report of the database management system course project titled Hostel Management. The project is about understanding the various data that can come under a hostel database and applying the concepts learnt in the course to represent and access this data better. As the database management course is all about storing and accessing data efficiently, our attempt was to show the same.

Problem Statement

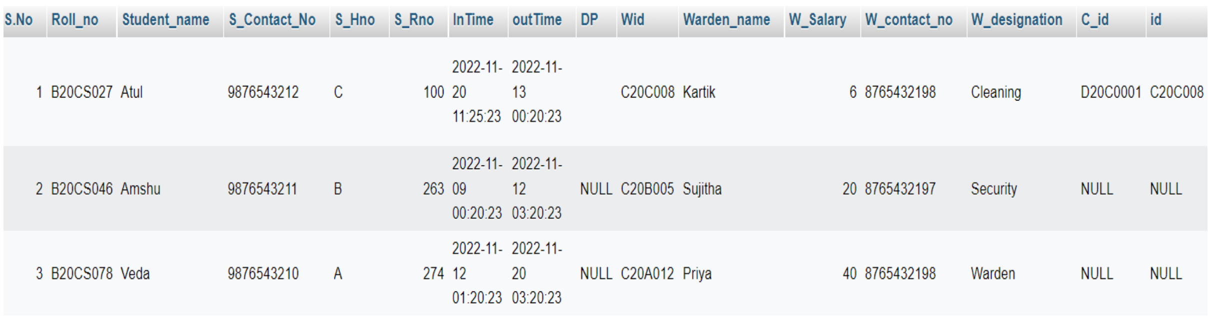
Hostel Management Project ultimately aims to collect , distribute , store, access and manage the hostel data efficiently and effectively.

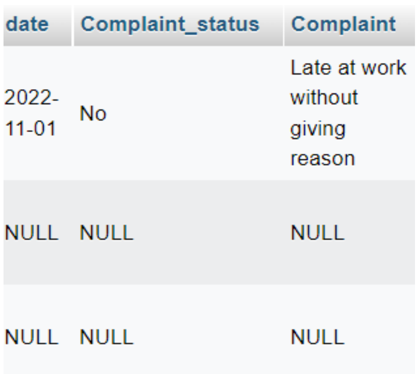
The main goal of this project is to show the following outcomes:

- Store as much data as possible under the hostel global data which basically contains every single information possible under hostel management.
- Normalize this global data and divide it into sub tables.
- Represent the ER diagram for these tables.
- Help access the hidden data within the given information using basic SQL queries.
- Access data faster using Hashing technique.
- Show various possible use cases and the problems that can be generated in the system and give solutions to them. (using of triggers)
- To show the basic blueprint of the architectural design.

Hostel Global Data

(Only few rows have been inserted, the purpose of this is to just show the columns of the global data, more tuples will be inserted in further sections)





S.No	Roll_no	Student_name	S_Contact_No	S_Hno	S_Rno	InTime	outTime	DP	Wid	Warden_name	W_Salary	W_contact_no	W_designation	C_id	id	date	Complaint_status	Complaint
1	B20CS027	Atul	9876543212	C	100	2022-11-20 11:25:23	2022-11-13 00:20:23		C20C008	Kartik	6	8765432198	Cleaning	D20C0001	C20C008	2022-11-01	No	Late at work without giving reason
2	B20CS046	Amshu	9876543211	B	263	2022-11-09 00:20:23	2022-11-12 03:20:23	NULL	C20B005	Sujitha	20	8765432197	Security	NULL	NULL	NULL	NULL	NULL
3	B20CS078	Veda	9876543210	A	274	2022-11-12 01:20:23	2022-11-20 03:20:23	NULL	C20A012	Priya	40	8765432198	Warden	NULL	NULL	NULL	NULL	NULL

Normalization

Functional dependencies In the Hostel global data found between different attributes are as follows:

- Roll_No->S_contact,
- Roll_No->S_Rno,
- Student_name->S_Rno,
- Wid->W_contact_no
- Wid->Warden_name ,
- W_salary->W->designation

=> Normalization of data on the basis of Functional dependancies;

As here if we compare the first 3 functional dependencies then we get to know that S_contact,S_Rno will be the non candidate attributes and Roll_no , student_name are candidate keys and we can take roll_no as primary key. So Based on the property of functional dependencies.

Roll_no, Student_name, ——--> S_contact,S_Rno

As S_Hno not in any of the side so it will come on both sides and so,

Roll_no, Student_name, S_Hno ——--> S_contact,S_Rno,S_hno

So here comes one normalized form.

=> Now coming to 2nd set of functional dependencies

- Wid->W_contact_no
- Wid->Warden_name ,
- W_salary->W_designation

We can assume that Wid,W_salary—>W_contact_no,warden_name,w_designation.Here these set of 5 attributes closely linked and no functional dependency so we can split these closely related attributes into another table of the following attributes

{ Wid,W_salary,W_contact_no,warden_name,w_designation}

=> Now the left attributes their is neither partial dependency nor functional dependency, so clubbed together as no one dependent on other so a 3rd split done and the table form is

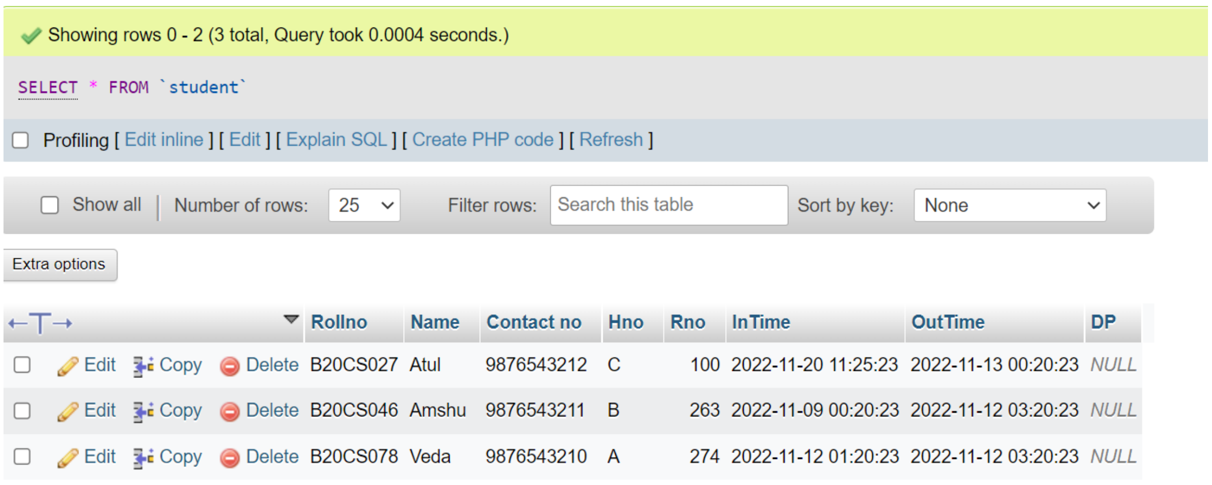
{C_id,id,date,Complaint_status,Complaint}

Hence the table gets normalized after splitting the Global data into 3 lists. As mentioned below.

Normalized tables

After Normalization, the 3 tables that we got are:

- **STUDENT**



Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT * FROM `student`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	Rollno	Name	Contact no	Hno	Rno	InTime	OutTime	DP
Edit Copy Delete	B20CS027	Atul	9876543212	C	100	2022-11-20 11:25:23	2022-11-13 00:20:23	NULL
Edit Copy Delete	B20CS046	Amshu	9876543211	B	263	2022-11-09 00:20:23	2022-11-12 03:20:23	NULL
Edit Copy Delete	B20CS078	Veda	9876543210	A	274	2022-11-12 01:20:23	2022-11-12 03:20:23	NULL

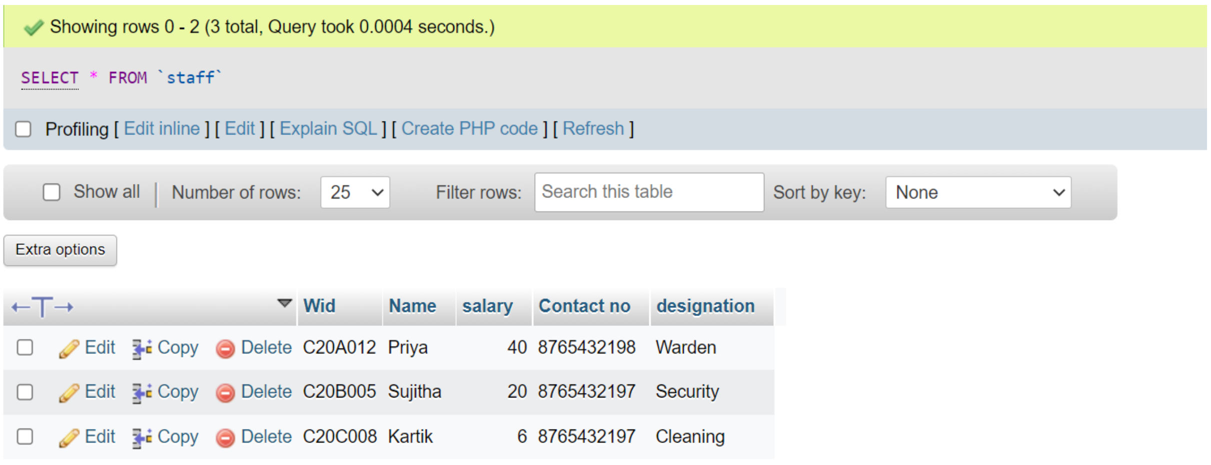
(sample values inserted)

INFORMATION ABOUT ATTRIBUTES:

- **Rollno** is the roll number of the student which is generally represented as B20CS001 where students' roll numbers start from B, the year of joining is the next 2 digits (2020 in this case) , department is the next 2 characters (CSE in this case) and the 3 digit roll number. It is represented as a string on the whole,

- **Name** is the name of the students which is obviously a string.
- **Contact no** is the contact number of the student, initially it was a multivariable attribute but one phone number has been removed to bring them to 1NF. It is represented as a string.
- **Hno** is the hostel number. It is a string, there are 2 hostels for girls A and B and 2 hostels for boys C and D.
- **Rno** is the room number of the person. It is a string.
- **InTime** and **OutTime** are the in time and out times of the student respectively. They are in datetime format {YYYY-MM-DD hh:mm:ss}
- The **DP column** has 2 possible values, YES and NULL. It basically is a non null value if there is any interdisciplinary action that the student has done. He would be given a warning

- **STAFF**



Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

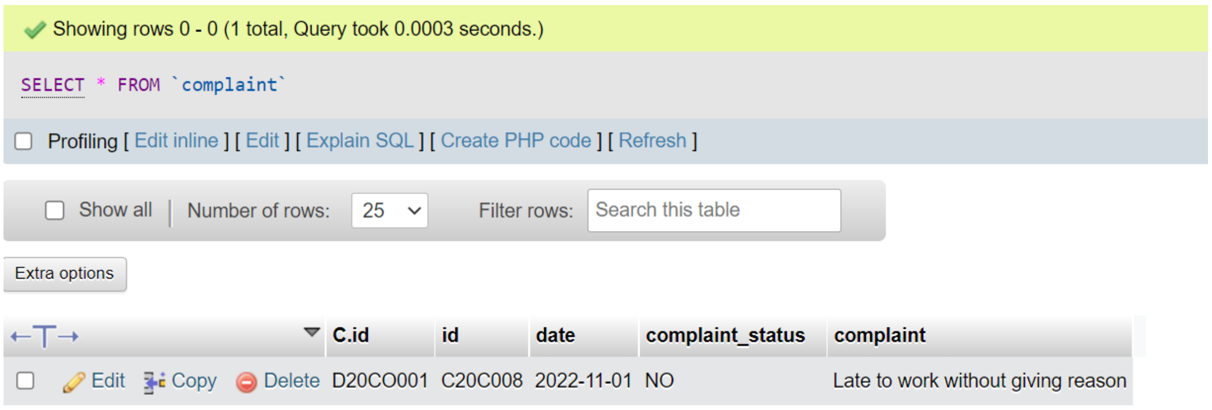
```
SELECT * FROM `staff`
```

Wid	Name	salary	Contact no	designation
C20A012	Priya	40	8765432198	Warden
C20B005	Sujitha	20	8765432197	Security
C20C008	Kartik	6	8765432197	Cleaning

INFORMATION ABOUT ATTRIBUTES

- **Name** is the name of the staff member
- **Wid** is the id of the staff member
- **salary** is the salary of the staff member in thousands (the number in the column * 1000 is the actual salary)
- **Contact no** is the contact number of the staff member.
- **designation** is the designation of the staff member

- **COMPLAINT**


Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

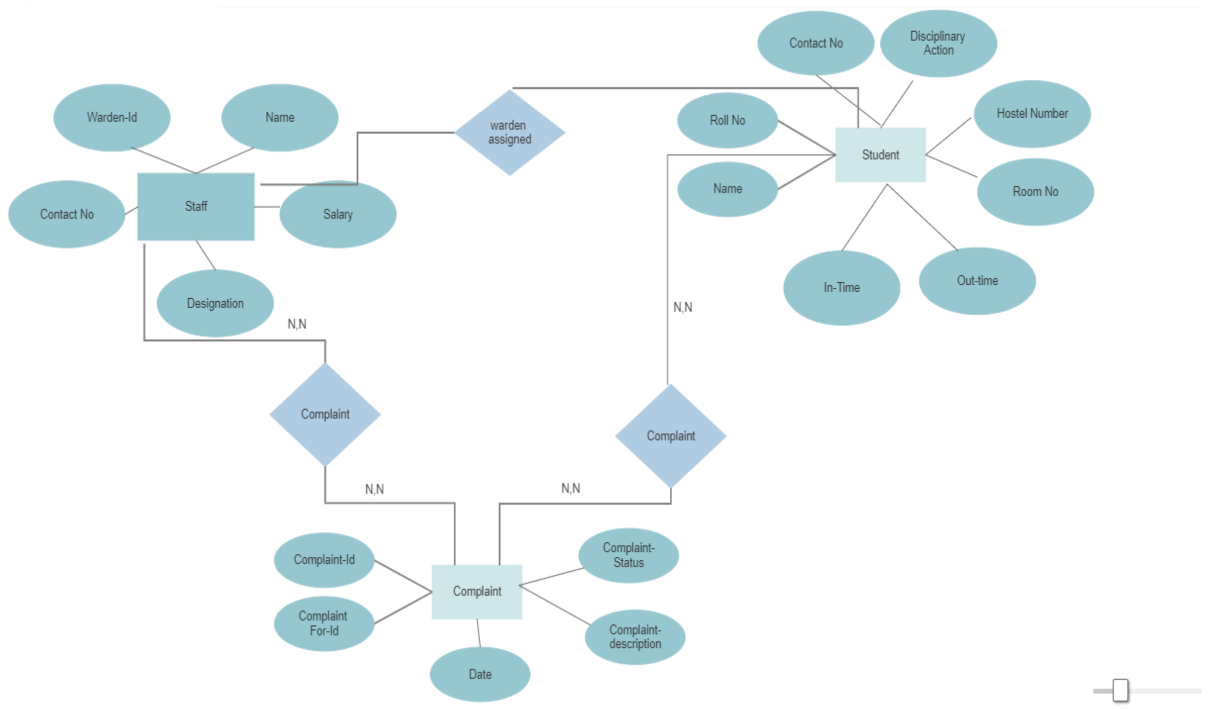
```
SELECT * FROM `complaint`
```

C.id	id	date	complaint_status	complaint
D20CO001	C20C008	2022-11-01	NO	Late to work without giving reason

INFORMATION ABOUT THE ATTRIBUTES

- **C.id** is the complaint id of the complaint.
- **Id** is the id of the complaint holder (the person on whom the complaint was lodged) who can be a student or a staff member. If this column is null in any particular row, it means that the complaint lodged is a general complaint regarding the hostel.
- The person who has complained will remain anonymous to avoid trouble, so the person who has complained will not give his/her details.
- **complaint_status** is the complaint status, it basically has 3 possibilities, YES which means that the complaint is still not resolved, NO, which means that there was a complaint in the past and it was closed, YES* means that the complaint is still ongoing but is a very serious issue.
- If a student gets a complaint under YES, he will be given a warning.
- If any student gets a complaint under YES*, he/she will be given DP.
- If a staff member gets a complaint under YES, his/her salary will be reduced by 8 thousand for a month.
- If a staff member gets a complaint under YES*, he/she will be removed from the campus, and the data will be deleted from the database.

ER diagram



Above attached is the Entity Relationship diagram between Student, Complaint and Staff schema.The schema are interrelated to each other using relation, which exist between them. Here in the diagram schema are mentioned in Rectangular block and the attributes are represented by Oval. The relationships are represented using diamond shapes. Connecting lines are used to represent the relationship between different entities using a relationship between them.

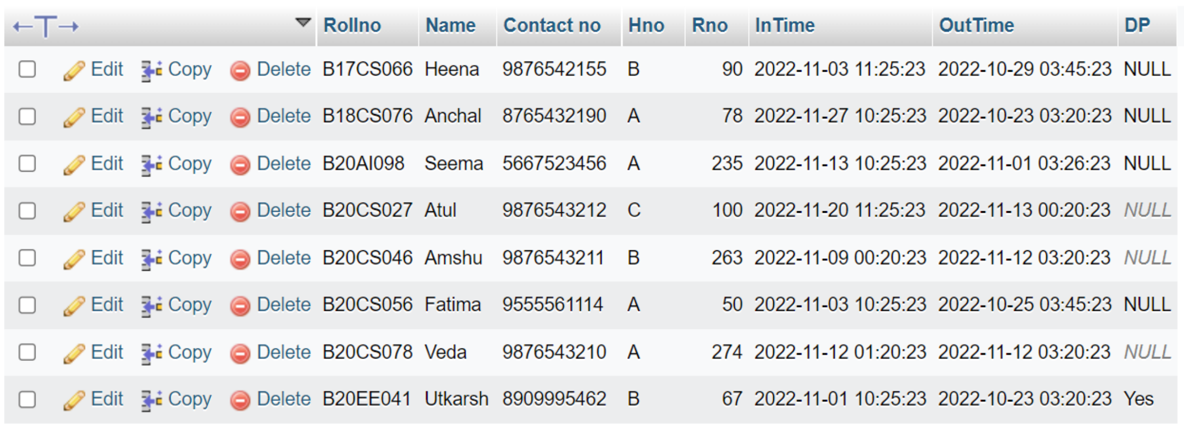
The relationships we used are:

1. Complained relationship: This relationship states that the student can file a complaint or against him/her complaint can be done. Similarly a staff complaint can be done or else he/she can file a complaint.
2. Warden Assigned: This relationship is between student and staff. Every hostel which is assigned to students, they have their respective staff members which include security, cleanliness staff and warden.

More values inserted into the tables

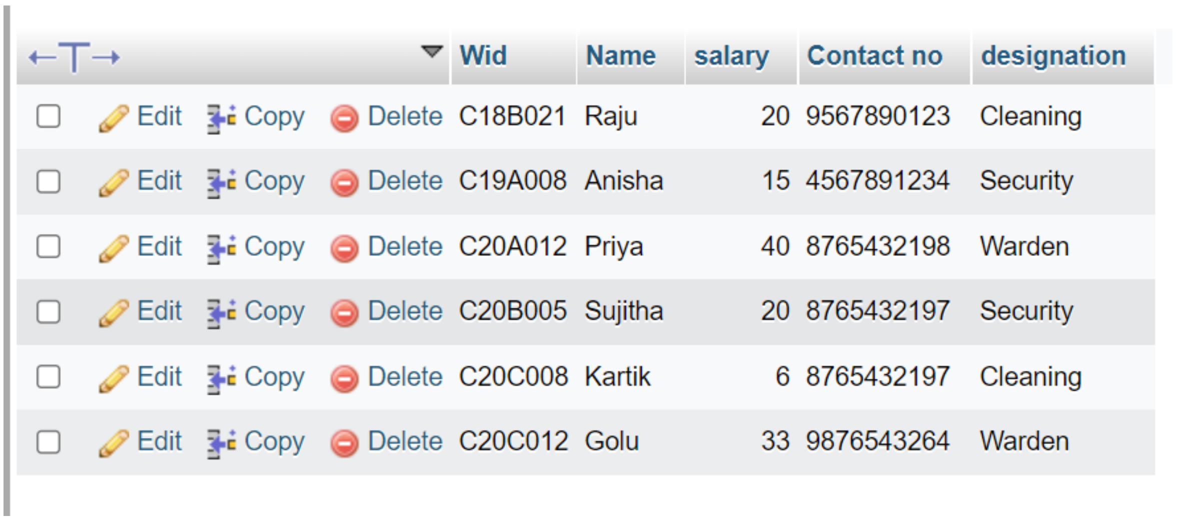
Final tables are :

Student Table



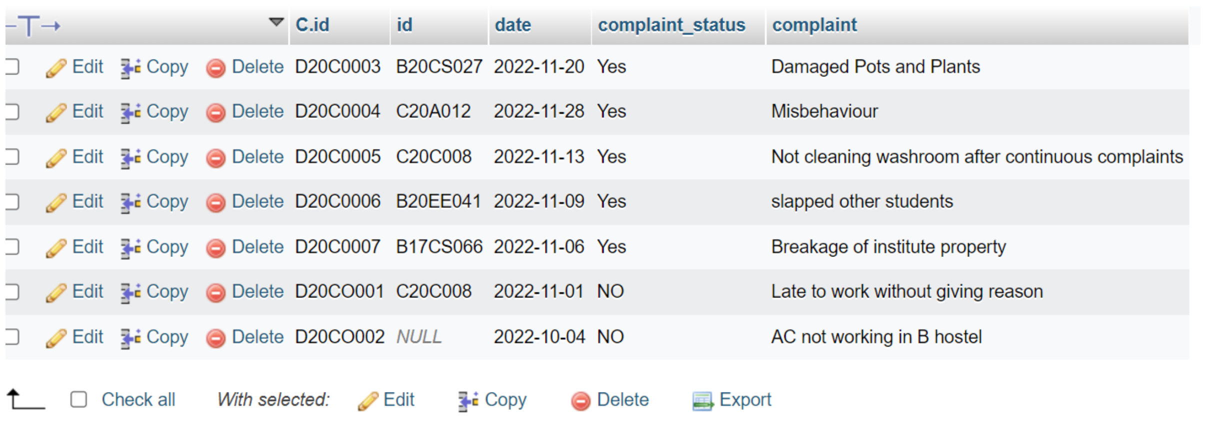
←T→	Rollno	Name	Contact no	Hno	Rno	InTime	OutTime	DP
☐ Edit Copy Delete	B17CS066	Heena	9876542155	B	90	2022-11-03 11:25:23	2022-10-29 03:45:23	NULL
☐ Edit Copy Delete	B18CS076	Anchal	8765432190	A	78	2022-11-27 10:25:23	2022-10-23 03:20:23	NULL
☐ Edit Copy Delete	B20AI098	Seema	5667523456	A	235	2022-11-13 10:25:23	2022-11-01 03:26:23	NULL
☐ Edit Copy Delete	B20CS027	Atul	9876543212	C	100	2022-11-20 11:25:23	2022-11-13 00:20:23	*NULL*
☐ Edit Copy Delete	B20CS046	Amshu	9876543211	B	263	2022-11-09 00:20:23	2022-11-12 03:20:23	*NULL*
☐ Edit Copy Delete	B20CS056	Fatima	9555561114	A	50	2022-11-03 10:25:23	2022-10-25 03:45:23	NULL
☐ Edit Copy Delete	B20CS078	Veda	9876543210	A	274	2022-11-12 01:20:23	2022-11-12 03:20:23	*NULL*
☐ Edit Copy Delete	B20EE041	Utkarsh	8909995462	B	67	2022-11-01 10:25:23	2022-10-23 03:20:23	Yes

Staff Table



←T→	Wid	Name	salary	Contact no	designation
☐ Edit Copy Delete	C18B021	Raju	20	9567890123	Cleaning
☐ Edit Copy Delete	C19A008	Anisha	15	4567891234	Security
☐ Edit Copy Delete	C20A012	Priya	40	8765432198	Warden
☐ Edit Copy Delete	C20B005	Sujitha	20	8765432197	Security
☐ Edit Copy Delete	C20C008	Kartik	6	8765432197	Cleaning
☐ Edit Copy Delete	C20C012	Golu	33	9876543264	Warden

Complaint Table

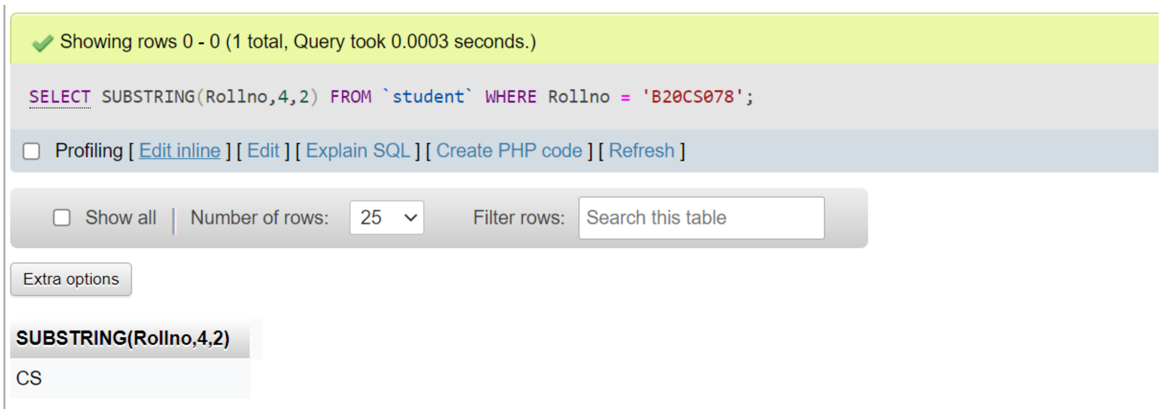


←T→	C.id	id	date	complaint_status	complaint
☐ Edit Copy Delete	D20C0003	B20CS027	2022-11-20	Yes	Damaged Pots and Plants
☐ Edit Copy Delete	D20C0004	C20A012	2022-11-28	Yes	Misbehaviour
☐ Edit Copy Delete	D20C0005	C20C008	2022-11-13	Yes	Not cleaning washroom after continuous complaints
☐ Edit Copy Delete	D20C0006	B20EE041	2022-11-09	Yes	slapped other students
☐ Edit Copy Delete	D20C0007	B17CS066	2022-11-06	Yes	Breakage of institute property
☐ Edit Copy Delete	D20CO001	C20C008	2022-11-01	NO	Late to work without giving reason
☐ Edit Copy Delete	D20CO002	*NULL*	2022-10-04	NO	AC not working in B hostel

☐ Check all *With selected:* Edit Copy Delete Export

Retrieval of data (SQL Queries)

- To get department from a student ID since there is no separate column as department and it is covered in the ID itself (the 4th and the 5th elements of the string are the department)



Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

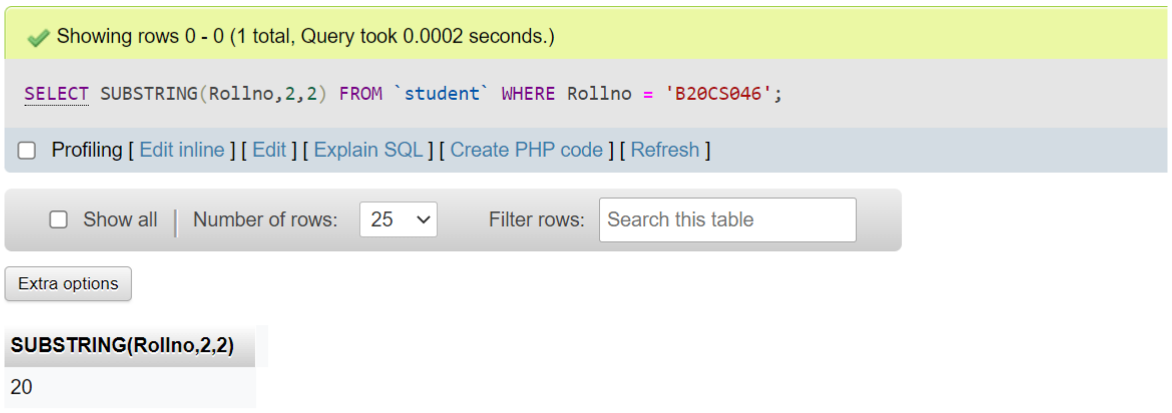
```sql
SELECT SUBSTRING(Rollno,4,2) FROM `student` WHERE Rollno = 'B20CS078';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

SUBSTRING(Rollno,4,2)
CS

So, the student belongs to CSE department

- To get joining year from a student ID since there is no separate column as joining year and it is covered in the ID itself (the 2nd and the 3rd elements of the string are the joining year)



Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

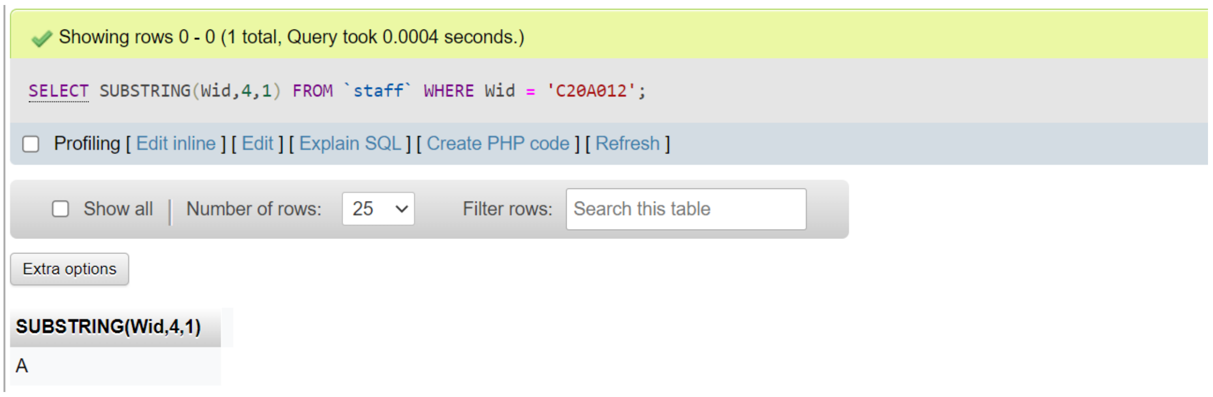
```sql
SELECT SUBSTRING(Rollno,2,2) FROM `student` WHERE Rollno = 'B20CS046';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

SUBSTRING(Rollno,2,2)
20

So the student joined in the year 2020.

- Similar approach can be used to get the joining year for the staff also using the staff id.
- To get the hostel number (A OR B OR C OR D) from the staff id since there is no separate column as hostel number for a particular staff and it is covered in the ID itself (4th element in the string is the hostel number)



Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

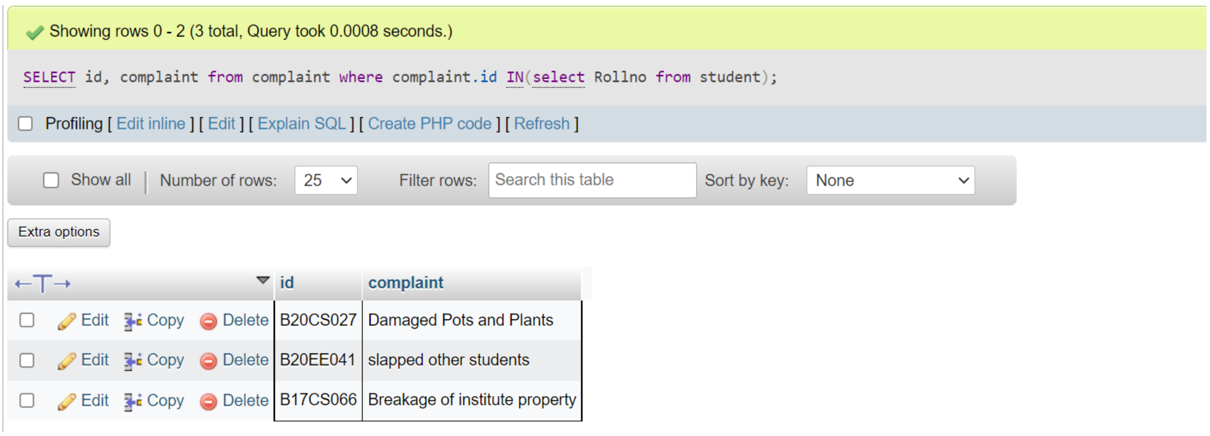
```sql
SELECT SUBSTRING(Wid,4,1) FROM `staff` WHERE Wid = 'C20A012';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

SUBSTRING(Wid,4,1)
A

So the staff member is allotted to the girls hostel A.

- Sql query to find if any complaint lodged against any student



Showing rows 0 - 2 (3 total, Query took 0.0008 seconds.)

```
SELECT id, complaint from complaint where complaint.id IN(select Rollno from student);
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

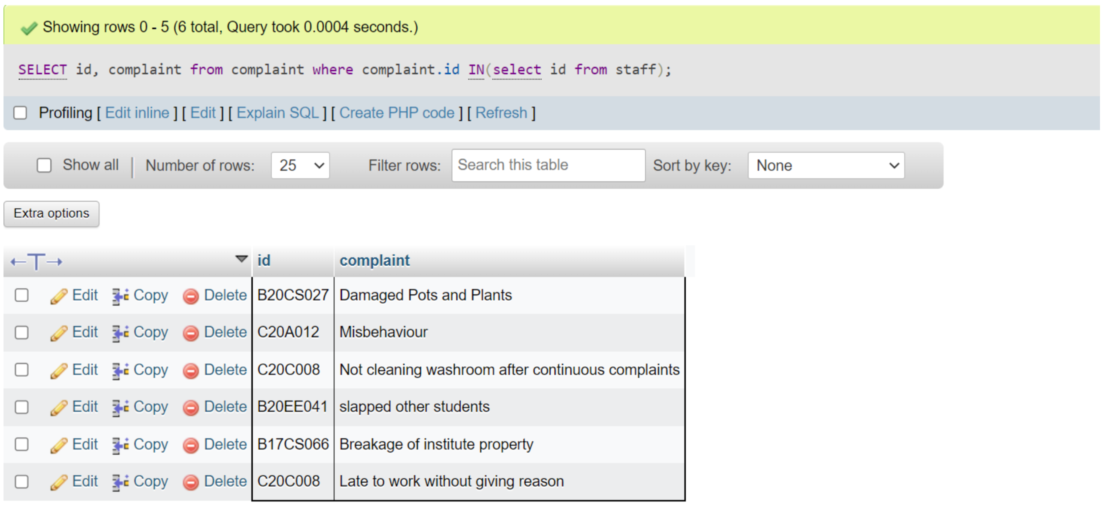
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	id	complaint
Edit Copy Delete	B20CS027	Damaged Pots and Plants
Edit Copy Delete	B20EE041	slapped other students
Edit Copy Delete	B17CS066	Breakage of institute property

So these students have complaints lodged against them

- Sql query to find if any complaint filed against any staff member



Showing rows 0 - 5 (6 total, Query took 0.0004 seconds.)

```
SELECT id, complaint from complaint where complaint.id IN(select id from staff);
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	id	complaint
Edit Copy Delete	B20CS027	Damaged Pots and Plants
Edit Copy Delete	C20A012	Misbehaviour
Edit Copy Delete	C20C008	Not cleaning washroom after continuous complaints
Edit Copy Delete	B20EE041	slapped other students
Edit Copy Delete	B17CS066	Breakage of institute property
Edit Copy Delete	C20C008	Late to work without giving reason

Hashing

APPROACH:

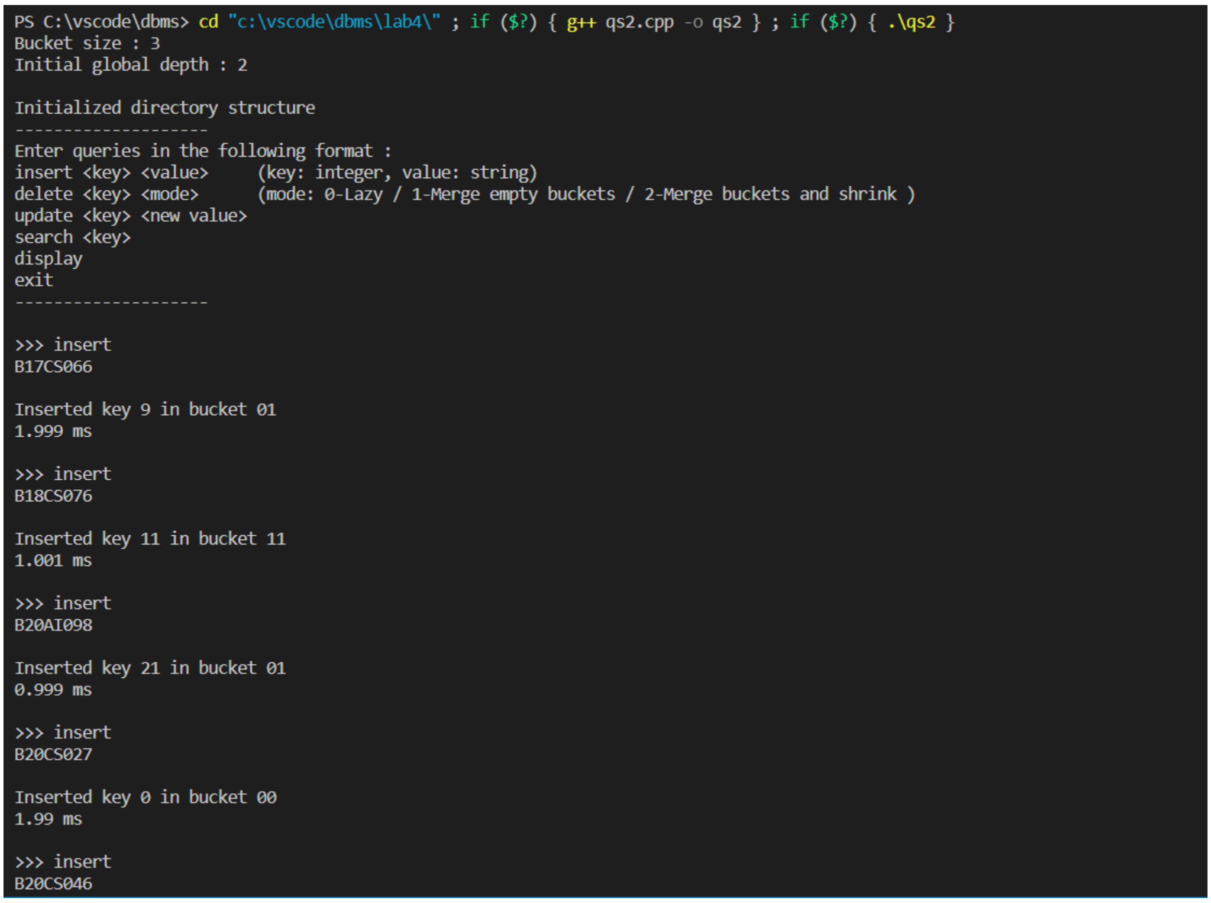
Bucket is a class which depicts the individual bucket of each global

bucket.

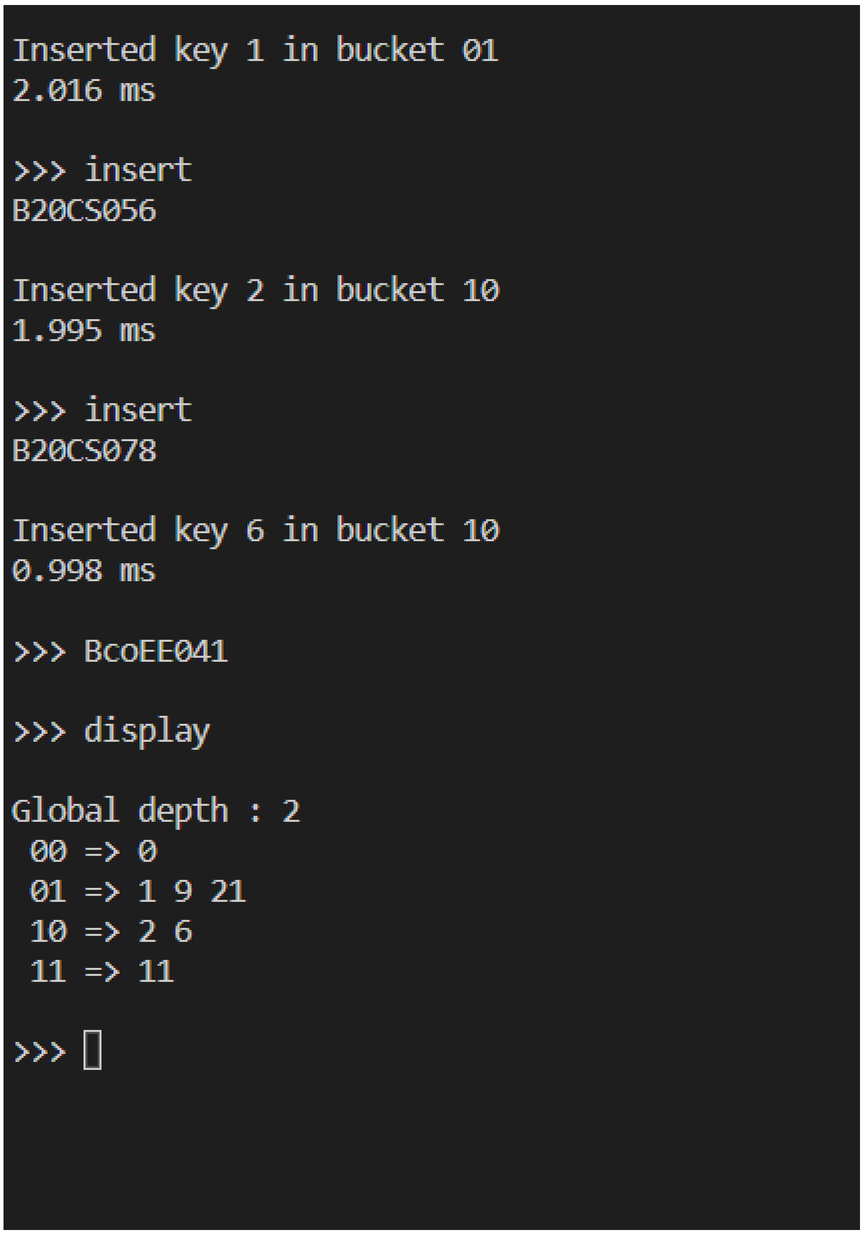
All these individual buckets go into class Directory. Insert function inserts values into the bucket. Delete deletes values. Update updates the values. Search is used to retrieve values. Display is used to display the buckets. And the functions used in Directory class are used to change the global bucket. The functions used in Bucket class are used to change the local bucket. In extendible hashing, initially we start with the global bucket size of 2 and local bucket size is fixed to 4. After there is an overflow, the bucket size increases to 2n which is 4,8,16...and so on. And the values rearrange according to the newly formed buckets.The same process repeats again when there is an overflow.

HASHING:

FOR STUDENT TABLE

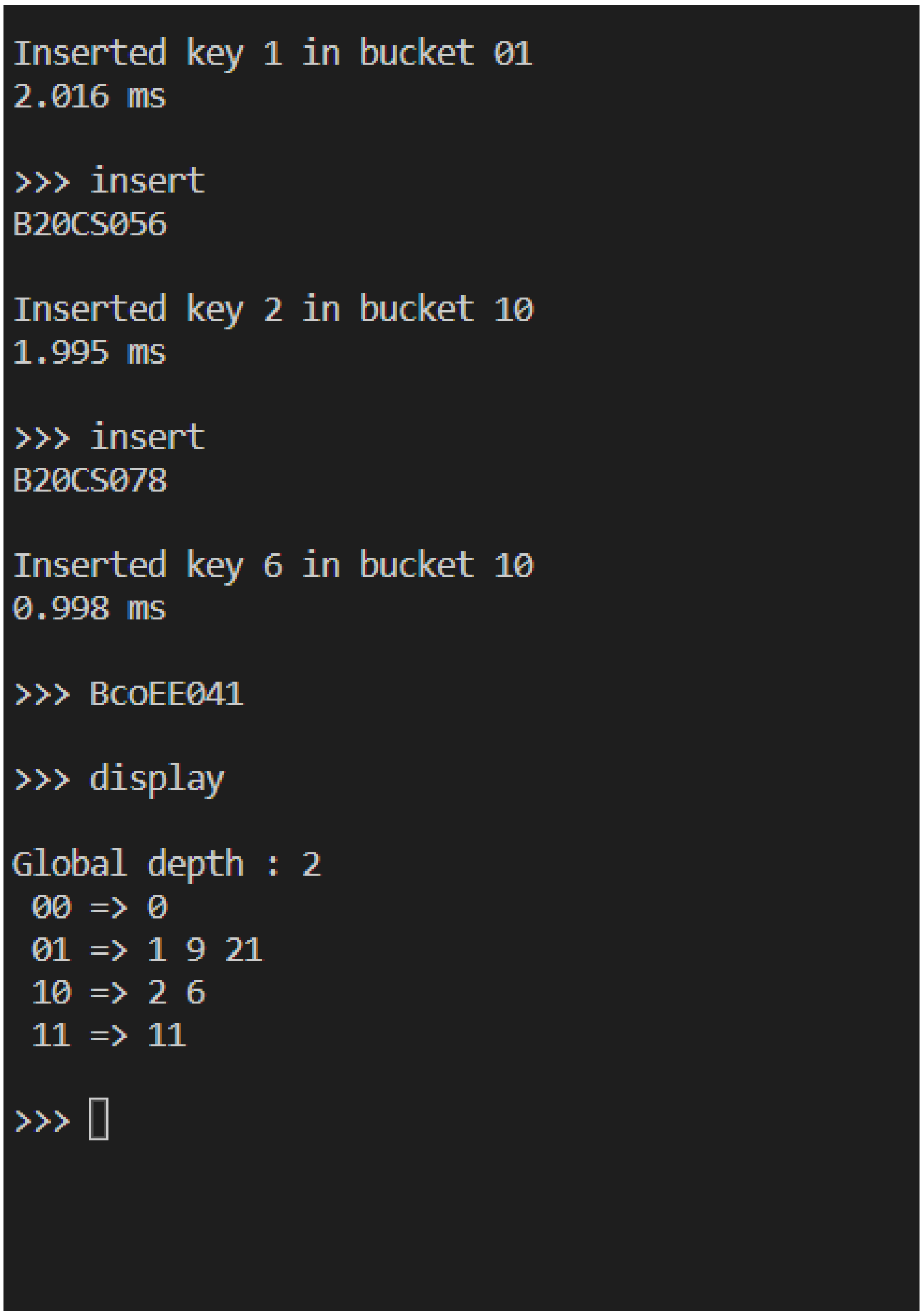


```
PS C:\vscode\dbms> cd "c:\vscode\dbms\lab4\" ; if ($?) { g++ qs2.cpp -o qs2 } ; if ($?) { .\qs2 }
Bucket size : 3
Initial global depth : 2

Initialized directory structure
--------------------
Enter queries in the following format :
insert <key> <value>     (key: integer, value: string)
delete <key> <mode>      (mode: 0-Lazy / 1-Merge empty buckets / 2-Merge buckets and shrink )
update <key> <new value>
search <key>
display
exit
--------------------

>>> insert
B17CS066

Inserted key 9 in bucket 01
1.999 ms

>>> insert
B18CS076

Inserted key 11 in bucket 11
1.001 ms

>>> insert
B20AI098

Inserted key 21 in bucket 01
0.999 ms

>>> insert
B20CS027

Inserted key 0 in bucket 00
1.99 ms

>>> insert
B20CS046
```



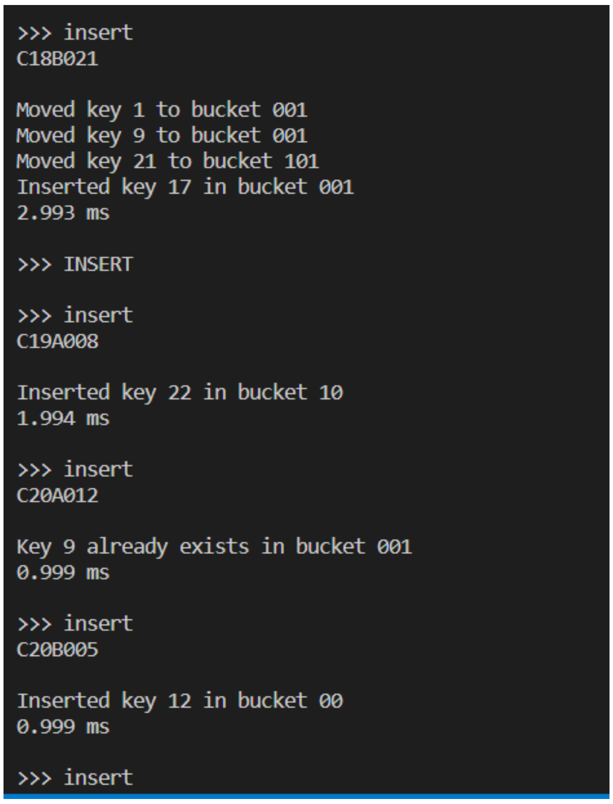
```
Inserted key 1 in bucket 01
2.016 ms

>>> insert
B20CS056

Inserted key 2 in bucket 10
1.995 ms

>>> insert
B20CS078

Inserted key 6 in bucket 10
0.998 ms

>>> BcoEE041

>>> display

Global depth : 2
 00 => 0
 01 => 1 9 21
 10 => 2 6
 11 => 11

>>> 
```

Final output:

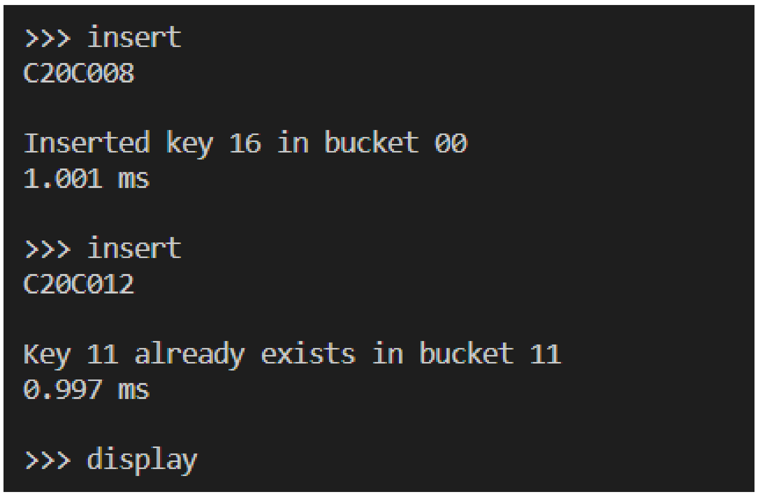


```
Inserted key 1 in bucket 01
2.016 ms

>>> insert
B20CS056

Inserted key 2 in bucket 10
1.995 ms

>>> insert
B20CS078

Inserted key 6 in bucket 10
0.998 ms

>>> BcoEE041

>>> display

Global depth : 2
 00 => 0
 01 => 1 9 21
 10 => 2 6
 11 => 11

>>> 
```

FOR STAFF TABLE

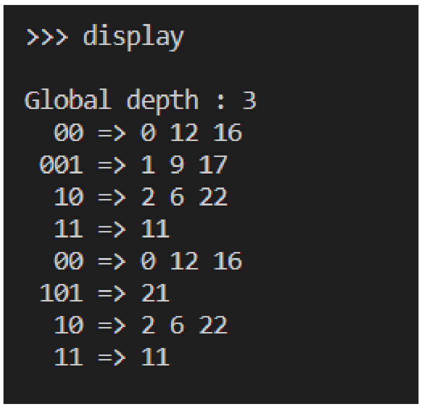


```
>>> insert
C18B021

Moved key 1 to bucket 001
Moved key 9 to bucket 001
Moved key 21 to bucket 101
Inserted key 17 in bucket 001
2.993 ms

>>> INSERT

>>> insert
C19A008

Inserted key 22 in bucket 10
1.994 ms

>>> insert
C20A012

Key 9 already exists in bucket 001
0.999 ms

>>> insert
C20B005

Inserted key 12 in bucket 00
0.999 ms

>>> insert
```



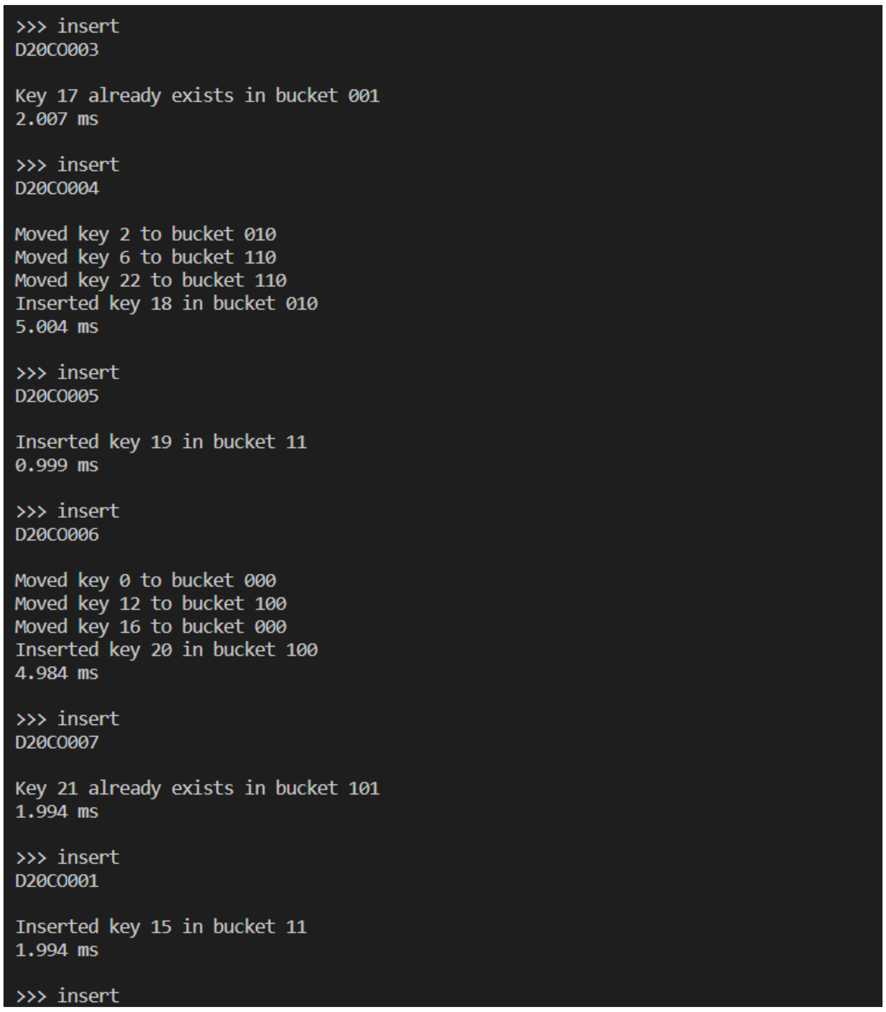
```
>>> insert
C20C008

Inserted key 16 in bucket 00
1.001 ms

>>> insert
C20C012

Key 11 already exists in bucket 11
0.997 ms

>>> display
```

Final Output:



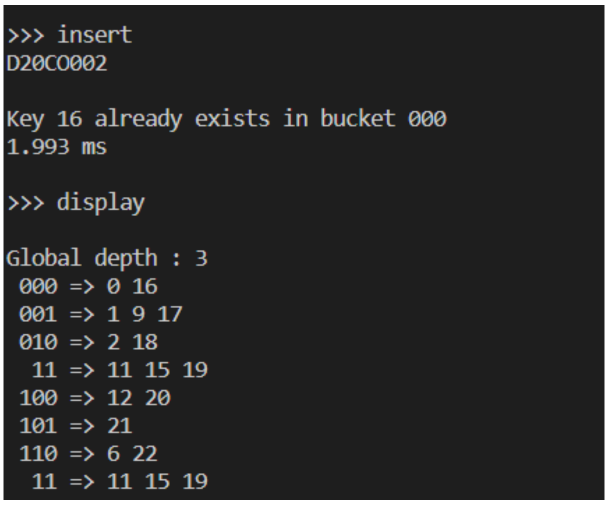
```
>>> display

Global depth : 3
  00 => 0 12 16
 001 => 1 9 17
  10 => 2 6 22
  11 => 11
  00 => 0 12 16
 101 => 21
  10 => 2 6 22
  11 => 11
```

FOR COMPLAINT TABLE



```
>>> insert
D20CO003

Key 17 already exists in bucket 001
2.007 ms

>>> insert
D20CO004

Moved key 2 to bucket 010
Moved key 6 to bucket 110
Moved key 22 to bucket 110
Inserted key 18 in bucket 010
5.004 ms

>>> insert
D20CO005

Inserted key 19 in bucket 11
0.999 ms

>>> insert
D20CO006

Moved key 0 to bucket 000
Moved key 12 to bucket 100
Moved key 16 to bucket 000
Inserted key 20 in bucket 100
4.984 ms

>>> insert
D20CO007

Key 21 already exists in bucket 101
1.994 ms

>>> insert
D20CO001

Inserted key 15 in bucket 11
1.994 ms

>>> insert
```

Final Output:

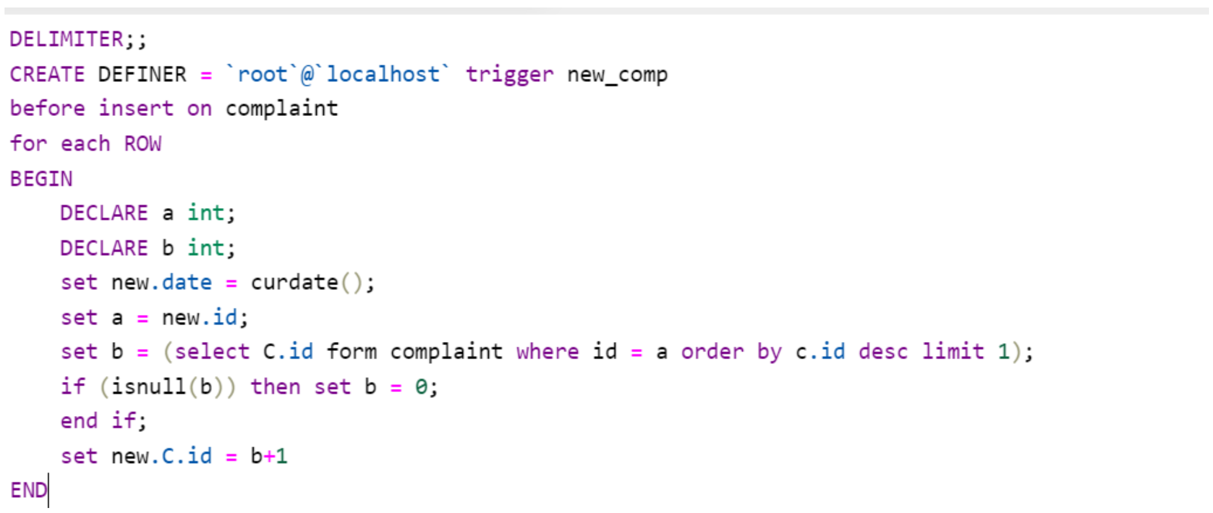


```
>>> insert
D20CO002

Key 16 already exists in bucket 000
1.993 ms

>>> display

Global depth : 3
 000 => 0 16
 001 => 1 9 17
 010 => 2 18
  11 => 11 15 19
 100 => 12 20
 101 => 21
 110 => 6 22
  11 => 11 15 19
```

Use Cases

1. **Automatic recording of datetime while lodging a complaint**

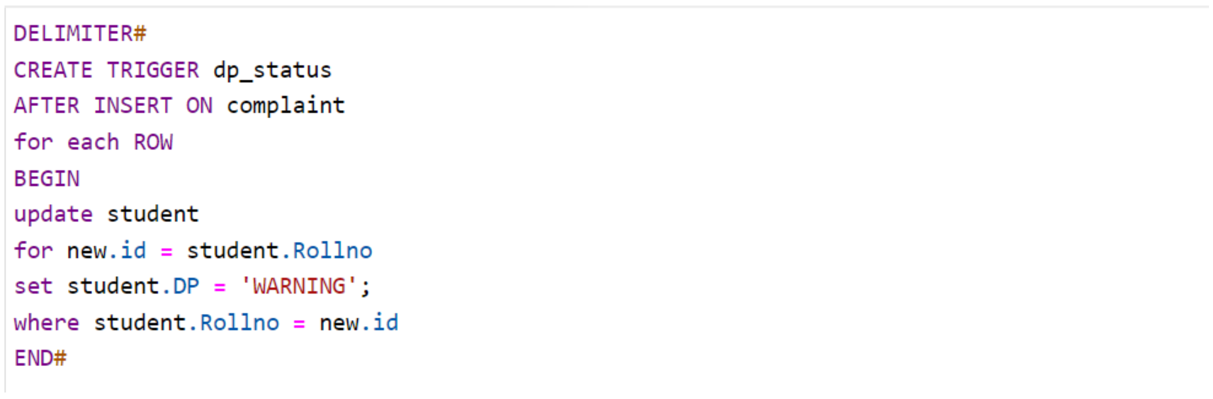
   Before lodging a new complaint, when we initiate a trigger such that it will set the date attribute using the current date from the pc local host, then it will be able to set the date before insert itself. Thus, before inserting every complaint, we don't have to separately insert the date of the complaint, it will automatically be detected and saved in the complaint table.



```
DELIMITER;;
CREATE DEFINER = `root`@`localhost` trigger new_comp
before insert on complaint
for each ROW
BEGIN
    DECLARE a int;
    DECLARE b int;
    set new.date = curdate();
    set a = new.id;
    set b = (select C.id form complaint where id = a order by c.id desc limit 1);
    if (isnull(b)) then set b = 0;
    end if;
    set new.C.id = b+1
END
```

2. **If there is a new registered complaint of a student, give him warning**

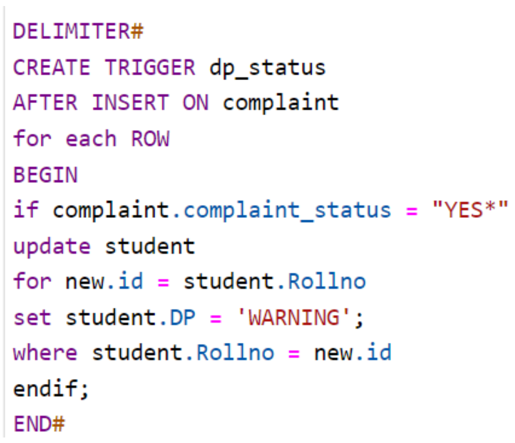
   If there is a new complaint for a student, that is for a particular student id; give him/her a warning, this must be implemented after insert because we need to at first record the complaint and then based on the "id" recorded, the student will be given the warning.



```
DELIMITER#
CREATE TRIGGER dp_status
AFTER INSERT ON complaint
for each ROW
BEGIN
update student
for new.id = student.Rollno
set student.DP = 'WARNING';
where student.Rollno = new.id
END#
```

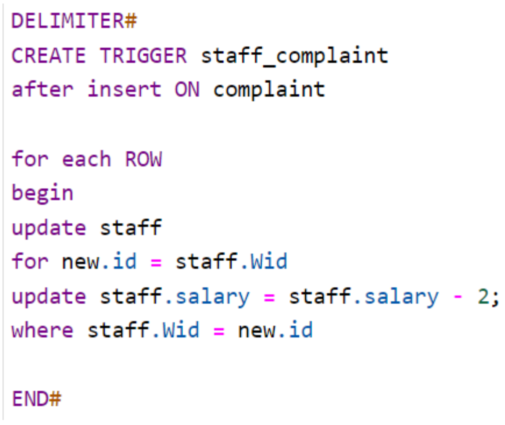
3. **If the new registered complaint of a student is very severe, give him/her a DP (update in the table under DP)**

   If there is a new complaint for a student, that is for a particular student id; give him/her a warning, this must be implemented after insert because we need to at first record the complaint and then based on the "id" recorded, the student will be given the warning.

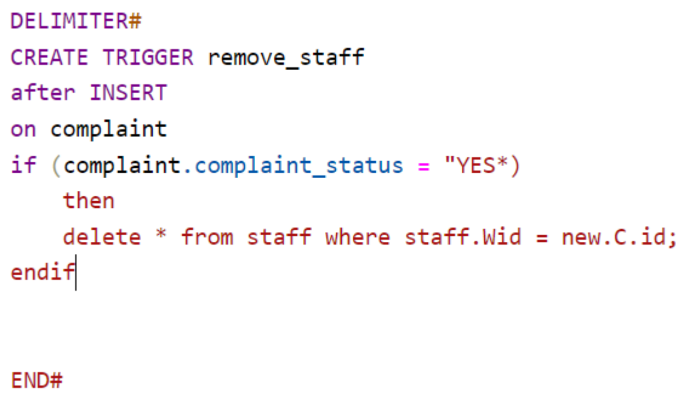

```
DELIMITER#
CREATE TRIGGER dp_status
AFTER INSERT ON complaint
for each ROW
BEGIN
if complaint.complaint_status = "YES*"
update student
for new.id = student.Rollno
set student.DP = 'WARNING';
where student.Rollno = new.id
endif;
END#
```

4. If there is a new registered complaint of a staff member, deduct 2 thousand from the salary and update the table.

Since it is for a new complaint, it should be after insert. So, after inserting the complaint, if the complaint if for a staff, deduct 2 thousand rupees and update the staff table


```
DELIMITER#
CREATE TRIGGER staff_complaint
after insert ON complaint

for each ROW
begin
update staff
for new.id = staff.Wid
update staff.salary = staff.salary - 2;
where staff.Wid = new.id

END#
```

5. If there is a new registered complaint of a staff member which is severe then remove the staff member.


```
DELIMITER#
CREATE TRIGGER remove_staff
after INSERT
on complaint
if (complaint.complaint_status = "YES*)
    then
    delete * from staff where staff.Wid = new.C.id;
endif


END#
```

6. If a student wants to change his hostel number before entering the hostel, update the room number in the database.

```
DELIMITER#
CREATE TRIGGER update_room
before INSERT
on student
UPDATE student.Rno = student.Rno + 10;
where student.Rollno = new.Rollno

END#
```

7. Delete the students who have passed out.

While entering a new tuple in the student database, if the year of joining in the roll number section is already less than or equal to 18, it means that the student has passed out of the college and we are entering useless data because the student doesn’t exist and hence we should use a trigger before insert which will make sure not to insert the data.

```
DELIMITER#
CREATE TRIGGER pass_out_student
BEFORE INSERT ON student
FOR EACH ROW
BEGIN
IF (INT(SELECT SUBSTRING(student.Rollno, 2,2) <=18)
THEN
    delete * from student where student.Rollno = new.Roll.no;
set MESSAGE_TEXT = "pass out student's data deleted";
endif
END#
```

8. Delete the phone number which starts with other digits apart from 9, 8, 7 or 6 because they are invalid and replace them with NULL.

Before updating the student table, it is important to validate the phone number because sometimes wrong phone numbers are retrieved. So, we are creating a trigger invalid_phone_number which updates the student table when the contact number’s first number is less than 6. For example, a number such as 678907892 is an invalid phone number.

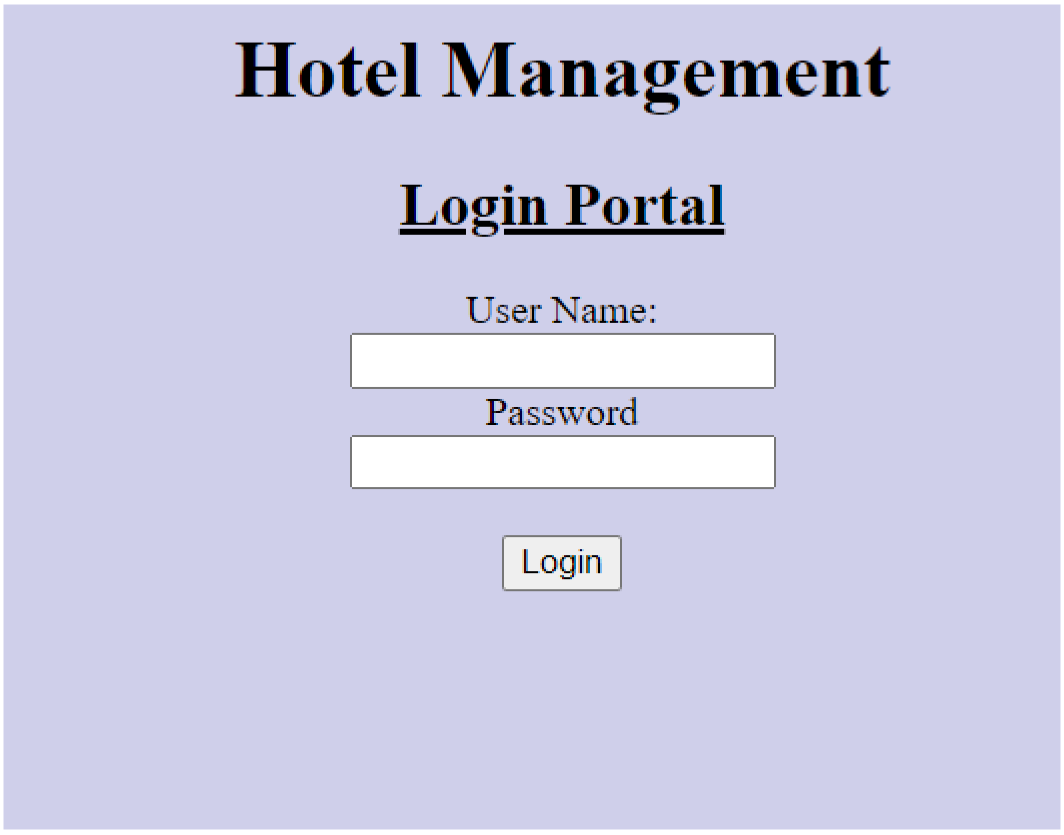
```
DELIMITER#
CREATE TRIGGER invalid_phone_number
ON student
before UPDATE
for each ROW
if ((select substring(student.Contact no,1,1)<6)
    THEN
    update student.Contact no = NULL;
endif

END#
```

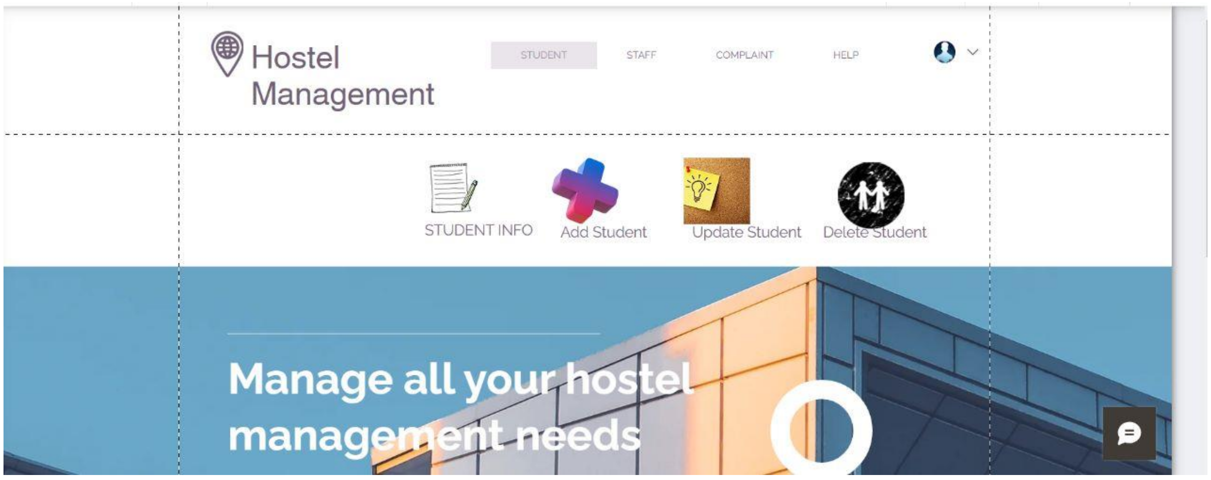
9. If a student’s intime is more than 2 am, give him a warning.

After the student enters the intime after coming to the hostel, we mist check whether he/she is following the rules or not. Because, entering the hostel after 2 am is not allowed and in such a case, a warning must be given to the student. So, create a trigger intime_warning to set the status of DP as “warning”

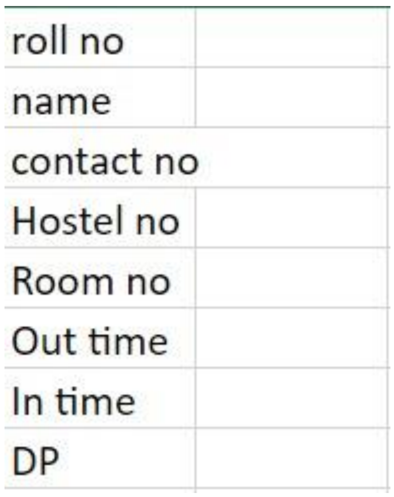
```
DELIMITER#
CREATE TRIGGER intime_warning
ON student
after insert
for each ROW
if (student.InTime(HOUR_MINUTE) > 02:00)
    THEN
    update student.DP  = "warning";
endif

END#
```

Basic Architectural Design (blueprint)

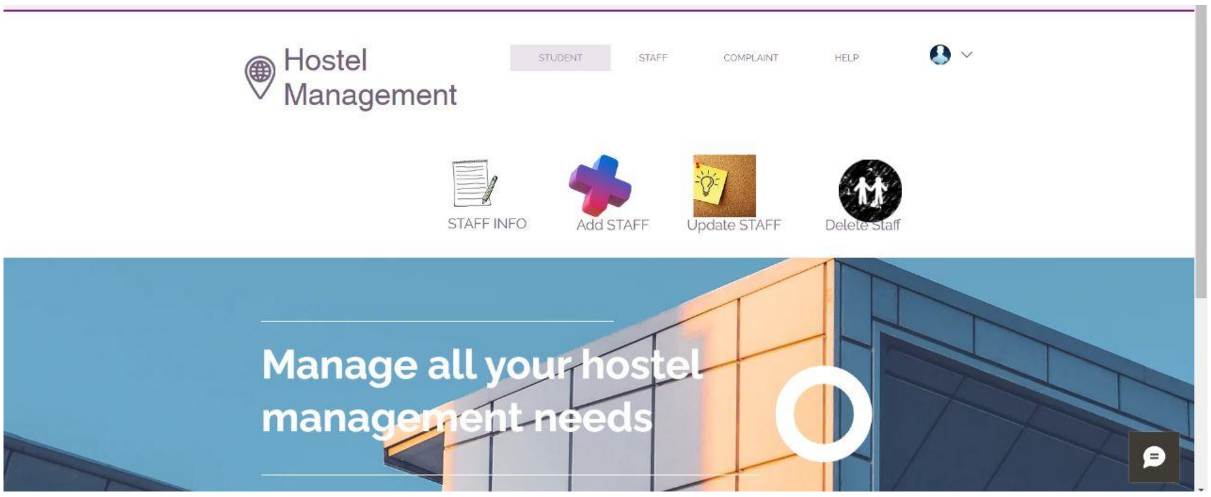




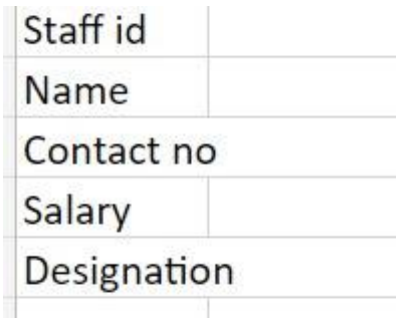
Inside this for each option



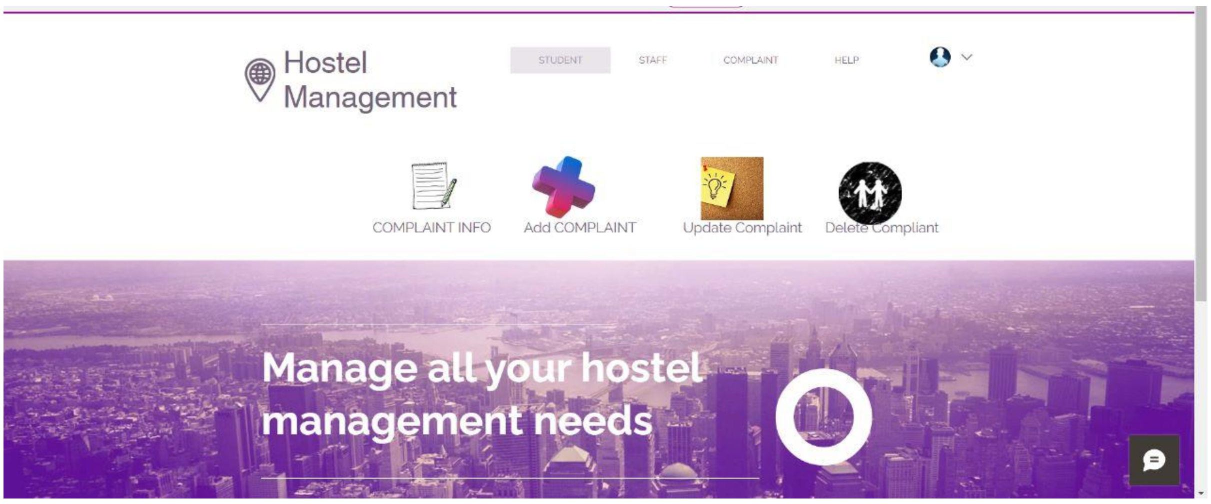
roll no	
name	
contact no	
Hostel no	
Room no	
Out time	
In time	
DP	



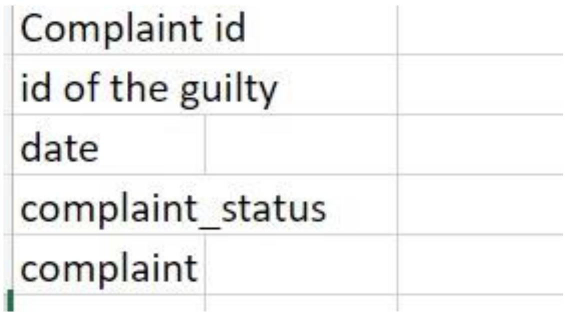
Inside this for each option



Staff id	
Name	
Contact no	
Salary	
Designation	



Inside this for each option



Complaint id	
id of the guilty	
date	
complaint_status	
complaint	

Contributions

All the team members Veda, Amshu and Atul contributed to the project putting in equal efforts. We started by exploring the various data that we can collect in a hostel database. Then, we chose a few main attributes and started to work on them. We ended up getting a very big table with many attributes. We brought them down to 3 main tables after normalization and made the ER diagram. And then we have added other things like a few important SQL queries, Triggers, tried showing a few Transactions and also explored hashing. Finally, we made the report and the video submission. We worked for about roughly 4-5 weeks on the project. We have put in maximum efforts and are hoping for a good score.

What we showed in the project in short

- Explored the various data that can be collected from a hostel database.
- Normalized and divided them into tables.
- Drew the ER diagram.
- Put in a lot of thought about how to represent this which results in immediate access and came up with hashing.
- Showed SQL queries on the database
- Did Hashing on the all the 3 entities
- Thought of use cases and the triggers for the cases.
- Showed the basic architectural design (blueprint)

SPECIAL THANKS TO ROMI MA’AM FOR HELPING US THROUGHOUT THE COURSE AND THE PROJECT!!

—THE END—

v