

**SEMESTER: - IV**

**COURSE NAME: - DATABASE MANAGEMENT SYSTEM**

**PROJECT TITLE: - HOTEL MANAGEMENT SYSYTEM**

**TEAM MEMBERS: - DEVARSH SHETH (AU1940189) (BTECH CSE)**

**MEET JHAVERI (AU1940284) (BTECH CSE)**

**KARAN SHAH (AU1940292) (BTECH CSE)**

* **DESCRIPTION OF PROJECT**

The main objective of this project is to create a database management system for a hotel. The hotel can have multiple chains, which can further have multiple hotels linked with it. Therefore, we need an organized management system, which can easily manage all the operations and data of the hotel chains and hotels respectively. We will be managing the below areas of the hotel database management system.

* The hotel chains and their details.
* The hotels in each chain and their details and other information like the rooms along with their description, ratings given to the hotels, discounts, etc.
* Information about employees and departments they work in.
* Information about guests.
* Managing bookings and other services used by the guests.

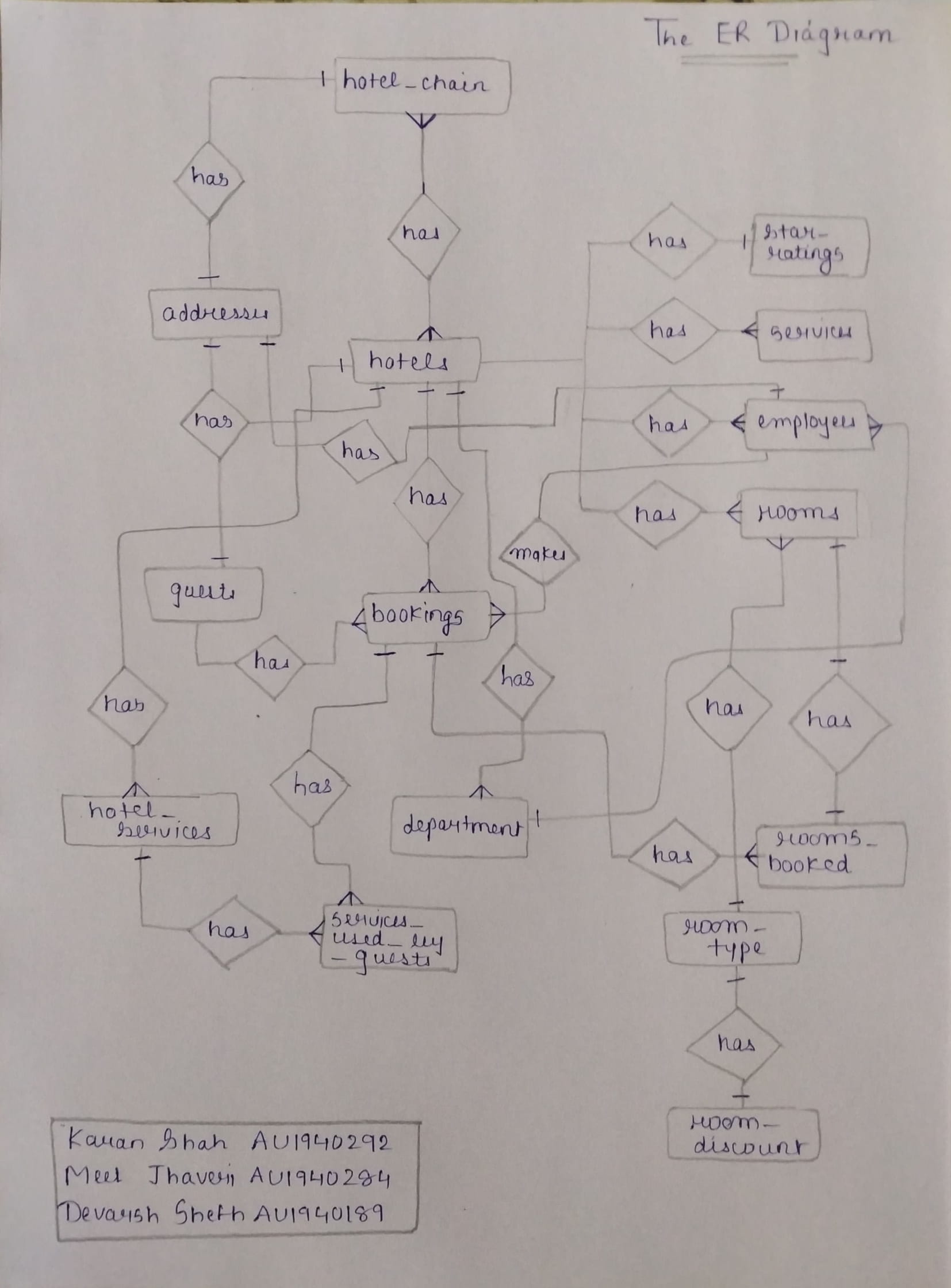
Apart from this we implemented triggers, procedures and functions on the given tables with the view to update, insert or delete the records of respective tables.

* **System Requirement Specification:**

Database created in: - MySQL Workbench

Frontend: - Trying to create in React

* **ENTITY-RELATIONSHIP DIAGRAM**

****

**One-to-One Relationship:**

This relationship is shown by connecting two entities by denoting two single dash ().

**For e.g.: -** The relation between ‘hotel\_chain’ and ‘addresses’ is one-to-one relation.

**One-to-Many Relationship:**

This relationship is shown by connecting two entities by denoting single dash () on one side and 3 dash () on other side.

**For e.g.: -** The relation between ‘Hotels’ and ‘Bookings’ is one-to-many relation.

**Many-to-One Relationship:**

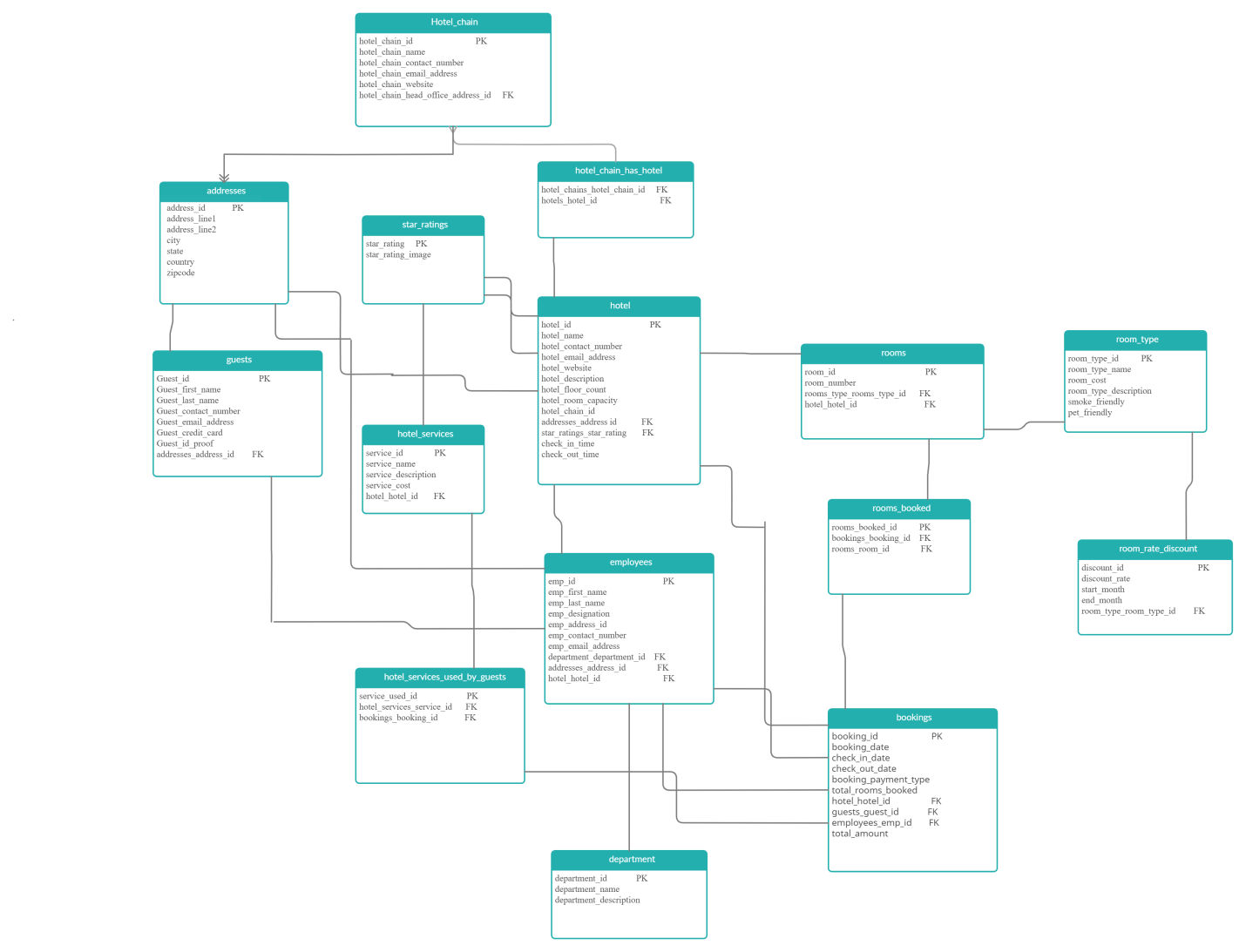
This relationship is shown by connecting two entities by denoting three dash () on one side and single dash () on other side

**Many-to-Many Relationship:**

This relationship is shown by connecting two entities by denoting three dash ().

**For e.g.: -** The relation between ‘hotel\_chain’ and ‘hotels’ is many-to-many relation.

* **TABLE DESIGN (DATA DICTIONARY)**

****

1. **Addresses**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Address\_id | INT | Primary key, Not null | 1234 | The unique id given to the address |
| Address\_line1 | VARCHAR (100) | Null | 9, Shnti Apts | Line 1 of the address |
| Address\_line2 | VARCHAR (100) | Null | Nr. Manekbaug | Line 2 of the address |
| City | VARCHAR (45) | Null | Ahmedabad | Shows the name of the city |
| State | VARCHAR (45) | Null | Gujarat | Shows the state name |
| Country | VARCHAR (45) | Null | India | Shows the country name |
| Zipcode | VARCHAR (8) | Null | 380015 | Provides zipcode of the area. |

1. **Hotel\_chain**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Hotel\_chain\_id | INT | Primary Key, Not Null | 123 | The unique id given to every hotel\_chain. |
| Hotel\_chain\_name | VARCHAR (45) | Null | ABC | Name given to the hotel chain |
| Hotel\_chain\_contact\_number | VARCHAR (12) | Null | 123456789 | Provides the contact number of hotel chain |
| Hotel\_chain\_email\_address | VARCHAR (45) | Null | abd@gmail.com | Provides the mail id |
| Hotel\_chain\_website | VARCHAR (45) | Null | www.abd.com | Provides the details about hotel chain on web page |
| Hotel\_chain\_head\_office\_address\_id | INT | Foreign Key, Not Null | 123 | Provides the unique id of head office in hotel chain. |

1. **Star\_ratings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraint** | **Format** | **Description** |
| Star\_rating | INT | Primary key, Not null | 102 | Describes the hotel’s rating |
| Star\_rating\_image | VARCHAR (100) | Null | Image | Shows the image in terms of stars |

1. **Hotel**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraint** | **Format** | **Description** |
| Hotel\_id | INT | Primary key, Not null | 23 | Provides the unique id of hotel |
| Hotel\_name | VARCHAR (45) | Null | ams | Shows the hotel name |
| Hotel\_contact\_number | VARCHAR (12) | Null | 982504447 | Provides contact no. of the hotel |
| Hotel\_email\_address | VARCHAR (45) | Null | asd@gmail.co.in | Provides the mail id of the hotel |
| Hotel\_website | VARCHAR (45) | Null | www.ansm.com | Provides the details about hotel on web page |
| Hotel\_description | VARCHAR (100) | Null | Details of the hotel | The description related to hotel is provided. |
| Hotel\_floor\_count | INT | Null | 34 | Gives the total no of floors |
| Hotel\_room\_capacity | INT | Null | 12 | Gives the total capacity of room |
| Hotel\_chain\_id | INT | Null | 1 | Provides id for the hotelchain |
| Addresses\_address\_id | INT | Not null , Foreign key | 1234 | The unique id given to the address |
| Star\_ratings\_star\_rating | INT | Not Null, Foreign key | 11 | Provides rating for the hotel |
| Check\_in\_time | TIME | Null | 02:00:00 | Describes the check in time |
| Check\_out\_time | TIME | Null | '23:00:00' | Describes the check out time |

1. **Room\_type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Room\_type\_id | INT | Not null, Primary key | 12 | Gives the room-type’s id |
| Room\_type\_name | VARCHAR (45) | Null | Standard room | Provides the name for the given type of room |
| Room\_cost | DECIMAL (10,2) | Null | 5000 | Shows the cost of particular room |
| Room\_type\_description | VARCHAR (100) | Null | ABCS | Provides the details of particular room type |
| Smoke\_friendly | TINY INT (1) | Null | 2 | Shows whether its smoke friendly or not |
| Pet\_friendly | TINY INT (1) | Null | 1 | Shows whether its pet friendly or not |

1. **Room**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Room\_id | INT | Not null, Primary key | 123 | Unique id of the room is provided |
| Room\_number | INT | Null | 123 | Gives the room number for the room |
| Rooms\_type\_rooms\_type\_id | INT | Not null, Foreign key | 123 | Gives the room-type’s id |
| hotel\_hotel\_id | INT | Not null, Foreign key | 1 | Provides the unique id of hotel |

1. **Guest**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Datatype** | **Constraints** | **Format** | **Description** |
| Guest\_id | INT | Not null, primary key | 101 | Provides unique id to guest |
| Guest\_first\_name | VARCHAR (45) | Null | ABC | Shows first name of the guest |
| Guest\_last\_name | VARCHAR (45) | Null | ABC | Shows last name of the guest |
| Guest\_contact\_number | VARCHAR (12) | Null | 982566647 | Shows contact no of the guest |
| Guest\_email\_address | VARCHAR (45) | Null | abc@gmail.com | Shows address of the guest |
| Guest\_credit\_card | VARCHAR (45) | Null | ABCS | Shows payment type of the guest |
| Guest\_id\_proof | VARCHAR (45) | Null | ABCD | Shows id proof of the guest |
| Addresses\_address\_id | INT | Not null, Foreign key | 1234 | The unique id given to the address |

1. **Department**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Department\_id | INT | Not null, Primary key | 14 | Provides the unique id for the respective department |
| Department\_name | VARCHAR (45) | Null | XYZ | Shows the name of department |
| Department\_description | VARCHAR (100) | Null | PQRS | Provide details of the particular department |

1. **Employees**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Emp\_id | INT | Not null, primary key | 24 | Provide unique id of the particular employee |
| Emp\_first\_name | VARCHAR (45) | Null | ABC | Provides first name of employee |
| Emp\_last\_name | VARCHAR (45) | Null | ABC | Provides first last of employee |
| Emp\_designation | VARCHAR (45) | Null | ABC | Provides designation of employee |
| Emp\_contact\_number | VARCHAR (12) | Null | 987454 | Provides contact number of employee |
| Emp\_email\_address | VARCHAR (45) | Null | abc@yahoo.com | Provides email id of employee |
| Department\_department\_id | INT | Not null, Foreign key | 36 | Provides the unique id for the respective department |
| Addresses\_address\_id | INT | Not null, Foreign key | 325 | Provides the unique id for the respective address |
| Hotel\_hotel\_id | INT | Not null, Foreign key | 21 | Provides the unique id for the respective hotel |

**(10)Bookings**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Booking\_id | INT | Not null, primary key | 30 | Provides the unique id for the respective bookings |
| Booking\_date | DATETIME | Null | 2020-08-07 | Shows the booking date |
| Check\_in\_date | DATETIME | Null | 2020-08-07 | Shows the check-in date |
| Check\_out\_date | DATETIME | Null | 2020-08-07 | Shows the check-out date |
| Booking\_payment\_type | VARCHAR (45) | Null | ABC | Shows the method of payment |
| Total\_rooms\_booked | INT | Null | 8 | Shows the total no of rooms booked |
| Hotel\_hotel\_id | INT | Not null, Foreign key | 45 | Provides the unique id for the respective hotel |
| Guests\_guest\_id | INT | Not null, Foreign key | 66 | Provides the unique id for the respective guests |
| Employees\_emp\_id | INT | Not null, Foreign key | 85 | Provides the unique id for the respective employees |
| Total\_amount | DECIMAL (10,2) | Null | 6000 | Shows the total expense done |

**(11)Hotel\_chain\_has\_hotel**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Hotel\_chains\_hotel\_chain\_id | INT | Not null, Foreign key | 87 | Provides the unique id for the respective hotel chains |
| Hotels\_hotel\_id | INT | Not null, Foreign key | 69 | Provides the unique id for the respective hotels |

**(12)rooms\_rate\_discount**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Discount\_id | INT | Not null, primary key | 6 | Provides the unique id for the discount |
| Discount\_rate | DECIMAL (10,2) | Null | 150 | Provides the discount rate |
| Start\_month | TINYINT (1) | Null | 1 | Provides the discount’s starting month |
| End\_month | TINYINT (1) | Null | 2 | Provides the discount’s ending month |
| Room\_type\_room\_type\_id | INT | Not null, foreign key | 66 | Gives the room-type’s id |

**(13)Rooms\_booked**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Rooms\_ booked\_id | INT | Not null, primary key | 98 | Provides the unique id for the booked rooms |
| Bookings\_booking\_id | INT | Not null, foreign key | 63 | Shows the total no of rooms booked |
| Rooms\_room\_id | INT | Not null, foreign key | 44 | Provides the unique id for the rooms |

**(14)hotel\_services**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Service\_id | INT | Not null, primary key | 7 | Provides the unique id for the services |
| Service\_name | VARCHAR (45) | Null | ABC | Provides the name for the service |
| Service\_description | VARCHAR (100) | Null | ABC | Provides the description of service |
| Service\_cost | DECIMAL (10,2) | Null | 6548 | Provides the service cost |
| Hotel\_hotel\_id | INT | Not null, foreign key | 1 | Provides the unique id for the hotel |

**(15) hotel\_services\_used\_by\_guests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Data type** | **Constraints** | **Format** | **Description** |
| Service\_used\_id | INT | Not null, primary key | 11 | Provides the unique id for the services |
| Hotel\_services\_service\_id | INT | Not null, foreign key | 99 | Provides the unique id for the hotel services |
| Bookings\_booking\_id | INT | Not null, foreign key | 87 | Provides the unique id for the bookings |

* **Triggers**

(1 )Before\_employee\_update

set sql\_notes=0;

create table employees\_audit(

id INT AUTO\_INCREMENT PRIMARY KEY,

emp\_id INT NOT NULL,

emp\_first\_name varchar(45) NOT NULL,

emp\_last\_name varchar(45) NOT NULL,

emp\_contact\_number varchar(12) NOT NULL,

changedat DATETIME DEFAULT NULL,

action VARCHAR(50) DEFAULT NULL);

DELIMITER //

CREATE TRIGGER before\_employee\_update

BEFORE UPDATE ON employees

FOR EACH ROW

INSERT INTO employees\_audit

SET action = 'update',

emp\_id = OLD.emp\_id,

emp\_last\_name = OLD.emp\_last\_name,

emp\_first\_name = OLD.emp\_first\_name,

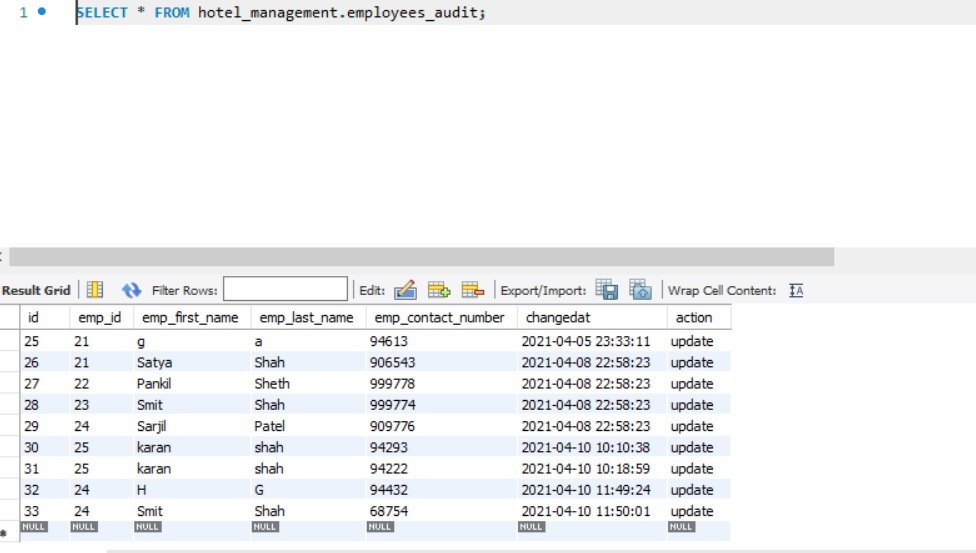
emp\_contact\_number = OLD.emp\_contact\_number,

changedat = NOW();

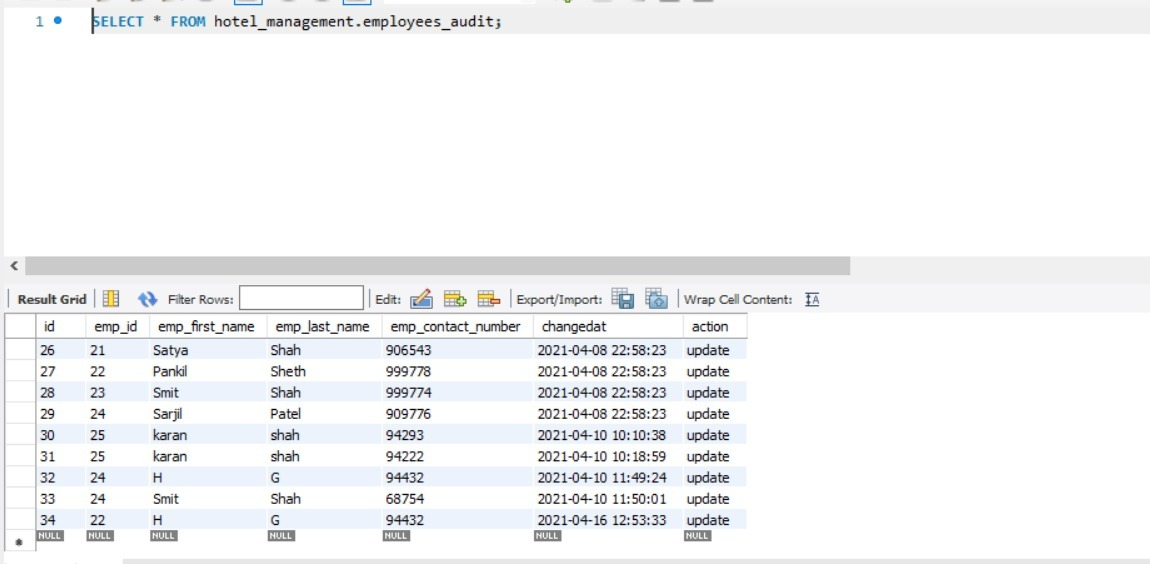
DELIMITER ;

set sql\_notes=1;

Before update



After update



1. **Chk\_empid**

DELIMITER //

CREATE TRIGGER chk\_empid

AFTER INSERT ON employees

FOR EACH ROW

BEGIN

DECLARE msg varchar(255);

IF new.emp\_id = null THEN

SET msg = 'INVALID EMPLOYEE ID';

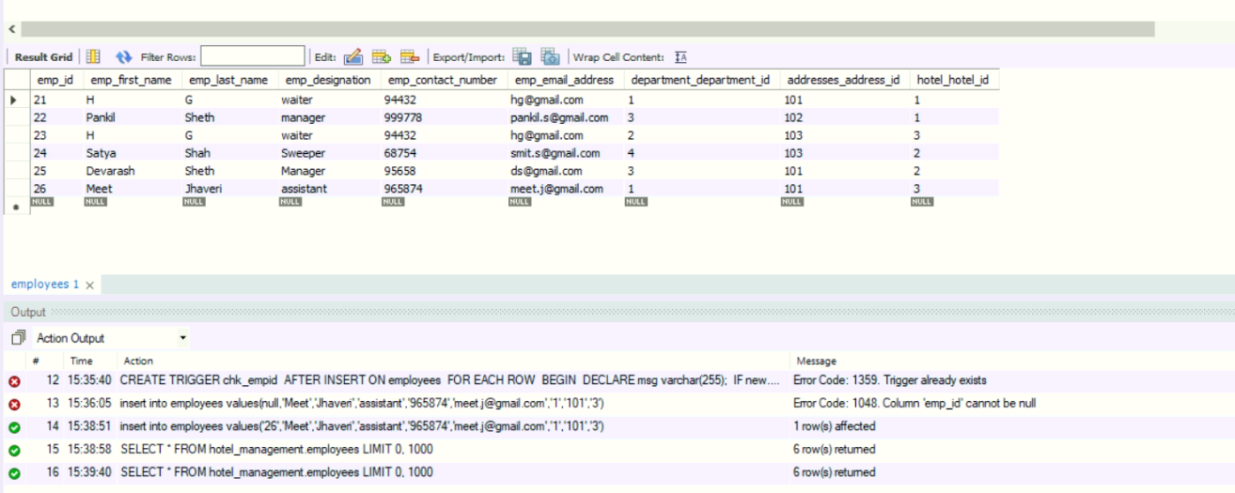
SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = msg;

END IF;

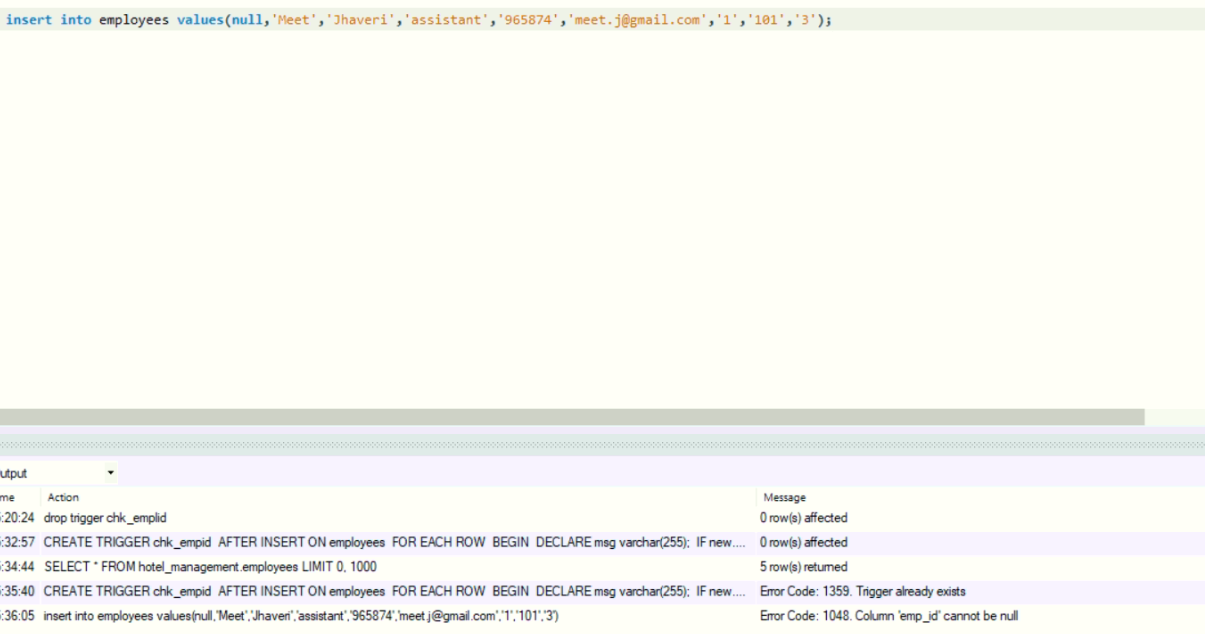
END

//

Before insert



After placing null value in emp\_id



1. **Service\_del**

create table service\_archives(

id INT PRIMARY KEY AUTO\_INCREMENT,

service\_id INT NOT NULL,

service\_name VARCHAR(45) NULL,

service\_description VARCHAR(100) NULL,

service\_cost DECIMAL(10,2) NULL,

deletedAt TIMESTAMP DEFAULT NOW());

DELIMITER $$

CREATE TRIGGER service\_del

BEFORE DELETE

ON hotel\_services FOR EACH ROW

BEGIN

INSERT INTO service\_archives (service\_id,service\_name,service\_description, service\_cost)

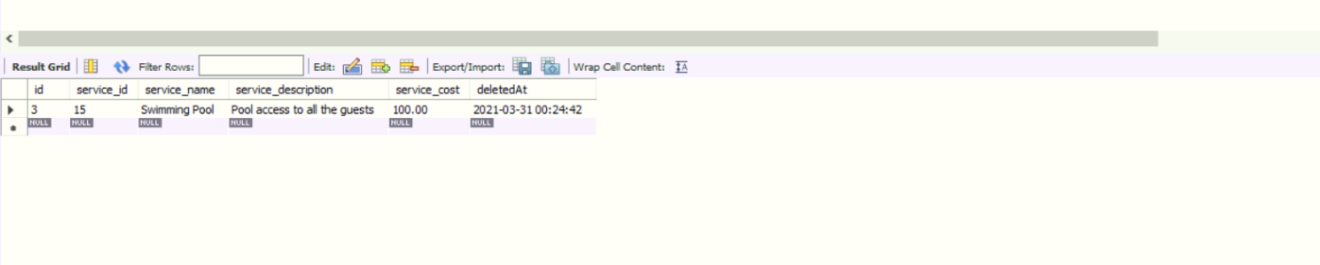
VALUES(old.service\_id, old.service\_name, old.service\_description, old.service\_cost);

END$$

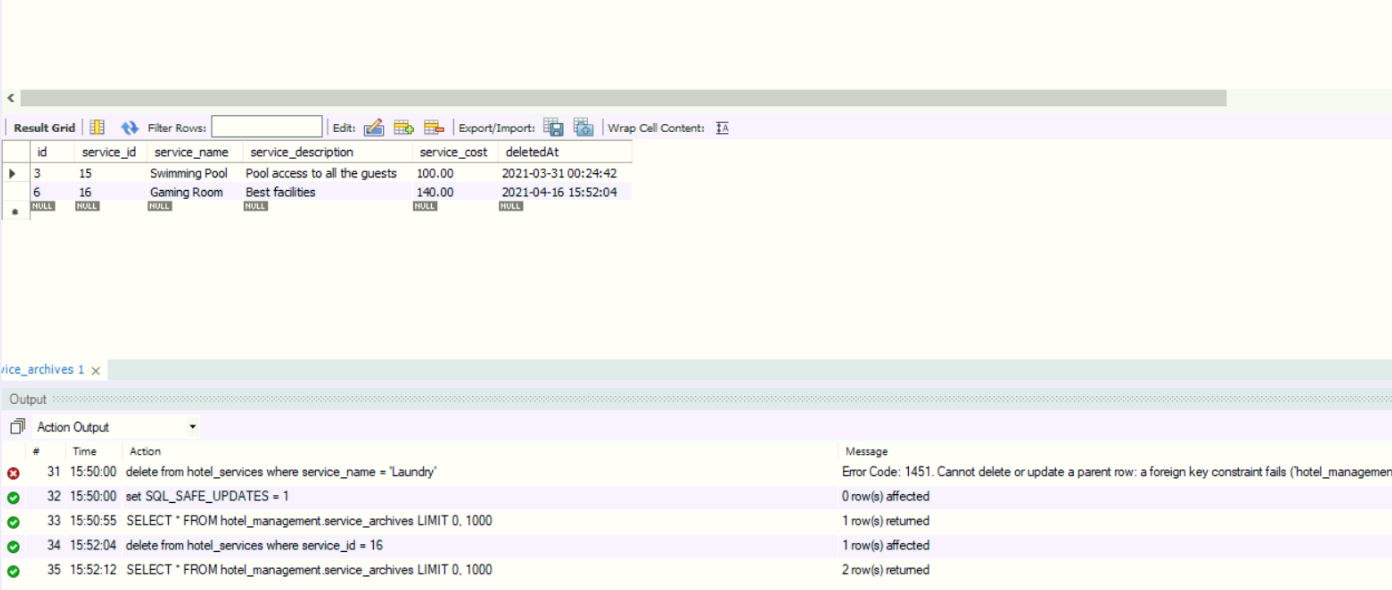
DELIMITER ;

set sql\_notes = 1;

Before deleting specified services



After deleting the specified service with service\_id=16



1. **After\_insert\_detail**

CREATE TABLE department\_detail (

department\_id int NOT NULL,

department\_name varchar(45) DEFAULT NULL,

department\_description varchar(100) DEFAULT NULL,

PRIMARY KEY (department\_id)

);

Delimiter $$

Create Trigger after\_insert\_details

AFTER INSERT ON department FOR EACH ROW

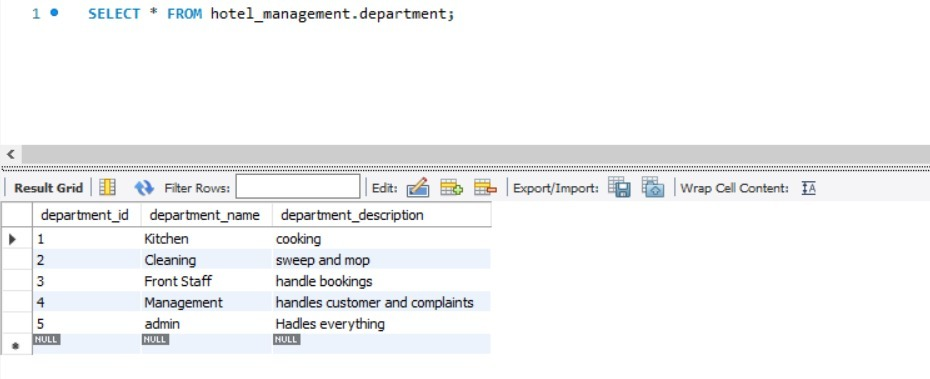
BEGIN

INSERT INTO department\_detail VALUES (new.department\_id, new.department\_name, new.department\_description);

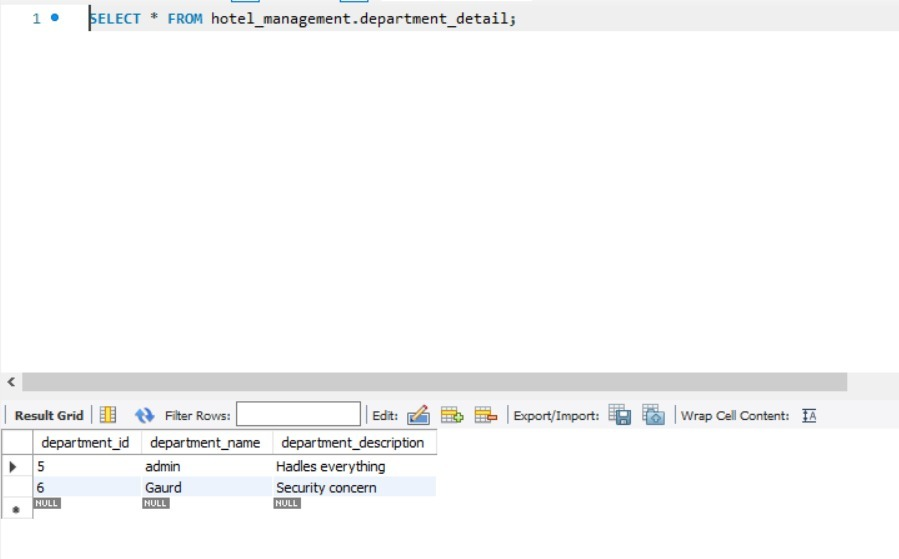
END $$

Delimiter ;

Before inserting the details of department



After inserting details into department



1. **Room\_history**

create table history\_table(room\_id INT, room\_number INT, rooms\_type\_rooms\_type\_id INT, hotel\_hotel\_id INT);

DELIMITER $$

CREATE TRIGGER `hotel\_management`.`room\_history`

BEFORE UPDATE ON `hotel\_management`.`rooms`

FOR EACH ROW

BEGIN

INSERT INTO history\_table

VALUES

(

OLD.room\_id,

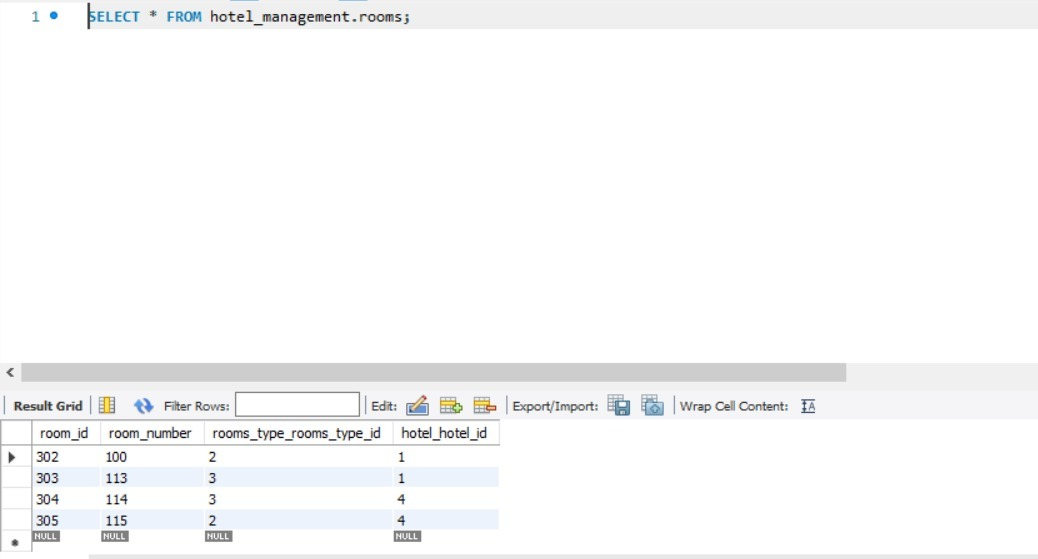
OLD.room\_number,

OLD.rooms\_type\_rooms\_type\_id,

OLD.hotel\_hotel\_id

);

END$$



updating the room table’s values

* **FUNCTIONS & PROCEDURES:**

1. **To count pet friendly rooms**

CREATE DEFINER=`root`@`localhost` PROCEDURE `countPetFriendlyRooms`(

isPetFriendly boolean )

BEGIN

IF isPetFriendly

then

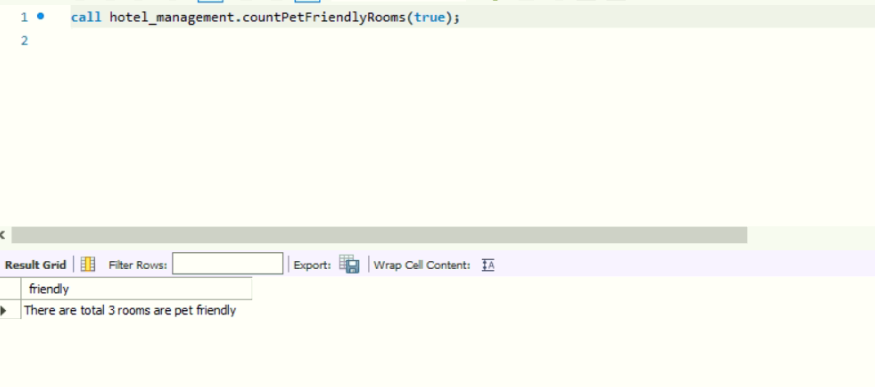
select concat('There are total ',COUNT(pet\_friendly),' rooms are pet friendly') as friendly FROM hotel\_management.room\_type where pet\_friendly = 1;

else

select concat('There are total ',COUNT(pet\_friendly),' rooms are pet friendly') as friendly FROM hotel\_management.room\_type where pet\_friendly = 0;

end if;

End



1. **To count smoke friendly rooms**

CREATE DEFINER=`root`@`localhost` PROCEDURE `countSomkeFriendlyRooms`(

isSmokeFriendly boolean )

BEGIN

IF isSmokeFriendly

then

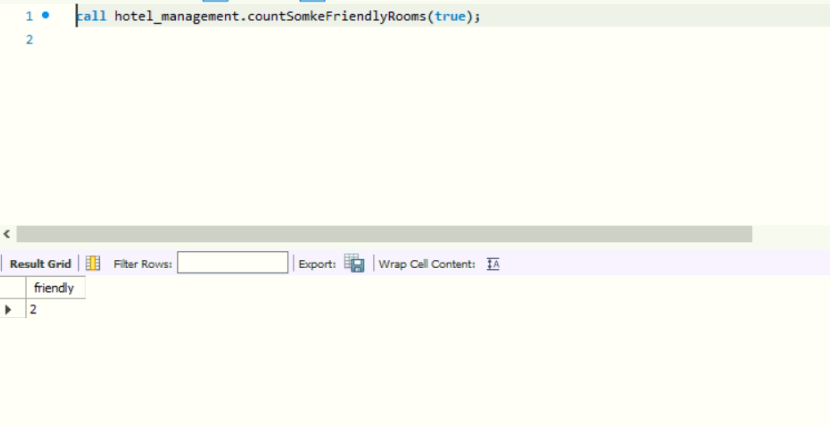
select COUNT(smoke\_friendly) as friendly FROM hotel\_management.room\_type where smoke\_friendly = 1;

else

select COUNT(smoke\_friendly) as friendly FROM hotel\_management.room\_type where smoke\_friendly = 0;

end if;

End



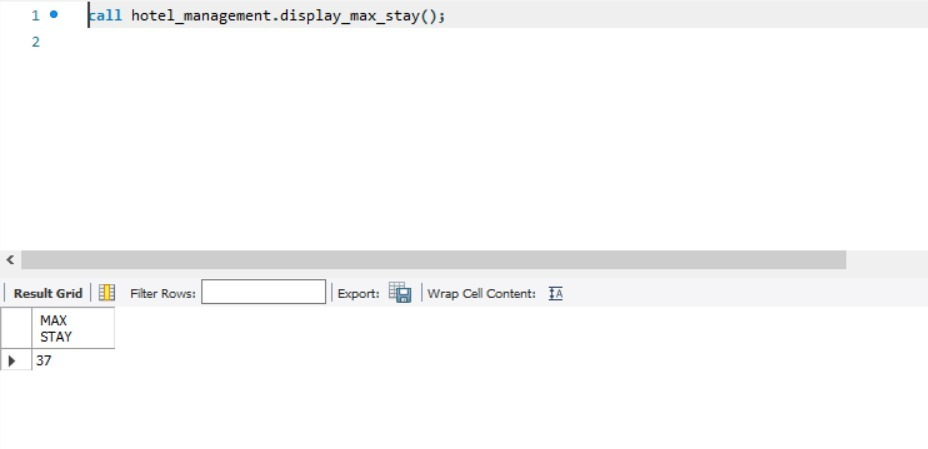
1. **The maximum days a person stays**

CREATE DEFINER=`root`@`localhost` PROCEDURE `display\_max\_stay`()

BEGIN

SELECT max(datediff(check\_out\_date,check\_in\_date)) as 'MAX STAY' FROM hotel\_management.bookings;

END



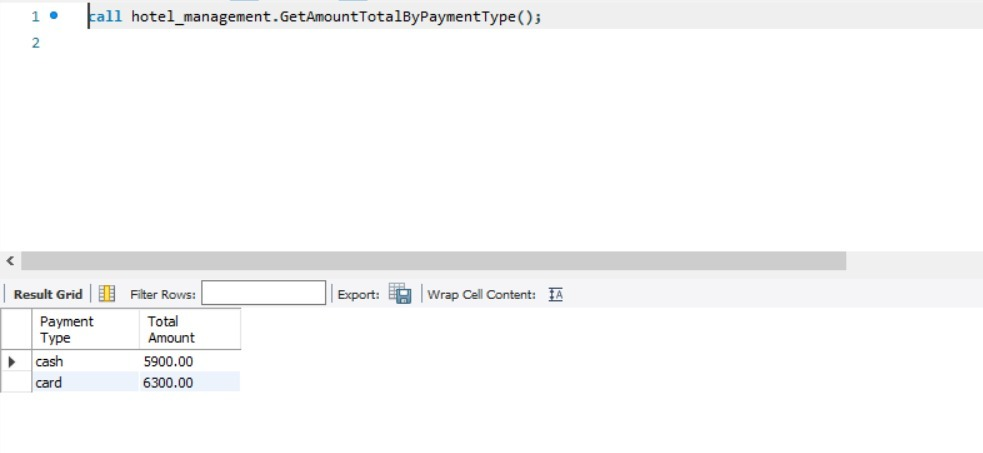
1. **Displays the amount earned by different modes of payment**

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetAmountTotalByPaymentType`()

BEGIN

select booking\_payment\_type as 'Payment Type',sum(total\_amount) as 'Total Amount' from hotel\_management.bookings group by booking\_payment\_type;

END



1. **Displays the information related to hotel by providing the name**

CREATE DEFINER=`root`@`localhost` PROCEDURE `GetHotelInfo`(

IN H\_name VARCHAR(255)

)

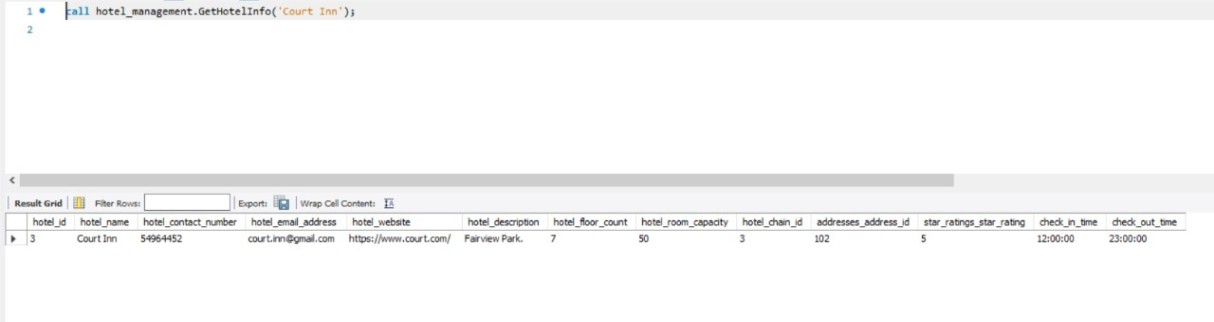
BEGIN

SELECT \*

FROM hotel

WHERE hotel\_name = H\_name;

END



1. **Provides the total amount earned between two dates**

CREATE DEFINER=`root`@`localhost` PROCEDURE `getTotalAmountBetweenTwoDates`(

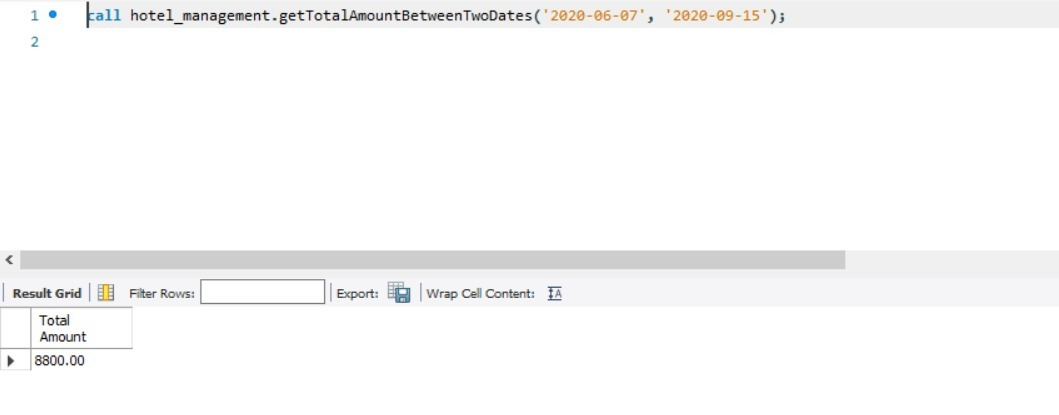
startingDate datetime,

endingDate datetime)

BEGIN

select sum(total\_amount) as 'Total Amount' from hotel\_management.bookings where booking\_date between startingDate and endingDate;

END



1. **Insert the details related to employee**

CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertEmployee`(

empid int,

fName varchar(50),

lName varchar(50),

designation varchar(20),

contactNo int,

eMail varchar(50),

deptId int,

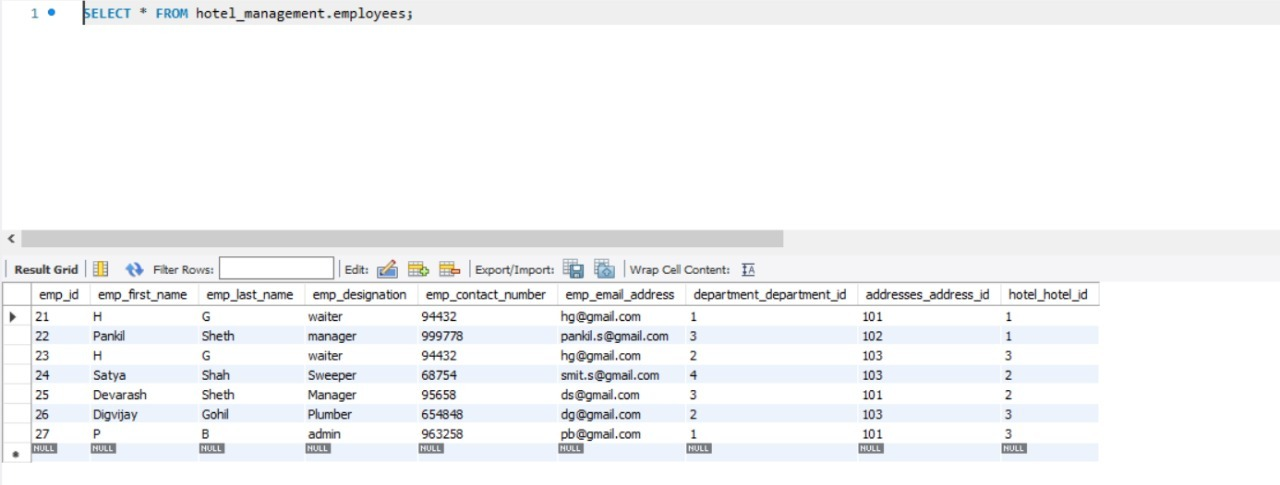
addressId int,

hotelId int)

BEGIN

Insert into hotel\_management.employees (emp\_id,emp\_first\_name, emp\_last\_name,emp\_designation,emp\_contact\_number,emp\_email\_address,department\_department\_id,addresses\_address\_id,hotel\_hotel\_id) values(empid,fName, lName,designation,contactNo,eMail,deptId,addressId,hotelId);

End

****

1. **This will update/insert the details related to employees**

CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertUpdateEmployee`(

empid int,

fName varchar(50),

lName varchar(50),

designation varchar(20),

contactNo int,

eMail varchar(50),

deptId int,

addressId int,

hotelId int)

BEGIN

if not exists (select emp\_id from hotel\_management.employees where emp\_id = empid)

Then

call hotel\_management.insertEmployee(empid,fname,lname,designation,contactNo,eMail,deptId,addressId,hotelId);

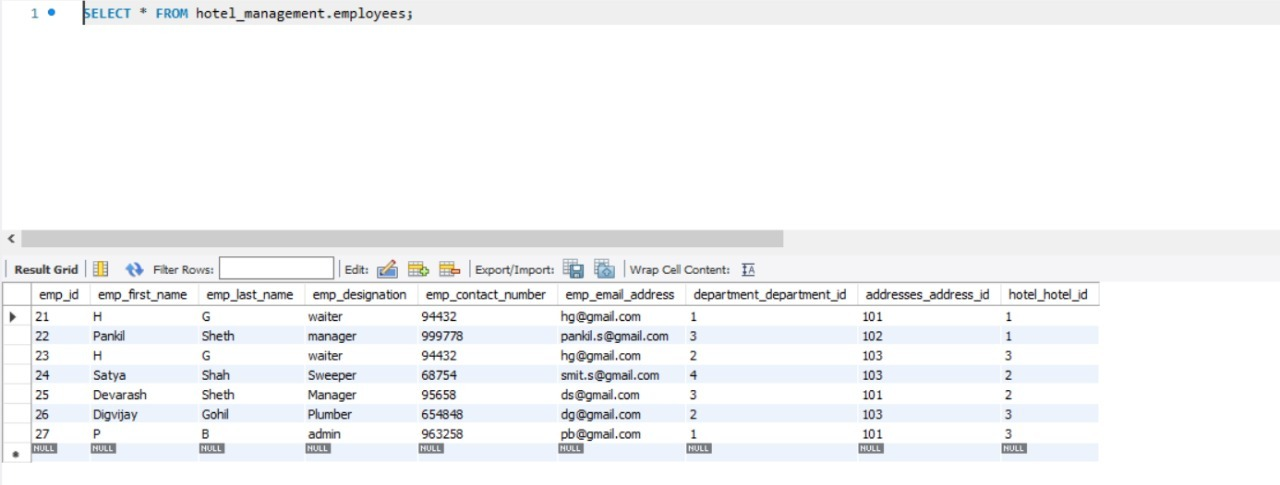
else

call hotel\_management.updateEmployee(empid,fname,lname,designation,contactNo,eMail,deptId,addressId,hotelId);

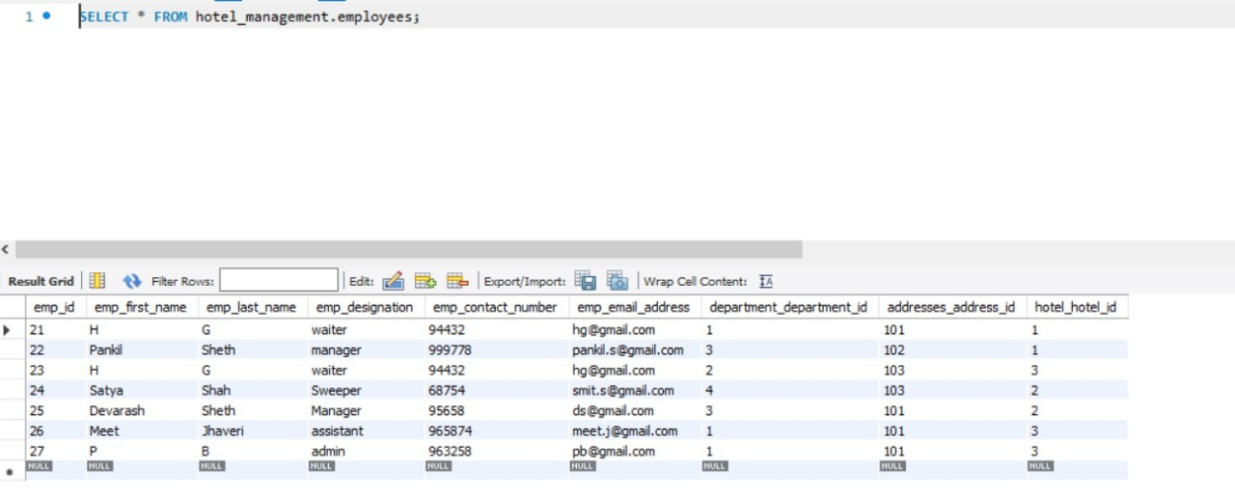
end if;

END

Before updating the id=26



After updating the id=26

****

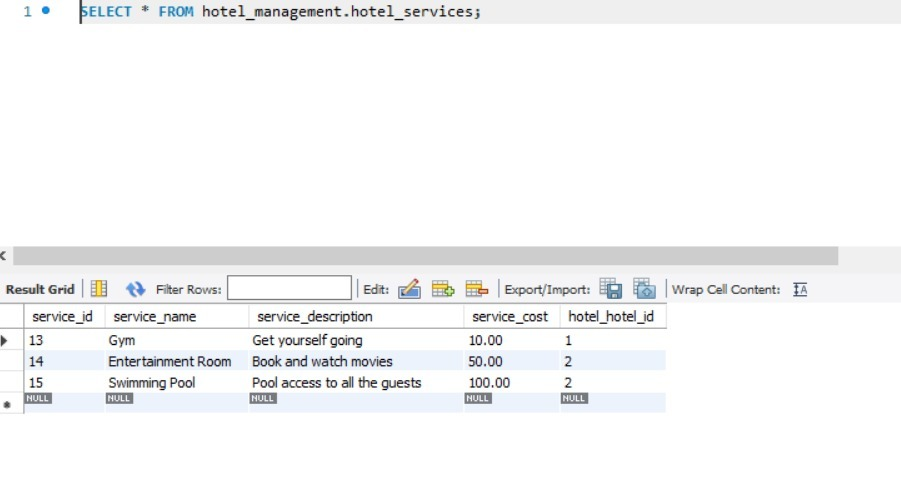
1. **Update hotel services**

CREATE DEFINER=`root`@`localhost` PROCEDURE `update\_hotel\_services`(IN temp\_service\_id INT, IN new\_service\_name varchar(45), IN new\_service\_description varchar(100))

begin

update hotel\_services set service\_name=new\_service\_name, service\_description=new\_service\_description where service\_id=temp\_service\_id;

end

****

Here the service\_id=13 having laundry gets updated to the Gym.

1. **Counts the total no of employees by providing department id**

CREATE DEFINER=`root`@`localhost` FUNCTION `emp\_dep`(department\_id INT) RETURNS int

DETERMINISTIC

begin

declare ans int default 0;

select count(emp\_id) from employees where department\_department\_id=department\_id into ans;

return ans;

end

