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

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.

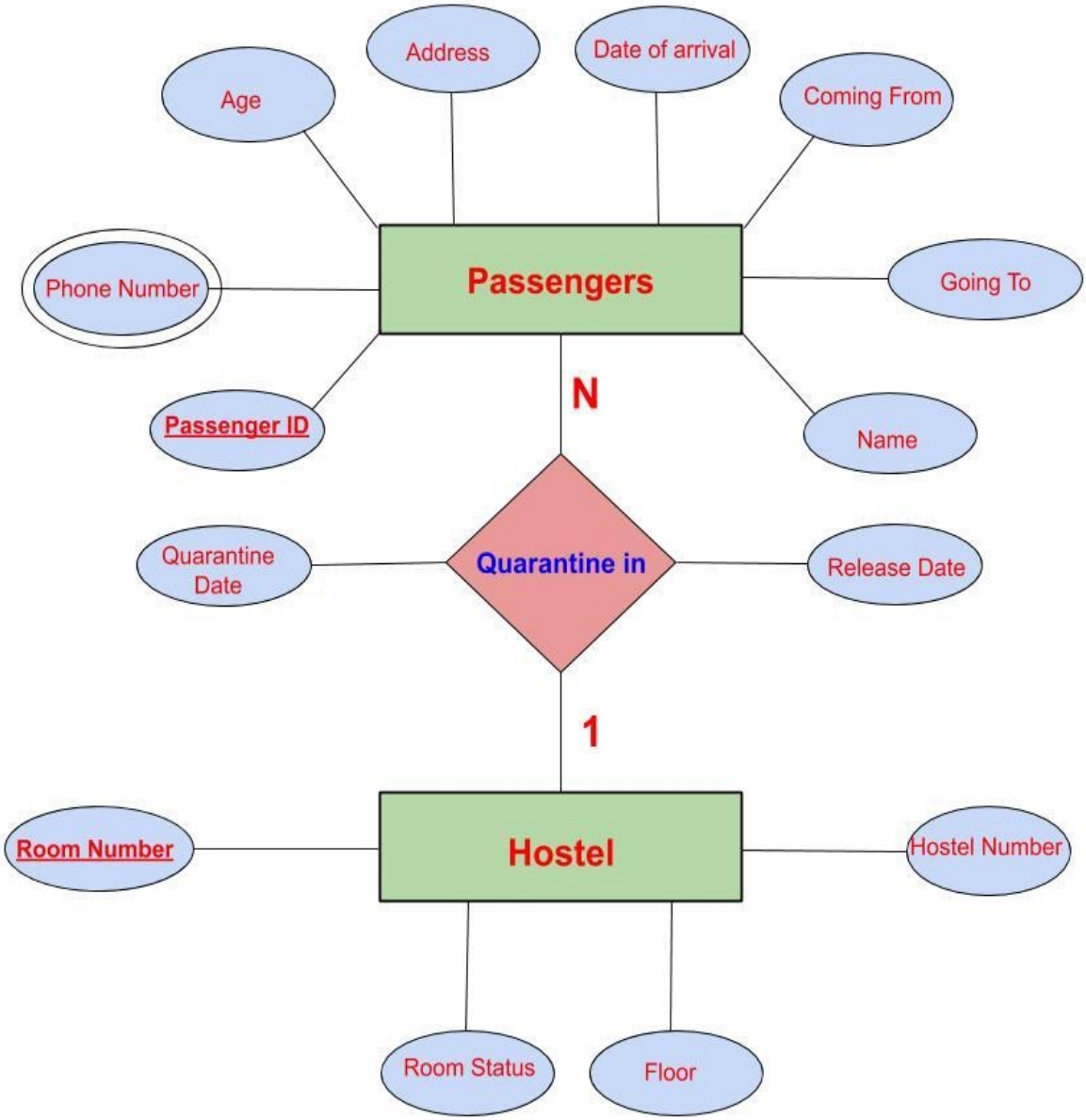
c) Many to One Relationship:

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.

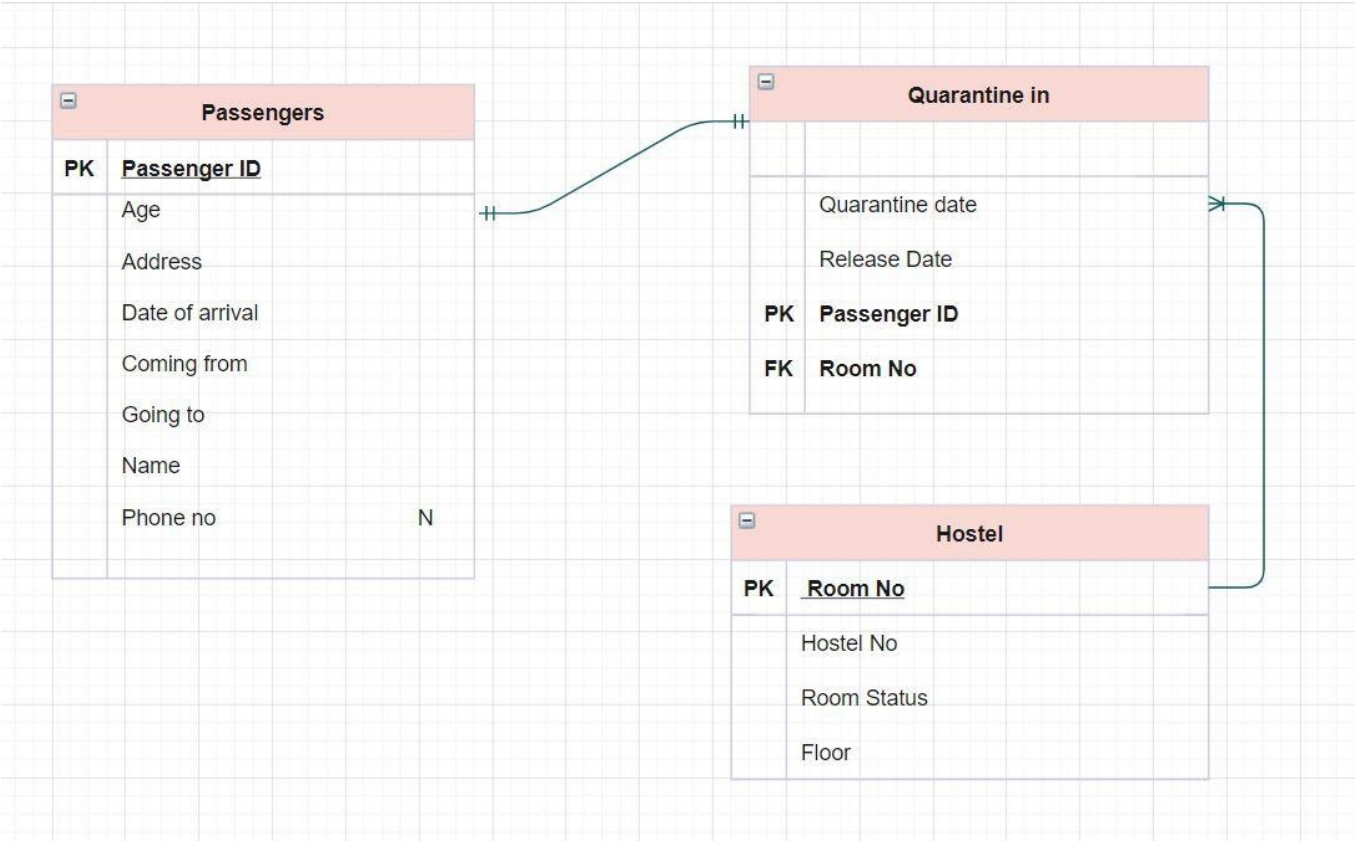
Relationship set name: **Quarantine in**

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

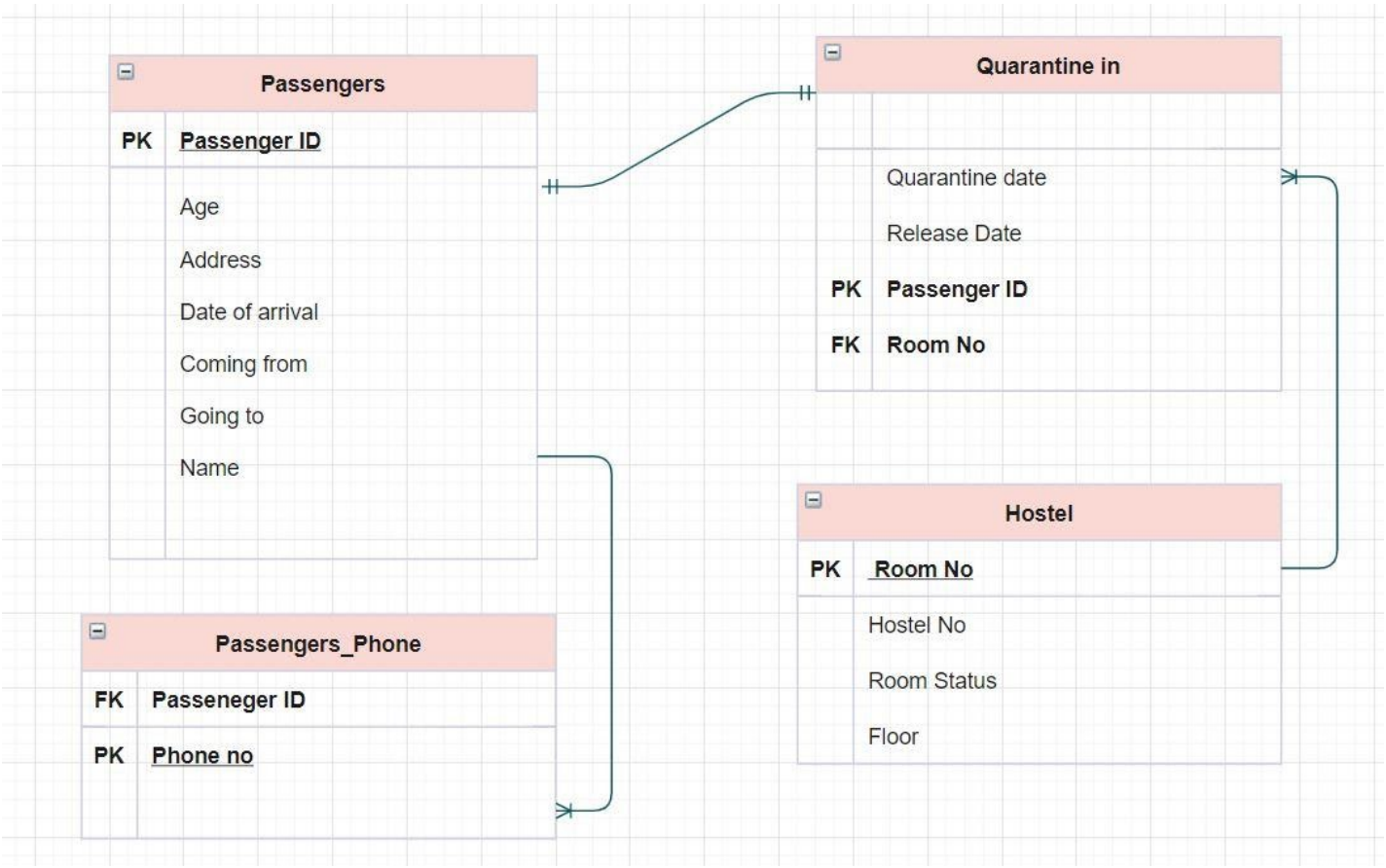
Passenger ID	Name
001	Ram
002	Shyam
005	Gaurav

Passengers_Phone

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

Passengers 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 dependancy 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 -> 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 thatfor 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.

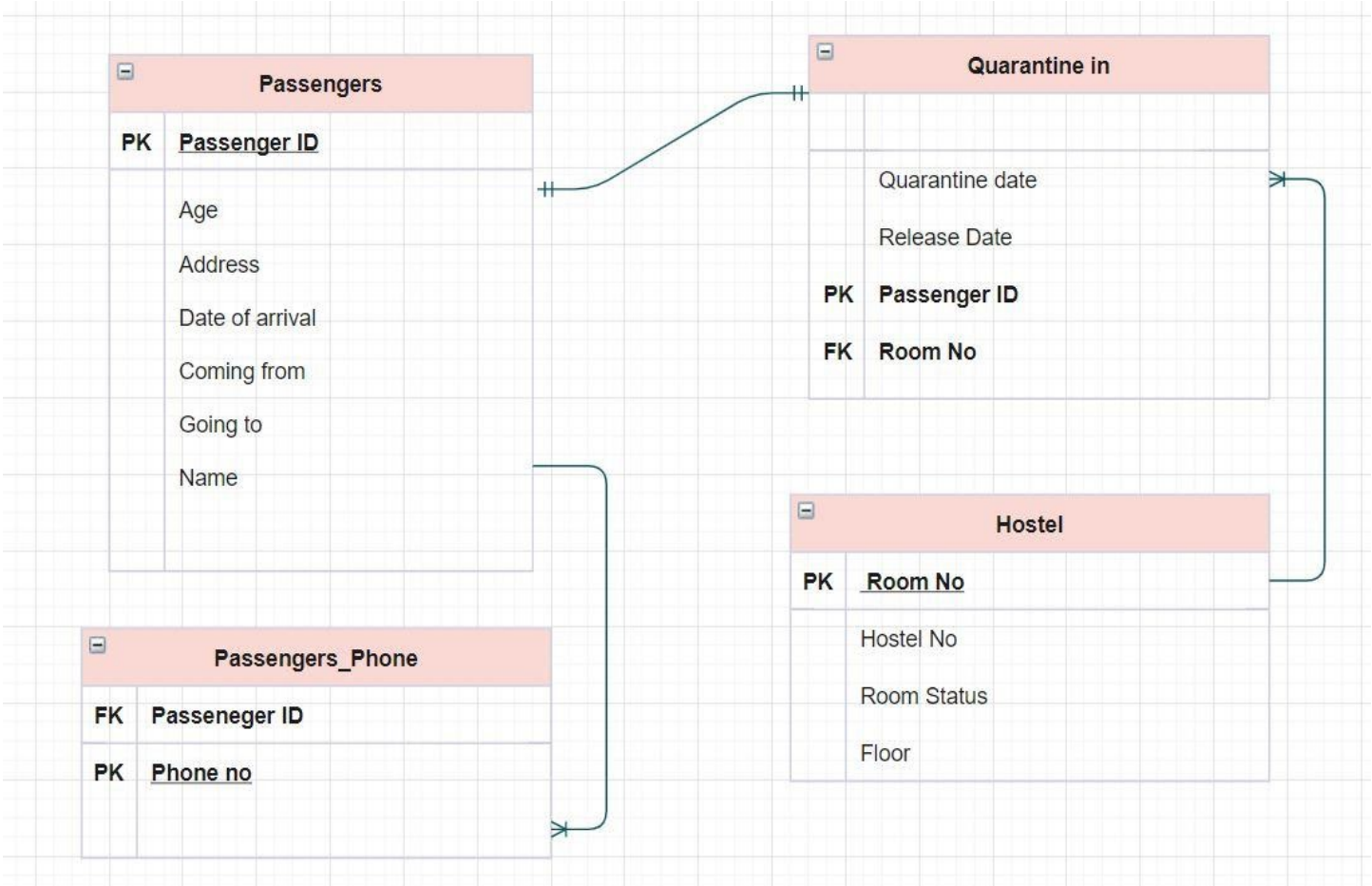
Hence, the normalized relational schema can be represented by the following relations:

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:



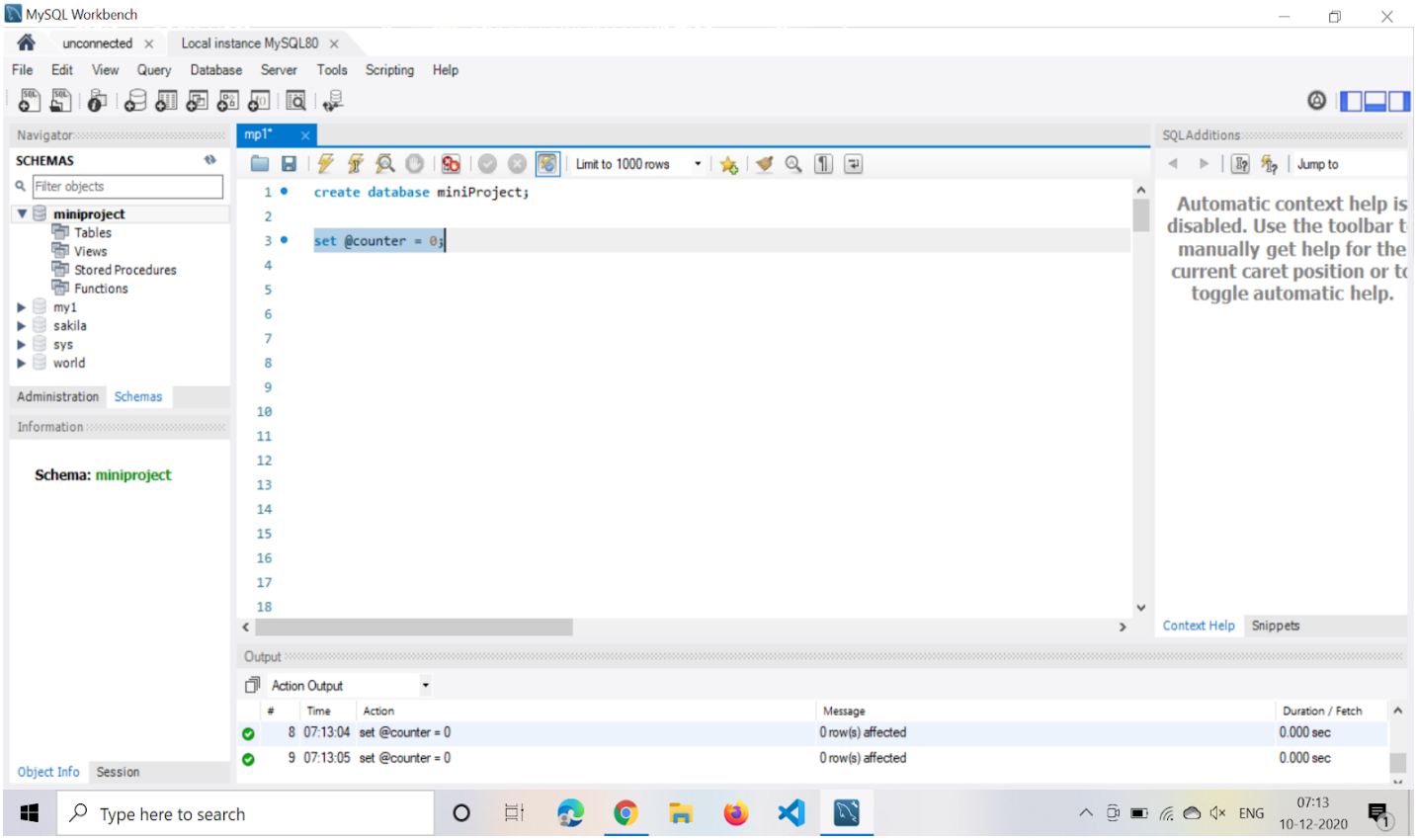
Assumptions

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.

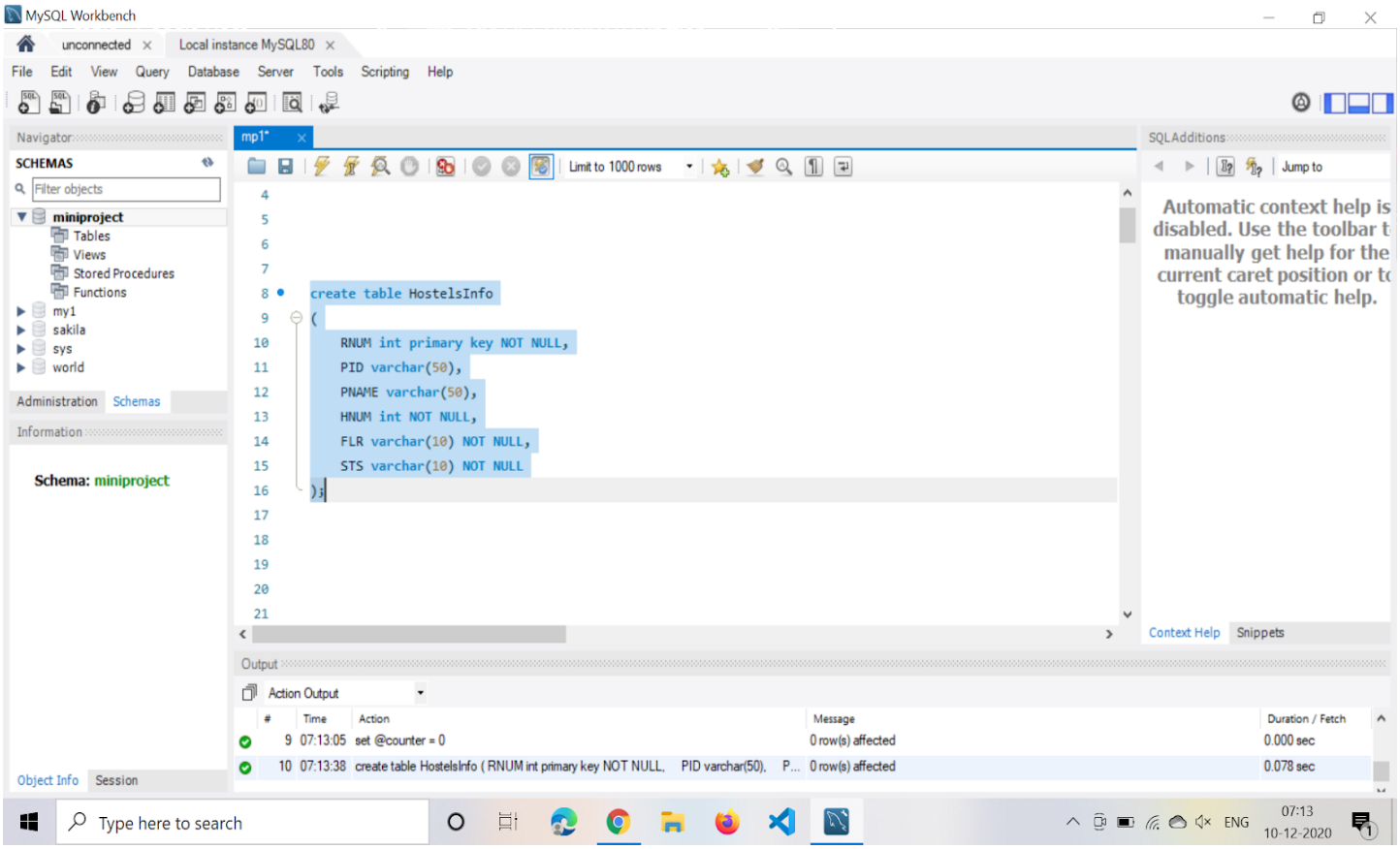
Code



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.

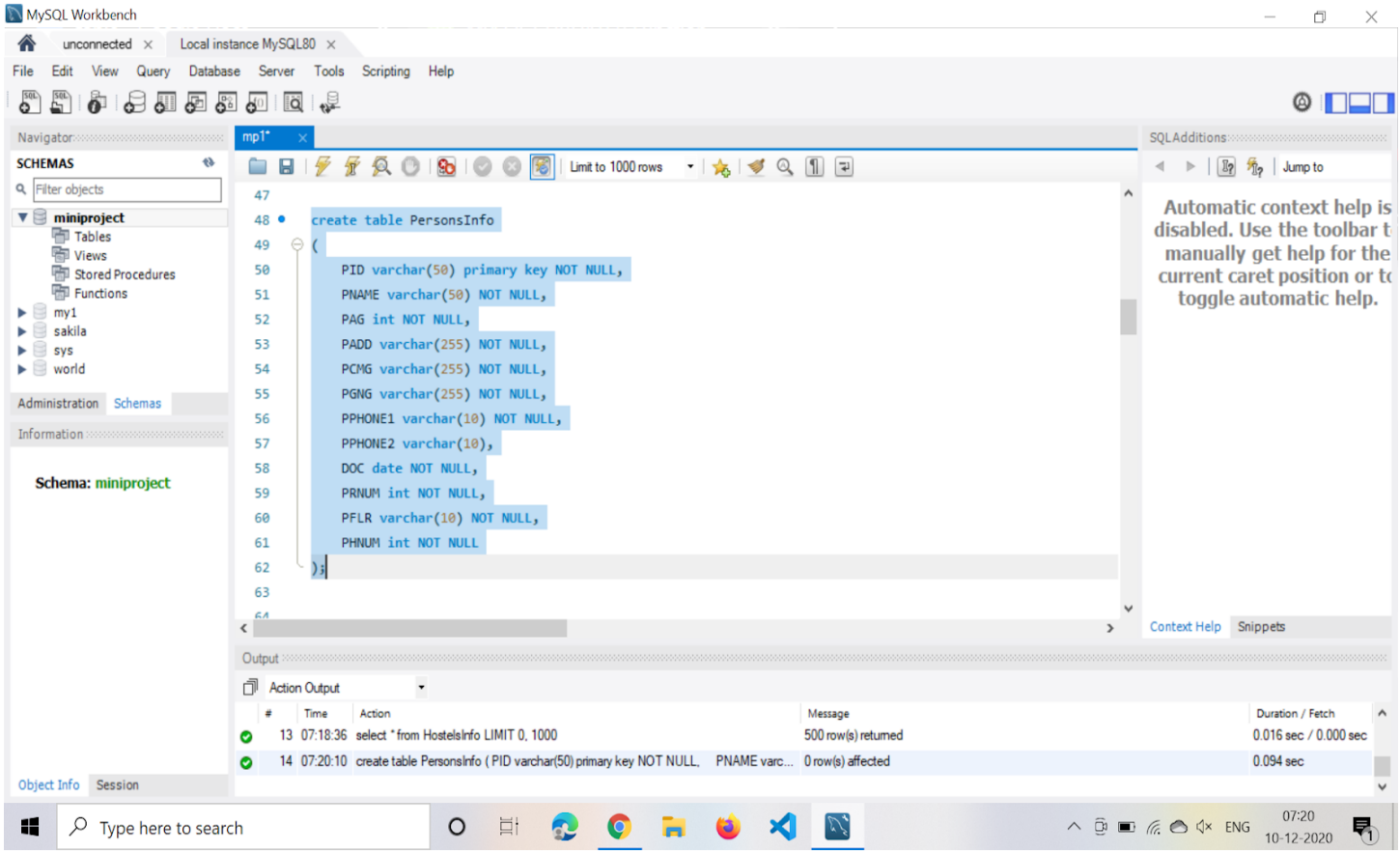
Table-I :



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.

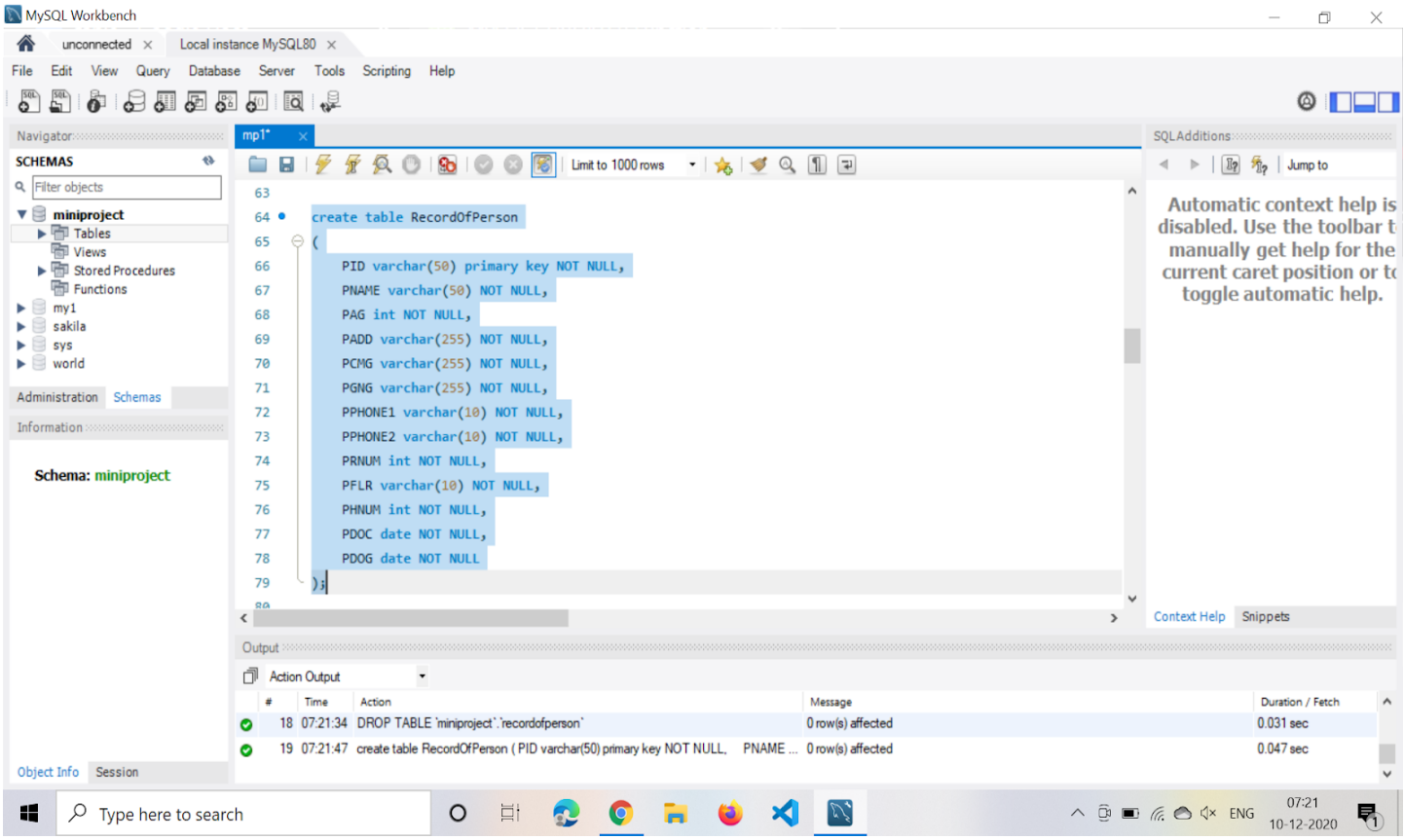
Table-2 :



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.

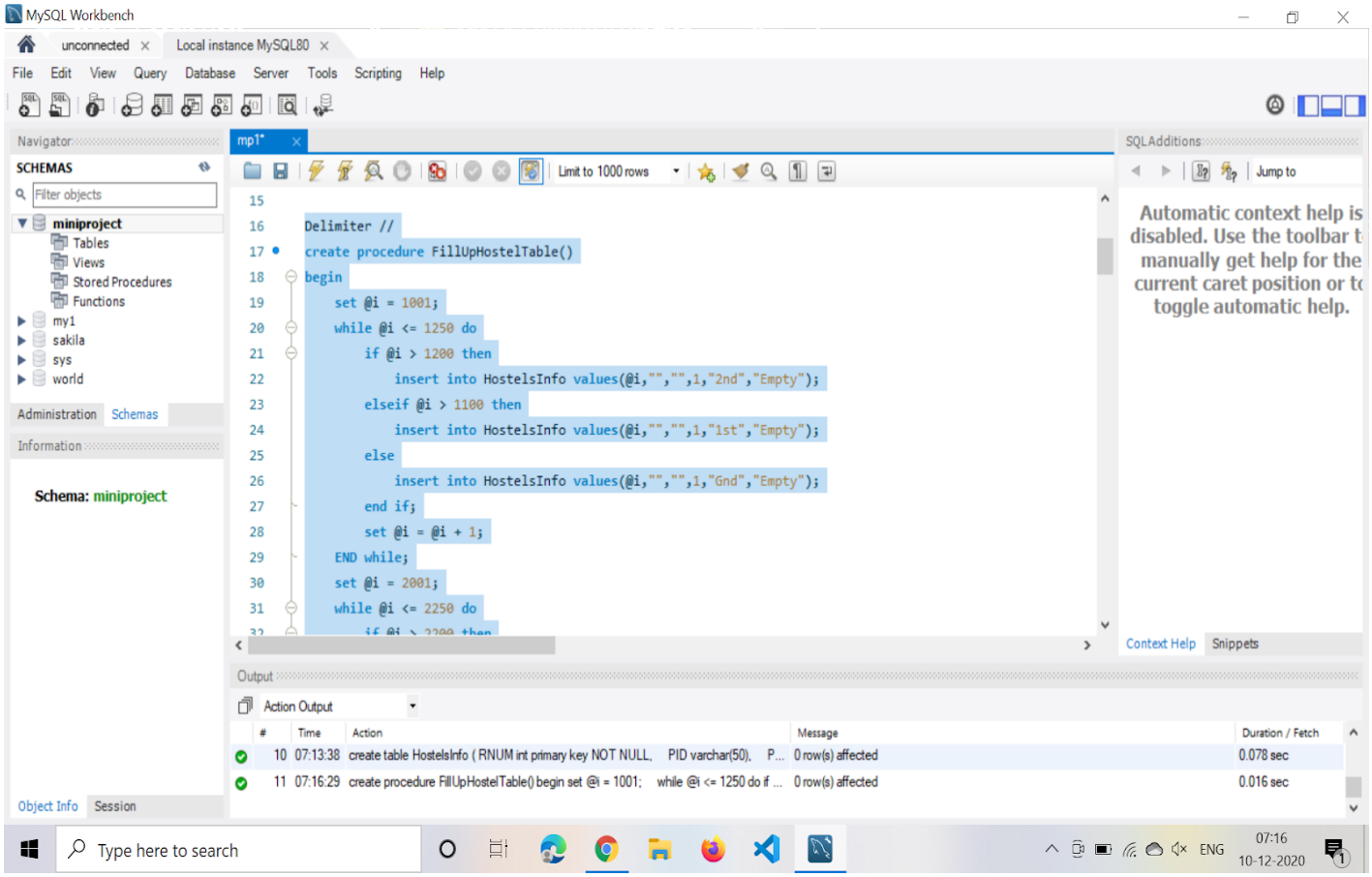
Table-3:

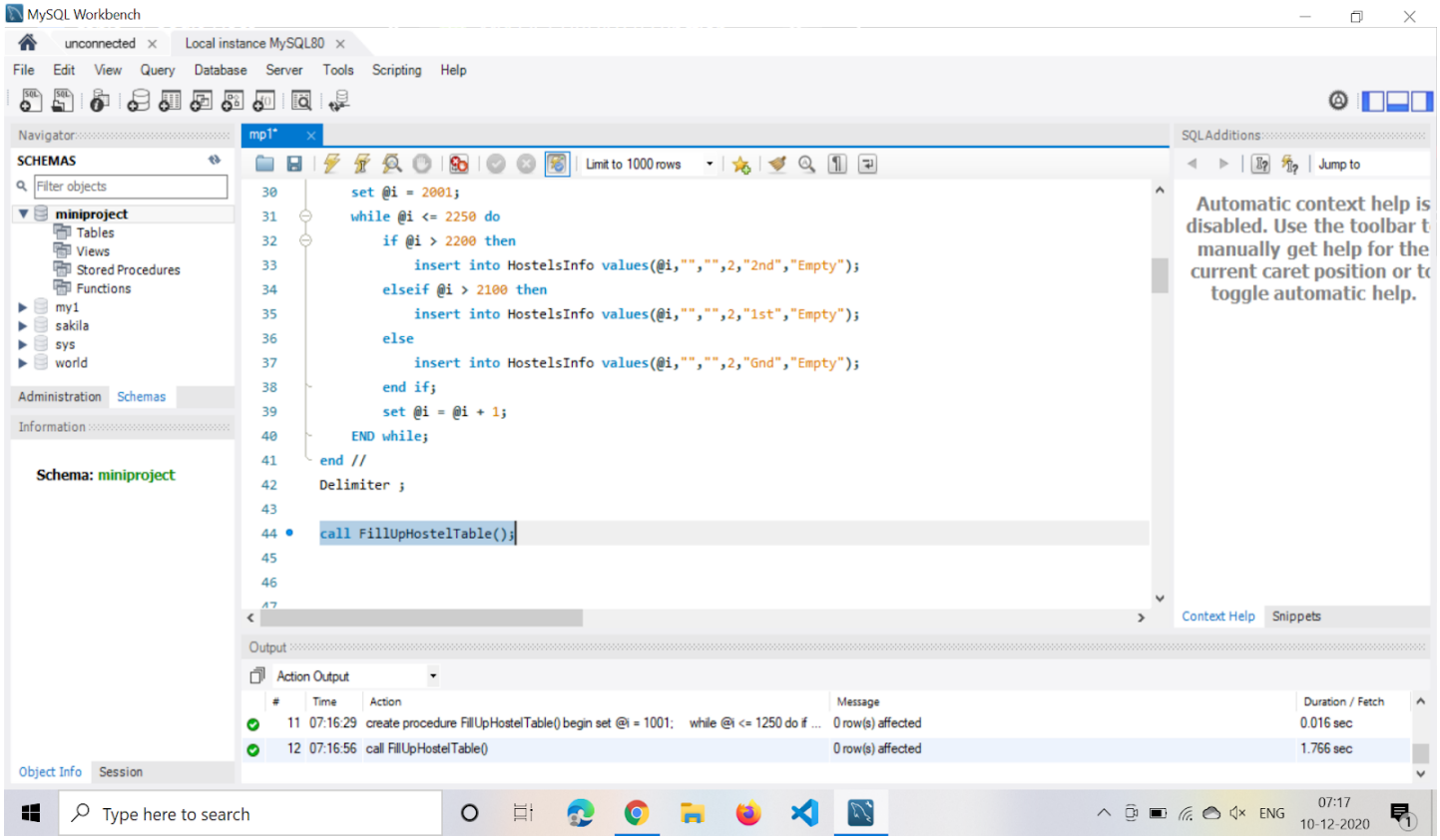


This table "**RecordOfPerson**" will store the information of every passenger who was in quarantine in NIT Silchar i.e, whose 14 days of quarantine is over. The attributes used here are:

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.

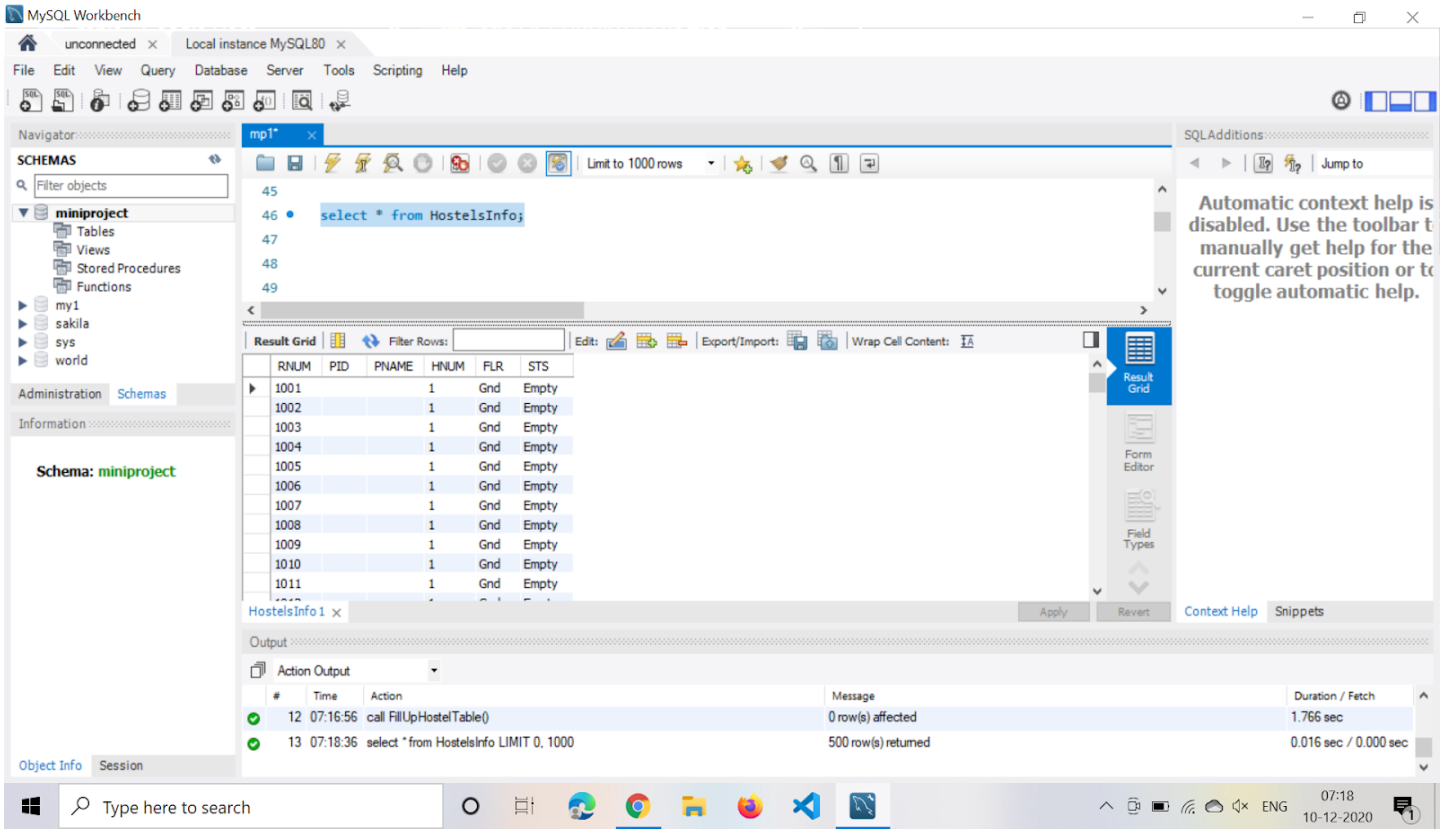
Procedure-1 :


MySQL Workbench
unconnected
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
miniproject
Tables
Views
Stored Procedures
Functions
my1
sakila
sys
world
Administration Schemas
Information
Schema: miniproject
Object Info Session
mp1*
Limit to 1000 rows
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
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
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
Type here to search
ENG
07:16
10-12-2020


MySQL Workbench
unconnected
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
miniproject
Tables
Views
Stored Procedures
Functions
my1
sakila
sys
world
Administration Schemas
Information
Schema: miniproject
Object Info Session
mp1*
Limit to 1000 rows
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();
45
46
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
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
Type here to search
ENG
07:17
10-12-2020

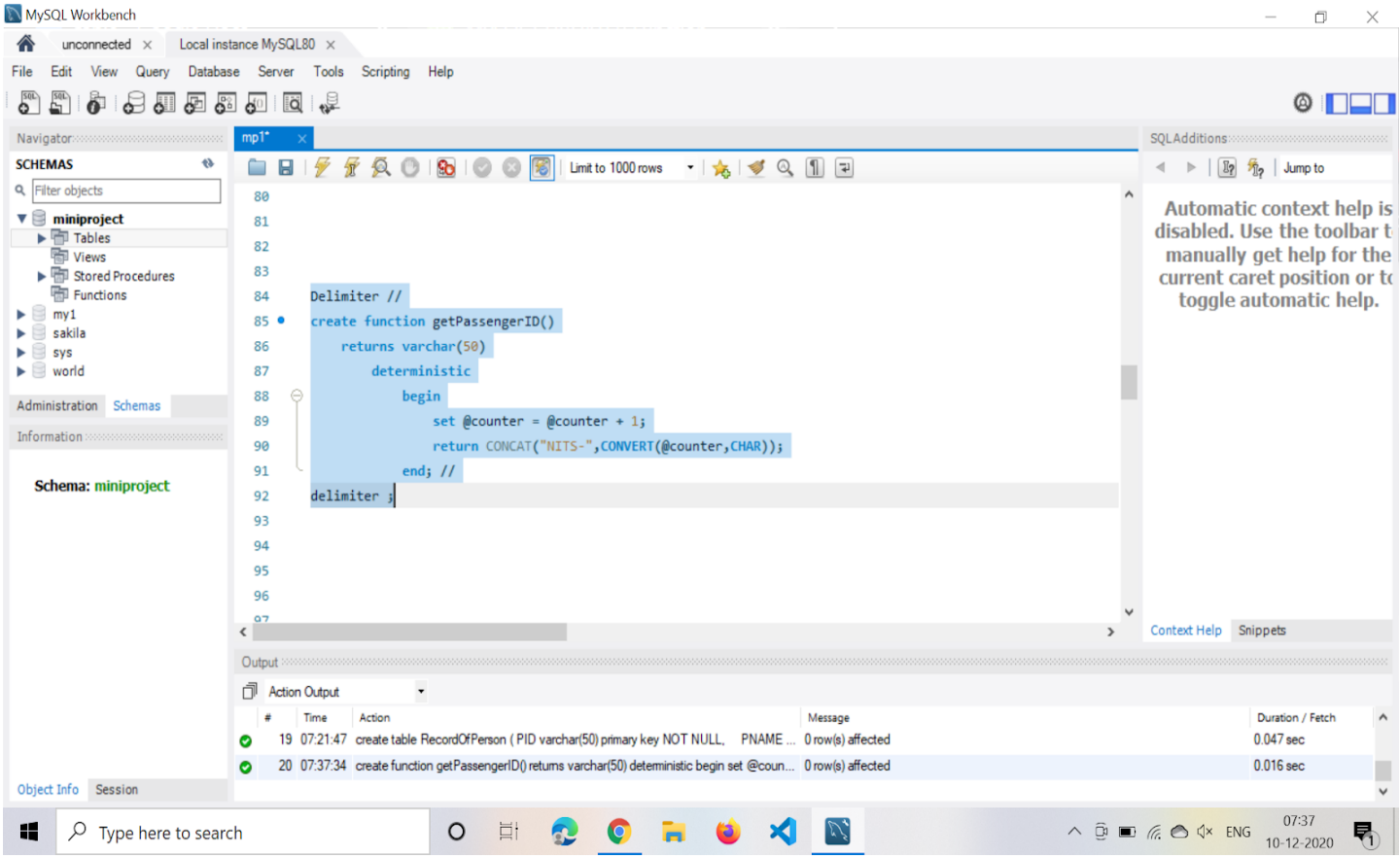
This procedure is created to fill the table "**HostelsInfo**". Initially, the table "**HostelsInfo**" will only contain the status of all the rooms i.e, EMPTY. Since, it was manually not possible to add 500(250 + 250) rooms in the table. So, we created a procedure for doing the same. The parameters (Passenger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

The table after executing the above procedure is shown below :



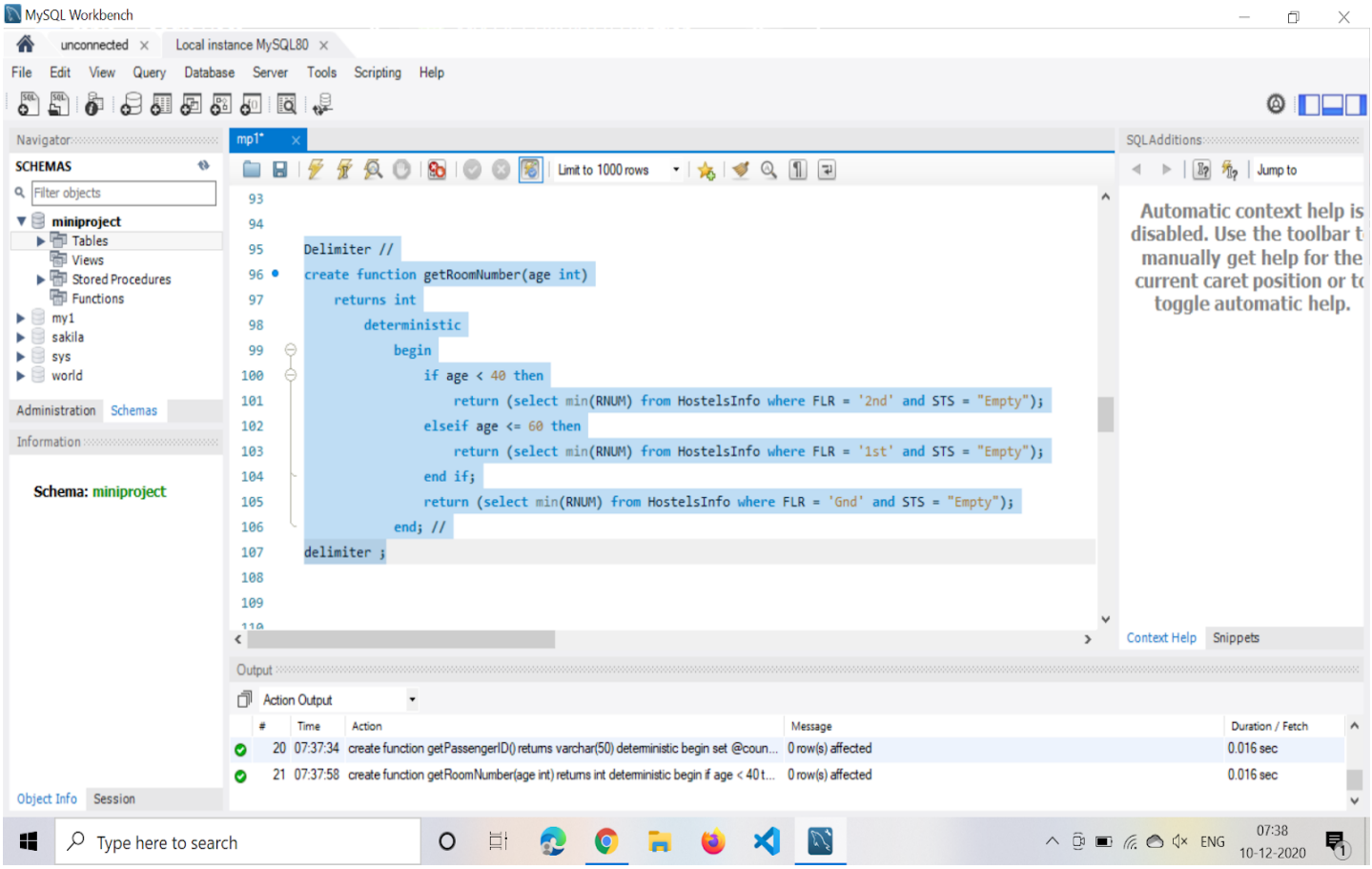
Initially all the rooms are empty. Therefore, the STS (status) of all the rooms is EMPTY.

Function-1:



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

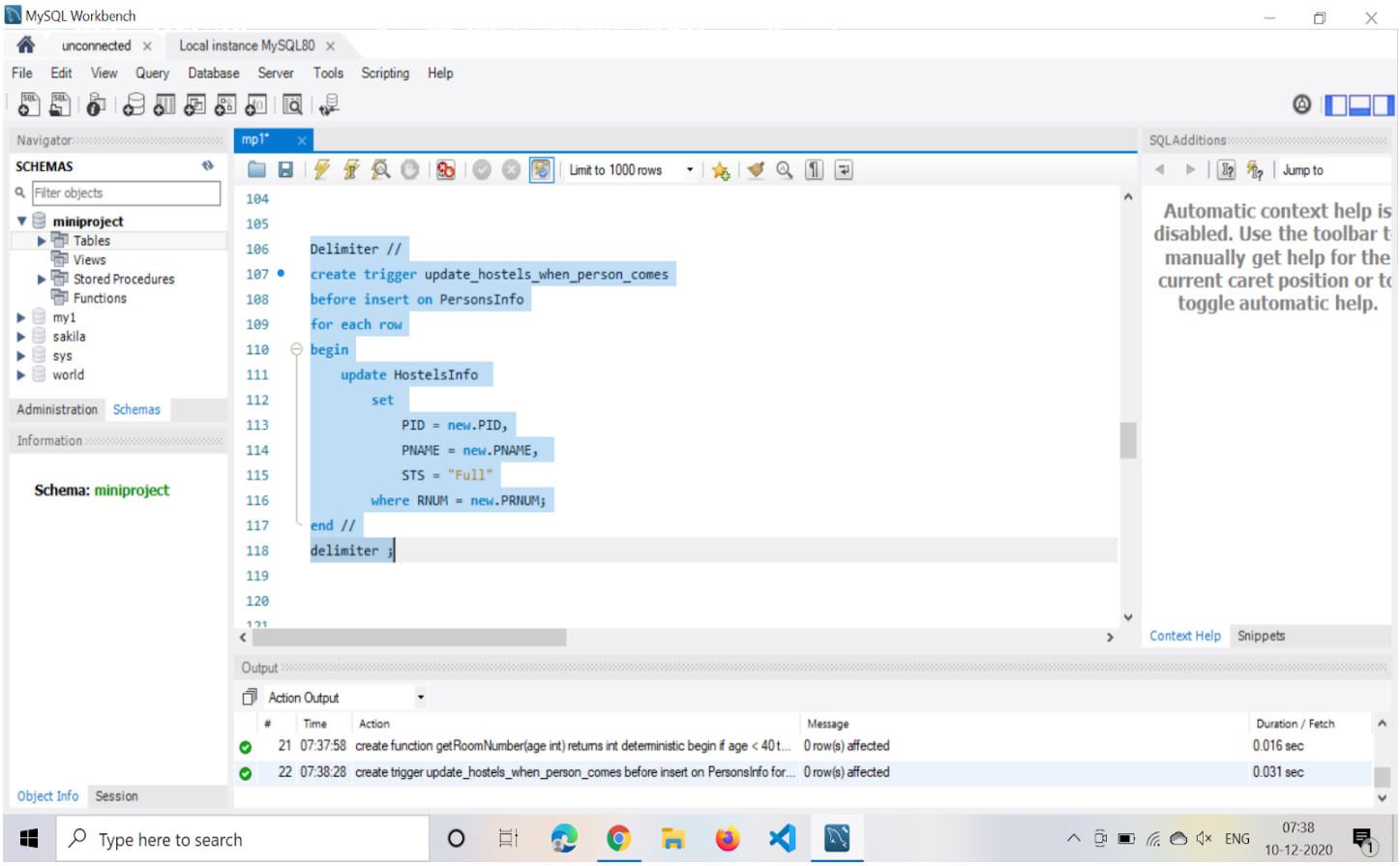
If age > 60, then Passenger gets room on the ground floor.

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.

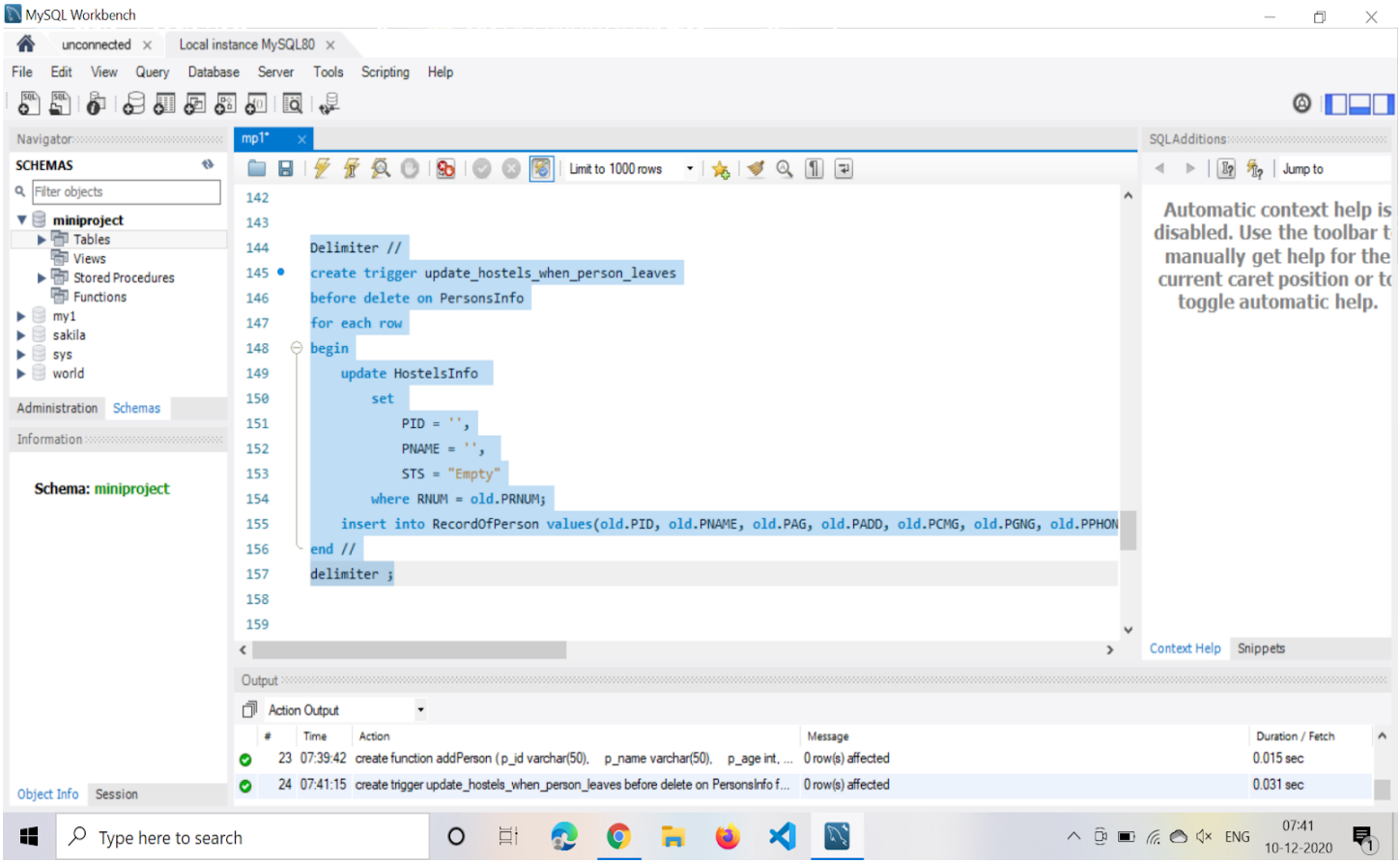
Trigger-1:



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:

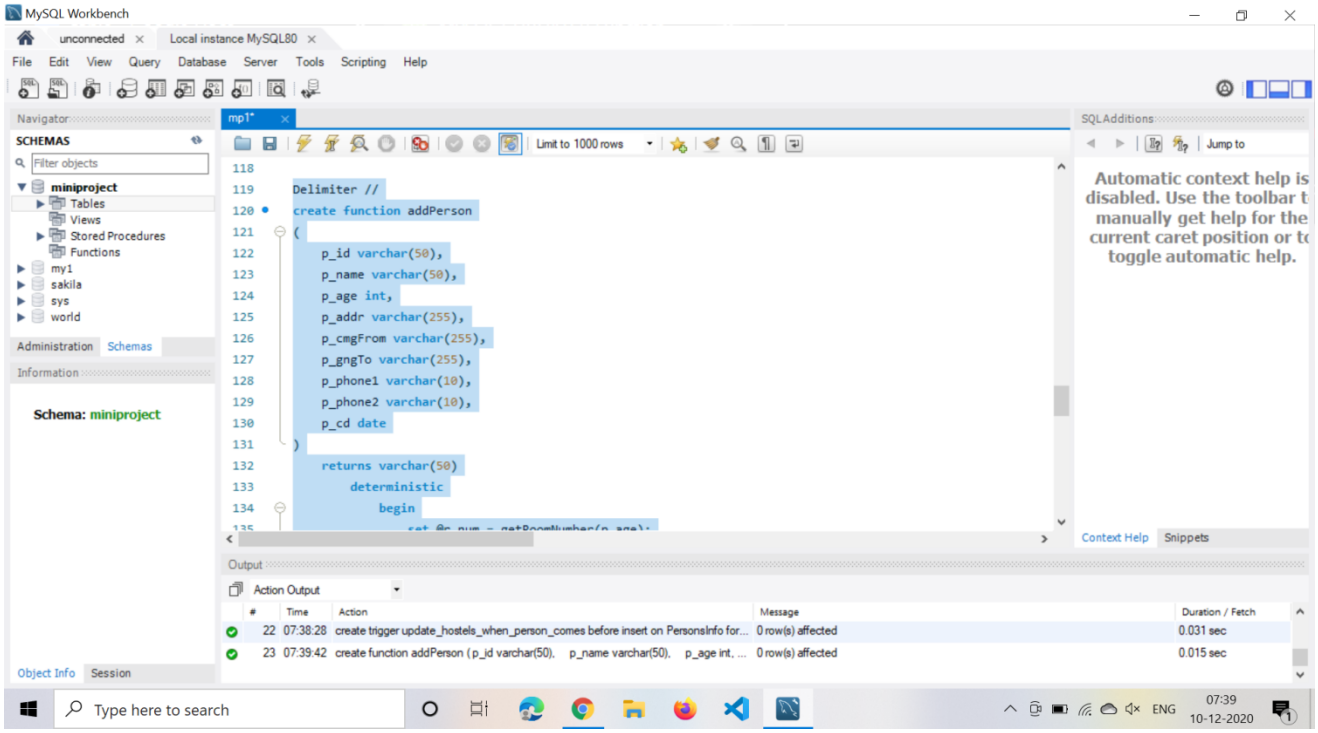


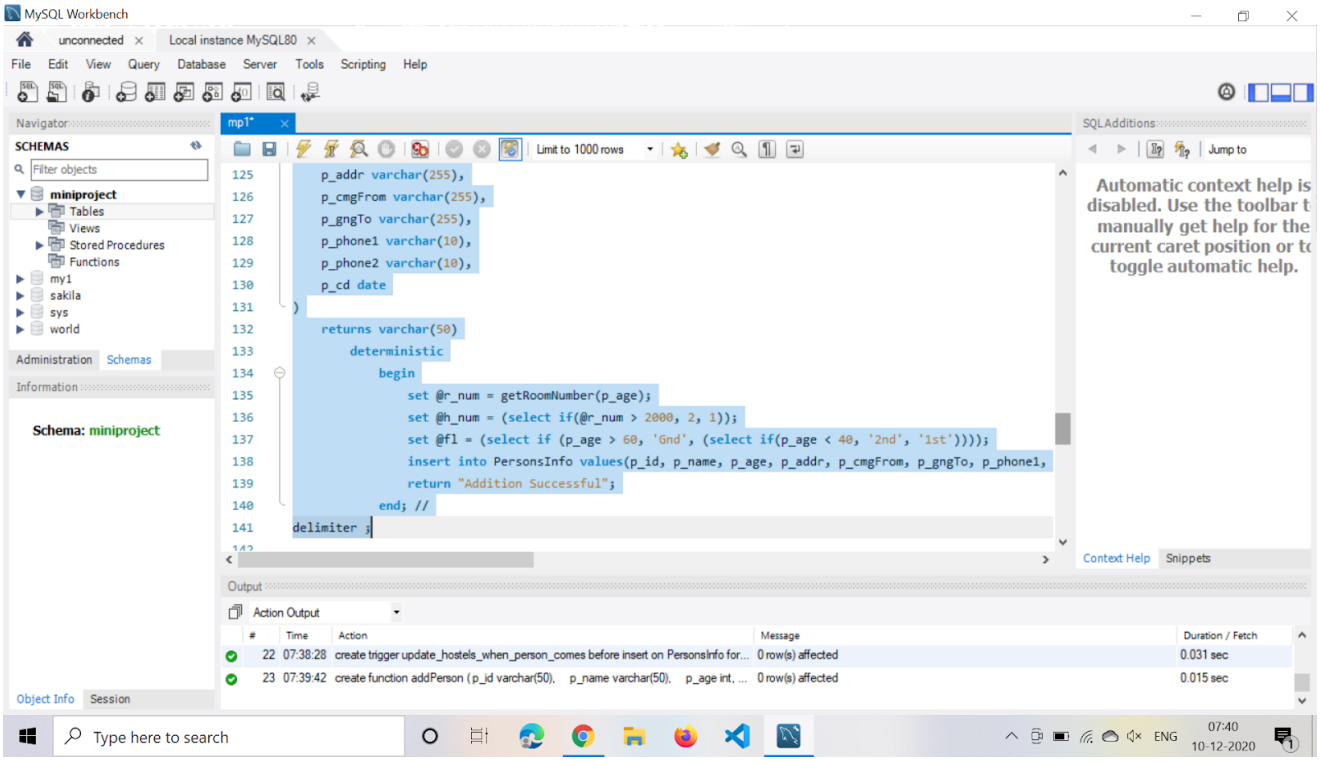
This **trigger** is written because as soon as a Passenger leaves, all his information has to be deleted from the **PersonsInfo** table.

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.

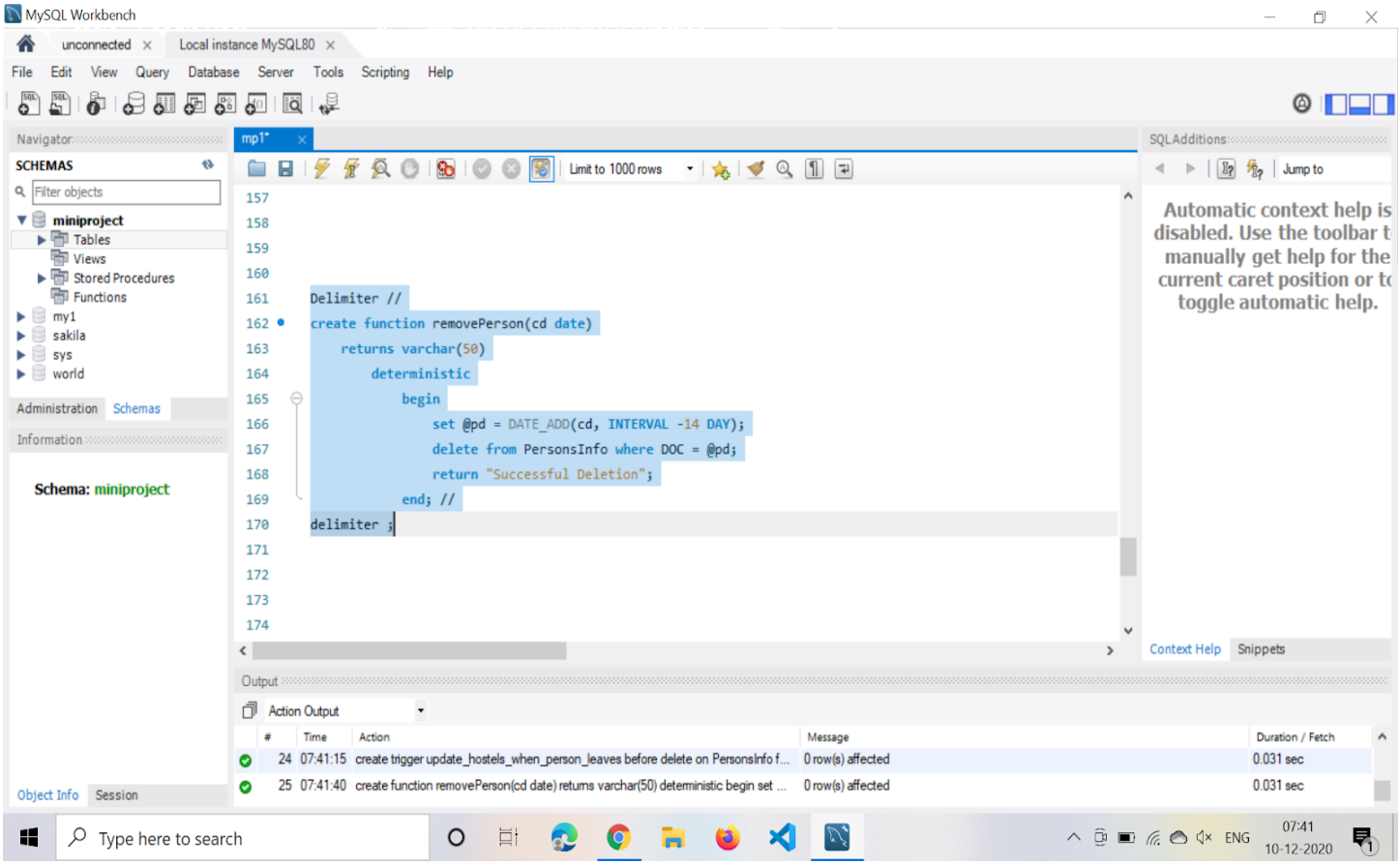
Function-3:



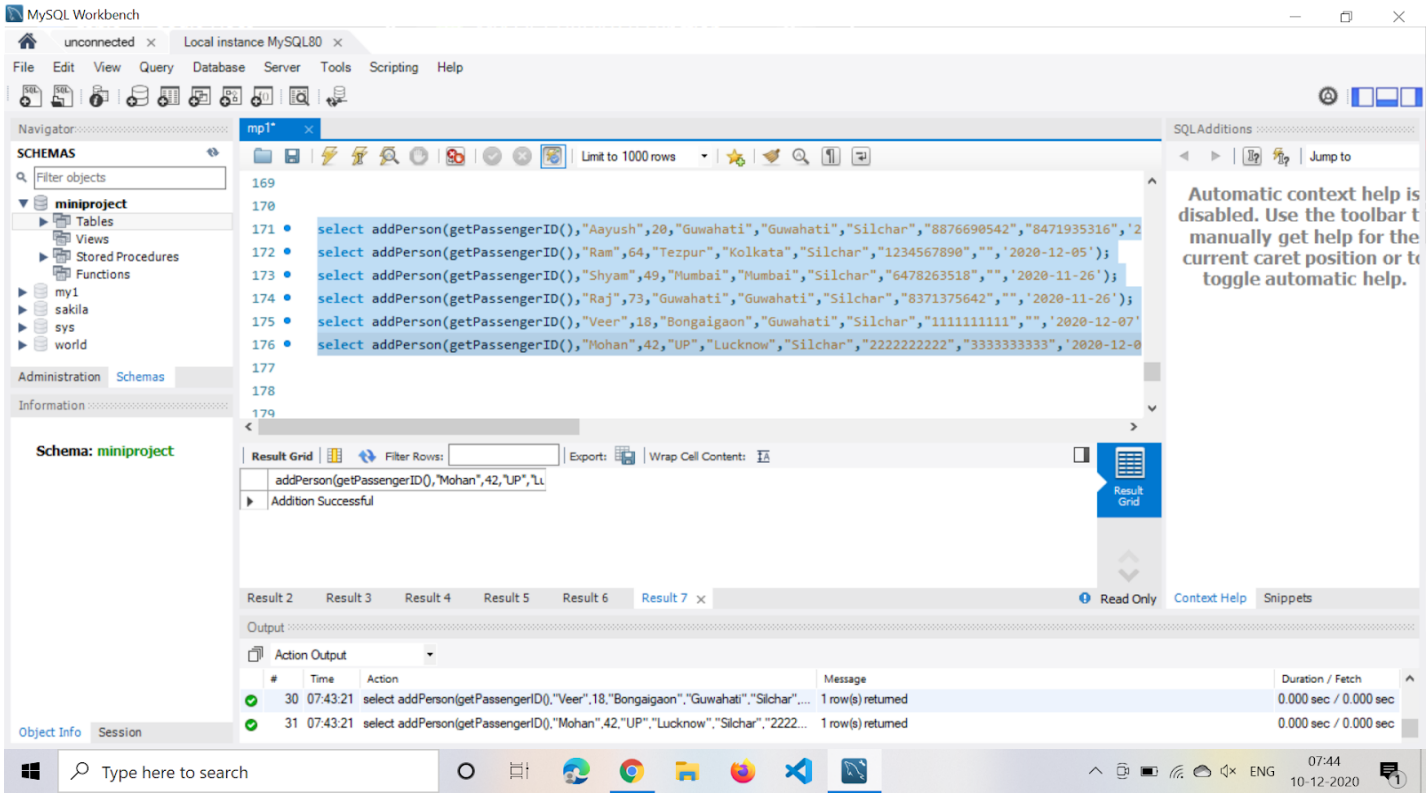


This function "**addPerson**" adds the details of the Passenger in the respective tables. It takes all the identification attributes as parameters.

Function-4:

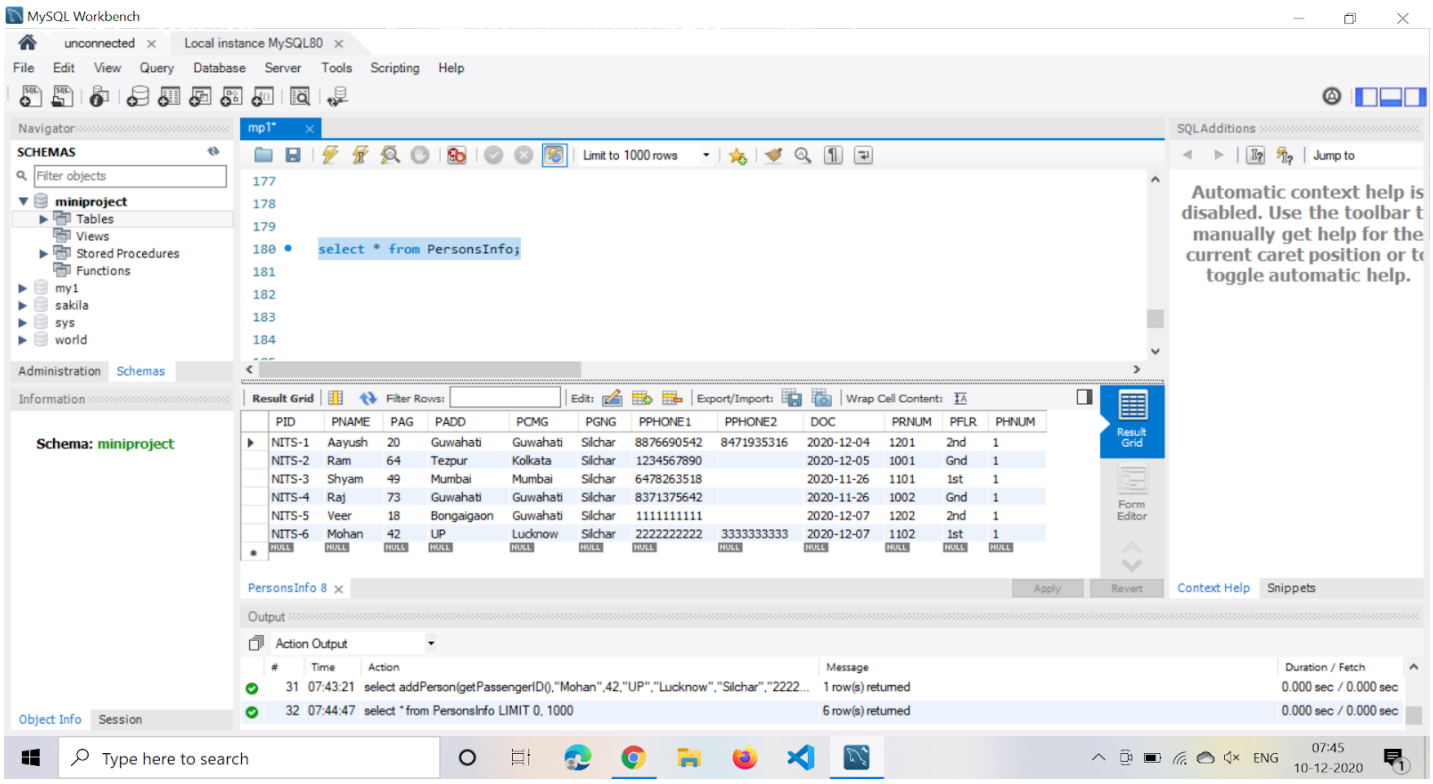


This function "**removePerson**" removes the details of the Passenger from the respective tables. It takes the current date as parameter which means it removes the list of Passengers whose 14 days of quarantine are over today.

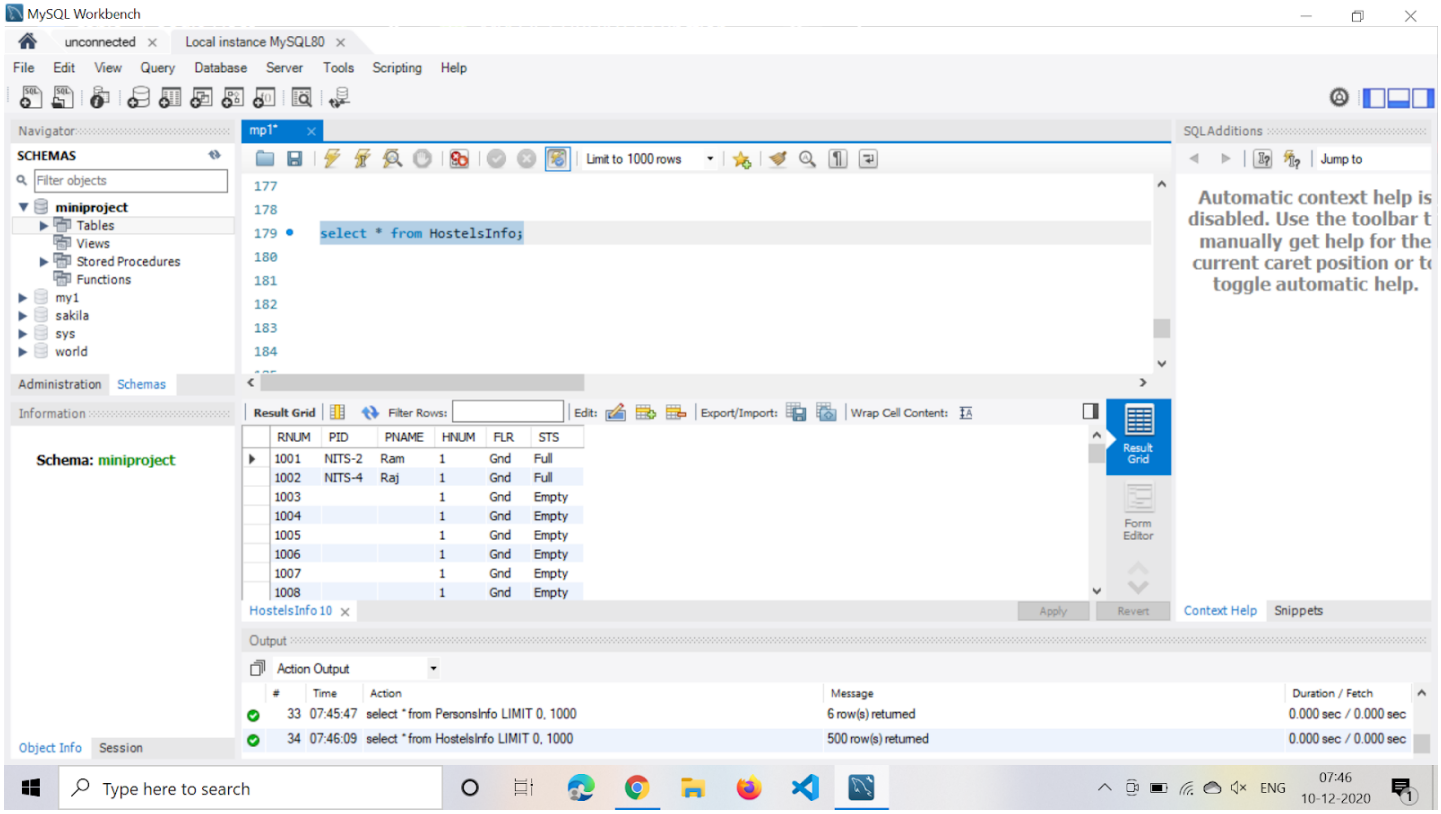


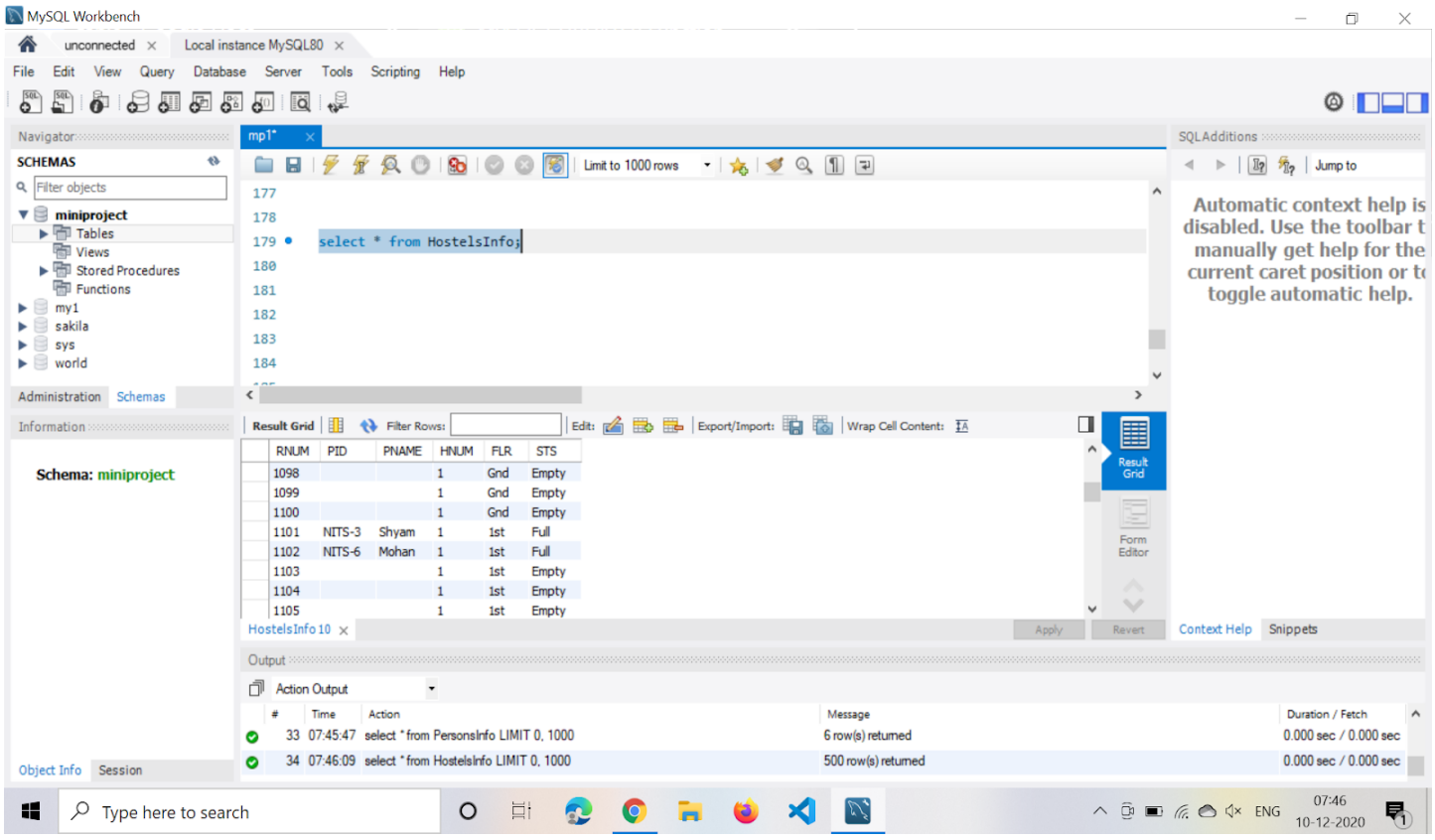
At first, we add the details of 5 Passengers of varying age in the database and observe the following tables:

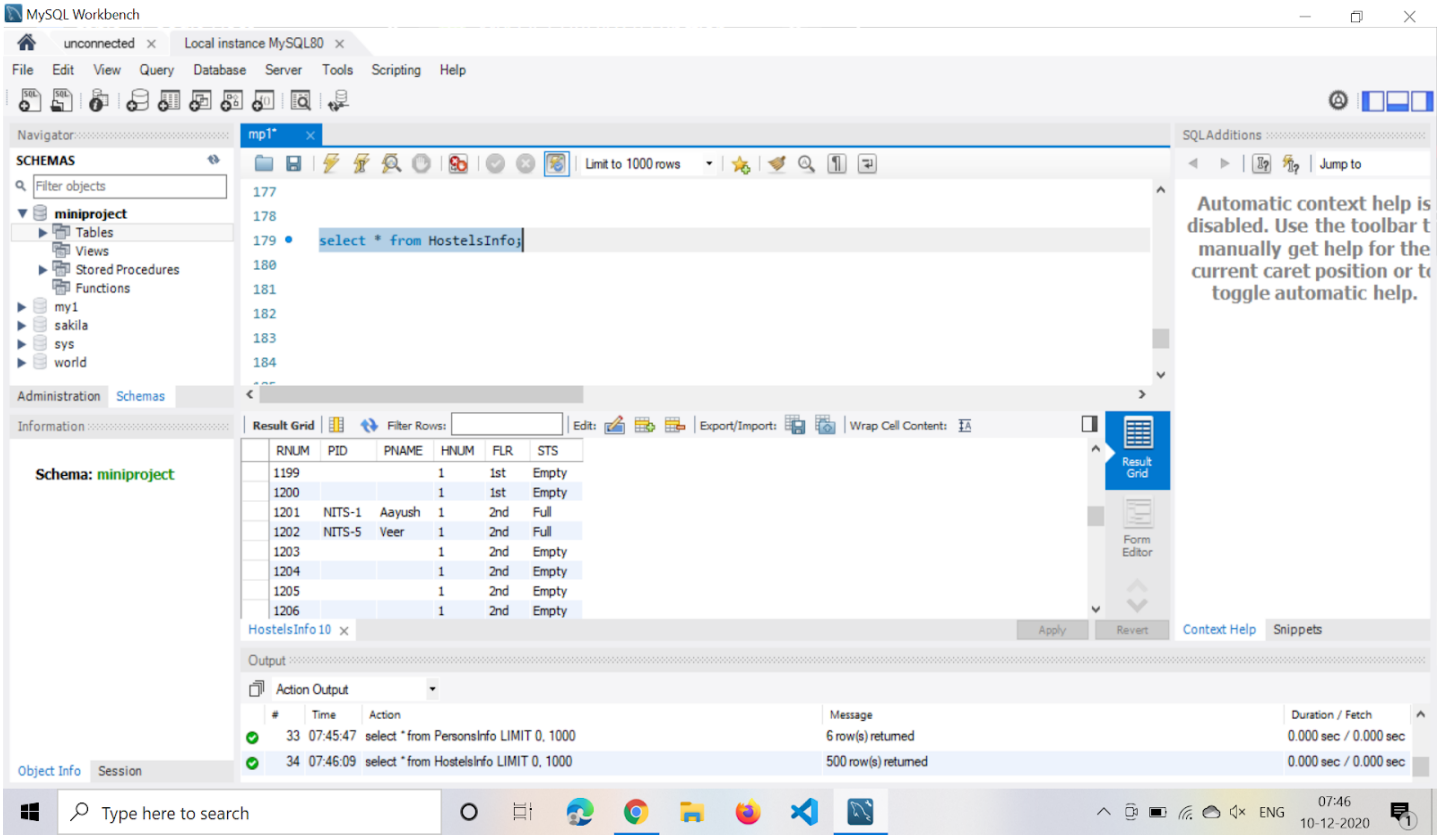
The **PersonsInfo** table --



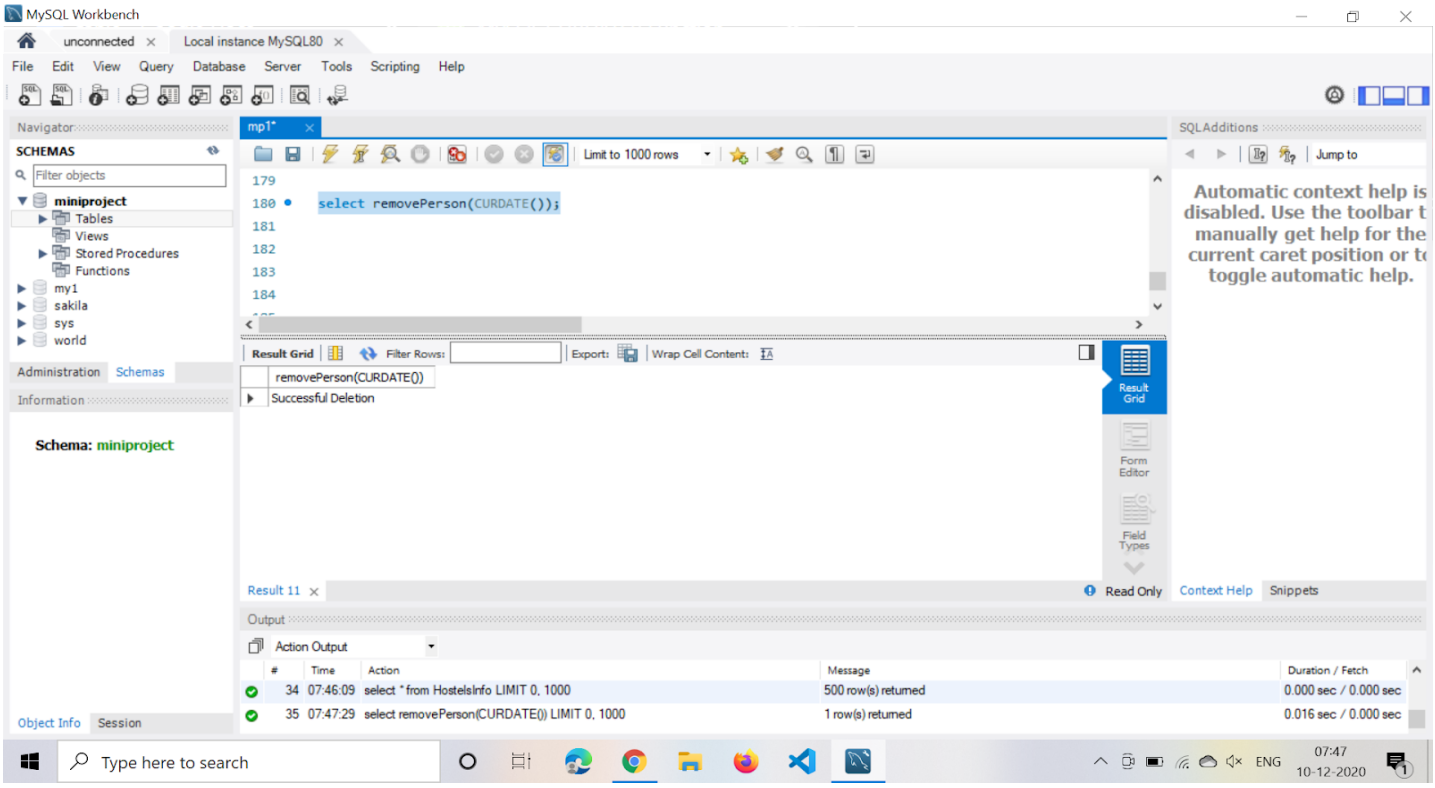
The **HostelsInfo** table –





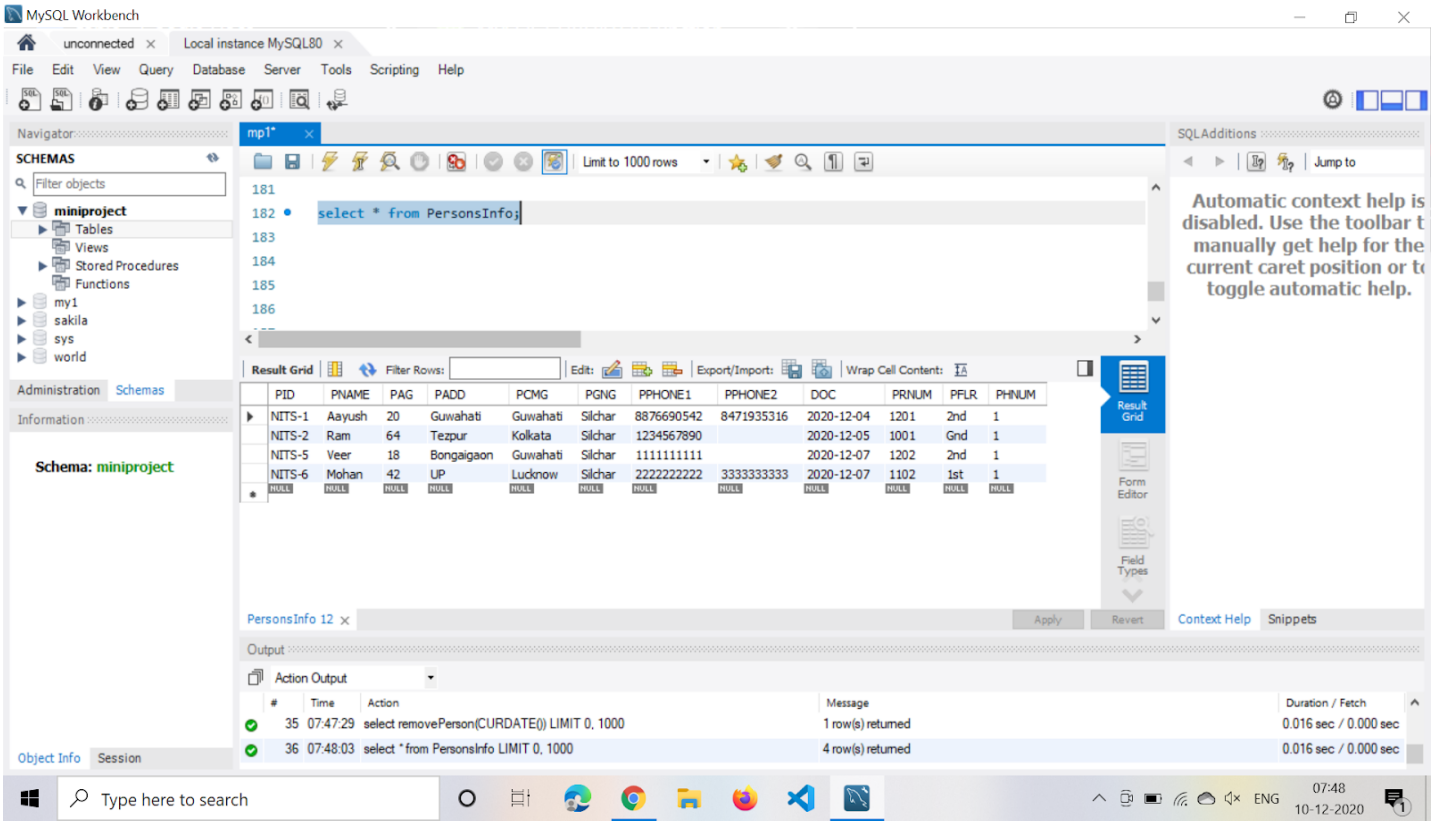


As we thought earlier, the **HostelsInfo** table is exactly like that.



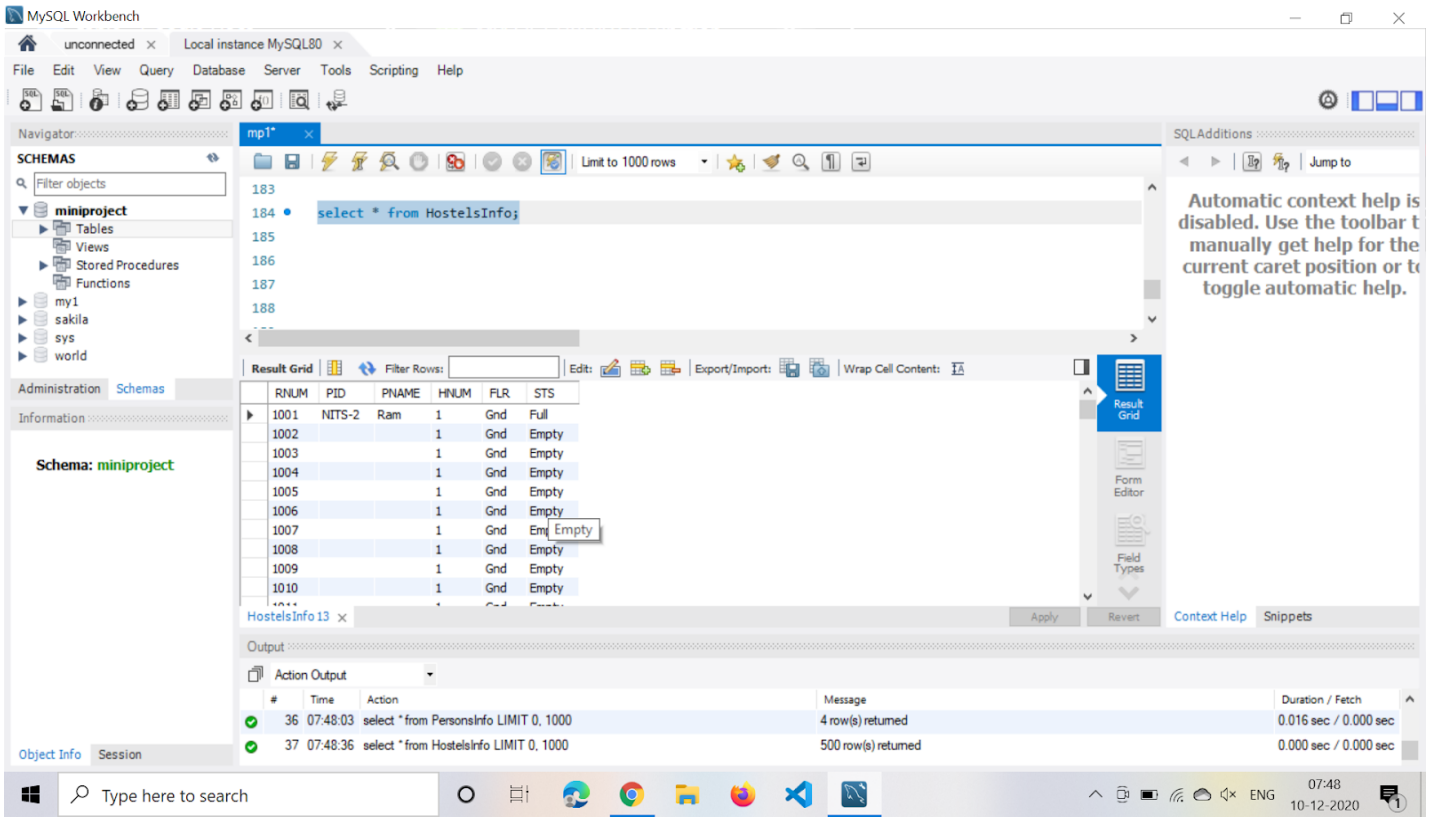
Now, we manage all the persons whose quarantine period is over today i.e, the current day and observe the tables again :

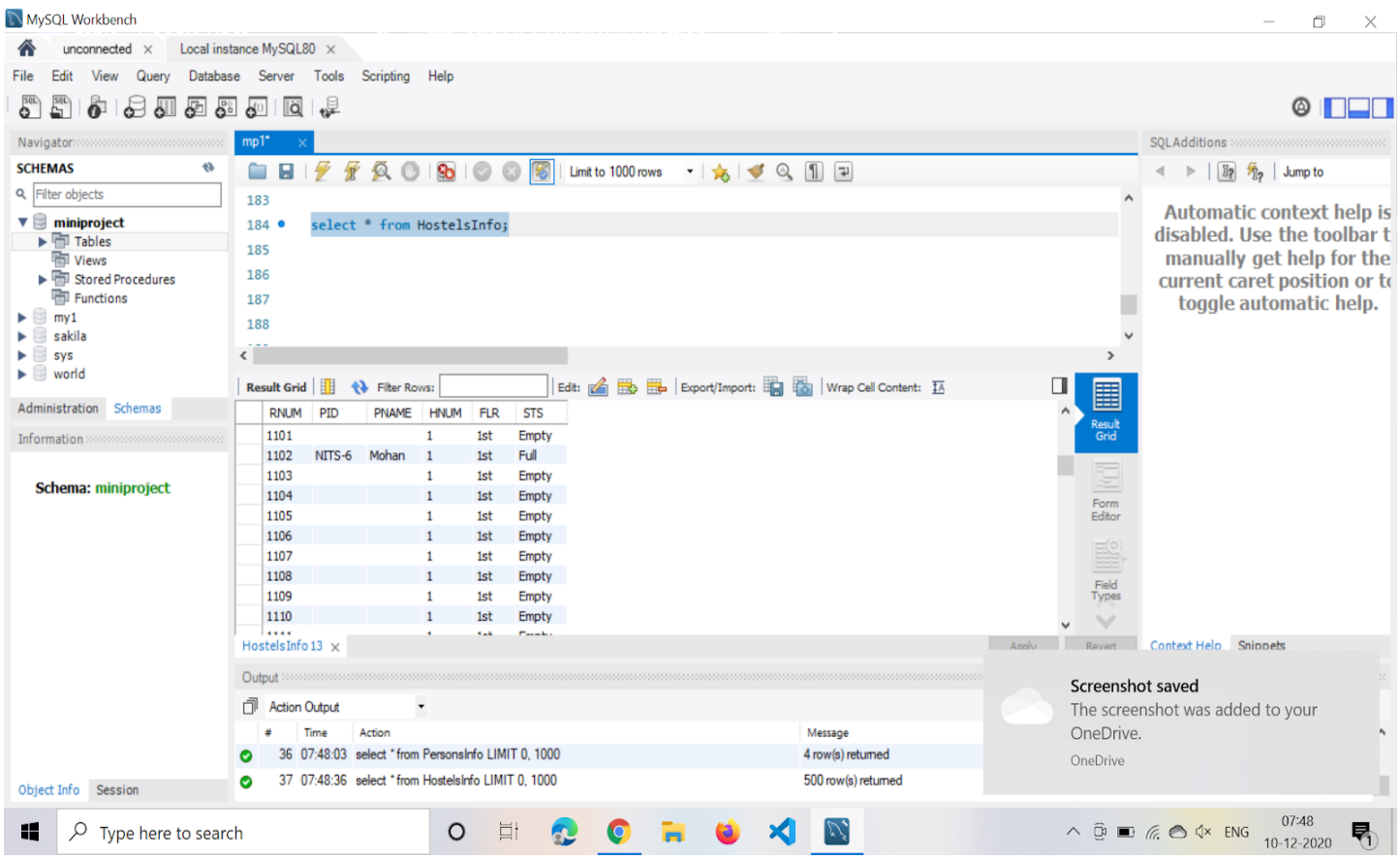
The **PersonsInfo** table --

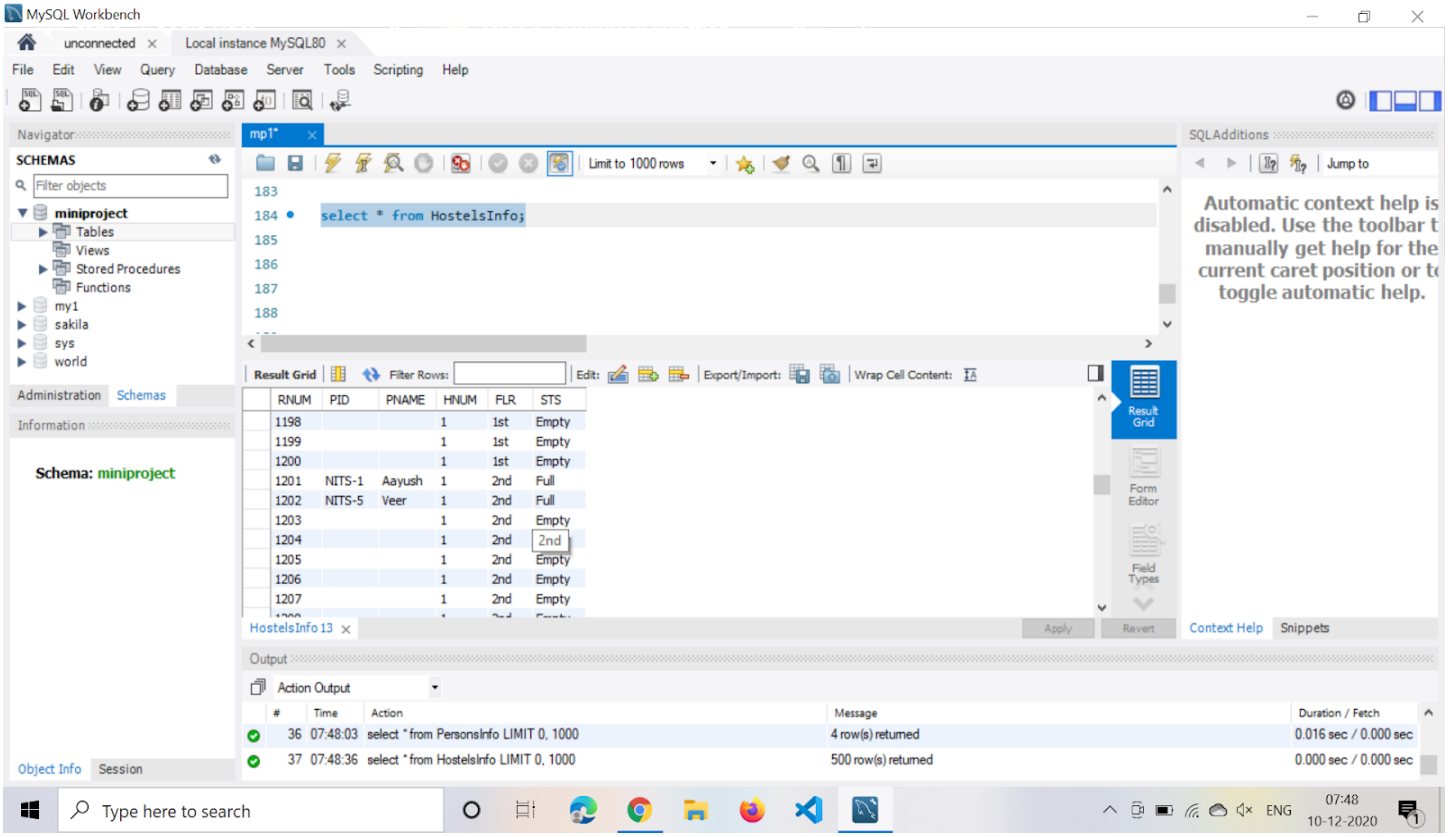


Two passengers are removed as their 14 days are over.

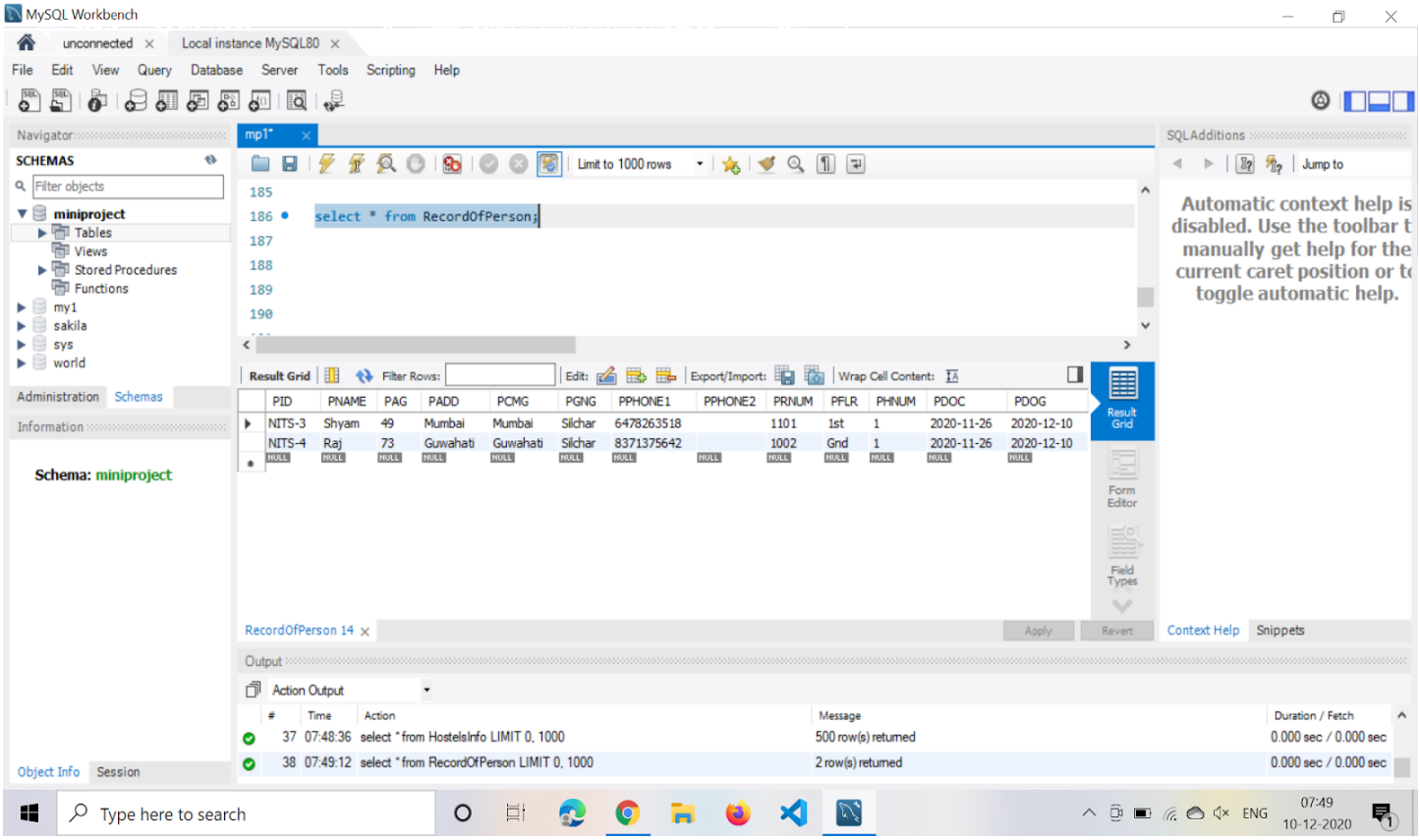
The **HostelsInfo** table --







The **RecordOfPerson** table--



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.