

COMPUTER SCIENCE & ENGINEERING DEPARTMENT
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
ASSAM 788010, INDIA



SUBJECT :::: DATABASE MANAGEMENT SYSTEM
SUB CODE:::::: CS 302

End Semester Mini Project

---: Submitted To :---

Dr. Samir Kumar Borgohain
Assistant Professor
Department: CSE

---: Submitted by :----

Aayush Jalan(1815027)
Nihar Jyoti Baishya(1815028)
Mainak Deb(1815029)
Swarnav Das (1815030)
Angshuman Sarma (1815031)

Contents

<u>Topic</u>	<u>Page No</u>
1. Entity Relationship(ER) Diagram.....	03
2. Conversion of ER Diagram to Relational Model.....	07
3. Normalization.....	08
4. Assumptions.....	16
5. Code.....	17
6. Conclusion.....	36

Entity Relationship(ER) Diagram

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in a database, so by showing relationships among tables and their attributes, the ER diagram shows the complete logical structure of a database.

An ER diagram has three main components:

1. Entity
2. Attribute
3. Relationship.

1. Entity:

An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.

2. Attribute:

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram.

● Key attribute:

A key attribute can uniquely identify an entity from an entity set. Key attribute is represented by oval same as other attributes however the **text of key attribute is underlined**. Such types of key attributes are called primary keys.

● Composite attribute:

An attribute that is a combination of other attributes is known as a composite attribute.

● Multi valued attribute:

An attribute that can hold multiple values is known as a multi valued attribute.

- **Derived attribute:**

A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram.

3. Relationship:

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

a) One to One Relationship:

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.

b) One to Many Relationship:

When a single instance of an entity is associated with more than one instance of another entity then it is called one to many relationship.

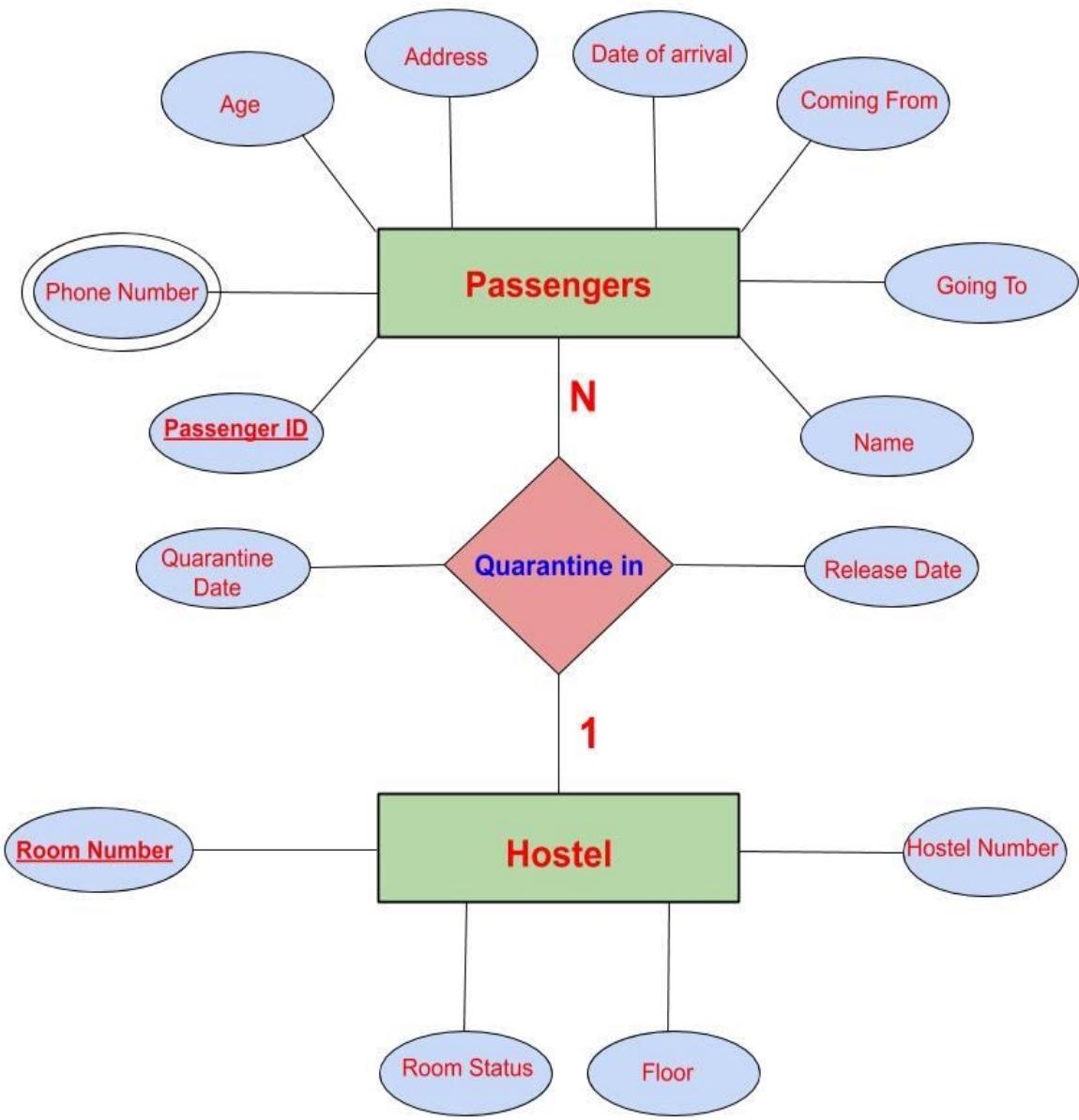
c) Many to One Relationship:

When more than one instance of an entity is associated with a single instance of another entity then it is called many to one relationship.

d) Many to Many Relationship:

When more than one instance of an entity is associated with more than one instance of another entity then it is called many to many relationship.

In our question we are asked to design a ER diagram of quarantine facility management system specifying constraints , cardinalities etc. The diagram drawn below is the required ER diagram for the quarantine facility management system.



In the ER diagram we make two entity sets, passengers and hostel. The relationship between the passengers and the hostel is many to one because no passenger can stay in more than one hostel but many passengers can live in the same hostel.

For the entity set “passenger”, we give 8 attributes where we consider passenger ID as the primary key. Phone number is the multi valued attribute in the passenger entity set.

For the relationship set “Quarantine in”, we declare two attributes named as quarantine date and release date.

For the entity set “hostel”, we give 4 attributes where we consider room number as the primary key.

Now let us describe the meaning of the attributes from the entity sets and the relationship set.

Entity set name: Passenger

Attributes of this entity set are :

- **Phone number:** It contains the phone numbers of the passengers. It is a multi valued attribute.
- **Age:** It contains the age of the passengers.
- **Address:** It contains the address of the passengers.
- **Date of arrival:** It contains that date at which the passengers has arrived .
- **Coming from:** It contains the name of the place from where the passengers are coming.
- **Going to:** It contains the name of the places to where the passengers are going.
- **Name:** It contains the name of the passengers.
- **Passenger id:** It contains the unique id of the passengers.

Entity set name: Hostel

Attributes of this entity set are :

- **Room number:** It contains the unique room numbers of the hostel rooms.
- **Hostel number:** It contains the number of hostels.
- **Floor:** It contains the floor number .
- **Room status:** It contains the status of the room whether it is vacant or not. If vacant then status is EMPTY else status is FULL.

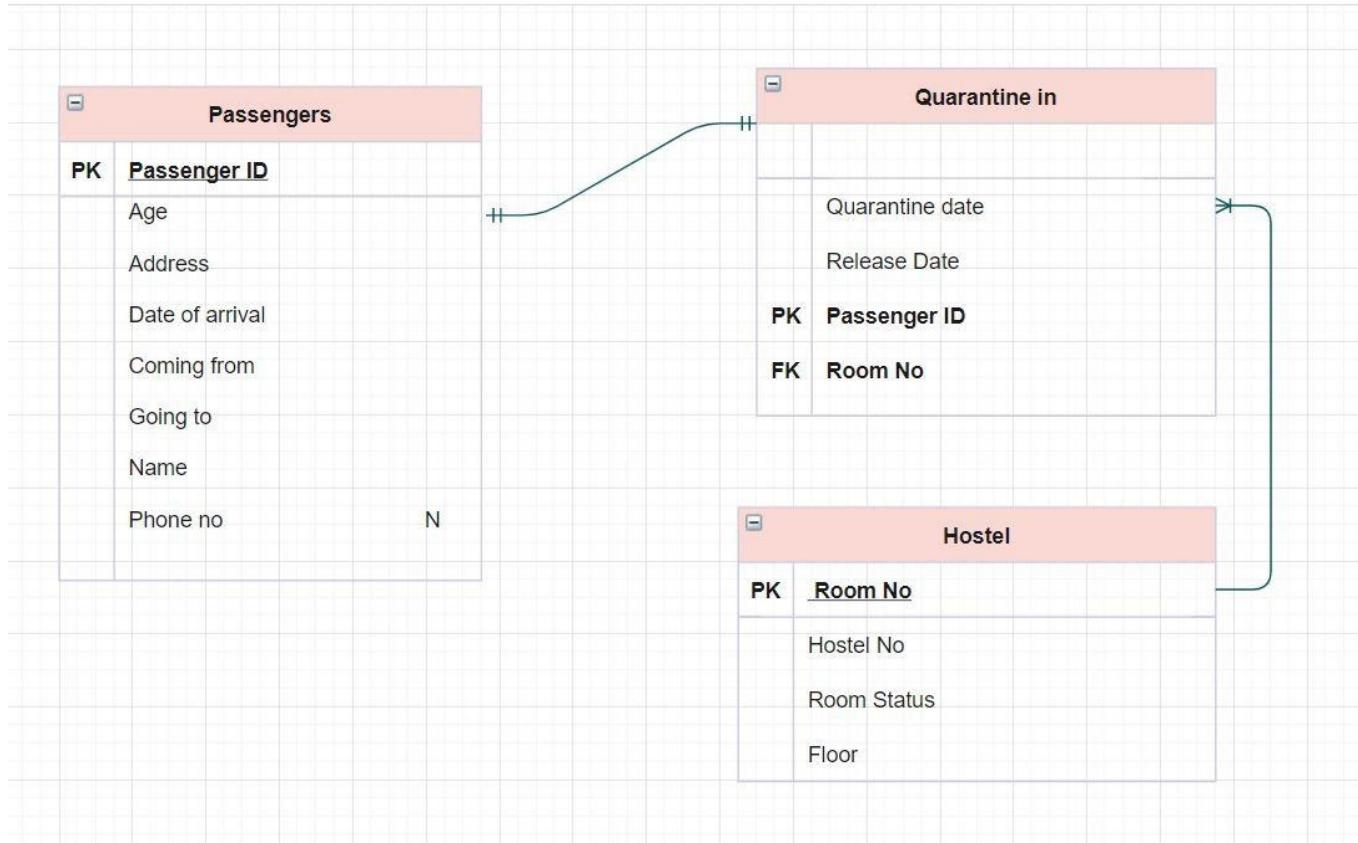
Relationship set name: Quarantine in

Attributes of this entity set are :

- **Quarantine date:** It contains the dates on which the passengers are kept in quarantine.
- **Release date:** It contains the dates on which the passengers are released from quarantine.

Conversion of ER Diagram to Relational Model

The Relational Model for our concerned ER Diagram is shown below:



The above relational model consists of many redundancies. Redundancy in a relation may cause insertion, deletion and update anomalies in the relational database.

Hence, in order to minimize the redundancies occurring in the above relational model, we need to perform normalization on it.

Normalization:

The process of minimizing redundancy from a relation or set of relations is called Normalization. Normal forms are used to eliminate or reduce redundancy in database tables.

In order to minimize the redundancies in our Relational model, we first convert it to 1st Normal Form, then to 2nd Normal Form and lastly to 3rd Normal Form.

Converting our Relational Model to 1st Normal Form:

A relational schema R is in 1st Normal Form if:

- (a) The domains of all attributes of R are atomic.
- (b) It does not contain any composite or multi-valued attribute.

In our relational model, we do not have any composite attribute. But, the “Passengers” entity consists of the multi-valued attribute “Phone Number” as shown in the following table.

Passengers

Passenger ID	Name	Phone Number
001	Ram	8842152345, 9256424532
002	Shyam	8126745241, 8912456721
005	Gaurav	9435725568

Thus, from the above table, it can be clearly seen that the above relational table is not present in 1st Normal Form and hence demands some suitable conversion.

We can consider the following methods to tackle this situation:

1. We can add a new attribute for every value of the multi-valued attribute.

This method is shown in the table below:

Passengers

Passenger ID	Name	Phone Number 1	Phone Number 2
001	Ram	8842152345	9256424532
002	Shyam	8126745241	8912456721
005	Gaurav	9435725568	

This is not a good method since we have two attributes representing the same thing, and the number of phone numbers for a particular entity may not be unique.

2. We can rewrite the values of other attributes for each of the values of multi-valued attributes.

This method is shown in the table below:

Passengers

Passenger ID	Name	Phone Number
001	Ram	8842152345
001	Ram	9256424532
002	Shyam	8126745241
002	Shyam	8912456721
005	Gaurav	9435725568

This is also a bad method since Passenger ID is no longer a primary key of the entity Passengers. It, in turn, introduces redundancy in the relationship.

3. We can separate the multi-valued attribute from the existing relation, thereby forming two different relations.

This method is shown in the tables below:

Passengers

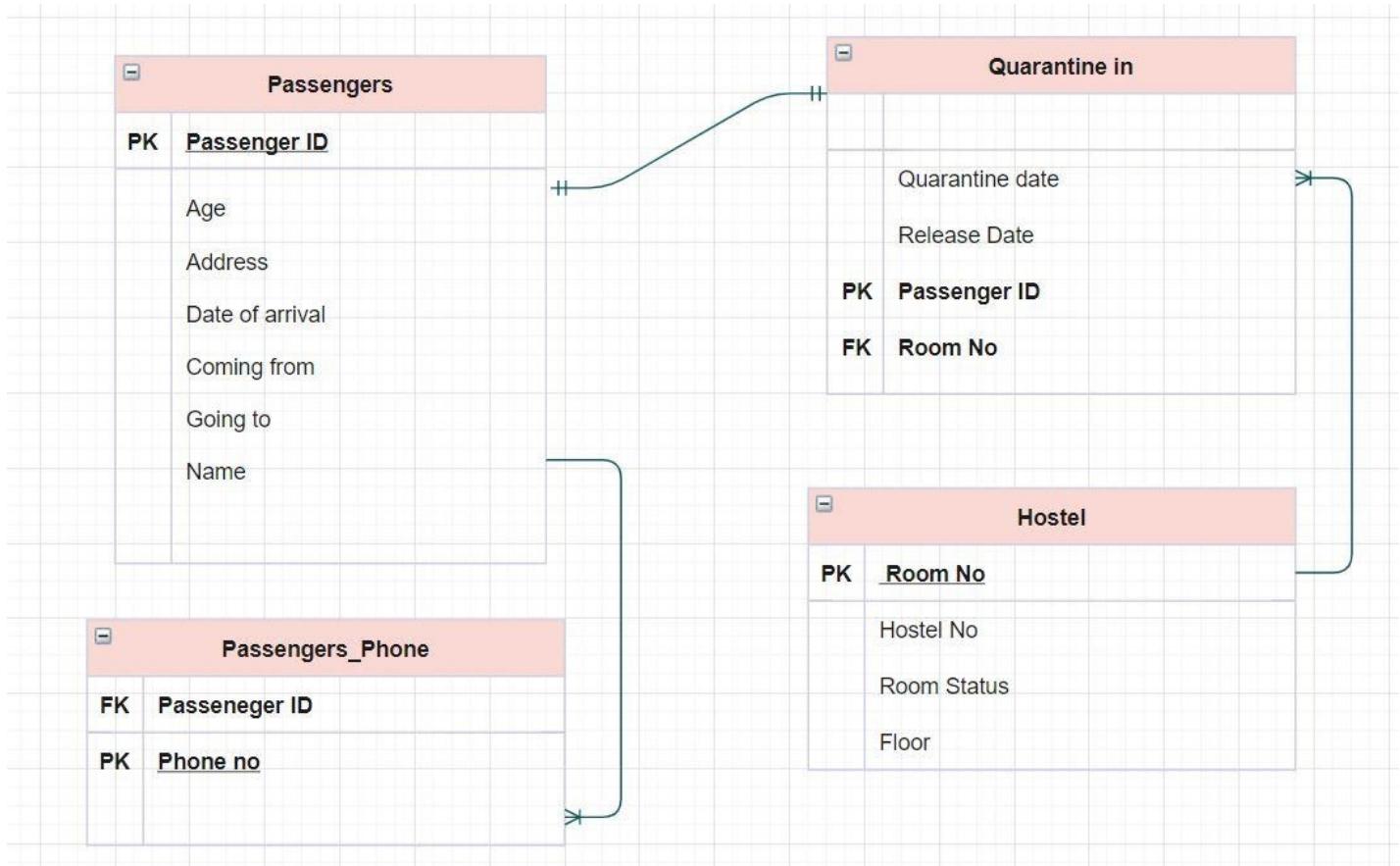
Passenger ID	Name
001	Ram
002	Shyam
005	Gaurav

Passengers_Phone

Passenger ID	Phone Number
001	8842152345
001	9256424532
002	8126745241
002	8912456721
005	9435725568

Passenger ID remains the primary key of the first relational table Passengers, while the primary key of the new relational table Passengers_Phone is {Phone Number, Foreign Key Passenger ID}.

This method can be represented by the following relational model:



The above relational model is one of the best possible solutions in 1st Normal Form since it does not contain any duplicate information, and hence is less redundant.

Converting our Relational Model to 2nd Normal Form:

A Relational schema R is in 2nd Normal Form iff:

- (a) It is in 1st Normal Form, and
- (b) It has no Partial Dependency, i.e., no non-prime attribute is dependent on any proper subset of any candidate key of R.

[Non-prime attribute: Attributes which are not part of any candidate key.]

In our previous section, we have already converted our relational model into 1st Normal Form. The relational tables now include “Passengers”, “Hostel” and “Passengers_Phone”.

Here, “Passengers” contains the primary key Passengers ID. Passengers ID is also the only candidate key in the relation, as all other attributes like Age, Name, Date of Arrival etc are non-prime attributes. Since the relational table “Passengers” contains only a single candidate key, hence there is no proper subset of the candidate key of “Passengers”. Therefore, there is no Partial dependency associated with it and hence, “Passengers” is in 2nd Normal Form.

Similarly, the relational table “Hostel” also contains a single candidate key Room No and does not have any proper subset of the candidate key, hence no partial dependency. Thus, “Hostel” is also in 2nd Normal Form.

In the relational table “Passengers_Phone”, both Phone Number and Passenger ID are candidate keys. There is no non-prime attribute and hence, no partial dependency. Therefore, “Passengers_Phone” is also in 2nd Normal Form.

From the above argument, it can be concluded that the obtained relational model was already in 2nd Normal Form.

Converting our Relational Model to 3rd Normal Form:

A Relational Schema R is in 3rd Normal Form iff:

- (a) It is in 2nd Normal Form, and
- (b) there is no transitive dependency for non-prime attributes.

[Transitive dependency: If A->B and B->C are two Functional Dependencies, then A->C is called transitive dependency.]

Also,

A relation is in 3rd Normal Form if at least one of the following conditions holds in every non-trivial functional dependency A → B:

- (a) A is a super key.
- (b) B is a prime attribute (each element of B is part of some candidate key).

In our previous section, we have already concluded that the obtained relational model of the concerned ER diagram is in 2nd Normal Form. Now, for determining whether our relational model is in 3rd Normal Form or not, we need to check if there exists any transitive dependency for non-prime attributes of the three relational tables “Passengers”, “Hostel” and “Passengers_Phone”.

- Let us consider the relational table “Passengers_Phone” first. The attributes of “Passengers_Phone” are both candidate keys. Thus, there are no non-prime attributes in this table and hence, no transitive dependency.

Therefore, “Passengers_Phone” is already in 3rd Normal Form.

- Let us consider the “Passengers” relational table now. We will be taking some pair of attributes and check if there exists a transitive dependency for any non-prime attributes in this relation.

As there always exists a functional dependency between the candidate key “Passengers ID” and any other non-prime attribute, we are going to check if there exists a functional dependency between any two pair of non-prime attributes in the relational table.

Passengers

Age	Date of Arrival
25	12/10/2020
25	09/10/2020
48	05/10/2020
62	05/10/2020

Taking an example of Age and Date of Arrival as shown in the above table, it can be clearly observed that for the same age there can be different dates of arrival depending upon the person. Here, for Age = 25, there are two possible (12/10/2020 and 09/10/2020) Dates of

Arrival. Thus, there cannot exist any Functional Dependency between the attributes Age and Date of Arrival. Similarly, by checking for Age and other non-prime attributes, it can be found that no functional dependency exists for the attribute Age and any other non-prime attribute of “Passengers” relation. This is shown by the following table:

Functional Dependency between Attribute 1 and Attribute 2

Attribute1	Attribute2	Valid
Age	Address	No
Age	Date of Arrival	No
Age	Coming from	No
Age	Going to	No
Age	Name	No

Similarly, we can check for all other non-prime attributes of “Passengers” relation by making each possible pair and then checking with the help of different possible examples.

It can be observed that for any two non-prime attributes X and Y of “Passengers” relation, the functional dependency $X \rightarrow Y$ does not hold. Hence, it can be concluded that there does not exist any transitive dependency for any non-prime attribute of this relation.

Therefore, the “Passengers” relational table is also in 3rd Normal Form.

- For the “Hostel” relational table, there always exists a functional dependency between the candidate key “Room No” and any other non-prime attribute. So, similar to the procedure followed for the “Passengers” relational table, we are going to check if there exists a functional dependency between any two pair of non-prime attributes in “Hostel” relation.

Hostel

Hostel No	Room Status
1	EMPTY
2	FULL
1	FULL
2	FULL

Taking an example of Hostel No and Room Status as shown in the above table, it can be observed that for Hostel No = 1, there are two possible Room Status (EMPTY and FULL). Thus, there cannot exist any Functional Dependency between the attributes Hostel No and Room Status.

Similarly, by checking for Hostel No and other non-prime attributes, it can be found that no functional dependency exists for this non-prime attribute in “Hostel” relation. This is shown by the following table:

Functional Dependency between Attribute 1 and Attribute 2

Attribute1	Attribute2	Valid
Hostel No	Room Status	No
Hostel No	Floor	No

As for the other non-prime attributes of “Hostel” relation, we can also check and find that in fact there does not exist any functional dependency between its non-prime attributes. Hence, it can be concluded that transitive dependency is not possible for any non-prime attributes in this relation.

Therefore, the “Hostel” relational table is already in 3rd Normal Form.

Since all three relational tables “Passengers”, “Hostel” and “Passengers_Phone” already exist in 3rd Normal Form, we need not decompose our relational model further. The redundancy in the relational database (which gives rise to insert, delete and update anomalies) is now completely minimized.

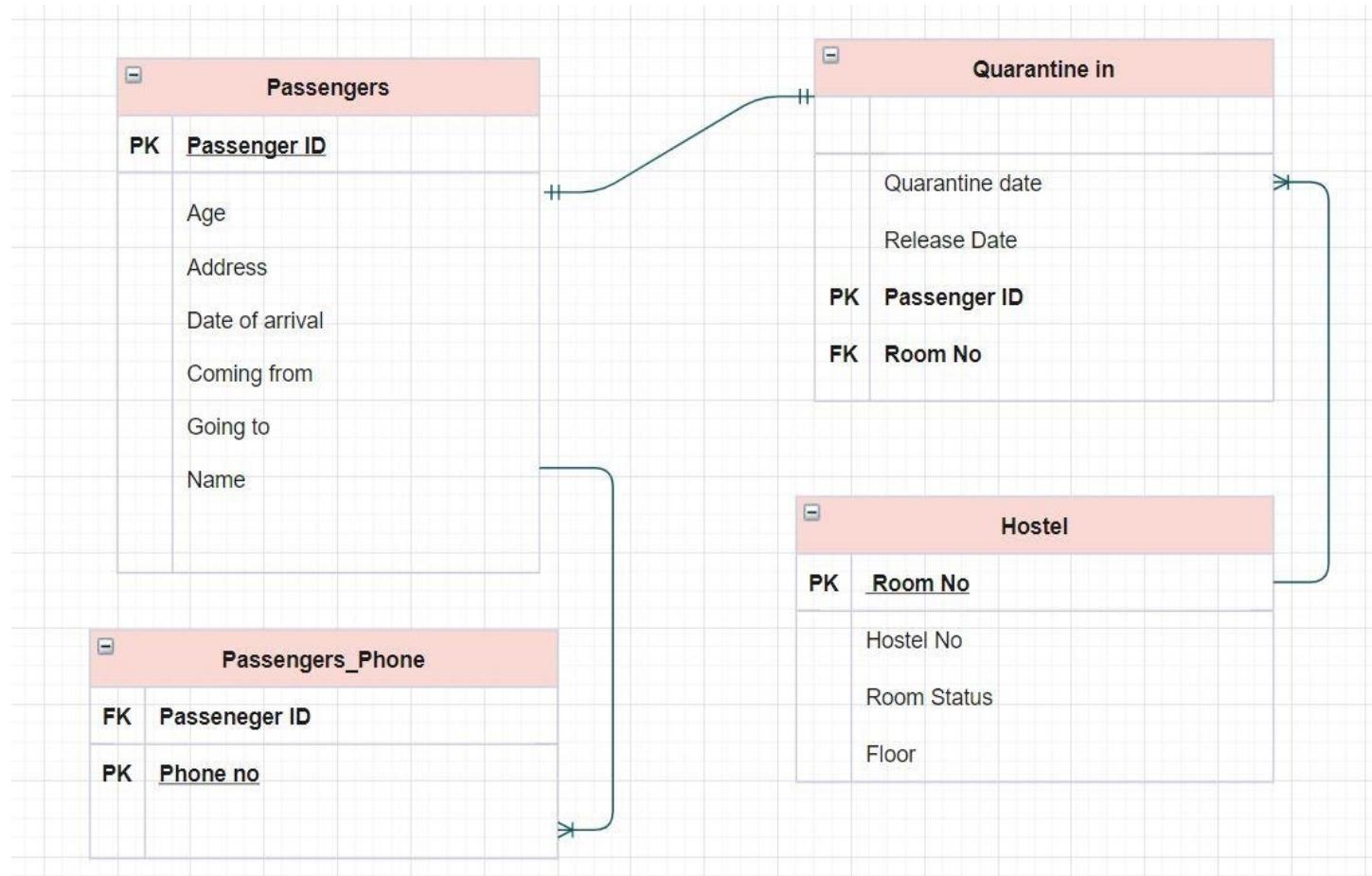
Hence, the normalized relational schema can be represented by the following relations:

Passengers (Passenger ID, Age, Address, Date of arrival, Coming from, Name, Going to)

Passengers_Phone (Passenger ID, Phone Number)

Hostel (Room No, Hostel No, Room Status, Floor)

The above normalized relational schema can be shown by the diagram below:



Assumptions

1. We assume that the database is managed every day i.e, we find the list of Passengers whose quarantine period is over on that day and all of them are discharged on that day. For this the database has to be managed every day at least once each day.
2. We also assume that passengers are not overloaded i.e,at max 200 Passengers of age 60+ will be there at the institute at a particular time because there are only 200(100 + 100) ground floor rooms present. This assumption is made to remove the complexity of code.
3. We also assume that hostels room are instantly sanitized after the Passenger staying in that room leaves.

Code

The screenshot shows the MySQL Workbench interface. In the top-left corner, it says "MySQL Workbench" and "unconnected" with "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, Help. Below the menu is a toolbar with various icons. The left sidebar is titled "Navigator" and shows "SCHEMAS" with "miniproject" selected, containing Tables, Views, Stored Procedures, and Functions. It also lists other databases: my1, sakila, sys, and world. The main area is titled "mp1" and contains the following SQL code:

```
1 •  create database miniProject;
2
3 •  set @counter = 0;
```

To the right of the code editor is a panel titled "SQLAdditions..." with the message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." Below the code editor is an "Output" section titled "Action Output" which displays two rows of execution results:

#	Time	Action	Message	Duration / Fetch
8	07:13:04	set @counter = 0	0 row(s) affected	0.000 sec
9	07:13:05	set @counter = 0	0 row(s) affected	0.000 sec

At the bottom of the window, there is a search bar with "Type here to search" and a system tray with icons for network, battery, and time (07:13, 10-12-2020).

First of all, we create a database named “**miniProject**” which will store the details of all the persons and hostels.

We have also created a global variable “**counter**” which will be used in calculation of the **Passenger ID**. More discussion about **Passenger ID** is given below.

Table-I :

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, the 'miniproject' schema is selected. In the main query editor window, the following SQL code is being typed:

```
4  
5  
6  
7  
8 •  create table HostelsInfo  
9  {  
10     RNUM int primary key NOT NULL,  
11     PID varchar(50),  
12     PNAME varchar(50),  
13     HNUM int NOT NULL,  
14     FLR varchar(10) NOT NULL,  
15     STS varchar(10) NOT NULL  
16 }  
17  
18  
19  
20  
21
```

The 'Output' pane at the bottom shows the results of the execution:

Action	Time	Message	Duration / Fetch
set @counter = 0	07-13-05	0 row(s) affected	0.000 sec
create table HostelsInfo (RNUM int primary key NOT NULL, PID varchar(50), P...)	07-13-05	0 row(s) affected	0.078 sec

Here, we create a table “**HostelsInfo**” which contains the status of each of the rooms of the hostel. The attributes used are :

1. *RNUM - the room number.
2. PID - the ID of the Passenger residing in that room.
3. PNAME - the name of the Passenger in that room.
4. HNUM - the hostel number in which this room is present.
5. FLR - the floor in which this room is present.
6. STS - the status of this room i.e, FULL or EMPTY.

Table-2 :

The screenshot shows the MySQL Workbench interface. In the central SQL editor window, a CREATE TABLE statement for 'PersonsInfo' is being typed. The schema 'miniproject' is selected. The table structure includes fields: PID (primary key), PNAME, PAG, PADD, PCMG, PGNG, PPHONE1, PPHONE2, DOC, PRNUM, PFLR, and PHNUM. The right-hand panel displays context help about automatic context help being disabled. The bottom status bar shows the date and time as 10-12-2020 07:20.

```
47
48 • create table PersonsInfo
49 {
50     PID varchar(50) primary key NOT NULL,
51     PNAME varchar(50) NOT NULL,
52     PAG int NOT NULL,
53     PADD varchar(255) NOT NULL,
54     PCMG varchar(255) NOT NULL,
55     PGNG varchar(255) NOT NULL,
56     PPHONE1 varchar(10) NOT NULL,
57     PPHONE2 varchar(10),
58     DOC date NOT NULL,
59     PRNUM int NOT NULL,
60     PFLR varchar(10) NOT NULL,
61     PHNUM int NOT NULL
62 }
63
```

Another table named "**PersonsInfo**" is created and it contains information of each and every person that is currently in quarantinsation. The attributes used here are :

1. *PID - Passenger's ID.
2. PNAME - Passenger's Name.
3. PAG - Passenger's Age.
4. PADD - Passenger's Address.
5. PCMG - the place from the Passenger is coming.
6. PGNG - the place where the Passenger is going.
7. PPHONE1 - Passenger's Phone Number.
8. PPHONE2 - Passenger's Phone Number.
9. DOC - date from where quarantinsation starts.
10. PRNUM - Passenger's Room Number.
11. PFLR - Passenger's Room's Floor.
12. PHNUM - Passenger's Hostel Number.

Table-3:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, it says "unconnected" and "Local instance MySQL80". The "Schemas" pane on the left shows the "miniproject" schema selected. The main query editor window contains the following SQL code:

```
63
64 • create table RecordOfPerson
65 (
66     PID varchar(50) primary key NOT NULL,
67     PNAME varchar(50) NOT NULL,
68     PAG int NOT NULL,
69     PADD varchar(255) NOT NULL,
70     PCMG varchar(255) NOT NULL,
71     PGNG varchar(255) NOT NULL,
72     PPHONE1 varchar(10) NOT NULL,
73     PPHONE2 varchar(10) NOT NULL,
74     PRNUM int NOT NULL,
75     PFLR varchar(10) NOT NULL,
76     PHNUM int NOT NULL,
77     PDOC date NOT NULL,
78     PDOG date NOT NULL
79 )
```

The "Output" pane below shows the execution history:

#	Time	Action	Message	Duration / Fetch
18	07:21:34	DROP TABLE 'miniproject'.`recordofperson`	0 row(s) affected	0.031 sec
19	07:21:47	create table RecordOfPerson (PID varchar(50) primary key NOT NULL, PNAME ...	0 row(s) affected	0.047 sec

The status bar at the bottom right shows the date and time: 10-12-2020 07:21.

This table “**RecordOfPerson**” will store the information of every passenger who was in quarantine in NIT Silchar i.e, whose 14 days of quarantine is over. The attributes used here are:

1. *PID - Passenger's ID.
2. PNAME - Passenger's Name.
3. PAG - Passenger's Age.
4. PADD - Passenger's Address.
5. PCMG - the place from the Passenger is coming.
6. PGNG - the place where the Passenger is going.
7. PPHONE1 - Passenger's Phone Number.
8. PPHONE2 - Passenger's Phone Number.
9. PRNUM - Passenger's Room Number.
10. PFLR - Passenger's Room's Floor.
11. PHNUM - Passenger's Hostel Number.
12. PDOC - Passenger's starting quarantinsation date.
13. PDOG - Passenger's ending quarantinsation date.

Procedure-1 :

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (miniproject selected), Tables, Views, Stored Procedures, Functions, my1, sakila, sys, world.
- SQL Editor:** mp1* tab, SQL code for creating a stored procedure:

```
15
16  Delimiter //
17  • create procedure FillUpHostelTable()
18  begin
19      set @i = 1001;
20      while @i <= 1250 do
21          if @i > 1200 then
22              insert into HostelsInfo values(@i,"","","1","2nd","Empty");
23          elseif @i > 1100 then
24              insert into HostelsInfo values(@i,"","","1","1st","Empty");
25          else
26              insert into HostelsInfo values(@i,"","","1","Gnd","Empty");
27          end if;
28          set @i = @i + 1;
29      END while;
30      set @i = 2001;
31      while @i < 2250 do
32          if @i > 2200 then
```
- Output:** Action Output table showing two log entries:

#	Time	Action	Message	Duration / Fetch
10	07:13:38	create table HostelsInfo (RNUM int primary key NOT NULL, PID varchar(50), P...	0 row(s) affected	0.078 sec
11	07:16:29	create procedure FillUpHostelTable() begin set @i = 1001; while @i <= 1250 do if ...	0 row(s) affected	0.016 sec
- System Bar:** Type here to search, taskbar icons, system status.

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (miniproject selected), Tables, Views, Stored Procedures, Functions, my1, sakila, sys, world.
- SQL Editor:** mp1* tab, SQL code for calling the stored procedure:

```
30      set @i = 2001;
31      while @i <= 2250 do
32          if @i > 2200 then
33              insert into HostelsInfo values(@i,"","","2","2nd","Empty");
34          elseif @i > 2100 then
35              insert into HostelsInfo values(@i,"","","2","1st","Empty");
36          else
37              insert into HostelsInfo values(@i,"","","2","Gnd","Empty");
38          end if;
39          set @i = @i + 1;
40      END while;
41  end //
42  Delimiter ;
43
44  • call FillUpHostelTable();
```
- Output:** Action Output table showing two log entries:

#	Time	Action	Message	Duration / Fetch
11	07:16:29	create procedure FillUpHostelTable() begin set @i = 1001; while @i <= 1250 do if ...	0 row(s) affected	0.016 sec
12	07:16:56	call FillUpHostelTable()	0 row(s) affected	1.766 sec
- System Bar:** Type here to search, taskbar icons, system status.

This procedure is created to fill the table “**HostelsInfo**”. Initially, the table “**HostelsInfo**” will only contain the status of all the rooms i.e, EMPTY. Since, it was manually not possible to add 500(250 + 250) rooms in the table. So, we created a procedure for doing the same. The parameters (Passenger ID and Passenger Name) are initialised with NULL as there are no entries initially.

The table after executing the above procedure is shown below :

The screenshot shows the MySQL Workbench interface. In the central pane, a query editor window titled 'mp1' displays the following SQL code:

```
45
46 • select * from HostelsInfo;
47
48
49
```

Below the query editor is a 'Result Grid' showing the data from the 'HostelsInfo' table. The table has columns: RNUM, PID, PNAME, HNUM, FLR, and STS. The data consists of 500 rows, each with RNUM values from 1001 to 1050. All rows have PID and PNAME as NULL, HNUM as 1, FLR as Gnd, and STS as Empty.

RNUM	PID	PNAME	HNUM	FLR	STS
1001			1	Gnd	Empty
1002			1	Gnd	Empty
1003			1	Gnd	Empty
1004			1	Gnd	Empty
1005			1	Gnd	Empty
1006			1	Gnd	Empty
1007			1	Gnd	Empty
1008			1	Gnd	Empty
1009			1	Gnd	Empty
1010			1	Gnd	Empty
1011			1	Gnd	Empty
...					

In the bottom right corner of the result grid, there is a toolbar with icons for 'Result Grid', 'Form Editor', and 'Field Types'. The 'Result Grid' icon is currently selected.

On the right side of the interface, there is a 'SQL Additions' panel with a message: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The bottom status bar shows the system tray icons and the date/time: 07:18 10-12-2020.

Initially all the rooms are empty. Therefore, the STS (status) of all the rooms is EMPTY.

Function-1:

The screenshot shows the MySQL Workbench interface with a query editor window titled 'mp1'. The code in the editor is as follows:

```
80
81
82
83
84 Delimiter //
85 • create function getPassengerID()
86     returns varchar(50)
87     deterministic
88     begin
89         set @counter = @counter + 1;
90         return CONCAT("NITS-",CONVERT(@counter,CHAR));
91     end; //
92 delimiter ;
```

The line 'create function getPassengerID()' is highlighted with a blue dot. The status bar on the right indicates: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

The 'Output' pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
19	07:21:47	create table RecordOfPerson (PID varchar(50) primary key NOT NULL, PNAME ...	0 row(s) affected	0.047 sec
20	07:37:34	create function getPassengerID() returns varchar(50) deterministic begin set @coun...	0 row(s) affected	0.016 sec

This function named "getPassengerID" returns a **unique** Passenger ID for every person coming to NIT Silchar for quarantinsation. The Passenger ID is of the format "**NITS-X**" where X is an integer. The globally declared variable "**counter**" represents X.

Function-2:

The screenshot shows the MySQL Workbench interface with a query editor window titled 'mp1'. The code in the editor is as follows:

```
93
94
95 Delimiter //
96 • create function getRoomNumber(age int)
97     returns int
98     deterministic
99     begin
100        if age < 40 then
101            return (select min(RNUM) from HostelsInfo where FLR = '2nd' and STS = "Empty");
102        elseif age <= 60 then
103            return (select min(RNUM) from HostelsInfo where FLR = '1st' and STS = "Empty");
104        end if;
105        return (select min(RNUM) from HostelsInfo where FLR = 'Gnd' and STS = "Empty");
106    end; //
107 delimiter ;
```

The 'Output' pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
20	07:37:34	create function getPassengerID() returns varchar(50) deterministic begin set @coun...	0 row(s) affected	0.016 sec
21	07:37:58	create function getRoomNumber(age int) returns int deterministic begin if age < 40 t...	0 row(s) affected	0.016 sec

The function named “**getRoomNumber**” returns the room number of the first room which the Passenger can get as per his age.

If age > 60, then Passenger gets room on the ground floor.

If age = [40,60], then Passenger gets room on the 1st floor.

If age < 40, then Passenger gets room on the 2nd floor.

First Hostel 1 is filled and then Hostel 2.

Trigger-1:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **miniproject** containing tables, views, stored procedures, functions, and other objects.
- SQL Editor:** A query window titled "mp1" containing the following SQL code:

```
104  
105  
106 Delimiter //  
107 • create trigger update_hostels_when_person_comes  
108 before insert on PersonsInfo  
109 for each row  
110 begin  
111     update HostelsInfo  
112         set  
113             PID = new.PID,  
114             PNAME = new.PNAME,  
115             STS = "Full"  
116         where RNUM = new.PRNUM;  
117 end //  
118 delimiter ;
```
- Output:** An "Action Output" table showing two log entries:

#	Time	Action	Message	Duration / Fetch
21	07:37:58	create function getRoomNumber(age int) returns int deterministic begin if age < 40 t...	0 row(s) affected	0.016 sec
22	07:38:28	create trigger update_hostels_when_person_comes before insert on PersonsInfo for...	0 row(s) affected	0.031 sec
- System Bar:** Includes a search bar, taskbar icons (File Explorer, Task View, Start, Taskbar, Task View, Taskbar, Taskbar), system status (Battery, Wi-Fi, Volume, ENG), and a date/time stamp (07:38 10-12-2020).

This **trigger** is written because as soon as a Passenger comes, all his information has to be added in the **PersonsInfo** table.

But at the same time the **HostelsInfo** table needs to be modified. The status of the room must be FULL now so that no other passenger gets this room as long as the respective person is there(14 days).

Trigger-2:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** unconnected x Local instance MySQL80 x File Edit View Query Database Server Tools Scripting Help
- Navigator:** Schemas (miniproject selected), Tables, Views, Stored Procedures, Functions, my1, sakila, sys, world.
- SQL Editor:** Title: mp1* x. Content:

```
142
143
144  Delimiter //
145  • create trigger update_hostels_when_person_leaves
146    before delete on PersonsInfo
147    for each row
148    begin
149      update HostelsInfo
150      set
151        PID = '',
152        PNAME = '',
153        STS = "Empty"
154        where RNUM = old.PRNUM;
155        insert into RecordOfPerson values(old.PID, old.PNAME, old.PAG, old.PADD, old.PCMG, old.PGNG, old.PPHON)
156    end //
157  delimiter ;
```

A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Output:** Action Output table showing two entries:

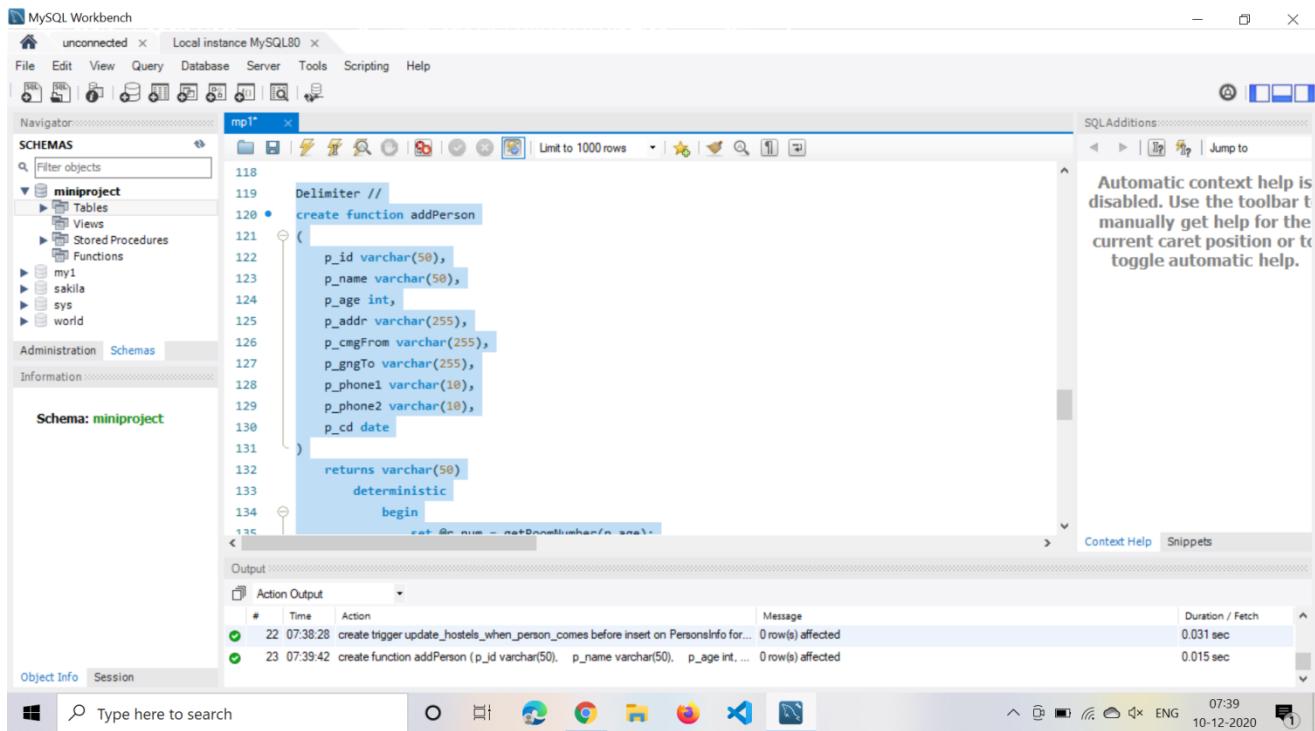
#	Time	Action	Message	Duration / Fetch
23	07:39:42	create function addPerson (p_id varchar(50), p_name varchar(50), p_age int, ...)	0 row(s) affected	0.015 sec
24	07:41:15	create trigger update_hostels_when_person_leaves before delete on PersonsInfo f...	0 row(s) affected	0.031 sec
- System Bar:** Type here to search, Task View, Taskbar icons (Google Chrome, Mozilla Firefox, Microsoft Edge, File Explorer, Task Manager), Date/Time (07:41 10-12-2020), and a notification icon.

This **trigger** is written because as soon as a Passenger leaves, all his information has to be deleted from the **PersonsInfo** table.

But at the same time the **HostelsInfo** table needs to be modified. The status of the room must be EMPTY now so that any other new passenger gets this room.

Just before leaving, the Passengers details are stored in the table **RecordOfPerson**. This trigger also manages this thing.

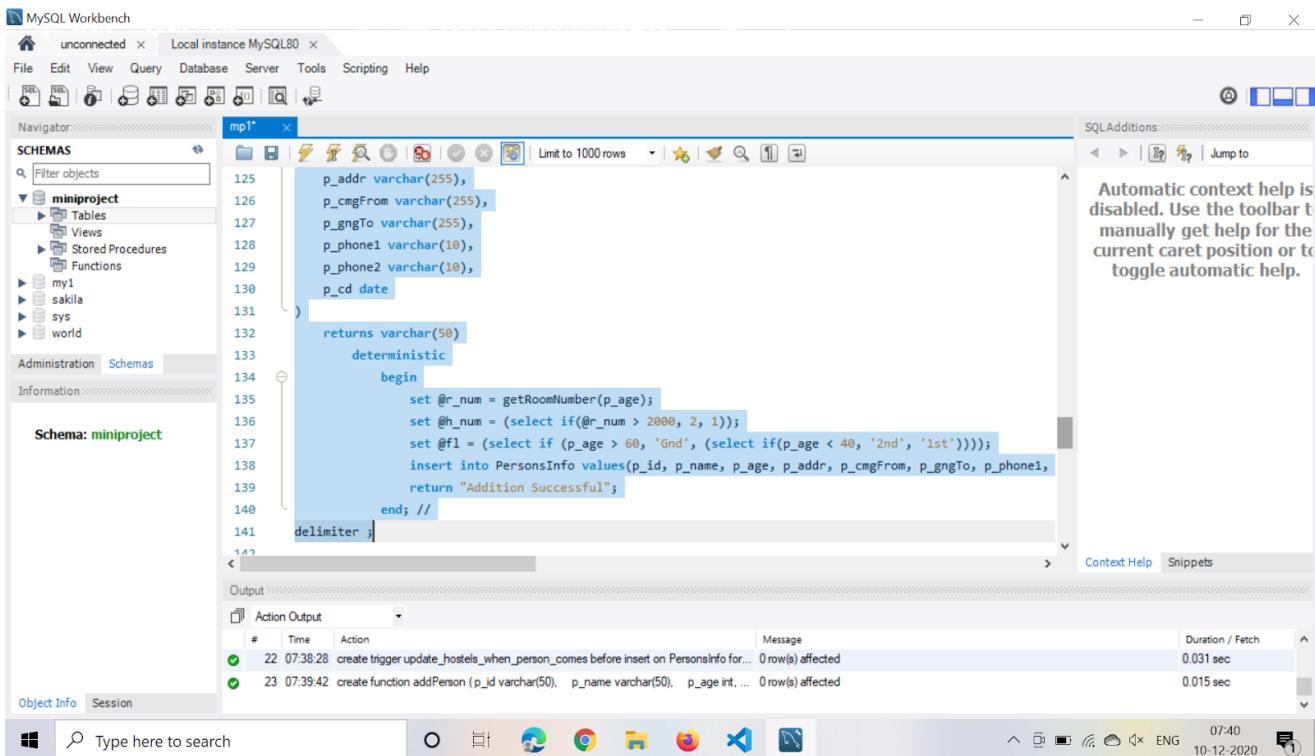
Function-3:



The screenshot shows the MySQL Workbench interface with the 'miniproject' schema selected. In the central query editor, a function named 'addPerson' is being created. The code defines parameters for p_id, p_name, p_age, p_addr, p_cmgrFrom, p_gngTo, p_phone1, p_phone2, and p_cd. It uses a trigger to update 'hostels_when_person_comes' before inserting into 'PersonsInfo'. The function returns a message indicating success or failure.

```
118
119  Delimiter //
120  •  create function addPerson
121  (
122      p_id varchar(50),
123      p_name varchar(50),
124      p_age int,
125      p_addr varchar(255),
126      p_cmgrFrom varchar(255),
127      p_gngTo varchar(255),
128      p_phone1 varchar(10),
129      p_phone2 varchar(10),
130      p_cd date
131  )
132      returns varchar(50)
133      deterministic
134      begin
135          set @r_num = getRoomNumber(p_age);
136          set @_num = (select if(@r_num > 2000, 2, 1));
137          set @f1 = (select if (p_age > 60, 'Gnd', (select if(p_age < 40, '2nd', '1st'))));
138          insert into PersonsInfo values(p_id, p_name, p_age, p_addr, p_cmgrFrom, p_gngTo, p_phone1,
139                                         return "Addition Successful";
140      end;
141  delimiter ;
```

The 'Output' pane shows two log entries: one for creating the trigger and one for creating the function itself. The status bar at the bottom right indicates the time as 07:39 and the date as 10-12-2020.



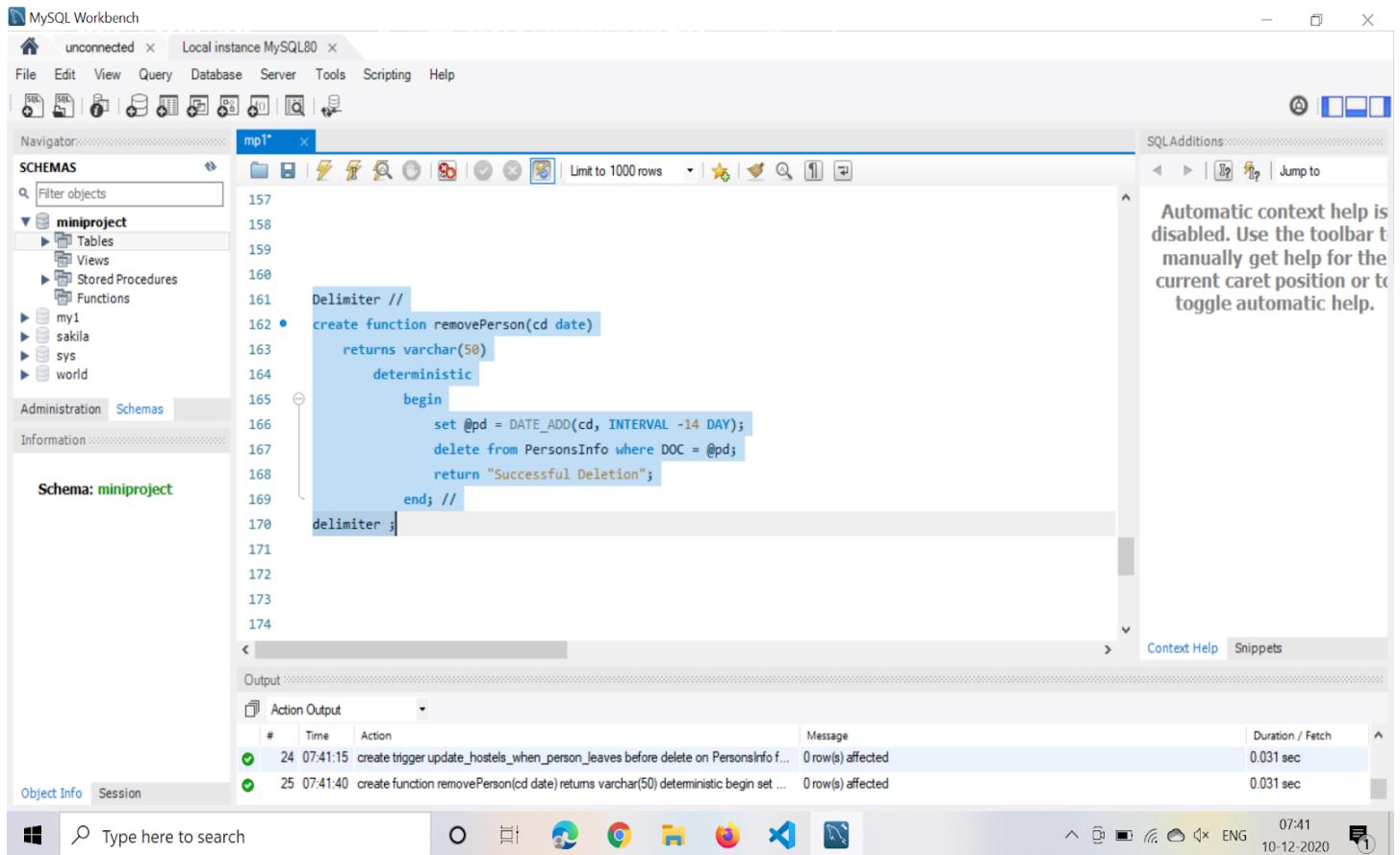
This screenshot shows the same MySQL Workbench session after the function 'addPerson' has been fully implemented. The code now includes logic to determine room numbers based on age and to insert into the 'PersonsInfo' table. The 'Output' pane shows the successful creation of the function and the trigger.

```
125
126      p_addr varchar(255),
127      p_cmgrFrom varchar(255),
128      p_gngTo varchar(255),
129      p_phone1 varchar(10),
130      p_phone2 varchar(10),
131      p_cd date
132  )
133  returns varchar(50)
134  deterministic
135  begin
136      set @r_num = getRoomNumber(p_age);
137      set @_num = (select if(@r_num > 2000, 2, 1));
138      set @f1 = (select if (p_age > 60, 'Gnd', (select if(p_age < 40, '2nd', '1st'))));
139      insert into PersonsInfo values(p_id, p_name, p_age, p_addr, p_cmgrFrom, p_gngTo, p_phone1,
140                                     return "Addition Successful";
141  end;
142  delimiter ;
```

The 'Output' pane shows the log entries for the trigger creation and the function creation. The status bar at the bottom right indicates the time as 07:40 and the date as 10-12-2020.

This function “**addPerson**” adds the details of the Passenger in the respective tables. It takes all the identification attributes as parameters.

Function-4:



The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (miniproject selected), Tables, Views, Stored Procedures, Functions, my1, sakila, sys, world.
- Current Editor:** mp1* tab, SQL code for creating a function named removePerson(cd date). The code uses a cursor to iterate through rows in the PersonsInfo table where the DOC date is 14 days from the input cd date. It then deletes these rows and returns a success message.
- Output Tab:** Action Output table showing two actions:

#	Time	Action	Message	Duration / Fetch
24	07:41:15	create trigger update_hostels_when_person_leaves before delete on PersonsInfo f...	0 row(s) affected	0.031 sec
25	07:41:40	create function removePerson(cd date) returns varchar(50) deterministic begin set ...	0 row(s) affected	0.031 sec
- System Tray:** Shows icons for Task View, Start, Taskbar, Google Chrome, Microsoft Edge, Firefox, Visual Studio Code, and File Explorer.
- Bottom Bar:** Type here to search, system status icons, and a timestamp: 07:41 10-12-2020.

This function “**removePerson**” removes the details of the Passenger from the respective tables. It takes the current date as parameter which means it removes the list of Passengers whose 14 days of quarantine are over today.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: mp1*

SCHEMAS: miniproject

```

169
170
171 • select addPerson(getPassengerID(),"Aayush",20,"Guwahati","Guwahati","Silchar","8876690542","8471935316",'2
172 • select addPerson(getPassengerID(),"Ram",64,"Tezpur","Kolkata","Silchar","1234567890","","'2020-12-05');
173 • select addPerson(getPassengerID(),"Shyam",49,"Mumbai","Mumbai","Silchar","6478263518","","'2020-11-26");
174 • select addPerson(getPassengerID(),"Raj",73,"Guwahati","Guwahati","Silchar","8371375642","","'2020-11-26");
175 • select addPerson(getPassengerID(),"Veer",18,"Bongaigaon","Guwahati","Silchar","1111111111","","'2020-12-07"
176 • select addPerson(getPassengerID(),"Mohan",42,"UP","Lucknow","Silchar","2222222222","3333333333","2020-12-0
177
178
179

```

Result Grid | Filter Rows: Export: Wrap Cell Content: Result 7 ×

addPerson(getPassengerID(),"Mohan",42,"UP","Lucknow","Silchar","2222222222","3333333333","2020-12-07")
Addition Successful

Output: Action Output

#	Time	Action	Message	Duration / Fetch
30	07:43:21	select addPerson(getPassengerID(),"Veer",18,"Bongaigaon","Guwahati","Silchar","1111111111","2020-12-07")	1 row(s) returned	0.000 sec / 0.000 sec
31	07:43:21	select addPerson(getPassengerID(),"Mohan",42,"UP","Lucknow","Silchar","2222222222","3333333333","2020-12-07")	1 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Type here to search

Result 2 Result 3 Result 4 Result 5 Result 6 Result 7 × Context Help Snippets

Read Only

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

At first, we add the details of 5 Passengers of varying age in the database and observe the following tables:

The PersonsInfo table --

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: mp1*

SCHEMAS: miniproject

```

177
178
179
180 • select * from PersonsInfo;
181
182
183
184

```

PersonsInfo 8 ×

Result Grid | Filter Rows: Edit: Export/Imports: Wrap Cell Content: Result 8 ×

PID	PNAME	PAG	PADD	PCMG	PGNG	PPHONE1	PPHONE2	DOC	PRNUM	PFLR	PHNUM
NITS-1	Aayush	20	Guwahati	Guwahati	Silchar	8876690542	8471935316	2020-12-04	1201	2nd	1
NITS-2	Ram	64	Tezpur	Kolkata	Silchar	1234567890		2020-12-05	1001	Gnd	1
NITS-3	Shyam	49	Mumbai	Mumbai	Silchar	6478263518		2020-11-26	1101	1st	1
NITS-4	Raj	73	Guwahati	Guwahati	Silchar	8371375642		2020-11-26	1002	Gnd	1
NITS-5	Veer	18	Bongaigaon	Guwahati	Silchar	1111111111		2020-12-07	1202	2nd	1
NITS-6	Mohan	42	UP	Lucknow	Silchar	2222222222	3333333333	2020-12-07	1102	1st	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

PersonsInfo 8 ×

Output: Action Output

#	Time	Action	Message	Duration / Fetch
31	07:43:21	select addPerson(getPassengerID(),"Mohan",42,"UP","Lucknow","Silchar","2222222222","3333333333","2020-12-07")	1 row(s) returned	0.000 sec / 0.000 sec
32	07:44:47	select * from PersonsInfo LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Type here to search

Result 2 Result 3 Result 4 Result 5 Result 6 Result 7 × Context Help Snippets

Read Only

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

The HostelsInfo table –

This screenshot shows the MySQL Workbench interface with the 'HostelsInfo' table selected. The table has columns: RNUM, PID, PNAME, HNUM, FLR, and STS. The data shows 1000 rows, with the first few rows being:

RNUM	PID	PNAME	HNUM	FLR	STS
1001	NITS-2	Ram	1	Gnd	Full
1002	NITS-4	Raj	1	Gnd	Full
1003			1	Gnd	Empty
1004			1	Gnd	Empty
1005			1	Gnd	Empty
1006			1	Gnd	Empty
1007			1	Gnd	Empty
1008			1	Gnd	Empty

The 'Output' pane shows two actions: selecting from PersonsInfo and selecting from HostelsInfo, both returning 1000 rows.

This screenshot shows the MySQL Workbench interface with the 'HostelsInfo' table selected. The table has columns: RNUM, PID, PNAME, HNUM, FLR, and STS. The data shows 1000 rows, with the first few rows being:

RNUM	PID	PNAME	HNUM	FLR	STS
1098			1	Gnd	Empty
1099			1	Gnd	Empty
1100			1	Gnd	Empty
1101	NITS-3	Shyam	1	1st	Full
1102	NITS-6	Mohan	1	1st	Full
1103			1	1st	Empty
1104			1	1st	Empty
1105			1	1st	Empty

The 'Output' pane shows two actions: selecting from PersonsInfo and selecting from HostelsInfo, both returning 1000 rows.

MySQL Workbench

unconnected x Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

miniproject

Tables Views Stored Procedures Functions

my1 sakila sys world

Administration Schemas

Information Schema: miniproject

HostelsInfo10 x

Result Grid | Filter Rows: [] | Edit: [] | Export/Imports: [] | Wrap Cell Content: []

RNUM PID PNAME HNUM FLR STS

1199		1	1st	Empty	
1200		1	1st	Empty	
1201	NITS-1	Aayush	1	2nd	Full
1202	NITS-5	Veer	1	2nd	Full
1203			1	2nd	Empty
1204			1	2nd	Empty
1205			1	2nd	Empty
1206			1	2nd	Empty

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
33	07:45:47	select * from PersonsInfo LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec
34	07:46:09	select * from HostelsInfo LIMIT 0, 1000	500 row(s) returned	0.000 sec / 0.000 sec

Type here to search

Object Info Session

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Form Editor

Context Help Snippets

07:46 10-12-2020

The screenshot shows the MySQL Workbench interface with a query editor window titled 'mp1'. The SQL query 'select * from HostelsInfo;' has been run, and the results are displayed in a grid. The grid has columns: RNUM, PID, PNAME, HNUM, FLR, and STS. The data shows six rows of hostels, each with a unique RNUM and PID, and varying values for PNAME, HNUM, FLR, and STS. Below the grid, the 'Action Output' section shows two log entries for the executed queries. The status bar at the bottom right indicates the time as 07:46 and the date as 10-12-2020.

As we thought earlier, the **HostelsInfo** table is exactly like that.

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar displays the Navigator, Schemas (with miniproject selected), Administration, and Information. The main workspace contains a query editor window titled 'mp1*' with the following content:

```
179
180 • select removePerson(CURDATE());
181
182
183
184
```

The 'Result Grid' tab shows the output of the query:

	removePerson(CURDATE())
▶	Successful Deletion

Below the result grid, the 'Result 11' tab is open, showing the following log entries:

#	Action	Message	Duration / Fetch
34	07:46:09 select * from HostelsInfo LIMIT 0, 1000	500 row(s) returned	0.000 sec / 0.000 sec
35	07:47:29 select removePerson(CURDATE()) LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

The status bar at the bottom shows the system tray icons and the time as 07:47 on 10-12-2020.

Now, we manage all the persons whose quarantine period is over today i.e, the current day and observe the tables again :

The PersonsInfo table --

The screenshot shows the MySQL Workbench interface. The top navigation bar includes 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. The left sidebar has sections for 'Administration' and 'Information', with the 'Schemas' section currently active, showing databases like 'miniproject', 'my1', 'sakila', 'sys', and 'world'. The main workspace displays a query editor with the following content:

```
181
182 • select * from PersonsInfo;
183
184
185
186
```

The results are shown in a 'Result Grid' table:

PID	PNAME	PAG	PADD	PCMG	PGNG	PPHONE1	PPHONE2	DOC	PRNUM	PFLR	PHNUM
NITS-1	Aayush	20	Guwahati	Guwahati	Silchar	8876690542	8471935316	2020-12-04	1201	2nd	1
NITS-2	Ram	64	Tezpur	Kolkata	Silchar	1234567890		2020-12-05	1001	Grid	1
NITS-5	Veer	18	Bongaigaon	Guwahati	Silchar	1111111111		2020-12-07	1202	2nd	1
NITS-6	Mohan	42	UP	Lucknow	Silchar	2222222222	3333333333	2020-12-07	1102	1st	1
*	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE

Below the grid, a log window titled 'PersonsInfo 12 x' shows the following entries:

Action Output	Time	Action	Message	Duration / Fetch
	#	Time	Action	
35	07:47:29	select removePerson(CURDATE()) LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
36	07:48:03	select * from PersonsInfo LIMIT 0, 1000	4 row(s) returned	0.016 sec / 0.000 sec

The bottom status bar shows system icons and the text 'Type here to search'.

Two passengers are removed as their 14 days are over.

The HostelsInfo table --

MySQL Workbench - Local instance MySQL80

Schema: miniproject

Output:

RNUM	PID	PNAME	HNUM	FLR	STS
1001	NITS-2	Ram	1	Gnd	Full
1002			1	Gnd	Empty
1003			1	Gnd	Empty
1004			1	Gnd	Empty
1005			1	Gnd	Empty
1006			1	Gnd	Empty
1007			1	Gnd	Empty
1008			1	Gnd	Empty
1009			1	Gnd	Empty
1010			1	Gnd	Empty

Action Output

#	Time	Action	Message	Duration / Fetch
36	07:48:03	select * from PersonsInfo LIMIT 0, 1000	4 row(s) returned	0.016 sec / 0.000 sec
37	07:48:36	select * from HostelsInfo LIMIT 0, 1000	500 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench - Local instance MySQL80

Schema: miniproject

Screenshot saved
The screenshot was added to your OneDrive.
OneDrive

Output:

RNUM	PID	PNAME	HNUM	FLR	STS
1101			1	1st	Empty
1102	NITS-6	Mohan	1	1st	Full
1103			1	1st	Empty
1104			1	1st	Empty
1105			1	1st	Empty
1106			1	1st	Empty
1107			1	1st	Empty
1108			1	1st	Empty
1109			1	1st	Empty
1110			1	1st	Empty

Action Output

#	Time	Action	Message	Duration / Fetch
36	07:48:03	select * from PersonsInfo LIMIT 0, 1000	4 row(s) returned	0.016 sec / 0.000 sec
37	07:48:36	select * from HostelsInfo LIMIT 0, 1000	500 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench

unconnected x Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

miniproject

Tables Views Stored Procedures Functions my1 sakila sys world

Administration Schemas

Information

Schema: miniproject

HostelsInfo13 x

Result Grid

RNUM	PID	PNAME	HNUM	FLR	STS
1198			1	1st	Empty
1199			1	1st	Empty
1200			1	1st	Empty
1201	NITS-1	Aayush	1	2nd	Full
1202	NITS-5	Veer	1	2nd	Full
1203			1	2nd	Empty
1204			1	2nd	2nd
1205			1	2nd	Empty
1206			1	2nd	Empty
1207			1	2nd	Empty
1208			1	2nd	Empty

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
36	07:48:03	select * from PersonsInfo LIMIT 0, 1000	4 row(s) returned	0.016 sec / 0.000 sec
37	07:48:36	select * from HostelsInfo LIMIT 0, 1000	500 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Type here to search

07:48 10-12-2020

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

The RecordOfPerson table--

MySQL Workbench

unconnected x Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: Schemas

miniproject

Tables Views Stored Procedures Functions my1 sakila sys world

Administration Schemas

Information

Schema: miniproject

RecordOfPerson14 x

Result Grid

PID	PNAME	PAG	PADD	PCMG	PGNG	PPHONE1	PPHONE2	PRNUM	PFLR	PHNUM	PDOC	PDOG
NITS-3	Shyam	49	Mumbai	Mumbai	Silchar	6478263518		1101	1st	1	2020-11-26	2020-12-10
NITS-4	Raj	73	Guwahati	Guwahati	Silchar	8371375642		1002	Gnd	1	2020-11-26	2020-12-10
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
37	07:48:36	select * from HostelsInfo LIMIT 0, 1000	500 row(s) returned	0.000 sec / 0.000 sec
38	07:49:12	select * from RecordOfPerson LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Type here to search

07:49 10-12-2020

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Since, the quarantine period of 2 people is over. Therefore, this table contains two records.

Conclusion

After doing this project, we got a good knowledge of ER diagram and Relational table. Normalisation was too studied in depth for this project. Moreover, we now also have a good knowledge of SQL commands such as procedure, trigger, etc.