Lab Project Report submitted

for Database Management system (UCS310)

By

**Jatin Malik 102103755**

**Ayesha Gupta 102103760**

**Himanshi 102103774**

COE27

Submitted to:

**Ms. Archana Singh**



**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,**

**(A DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB**

**INDIA**

**Jan-May 2023**

**Project: Hotel Management System**

**Intro****duction:**

A hotel management system is an integrated software platform designed to help hotel managers and staff manage their operations effectively. It helps automate a range of processes, from managing room reservations to billing and accounting. The system can also assist in managing staff, resources, and facilities, while providing critical business intelligence to help hotel managers make informed decisions.

A hotel management system typically consists of a database that stores information about guests, rooms, staff, inventory, and other essential aspects of hotel management. The system allows for data entry and retrieval, as well as reporting and analysis. Hotel managers can use the data collected to improve customer service, optimize room utilization, and increase overall revenue.

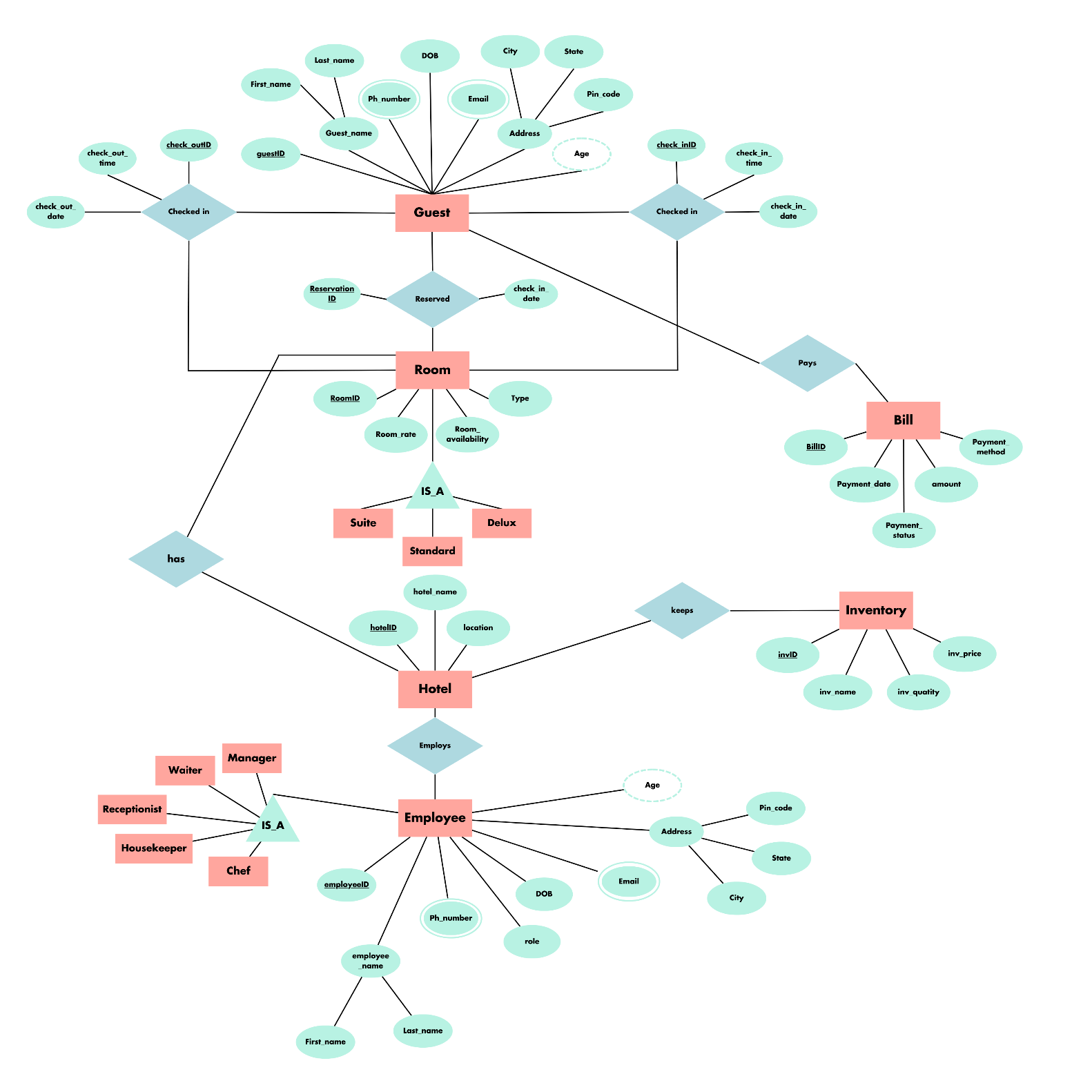
One of the key features of a hotel management system is its ability to manage room reservations. The system can be used to track room availability, room types, and room rates, and manage the entire reservation process, from booking to check-out. The system can also handle guest requests and preferences, and store guest information securely for future reference.

A hotel management system can also be used to manage inventory and other resources, such as food and beverage supplies, linens, and cleaning equipment. The system can help optimize inventory levels, reduce waste, and ensure that the hotel has the necessary resources to meet guest demand.

In terms of benefits, a hotel management system can help improve operational efficiency, reduce costs, and enhance guest satisfaction. By automating routine tasks and providing real-time data, the system can help hotel managers make better decisions, improve staff productivity, and increase revenue.

In terms of requirements, a hotel management system must be reliable, user-friendly, and scalable. It should be able to handle a large volume of data, support multiple users, and integrate with other hotel management systems, such as payment gateways and inventory management systems. The system should also provide data security, backup and recovery, and 24/7 support to ensure that it can be relied upon at all times. Finally, the system should be customizable to meet the specific needs of different hotels, and provide a range of reporting and analysis tools to help hotel managers make informed decisions.

**ER diagram:**

****

**Entities:**

1. Guest: (Attributes- id, name, phone number, email, address, date of birth, age)

A guest is a person that stays in the hotel with a reservation and stores information about the hotel's guests, including their guest ID, address, phone number, and email address.

Guest -> checks\_in / checks\_out -> Room: many to one relation

Guest -> pays -> Bill: one to one relation

Guest ->reserves -> Room: one to many

1. Room: (Attributes- id, availability, rate, type)

A room in the hotel can be a suite, deluxe or standard as per the guest choice. This entity stores all the rooms in the hotel along with their status- empty, occupied or reserved and their rates. A guest has to make a reservation in the hotel with the choice of their room to change its status to reserved. Once the guest checks in the status changes to occupied and back to empty after the guest checks out.

Room -> reserved by -> Guest: many to one

Room -> is in -> Hotel: many to one

1. Hotel: (Attributes- name, address, phone number)

The hotel is the entity that contains details about our hotel like name and address.

Hotel -> has -> Rooms: one to many

Hotel -> employs -> Employee: one to many

Hotel -> has -> inventory: one to one

1. Employee: (Attributes- id, name, address, phone number, email, salary, role, date of birth, age)

An employee works in the hotel and has a role. Employee can be a manager, waiter, receptionist and housekeeper. There can be multiple employees of a single role and have a fixed salary.

Employee -> works in -> Hotel: many to one

1. Bill: (Attributes- id, payment status, number of days, total amount payable)

A bill is generated once a guest checks in and the total bill amount is generated on check out which has to be paid in full before leaving the hotel. Bill can be paid in instalments before check out too.

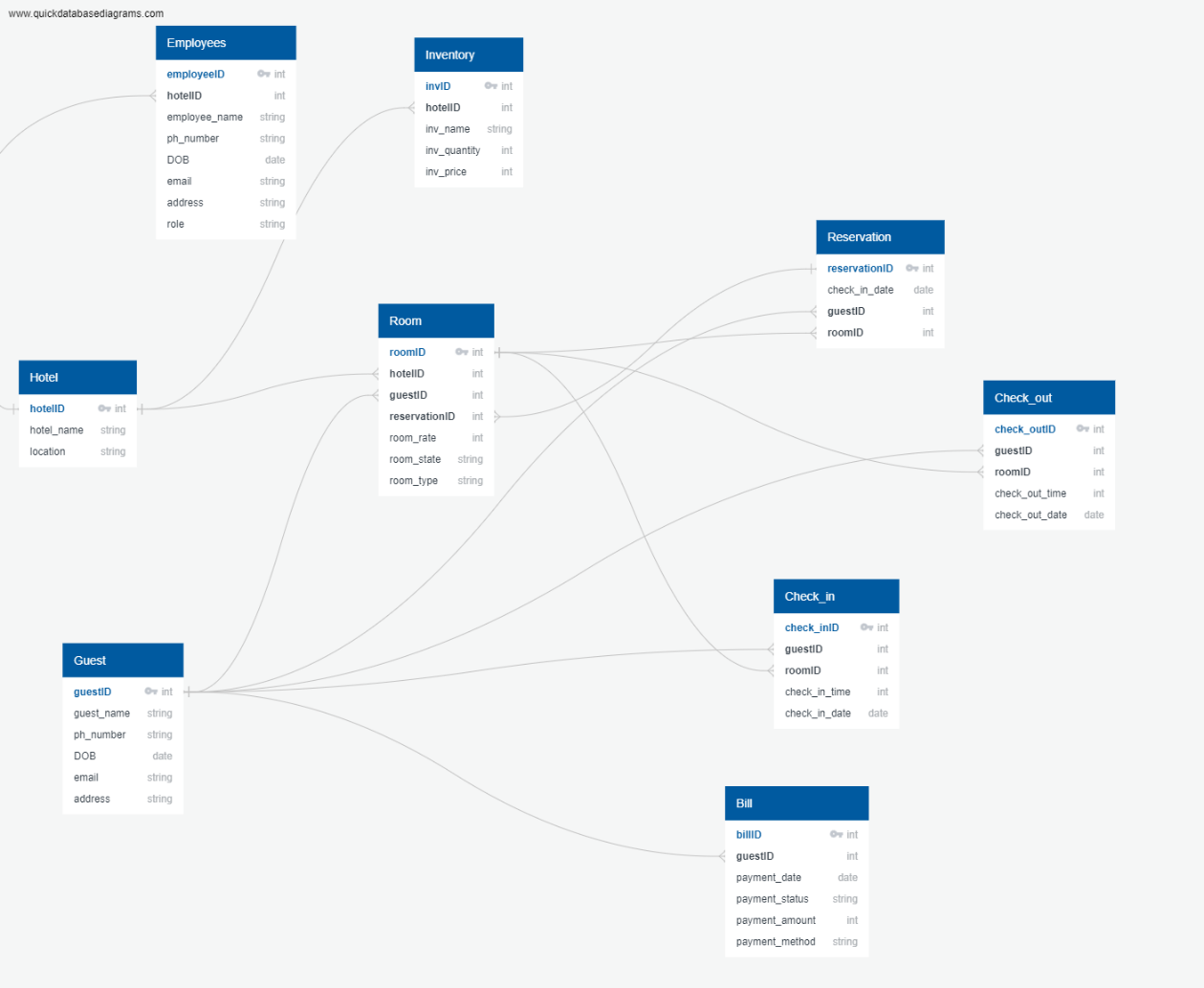
Bill -> paid by -> Guest: one to one

1. Inventory: (Attributes- id, name, quantity, rate)

The hotel inventory contains items like towels, shampoos, soaps, sheets, etc to provide to the guest. The room is equipped with some of these but the guests can request more.

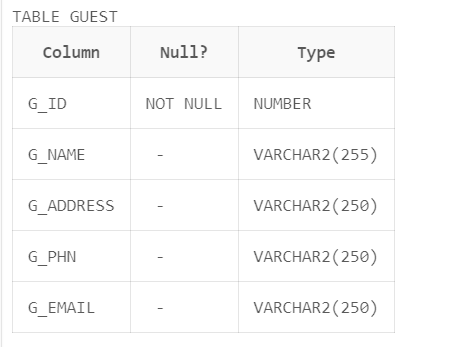
Inventory -> is in -> Hotel: one to one

**ER diagram to Tables:**

****

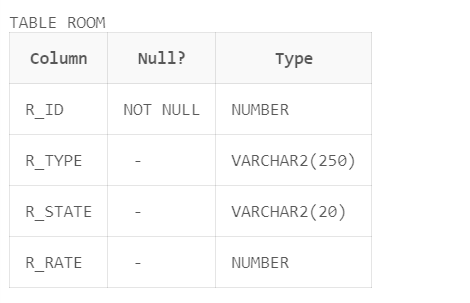
1. Entity guest to table

create table guest(g\_id int Primary Key,g\_name varchar(255), g\_address varchar(250), g\_phn varchar(250), g\_email varchar(250));



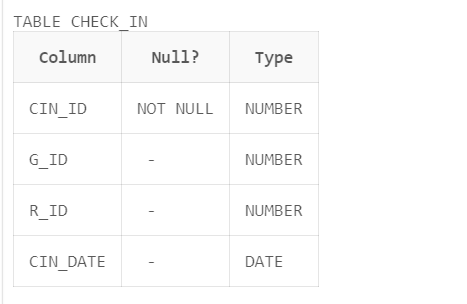
1. Entity room to table

create table room(r\_id int Primary Key, r\_type varchar(250), r\_state varchar(20) check(r\_state in ('reserved','occupied','empty')), r\_rate int);



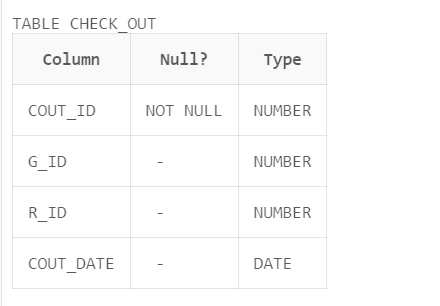
1. Relation checks in between guest and room to table

create table check\_in(cin\_id int Primary Key, g\_id int, r\_id int,cin\_date date, Foreign Key(r\_id) References room(r\_id), Foreign Key(g\_id) References guest(g\_id));



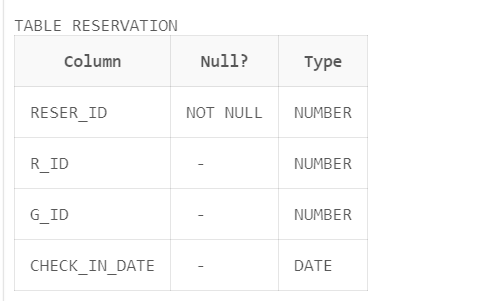
1. Relation checks out between guest and room to table

create table check\_out(cout\_id int Primary Key, g\_id int, r\_id int,cout\_date date, Foreign Key(r\_id) References room(r\_id), Foreign Key(g\_id) References guest(g\_id));



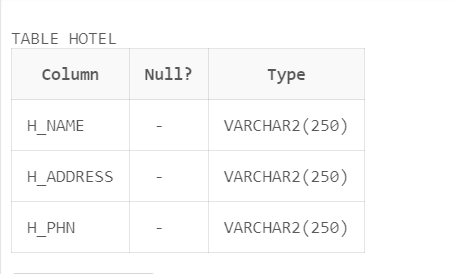
1. Relation reserve between guest and room to table

create table reservation(reser\_id int Primary Key, r\_id int, g\_id int, check\_in\_date date, Foreign Key(r\_id) References room(r\_id), Foreign Key(g\_id) References guest(g\_id));



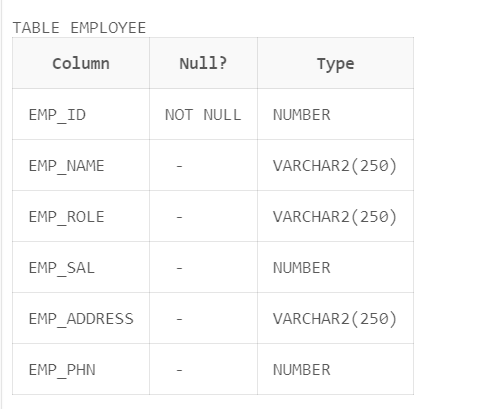
1. Entity hotel to table

create table hotel(h\_name varchar(250), h\_address varchar(250), h\_phn varchar(250));



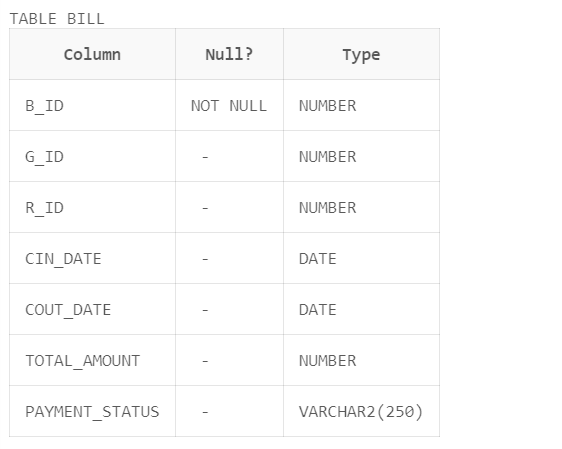
1. Entity employee to table

create table employee(emp\_id int Primary Key, emp\_name varchar(250), emp\_role varchar(250), emp\_sal int, emp\_address varchar(250), emp\_phn int);



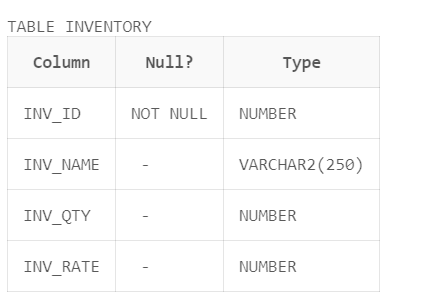
1. Entity bill to table

create table bill(b\_id int Primary Key, g\_id int, r\_id int, cin\_date date, cout\_date date, total\_amount int, payment\_status varchar(250), Foreign Key(r\_id) References room(r\_id), Foreign Key(g\_id) References guest(g\_id));



1. Entity inventory to table

create table inventory(inv\_id int Primary Key, inv\_name varchar(250), inv\_qty int, inv\_rate int);



**Normalization:**

* **First Normal Form (1NF)**

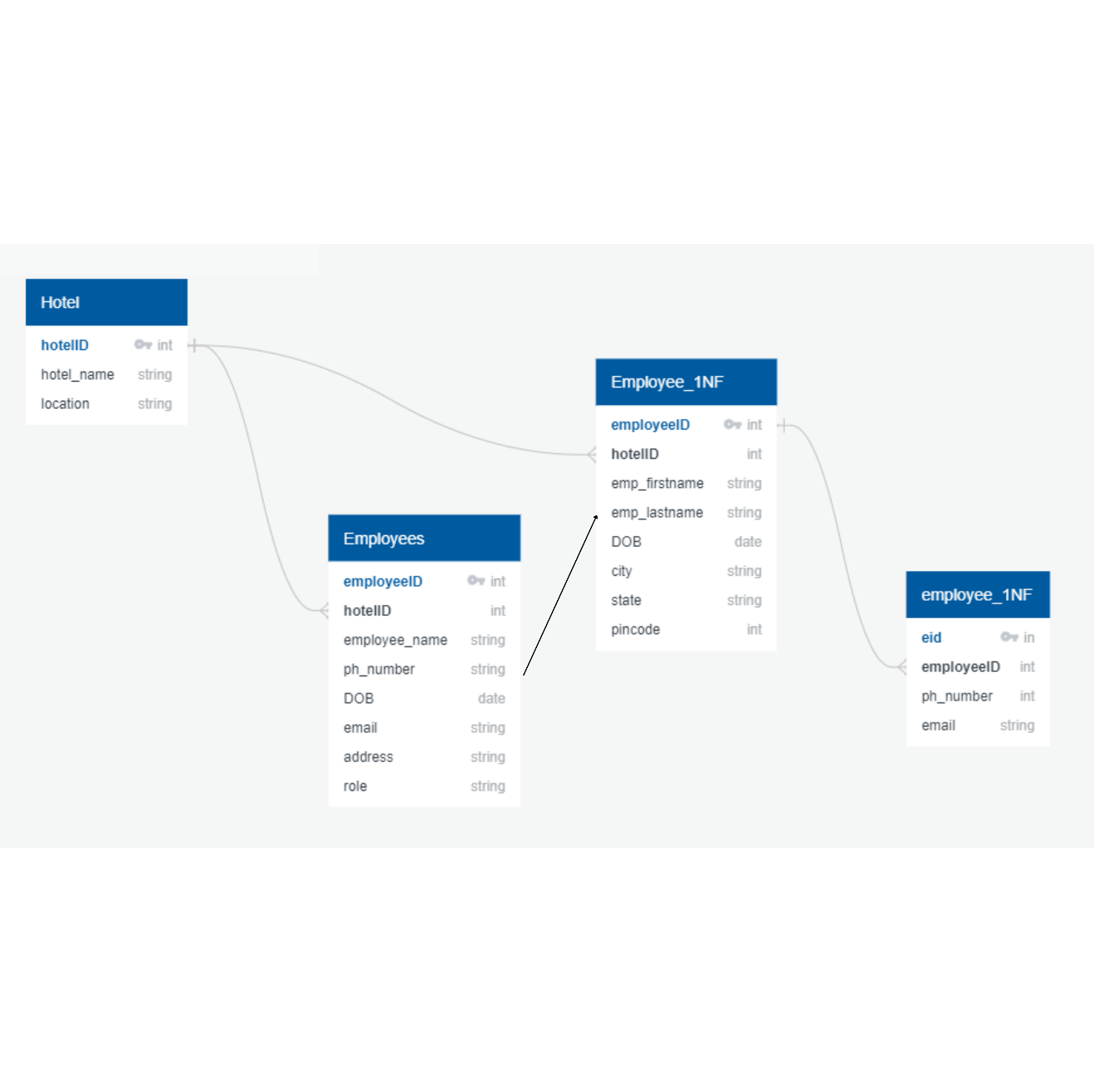
A relation R is in first normal form (1NF) if and only if it does not contain any composite attribute or multi-valued attributes or their combinations or all underlying domains contain atomic values only.

To normalize a table to 1NF we:

* + Divide composite attributes into number of sub-attributes and insert value in proper sub-attribute.
  + The first table contains all attributes except multi-valued attribute with same primary key and
  + Second table contains multi-valued attribute and place a primary key in it.
  + Insert the primary key of first table in the second table as a foreign key.



*1NF for Guest Table*

**

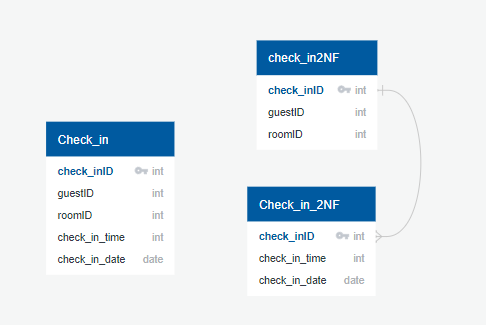
*1NF for Employee Table*

* **Second Normal Form (2NF)**

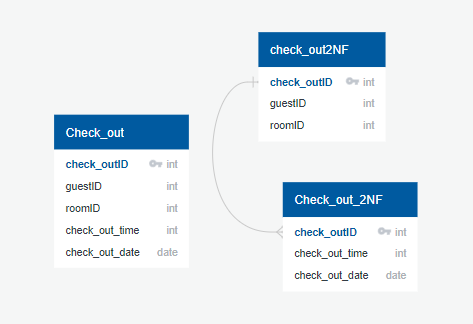
A relation R is in second normal form (2NF) if and only if it is in 1NF and every non-primary key attribute is fully dependent on the primary key or, in simpler words, no non-primary key attribute is partially dependent on the primary key.

To normalize a table to 2NF we:

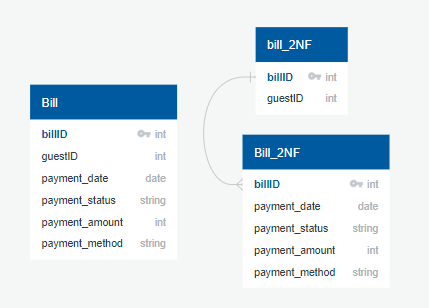
* + First normalize it to 1NF
  + Decompose relation in such a way that resultant relations do not have any partial FD.
  + Remove partial dependent attributes from the relation.
  + Place them in separate relation along with the prime attribute on which they are fully dependent.
  + The primary key of new relation will be the attribute on which it is fully dependent.
  + Keep other attributes same as in that table with the same primary key.

**

*2NF for Check in Table*



*2NF for the Check out Table*

**

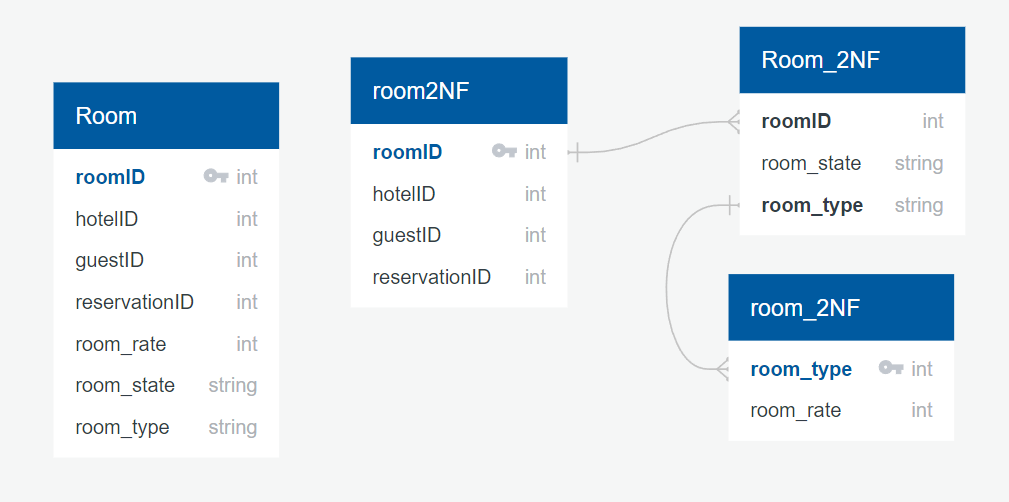
*2NF for Bill Table*

* **Third Normal Form (3NF)**

A relation R is in third normal form (3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent on the primary key or, no non-key attribute is transitively dependent on the primary key.

To normalize a table to 3NF we:

* + Decompose relation in such a way that resultant relations do not have any transitive FD.
  + Remove transitive dependent attributes from the relation.
  + Place them in a new relation along with the non-prime attributes due to which transitive dependency occurred.
  + The primary key of the new relation will be non-prime attributes due to which transitive dependency occurred.
  + Keep other attributes same as in the table with same primary key and add prime attributes of other relation into it as a foreign key

****

*2NF & 3NF for Room Table*

**PL/SQL:**

Some of the tables have been filled initially using both sql and pl/sql queries:

begin

add\_room(101,'standard','empty',10000);

add\_room(102,'standard','empty',10000);

add\_room(103,'standard','empty',10000);

add\_room(104,'standard','empty',10000);

add\_room(105,'standard','empty',10000);

add\_room(201,'standard','empty',10000);

add\_room(202,'standard','empty',10000);

add\_room(203,'standard','empty',10000);

add\_room(204,'standard','empty',10000);

add\_room(205,'standard','empty',10000);

add\_room(301,'deluxe','empty',15000);

add\_room(302,'deluxe','empty',15000);

add\_room(303,'deluxe','empty',15000);

add\_room(304,'deluxe','empty',15000);

add\_room(305,'deluxe','empty',15000);

add\_room(401,'deluxe','empty',15000);

add\_room(402,'deluxe','empty',15000);

add\_room(403,'deluxe','empty',15000);

add\_room(404,'deluxe','empty',15000);

add\_room(405,'deluxe','empty',15000);

add\_room(501,'suite','empty',25000);

add\_room(502,'suite','empty',25000);

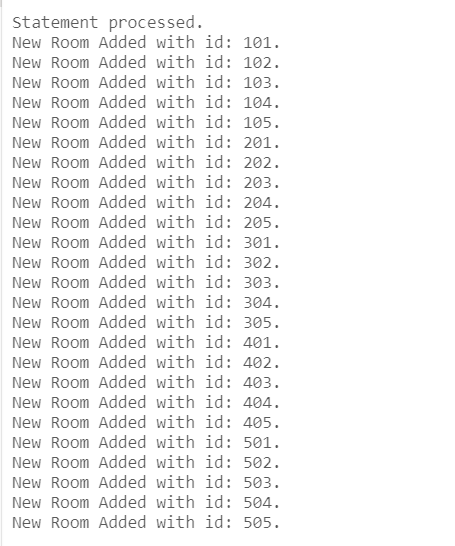
add\_room(503,'suite','empty',25000);

add\_room(504,'suite','empty',25000);

add\_room(505,'suite','empty',25000);

end;

/



begin

add\_employee(1,'Ayesha Gupta','Manager',80000,'A404 Garden Heights, Patiala',9646481684);

add\_employee(2,'Sandhya Gulati','Housekeeper',12000,'33 near rai bhadur road, patiala',9646481684);

add\_employee(3,'Shiv Kumar','Housekeeper',12000,'42 gulab nagar, patiala',9014820571);

add\_employee(4,'Rita Rani','Housekeeper',12000,'27 NTC school road, patiala',9057120148);

add\_employee(5,'Honey Singh','Housekeeper',12000,'1443 ward 27c, patiala',9978155423);

add\_employee(6,'Kamala','Housekeeper',12000,'273 gulab nagar, patiala',9787234561);

add\_employee(7,'Himanshi Midha','Receptionist',22000,'320 Hostel N, Patiala',8054220871);

add\_employee(8,'Virat Shah','Receptionist',22000,'',8008713452);

add\_employee(9, 'John Doe', 'Waiter', 10000, '123 Main St, Anytown', 555-1234),

add\_employee(10, 'Jane Smith', 'Waiter', 10000, '456 Oak St, Anytown', 555-5678),

add\_employee(11, 'Bob Johnson', 'Waiter', 10000, '789 Elm St, Anytown', 555-9012),

add\_employee(12, 'Sara Lee', 'Waiter', 10000, '321 Maple St, Anytown', 555-3456),

add\_employee(13, 'Mike Johnson', 'Waiter', 10000, '654 Pine St, Anytown', 555-7890);

add\_guest('Jatin Malik','13 Cornelia Street',7009854237,'jmalik@gmail.com');

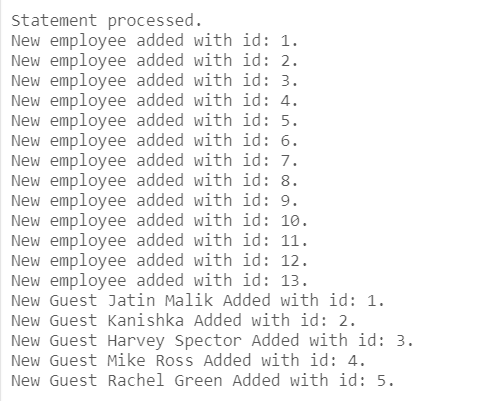
add\_guest('Kanishka','Sector 20, Panchkula',9763241876,'kani22@gmail.com');

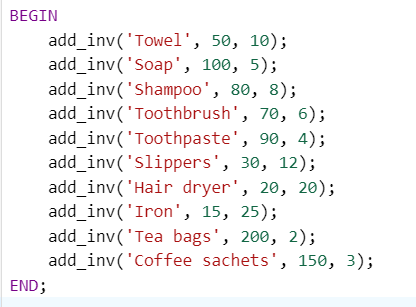
add\_guest('Harvey Spector','223, 5th Avenue',7634576344,'hs0308@gmail.com');

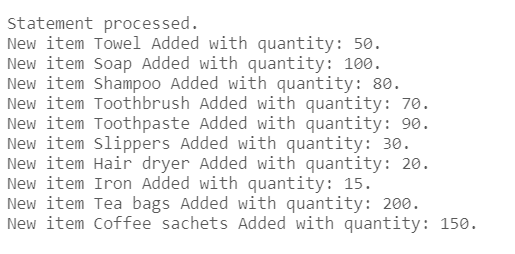
add\_guest('Mike Ross','#2407, 37th Street Midtown South',964640406,'r142mr@gmail.com');

add\_guest('Rachel Green','16th Avenue, NYC',964343406,'rg22@gmail.com');

end;







In Oracle PL/SQL, a stored procedure is a named block of code that performs one or more related database operations. It is a subroutine or function that is stored in the database and can be executed by calling its name. Stored procedures are compiled and stored in the database, which allows them to be executed repeatedly without having to recompile the code each time.

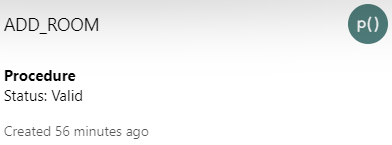
Stored procedures in PL/SQL can take input parameters and return output parameters, making them versatile and powerful tools for working with databases. They can be used to perform a wide variety of operations, such as updating or querying data, performing calculations, sending email notifications, or generating reports.

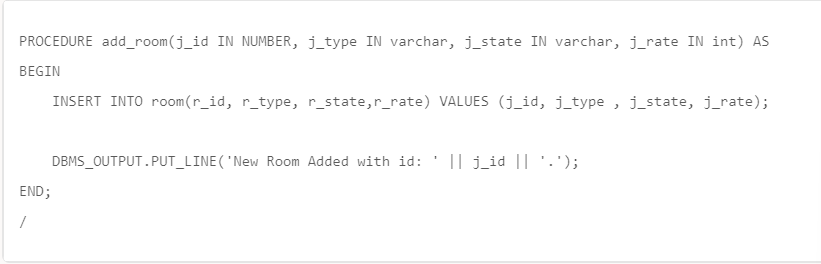
One of the main benefits of stored procedures is that they allow complex database operations to be encapsulated and managed as a single unit. This can simplify the programming process, improve performance, and increase security by limiting direct access to the underlying database. Additionally, stored procedures can be used to implement business rules and logic in a consistent and standardized way across different applications and users.

To create a stored procedure in PL/SQL, you use the **CREATE PROCEDURE** statement followed by the name of the procedure, any input parameters, and the block of code that defines the procedure's functionality.

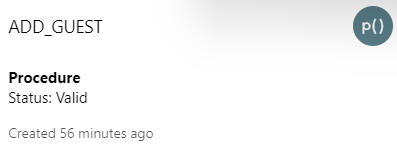
The following stored procedures have been used:

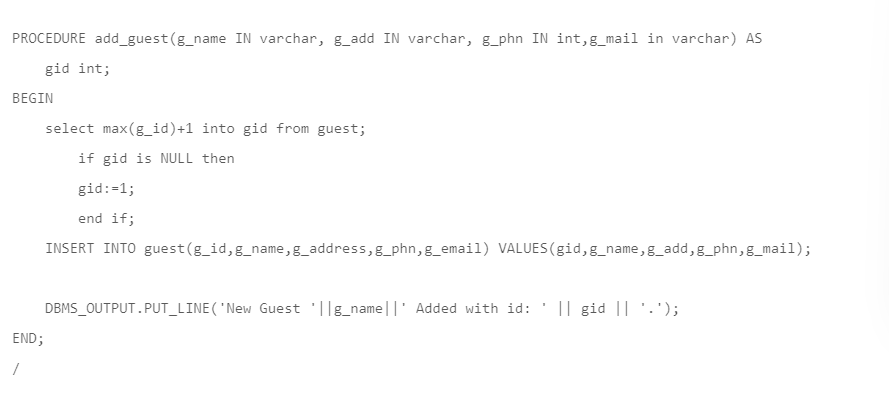
1. add\_room: This procedure simply inputs id, type, state and rate of a new room and adds it to the room table. Since the rooms can be on different floors and have no continuity in their ids, the room id is also taken in as a parameter to avoid any errors. The procedure also displays a message once the room is added along with the room id of that respective room.



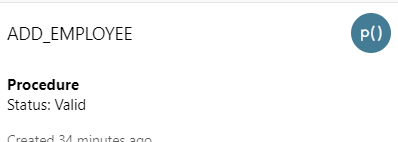


1. add\_guest: This procedure takes name, address , phone number and email id of a new guest as parameters and inserts it into guest table. The guest id is itself calculated by adding one to the last guest id and if the table is empty the id starts from 1 by itself. Therefore guest id is not taken as an input parameter. After successfully executing the procedure also displays a message that a new guest has been added with the name and guest id of that respective guest. The working of this procedure along with the output have been illustrated above.





1. add\_employee: This procedure takes in employee id, name, role(manager, housekeeper, etc), salary, address and phone number as input parameters and inserts it into employee table while displaying a message that a new employee has been added and the employee id is also printed along with it.



CREATE or replace PROCEDURE add\_employee

(emp\_id\_val in int, emp\_name\_val varchar,emp\_role\_val in varchar,emp\_sal\_val in int,emp\_address\_val in varchar,emp\_phn\_val in varchar) as

BEGIN

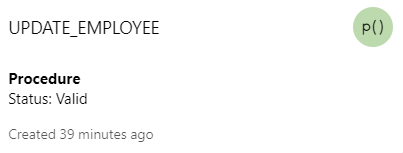
INSERT INTO employee VALUES (emp\_id\_val, emp\_name\_val, emp\_role\_val, emp\_sal\_val, emp\_address\_val, emp\_phn\_val);

DBMS\_OUTPUT.PUT\_LINE('New employee added with id: ' || emp\_id\_val || '.');

END;

/

1. update\_employee: This procedure takes in all the values of add employee procedure as input parameters. Any of these values can be changed to be updated other than the employee id since that is what the procedure uses to identify the employee. If the field remains the same it is sent in the same but it still has to be passed as a parameter.



CREATE or replace PROCEDURE update\_employee

(emp\_id\_val in int,emp\_name\_val in varchar,emp\_role\_val in varchar,emp\_sal\_val in int,emp\_address\_val in varchar,emp\_phn\_val in varchar) as

BEGIN

UPDATE employee SET

emp\_name = emp\_name\_val,

emp\_role = emp\_role\_val,

emp\_sal = emp\_sal\_val,

emp\_address = emp\_address\_val,

emp\_phn = emp\_phn\_val

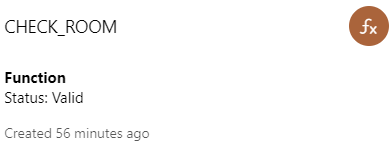
WHERE emp\_id = emp\_id\_val;

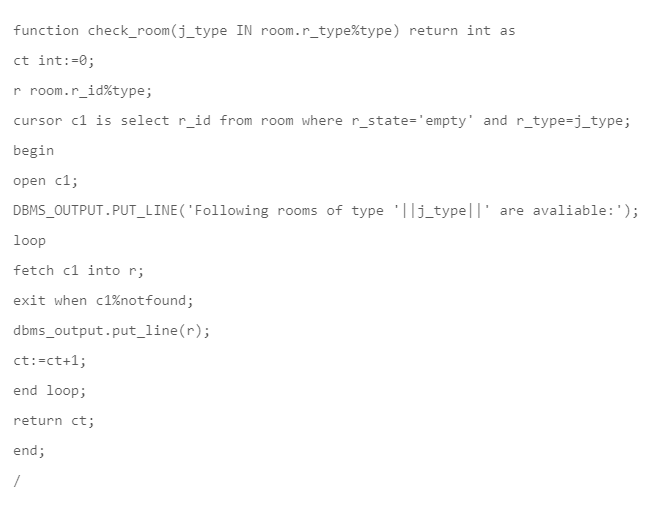
DBMS\_OUTPUT.PUT\_LINE('Employee with id: ' || emp\_id\_val || ' info has been updated.');

END;

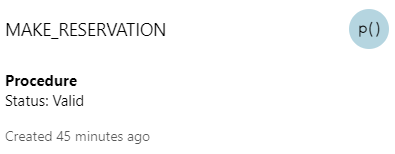
/

1. check\_room: This procedure takes in the type (deluxe, standard or suite) of the room as a parameter and finds out all the rooms of that type that are empty. It makes use of a cursor to fetch these records from the room table and prints them one by one as the room id unless no more rooms are available.





1. make\_reservation: The make reservation procedure takes in guest id, room type and check in date as input parameters. It finds an empty room of the input type and if found makes a reservation by inserting a new tuple in the reservation table. A new tuple in bill table is also added at the time of reservation. If the reservation is successful it displays a message along with the room number and in case an empty room is not found another message is displayed that the reservation was unsuccessfull.



create or replace procedure make\_reservation(g\_id in number,r\_typ in varchar,dt in date) as

r\_no int:=0;

res\_no int;

b\_no int;

begin

select r\_id into r\_no from room where r\_state='empty' and r\_type=r\_typ FETCH FIRST ROWS ONLY;

if r\_no!=0 then

select max(reser\_id)+1 into res\_no from reservation;

if res\_no is NULL then

res\_no:=1;

end if;

insert into reservation values(res\_no,r\_no,g\_id,dt);

update room set r\_state='reserved' where r\_id=r\_no;

select max(b\_id)+1 into b\_no from bill;

if b\_no is NULL then

b\_no:=1;

end if;

INSERT INTO bill(b\_id, g\_id, r\_id, cin\_date, payment\_status) VALUES(b\_no,g\_id,r\_no,dt,'pending');

dbms\_output.put\_line('Your reservation was successful. Room number is: '||r\_no);

else

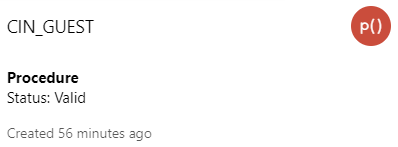
dbms\_output.put\_line('Your reservation was not successful. NO empty room!');

end if;

end;

/

1. cin\_guest: The check in guest procedure is used when the guest wants to check in his/her room. It just takes the guest id provided at the time of reservation as input and adds a tuple in the check in table. Since the guest has checked in the tuple of that respective guest is deleted from the reservation table and the status of their room is changed to ‘occupied’.



create or replace procedure cin\_guest(gid in int) as

room int:=0;

dt date;

cid int;

begin

select r\_id into room from reservation where g\_id=gid;

if room!=0 then

select check\_in\_date into dt from reservation where g\_id=gid;

select max(cin\_id)+1 into cid from check\_in;

if cid is NULL then

cid:=1;

end if;

insert into check\_in values(cid,gid,room,dt);

dbms\_output.put\_line('You can check in to Room no: '||room);

update room set r\_state='occupied' where r\_id=room;

delete from reservation where r\_id=room;

else

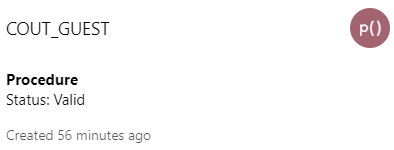
dbms\_output.put\_line('You dont have a reservation.');

end if;

end;

/

1. cout\_guest: The check out procedure takes guest id and check out date (could be the system date) as input parameter and adds a tuple to the check out table. It then updates the bill table to add the check out date passed in by the user for that same guest id and then calls the generate bill procedure to calculate the total bill which is then displayed. After the payment the status of bill is set to clear and a visit again message is displayed.



create or replace procedure cout\_guest(gid in int,ctdt date) as

ta int;

room int;

cid int;

begin

select max(cout\_id)+1 into cid from check\_out;

if cid is NULL then

cid:=1;

end if;

select r\_id into room from check\_in where g\_id=gid;

insert into check\_out values(cid,gid,room,ctdt);

update bill set cout\_date=ctdt where g\_id=gid;

generate\_bill(gid);

select total\_amount into ta from bill where g\_id=gid;

dbms\_output.put\_line('Checking Out...');

dbms\_output.put\_line('Your Total Bill is: '||ta);

update bill set payment\_status='clear' where g\_id=gid;

update room set r\_state='empty' where r\_id=room;

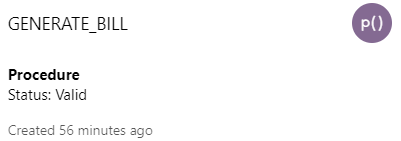
dbms\_output.put\_line('Payment Successful!');

dbms\_output.put\_line('Thanks for staying :-)');

end;

/

1. generate\_bill: This procedure takes guest id as input and then calculates the number of days between the check in date and the check out date. The rate of the room in which the guest is staying is found and multiplied with the number of days to produce the bill. This procedure is to be executed once the guest wants to check out.



create or replace procedure generate\_bill(gid in int) as

room int;

room\_rate int;

cin date;

cout date;

diff int;

ta int;

begin

select r\_id,cin\_date, cout\_date into room,cin,cout from bill where g\_id=gid;

select r\_rate into room\_rate from room where r\_id=room;

select (cout-cin) into diff from dual;

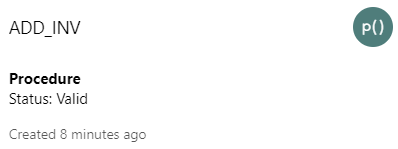
ta:=diff\*room\_rate;

update bill set total\_amount=ta where g\_id=gid;

end;

/

1. add\_inv: This procedure takes in the item name, quantity and its rate and adds a tuple to the inventory table while also displaying a message that a new item with its name and quantity has been added. It does not take item id as an input parameter as it is calculated by adding one to the last input id.



CREATE OR REPLACE PROCEDURE add\_inv(invname IN varchar, invqty IN int,invrate in int) AS

invid int;

BEGIN

select max(inv\_id)+1 into invid from inventory;

if invid is NULL then

invid:=1;

end if;

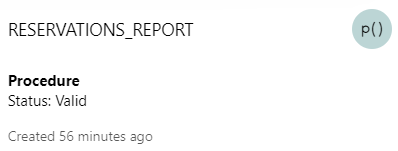
INSERTINTOinventory(inv\_id,inv\_name,inv\_qty,inv\_rate) VALUES(invid,invname,invqty,invrate);

DBMS\_OUTPUT.PUT\_LINE('New item '||invname||' Added with quantity: ' || invqty || '.');

END;

/

1. reservation\_report: This procedure is used to display a report of all the active reservations at the time. It takes no input parameters and makes use of a system cursor to fetch data from guest, reservation and room tables. This data is then displayed tuple by tuple in a report format.



CREATE OR REPLACE PROCEDURE reservations\_report

IS

cursor\_reservations SYS\_REFCURSOR;

r\_guest\_id guest.g\_id%TYPE;

r\_guest\_name guest.g\_name%TYPE;

r\_room\_id room.r\_id%TYPE;

r\_checkin\_date reservation.check\_in\_date%TYPE;

BEGIN

OPEN cursor\_reservations FOR

SELECT guest.g\_id, guest.g\_name, reservation.r\_id, reservation.check\_in\_date

FROM guest

JOIN reservation ON guest.g\_id = reservation.g\_id

JOIN room ON reservation.r\_id = room.r\_id

ORDER BY guest.g\_id;

DBMS\_OUTPUT.PUT\_LINE('RESERVATIONS REPORT');

DBMS\_OUTPUT.PUT\_LINE('------------------------------------');

DBMS\_OUTPUT.PUT\_LINE('GUEST ID | GUEST NAME | ROOM ID | CHECK-IN DATE ');

LOOP

FETCH cursor\_reservations INTO r\_guest\_id, r\_guest\_name, r\_room\_id, r\_checkin\_date;

EXIT WHEN cursor\_reservations%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(r\_guest\_id || ' | ' || r\_guest\_name || ' | ' || r\_room\_id || ' | ' || r\_checkin\_date );

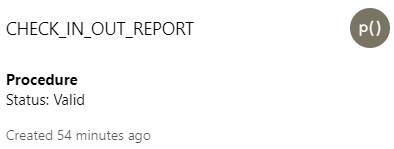
END LOOP;

CLOSE cursor\_reservations;

END;

/

1. check\_in\_out\_report: This procedure is used when you want a report of all the check ins and check outs within a given time range. It takes start and end dates as input parameters and uses a cursor to fetch data from check in and check out tables and then display it in a report format.



CREATE or replace PROCEDURE check\_in\_out\_report(start\_date DATE, end\_date DATE)

AS

CURSOR c1 IS SELECT check\_in.g\_id, check\_in.r\_id, check\_in.cin\_date,

check\_out.cout\_date

FROM check\_in

JOIN check\_out ON check\_in.g\_id = check\_out.g\_id AND check\_in.r\_id = check\_out.r\_id

WHERE check\_in.cin\_date BETWEEN start\_date AND end\_date;

BEGIN

-- Print report header

DBMS\_OUTPUT.PUT\_LINE('CHECK-IN/OUT REPORT');

DBMS\_OUTPUT.PUT\_LINE('Date Range: ' || start\_date || ' - ' || end\_date);

DBMS\_OUTPUT.PUT\_LINE('-----------------------');

-- Print report rows

FOR row IN c1 LOOP

DBMS\_OUTPUT.PUT\_LINE(row.g\_id || ' - ' || row.r\_id || ' - ' || row.cin\_date || ' - ' || row.cout\_date);

END LOOP;

END;

/

An outer while loop is used to call these functions as per a choice:

declare

a int:=5;

ch int:=0; --Enter choice here(leave default at 0)

n int;

begin

while a!=0 loop

dbms\_output.put\_line('Welcome to THE PLAZA!');

dbms\_output.put\_line('---Menu---');

dbms\_output.put\_line('1. Settings');

dbms\_output.put\_line('2. Check available rooms');

dbms\_output.put\_line('3. Add Guest');

dbms\_output.put\_line('4. Reserve Room');

dbms\_output.put\_line('5. Check In Guest');

dbms\_output.put\_line('6. Check out Guest');

dbms\_output.put\_line(' ');

dbms\_output.put\_line('Your Choice is: '||ch);

if ch=1 then

settings(0); --Enter choice here(leave default at 0)

elsif ch=2 then

n:=check\_room('standard');

elsif ch=3 then

add\_guest('Ayesha','304 Sector 3, Panchkula',9876631792,'agupta@gmail.com');

elsif ch=4 then

make\_reservation(1,'standard',date'2023-04-25');

elsif ch=5 then

cin\_guest(1);

elsif ch=6 then

cout\_guest(1,date'2023-04-26');

else

dbms\_output.put\_line('Invalid Choice');

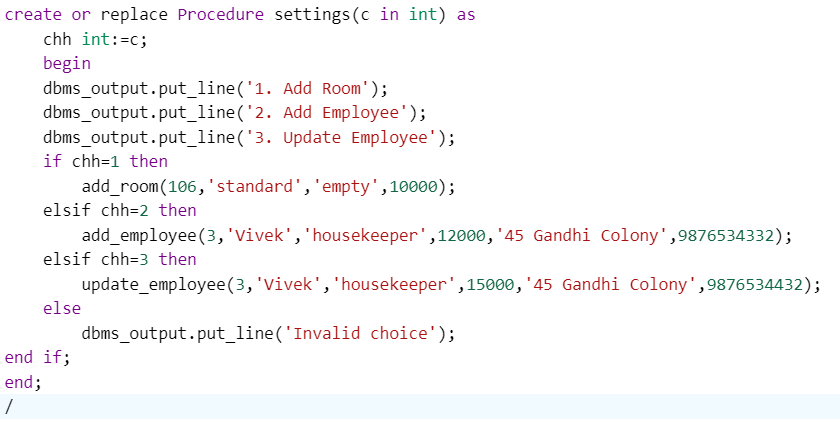
end if;

a:=0;

end loop;

end;

Since some of the procedures are rarely called they have been added to another function called settings:



The values in these above code snippets are to be taken from the user but since live sql oracle doesn’t support user inputs values have already been inserted to show what it might look like.

**Conclusion:**

Hotel needs to maintain the record of guests and reserve rooms beforehand. Customers should be able to know the availability of the rooms on a particular date. They should be able to reserve the available rooms according to their need in advance so as to make their stay comfortable. No hotel reservation system can operate alone in the hotel because it concentrates mainly in reservation process, the system needs an accounting system and a management information system, so it can serve all the needs of the hotel. It is therefore very important to build new and modern, flexible, dynamic and compatible reusable information systems including database to help manipulate different processes and operations that are carried out in the hotels.