Mini Project Synopsis

HOTEL MANAGEMENT SYSTEM

Submitted as a part of course curriculum for

CORE COURSE IN

DATABASE MANAGEMENT SYSTEM

Under the guidance of
Prof Nivedita Kasturi

Submitted by -:
1) B Pravena – PES2UG19CS076
2) Bharath Kumar S P-PES2UG19CS087
3) Bhuvantej R – PES2UG19CS092

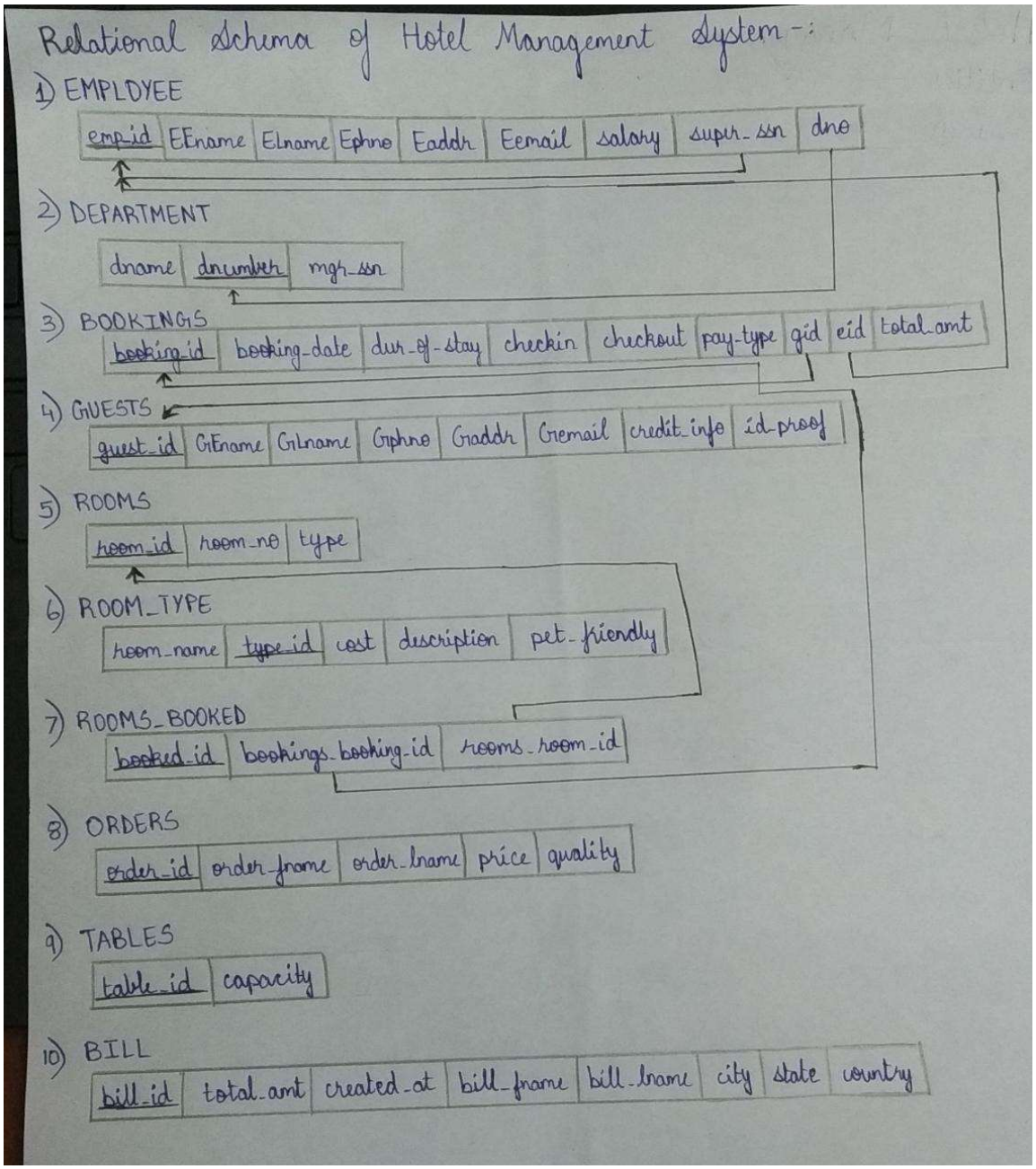
Department of Computer Science and Engineering

Pes University

Problem Statement -:

To create an efficient Hotel (lodge + restaurant) management system. The main objective is to manage the details of employees, bookings, guests at the lodge and customers at the restaurant, rooms, order, bill, etc. A customer can make reservations, change, or cancel reservations through the hotel website. When a customer makes reservations, based on availability employee allots room.

Relational Schema -:



Queries -:

1) Simple Queries

```
postgres=# \c hotel_management;
You are now connected to database "hotel_management" as user "postgres".
hotel_management=# SELECT* FROM EMPLOYEE;
 emp_id |  efname  | elname  |   ephno    |          eaddr          |          eemail          | salary | super_ssn |
no
--------+----------+---------+------------+-------------------------+--------------------------+--------+-----------+-
---
      1 | James    | Borg    | 888665555  | 450 Stone, Houston,TX   | james.borg@gmail.com     |  55000 |           |
 1
      2 | John     | Smith   | 998665545  | 731 Fondren,Houston,TX  | john.smith@gmail.com     |  35000 |         1 |
 3
      8 | Ahmed    | Jabber  | 7786652533 | 980 Dallas, Houston,TX  | ahmed.jabber@gmail.com   |  35000 |           |
 3
      3 | Franklin | Wong    | 778665533  | 638 voss,Houston,TX     | franklin.wong@gmail.com  |  30000 |         8 |
 3
      4 | Alicia   | Zelaya  | 988455255  | 3321 Castle,Spring,TX   | alica.zelaya@yaahoo.com  |  65000 |           |
 2
      5 | Jennifer | Wallace | 9867585821 | 291 Berry, Bellaire,TX  | jennifer@gmail.com       |  45000 |         2 |
 2
      6 | Ramesh   | Narayan | 8884455920 | 975 Fire Oak, Humble, TX | ramesh.narayan@gmail.com |  60000 |         1 |
 1
      7 | Jonny    | English | 998665533  | 5631 Rice,Houston,TX    | johnny.english@gmail.com |  60000 |         1 |
 1
(8 rows)
```

```
hotel_management=# SELECT* FROM bill;
 bill_id | total_amt | created_at | bill_fname | bill_lname |   city    |   state   | country
---------+-----------+------------+------------+------------+-----------+-----------+---------
       1 |      2400 | 2021-08-20 | Sachin     | Tendulkar  | Bangalore | Karnataka | India
       2 |      9900 | 2021-07-14 | Virat      | Kohli      | Bangalore | Karnataka | India
       3 |     12700 | 2021-06-14 | Rohit      | Sharma     | Bangalore | Karnataka | India
       4 |     17500 | 2021-05-25 | Gautham    | Gambhir    | Bangalore | Karnataka | India
       5 |     22500 | 2021-04-29 | MS         | Dhoni      | Bangalore | Karnataka | India
       6 |      9000 | 2021-03-24 | Smrithi    | Mandhana   | Bangalore | Karnataka | India
       7 |     15000 | 2021-02-10 | Mithali    | Raj        | Bangalore | Karnataka | India
       8 |     10000 | 2021-01-17 | Salman     | Khan       | Bangalore | Karnataka | India
(8 rows)
```

```
hotel_management=# DELETE FROM bill WHERE total_amt=2400;
DELETE 1
hotel_management=# SELECT* FROM bill;
 bill_id | total_amt | created_at | bill_fname | bill_lname |   city    |   state   | country
---------+-----------+------------+------------+------------+-----------+-----------+---------
       2 |      9900 | 2021-07-14 | Virat      | Kohli      | Bangalore | Karnataka | India
       3 |     12700 | 2021-06-14 | Rohit      | Sharma     | Bangalore | Karnataka | India
       4 |     17500 | 2021-05-25 | Gautham    | Gambhir    | Bangalore | Karnataka | India
       5 |     22500 | 2021-04-29 | MS         | Dhoni      | Bangalore | Karnataka | India
       6 |      9000 | 2021-03-24 | Smrithi    | Mandhana   | Bangalore | Karnataka | India
       7 |     15000 | 2021-02-10 | Mithali    | Raj        | Bangalore | Karnataka | India
       8 |     10000 | 2021-01-17 | Salman     | Khan       | Bangalore | Karnataka | India
(7 rows)
```

```
hotel_management=# DROP ROLE customer;
ERROR:  role "customer" cannot be dropped because some objects depend on it
DETAIL:  privileges for table bookings
privileges for table guests
privileges for table rooms
privileges for table room_type
privileges for table rooms_booked
privileges for table orders
privileges for table tables
privileges for table bill
hotel_management=#
```

```
hotel_management=# SELECT * FROM information_schema. table_privileges LIMIT 5;
 grantor  | grantee  |  table_catalog   | table_schema | table_name | privilege_type | is_grantable | with_hierarchy
----------+----------+------------------+--------------+------------+----------------+--------------+----------------
 postgres | postgres | hotel_management | public       | bill       | INSERT         | YES          | NO
 postgres | postgres | hotel_management | public       | bill       | SELECT         | YES          | YES
 postgres | postgres | hotel_management | public       | bill       | UPDATE         | YES          | NO
 postgres | postgres | hotel_management | public       | bill       | DELETE         | YES          | NO
 postgres | postgres | hotel_management | public       | bill       | TRUNCATE       | YES          | NO
(5 rows)
```

Simple Queries are those which are used to retrieve/perform very simple operations on the database. Above, using queries we were able to view the employee, bill table and then delete a record, remove a role ‘customer’ and then check all privileges used.

2) Complex Queries

```
postgres=# \c hotel_management;
You are now connected to database "hotel_management" as user "postgres".
hotel_management=# SELECT* FROM employee WHERE salary > 55000;
 emp_id | efname | elname  |   ephno    |          eaddr           |          eemail          | salary | super_ssn |
--------+--------+---------+------------+--------------------------+--------------------------+--------+-----------+-
-
      4 | Alicia | Zelaya  | 988455255  | 3321 Castle,Spring,TX    | alica.zelaya@yaahoo.com  |  65000 |           |
      6 | Ramesh | Narayan | 8884455920 | 975 Fire Oak, Humble, TX | ramesh.narayan@gmail.com |  60000 |         1 |
      7 | Jonny  | English | 998665533  | 5631 Rice,Houston,TX     | johnny.english@gmail.com |  60000 |         1 |
(3 rows)


hotel_management=# SELECT* FROM employee WHERE salary > 55000 and dno=1;
 emp_id | efname | elname  |   ephno    |          eaddr           |          eemail          | salary | super_ssn |
--------+--------+---------+------------+--------------------------+--------------------------+--------+-----------+-
-
      6 | Ramesh | Narayan | 8884455920 | 975 Fire Oak, Humble, TX | ramesh.narayan@gmail.com |  60000 |         1 |
      7 | Jonny  | English | 998665533  | 5631 Rice,Houston,TX     | johnny.english@gmail.com |  60000 |         1 |
(2 rows)
```

```
hotel_management=# SELECT room_name, cost FROM room_type where cost>2000;
    room_name    |  cost
-----------------+---------
 Double          |  2000.5
 Triple          | 3500.25
 Queen           |    5000
 Double+Balcony  |    2300
 Double+lakeview |    2800
(5 rows)
```

```
hotel_management=# SELECT order_fname, order_lname FROM orders WHERE capacity>=2;
 order_fname | order_lname
-------------+-------------
 Tomato      | soup
 Roti        | Curry
 Vegetable   | salad
 Tandoori    | Chicken
 Fish        | Fry
(5 rows)
```

```
hotel_management=# SELECT checkin, checkout FROM bookings where dur_of_stay<10;
  checkin   |  checkout
------------+------------
 2021-01-14 | 2021-01-17
 2021-02-02 | 2021-02-10
 2021-03-22 | 2021-03-24
 2021-05-20 | 2021-05-25
 2021-06-11 | 2021-06-14
 2021-07-05 | 2021-07-14
 2021-08-19 | 2021-08-20
(7 rows)
```

```
hotel_management=# SELECT checkin, total_amt FROM bookings where checkin>'2021-01-14';
  checkin   | total_amt
------------+-----------
 2021-02-02 |     15000
 2021-03-22 |      9000
 2021-04-18 |     22500
 2021-05-20 |     17500
 2021-06-11 |     12700
 2021-07-05 |      9900
 2021-08-19 |      2400
(7 rows)
```

In the above complex queries we have used query along with some keyword to obtain more information from the database than what simple queries provide.

3) Nested Queries

It refers to those queries in which there is a query or an action being performed inside another query.

```
hotel_management=# SELECT gfname,glname FROM guests WHERE credit_info='visa'
hotel_management-# ORDER BY gfname;
 gfname | glname
--------+--------
 Rohit  | Sharma
 Salman | Khan
 Virat  | Kholi
(3 rows)
```

```
hotel_management=# SELECT type_id,room_name FROM room_type WHERE pet_friendly=0
hotel_management-# ORDER BY type_id;
 type_id |   room_name
---------+---------------
      21 | Single
      22 | Double
      25 | Single+Balcony
      27 | Double+Balcony
(4 rows)
```

```
postgres=# \c hotel_management;
You are now connected to database "hotel_management" as user "postgres".
hotel_management=# SELECT COUNT (DISTINCT gid) FROM bookings
hotel_management-# WHERE (checkin>='2021-05-20') OR
hotel_management-# (checkout<='2021-10-15');
 count
-------
     8
(1 row)
```

```
hotel_management=# SELECT rooms.room_no, COUNT(rooms.type) AS COUNT from rooms
hotel_management-# GROUP BY rooms.room_no;
 room_no | count
---------+-------
     304 |     1
     101 |     1
     103 |     1
     302 |     1
     104 |     1
     305 |     1
     303 |     1
     201 |     2
     102 |     1
     301 |     1
     203 |     1
(11 rows)
```

```
hotel_management=# SELECT bill_id,bill_fname,SUM(total_amt)
hotel_management-# FROM bill
hotel_management-# GROUP BY bill_id;
 bill_id | bill_fname |  sum
---------+------------+-------
       4 | Gautham    | 17500
       7 | Mithali    | 15000
       6 | Smrithi    |  9000
       3 | Rohit      | 12700
       5 | MS         | 22500
       2 | Virat      |  9900
       8 | Salman     | 10000
(7 rows)
```

Users with different access privilege levels -:

The users we have created are admin with all privileges, Employee with SELECT on tables employee and department, Manager with INSERT, UPDATE privilege on department, bookings, bill. Finally we gave INSERT and UPDATE privilege on the tables guests, rooms, room_type, orders and tables and SELECT privilege on bill, rooms_booked, bookings to CUSTOMER.

```
hotel_management=# CREATE user admin with encrypted password 'admin8055';
CREATE ROLE
hotel_management=# GRANT all privileges on database hotel_management to admin;
GRANT
hotel_management=#
```

```
hotel_management=# CREATE user Employee with encrypted password '123456';
CREATE ROLE
hotel_management=# \d hotel_management
Did not find any relation named "hotel_management".
hotel_management=# GRANT SELECT on employee,department;
ERROR:  syntax error at or near ";"
LINE 1: GRANT SELECT on employee,department;
                                            ^
hotel_management=# GRANT SELECT on employee,department to Employee;
GRANT
```

```
hotel_management=# CREATE user Manager with encrypted password '909090';
CREATE ROLE
hotel_management=# GRANT INSERT,UPDATE on department,bookings,bill to Manager;
GRANT
```

```
hotel_management=# CREATE user Customer with encrypted password '10007';
CREATE ROLE
hotel_management=# GRANT INSERT,UPDATE on guests,rooms,rooms_type,orders,tables to Customer;
ERROR:  relation "rooms_type" does not exist
hotel_management=# GRANT INSERT,UPDATE on guests,rooms,room_type,orders,tables to Customer;
GRANT
hotel_management=#
hotel_management=# GRANT SELECT on bill,rooms_booked,bookings to Customer;
GRANT
hotel_management=#
```

Triggers with functions -:

triggers.sql

```sql
create trigger changes_lastname
    before update on employee
    for each row
    execute procedure log_changes_lastname();



create trigger inform_changes
    after update on employee
    for each row
    execute procedure notify_changes();



create trigger afterDelete_guest
    after delete on guests
    for each row
    execute procedure log_guest_delete();


create trigger newBooking_trigger
    after insert on bookings
    for each row
    execute procedure log_bookings();
```

Functions.sql

```sql
-- Trigger functions

create or replace function log_changes_lastname()
    returns trigger
    language plpgsql
    as
    $$
        begin
            if NEW.ELname <> OLD.ELname then
                insert into employee_audits values(OLD.emp_id, OLD.ELname, now() );
            end if;

            return NEW;
        end;
    $$;


create or replace function log_guest_delete()
    returns trigger
    language plpgsql
    as
    $$
        begin
            insert into guest_audit values (new.guest_id, new.GFname, new.GLname, new.Gphno, new.Gaddr, new.Gemail, now());

            return NEW;
        end;
    $$;


create or replace function log_bookings()
    returns trigger
    language plpgsql
    as
    $$
        begin
            insert into bookings_audit values (new.booking_id, new.gid, current_timestamp);

            return NEW;
        end;
```

```sql
-- trigger procedures

create or replace procedure notify_changes()
    language plpgsql
    as
    $$
    declare
        cur cursor for select * from employee_audits where emp_id=NEW.emp_id;
        rec record;
    begin

        open cur;
        fetch 1 into rec;

        raise notice 'Updated name to %',rec.ELname,' on %',rec.changed_on;
        close cur;


    end;
    $$
```

Function with usage of cursors -:

Functions.sql

```
create or replace function book_room(rno int, book_id int, book_date date, stay_dur int, checkin date, checkout date, payType varchar(255), guestID int, em
    returns int
    language plpgsql
    as
    $$
    declare
        cur cursor for select * from rooms where vacant=1 and room_no=rno;
        rec record;

        status integer;
        booking_cost float;
```

```
    begin
        open cur;
        fetch cur into rec;

        status:=0
        booking_cost := 0.00;

        if(rec == NULL)
            begin
                raise notice 'Room-% already occupied',rno;
            end
        else
            begin
                update rec if exists set vacant=0;
                -- update rooms_booked SET booked_id=book_id, rooms_room_id=rec.room_id, bookings_booking_id=rno;

                select cost into booking_cost from room_type as T, rooms as R where R.r_type=T.room_name;

                insert into rooms_booked values(rno, book_id, rec.room_id);
                CALL update_bookings(book_id, book_date, stay_dur, checkin, checkout, payType, guestID, empID, booking_cost);

                status:=1;
            end

        raise notice 'Booked Status: %',status;
        raise notice 'Booking Charge: %',booking_cost;


        close cur;
        return status;
```

Functions.sql

```
48  create or replace function vacant_room(rno int)
49      returns int
50      language plpgsql
51      as
52      $$
53      declare
54          cur cursor for select * from rooms where room_no=rno;
55          rec record;
56
57          status integer;
58          room_booking_id integer;
59      begin
60          open cur;
61          fetch 1 into rec; --first record
62
63          if(rec == NULL)
64              begin
65                  raise notice 'Room-% does not exists!',rno;
66              end
67          else
68              begin
69                  if(rec.vacant == 1)
70                      begin
71                          raise notice 'Room-% is already vacated.',rno;
72                      end
73                  else
74                      begin
75                          update rec if exists set vacant=1;
76
77                          select bookings_booking_id into room_booking_id from rooms_booked as B where B.book_id=rno;
78
79                          delete from rooms_booked as R where R.rooms_room_id=rec.room_id;
80                          delete from bookings as B where B.booking_id=bookings_booking_id;
81                          status:=1;
82                      end
83              end;
84
85          raise notice 'Successfully vacated Room-% ',rno;
86
87          close cur;
88          return status;
```

Procedures -:

procedure.sql

```sql
create or replace procedure show_available_rooms()
    language plpgsql
    as
    $$
    declare
        -- cur cursor for select * from rooms as R, rooms_booked as B where R.room_id - B.rooms_room_id;
        -- rec record;
    begin

        -- select * from rooms as R, rooms_booked as B where R.room_id - B.rooms_room_id;
        -- select * from rooms EXCEPT select * from rooms_booked;
        select DISTINCT rooms.* from (rooms as R LEFT OUTER JOIN rooms_booked as B on R.room_id=B.rooms_room_id) where B.rooms_room_id is NULL;

        -- select room_no, r_type, cost from ROOMS as R, ROOM_TYPE as T  where rooms.empty='true';

    end;
    $$
```

```sql
-- write new booking records
create or replace procedure update_bookings(book_id int, book_date date, stay_dur int, checkin date, checkout date, payType varchar(255), guestID int, empI
    language plpgsql
    as
    $$
    declare

    begin
        INSERT into BOOKINGS values(book_id, book_date, stay_dur, checkin, checkout, payType, guestID, empID, amt);
        raise notice 'Updated bookings data';
    end;
    $$
```

```sql
-- to insert new departments
create or replace procedure add_new_department(dep_name varchar(25), dep_no int, mSSN int)
    language plpgsql
    as
    $$
    declare
        check_dep_no integer;
    begin
        select count(*) into check_dep_no from department where dnumber=dep_no;
        -- if no duplicate department exists
        if(check_dep_no == 0)
            begin
                if((dep_name is not NULL) and (dep_no is not NULL))
                    begin
                        insert into department values(dep_name, dep_no, mSSN);
                    end
                else
                    begin
                        raise notice "Department values cannot be NULL";
                    end
            end
        else
            begin
                raise notice 'Department already exists!';
            end
    end;
```

```sql
-- trigger procedures

create or replace procedure notify_changes()
    language plpgsql
    as
    $$
    declare
        cur cursor for select * from employee_audits where emp_id=NEW.emp_id;
        rec record;
    begin

        open cur;
        fetch 1 into rec;

        raise notice 'Updated name to %',rec.ELname,' on %',rec.changed_on;
        close cur;


    end;
    $$
```

```
hotel_management=#
hotel_management=# \df
                                                                                                                              List of func
tions
 Schema |         Name          | Result data type |
         Argument data types                                                                                  | Type
--------+-----------------------+------------------+-------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------+------
 public | add_new_department    |                  | IN dep_name character varying, IN dep_no integer, IN mssn integer
                                                                                                              | proc
 public | book_room             | integer          | rno integer, book_id integer, book_date date, stay_dur integer, checkin da
te, checkout date, paytype character varying, guestid integer, empid integer                                  | func
 public | log_bookings          | trigger          |
                                                                                                              | func
 public | log_changes_lastname  | trigger          |
                                                                                                              | func
 public | log_guest_delete      | trigger          |
                                                                                                              | func
 public | notify_changes        | trigger          |
                                                                                                              | func
 public | show_available_rooms  |                  |
                                                                                                              | proc
 public | update_bookings       |                  | IN book_id integer, IN book_date date, IN stay_dur integer, IN checkin dat
e, IN checkout date, IN paytype character varying, IN guestid integer, IN empid integer, IN amt double precision | proc
 public | vacant_room           | integer          | rno integer
                                                                                                              | func
(9 rows)
```

```
CREATE PROCEDURE
CREATE PROCEDURE
CREATE PROCEDURE
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE TRIGGER
CREATE TRIGGER
CREATE TRIGGER
CREATE TRIGGER

D:\Postgres_App\psql_10\bin>
```

Newly added Tables -:

We realized we required some more tables and values while doing the above functions and procedures and thus included these.

```
create table rooms(
 room_id int PRIMARY KEY,
 room_no int,
 type varchar(255),
 vacant int DEFAULT 1
);
```

```
create table bookings_audit(
    booking_id int PRIMARY KEY,
    guest_Id int,
    booked_date varchar(100) NOT NULL
)
```

```
create table guest_audit(
    guest_id int PRIMARY KEY,
    GFname varchar(255),
    GLname varchar(255),
    Gphno varchar(15),
    Gaddr varchar(255) NOT NULL,
    Gemail varchar(255)
)
```

```
INSERT into ROOMS values(1, 101, 'Single');
INSERT into ROOMS values(2, 102, 'Double');
INSERT into ROOMS values(3, 103, 'Double+Balcony');
INSERT into ROOMS values(4, 104, 'Single+Balcony');
INSERT into ROOMS values(5, 201, 'Single+lakeview');
INSERT into ROOMS values(6, 202, 'Double');
INSERT into ROOMS values(7, 203, 'Triple');
INSERT into ROOMS values(8, 301, 'Double+Balcony');
INSERT into ROOMS values(9, 302, 'Queen');
INSERT into ROOMS values(10, 303, 'Triple');
INSERT into ROOMS values(11, 304, 'Single+lakeview');
INSERT into ROOMS values(12, 305, 'Double+lakeview');
```

Contributions -:

1) B. Pravena – PES2UG19CS076 – simple, complex and nested queries, compilation of report (3.5hrs)
2) Bharath Kumar S P - PES2UG19CS087 – trigger with proper functions, function with usage of cursors, saved procedures (4.5hrs)
3) Bhuvantej R – PES2UG19CS092 –users and their different privileges (2.5hrs)