Provided By

Ahmed Sultan Asiri – 1636836

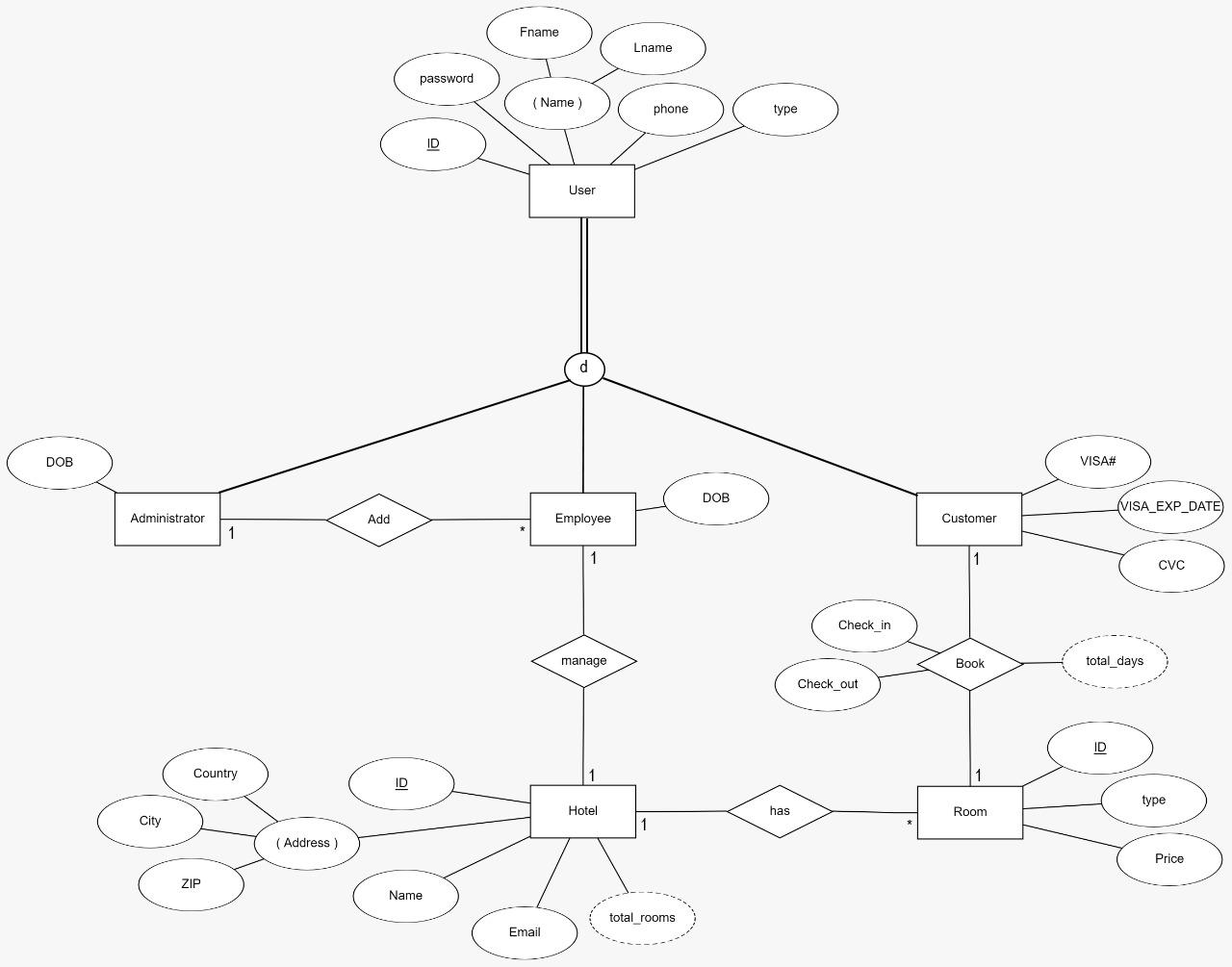
Provided For

Dr. Mutasem Mohammed Jarrah

ONLINE BOOKING DATABASE

CPIT-340 PROJECT

Enhanced-ERD

****

Schema

**USERS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Fname | Lname | Password | Phone | Type |

**CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| CID | Fname | Lname | Password |

**ADMINISTRATOR**

|  |  |
| --- | --- |
| AID | DOB |

**EMPLOYEE**

|  |  |
| --- | --- |
| EID | DOB |

**HOTEL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| HID | Name | Email | Country | City | ZIP |

**ROOM**

|  |  |  |  |
| --- | --- | --- | --- |
| RID | Type | Price | Hotel\_ID |

**BOOK**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BID | Customer\_ID | Hotel\_ID | Room\_ID | Check\_in | Check\_out |

Create Tables

**HOTEL**

create table hotel(id int primary key, name varchar2(50), email varchar2(50),

country varchar2(20), city varchar2(20), zip int, total\_rooms int);

**ROOM**

create table room(id int, type varchar2(10), price float, hotel\_id int,

foreign key (hotel\_id) references hotel(id), primary key(id, hotel\_id));

**USERS**

create table users(id int primary key, fname varchar2(20), lname varchar2(20),

password varchar2(20), phone varchar2(15), type varchar2(20));

**CUSTOMER**

create table customer(id int primary key, visa\_number int, visa\_exp\_date date,

cvc int, foreign key (id) references users(id));

**EMPLOYEE**

create table employee(id int primary key, dob date, admin\_id int, hotel\_id int,

foreign key (admin\_id) references administrator(id), foreign key (hotel\_id) references hotel(id),

foreign key (id) references users(id));

**AMINISTRATOR**

create table administrator(id int primary key, dob date,

foreign key (id) references users(id));

**BOOK**

create table book(id int, customer\_id int, room\_id int, hotel\_id int,

check\_in date, check\_out date, total\_days int, total\_price float,

foreign key (customer\_id) references customer(id),

foreign key (room\_id, hotel\_id) references room(id,hotel\_id),

primary key(id, customer\_id, room\_id, hotel\_id));

Insert into Tables

**HOTEL**

insert into hotel values(4865, 'Irbid Plaza', 'IrbidPlaza@gmail.com',

'Jordan', 'Irbid', 22699, 0);

insert into hotel values(9879, 'Al Joude Hotel', 'Al-Joude@gmail.com',

'Jordan', 'Irbid', 22987, 0);

insert into hotel values(2323, 'Seven Days Hotel', '7Days@gmail.com',

'Jordan', 'Irbid', 22984, 0);

**ROOM**

insert into room values(103, 'Suite', 400, 9879);

**USERS**

insert into users values(000598, 'Mutasem', 'Jarrah', '123456789', '0545454545', 'Administrator');

insert into users values(1746954, 'Mostafa', 'Ramadan', '232323', '054848484', 'Customer');

insert into users values(1636836, 'Ahmed', 'Asiri', '123456', '0562109709', 'Customer');

**ADMINISTRATOR**

Insert into administrator values(598, sysdate – 15000);

**CUSTOMER**

Insert into customer values(1636836, 48498757, ‘13-MAR-20’, 255);

**EMPLOYEE**

Insert into employee values(588784, sysdate – 10000, 000598, 1111);

**BOOK**

Insert into book values(87647988, 1746954, 105,1111, sysdate+7, sysdate+15, 0, 0);

**SQL/PL BLOCK**

**1- Biggest outcome**

**Description:** print the hotel information that have the biggest outcome in the hotels table.

**Code:**

declare

biggestOutcome book.total\_price%type:= 0;

biggestHotelId hotel.id%type:= 0.0;

hotelName hotel.name%type;

hotelCity hotel.city%type;

cursor cursHotelsOutcome

is

select hotel\_id, sum(total\_price) as total\_outcome from book group by hotel\_id;

hotelsOutcome cursHotelsOutcome%rowtype;

begin

open cursHotelsOutcome;

loop

fetch cursHotelsOutcome into hotelsOutcome;

if(cursHotelsOutcome%notfound) then

dbms\_output.put\_line('Hotel ID: ' || biggestHotelId || ', Hotel Name: ' || hotelName || ', Hotel City: ' || hotelCity || ', Total outcome: ' || biggestOutcome);

end if;

exit when(cursHotelsOutcome%notfound);

if (hotelsOutcome.total\_outcome > biggestOutcome) then

biggestOutcome:= hotelsOutcome.total\_outcome;

biggestHotelId:= hotelsOutcome.hotel\_id;

select name, city into hotelName, hotelCity from hotel where id = biggestHotelId;

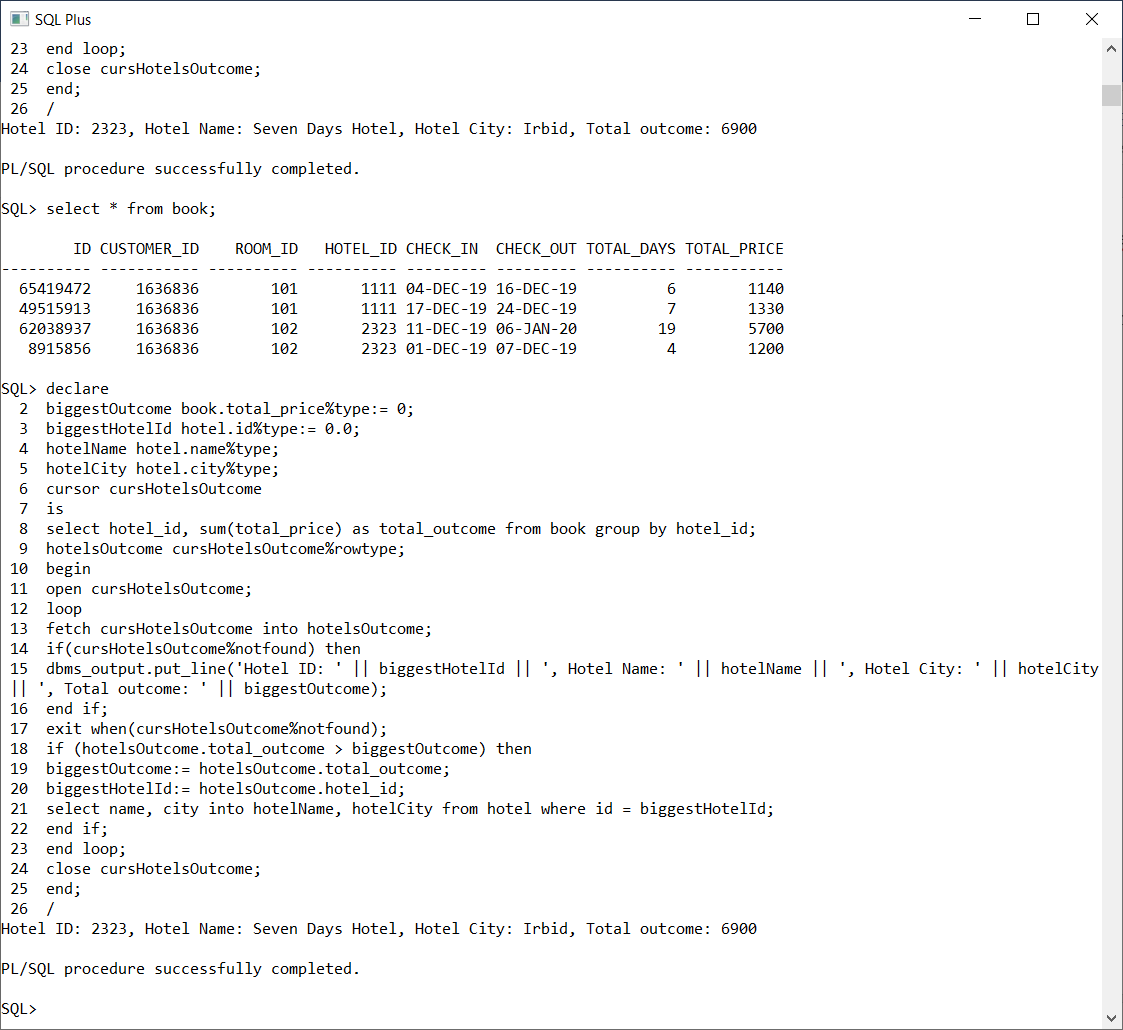
end if;

end loop;

close cursHotelsOutcome;

end;

**Biggest outcome execution**



**SQL/PL TRIGGER**

**1- before\_insert\_book**

**Description:** before inserting a new book, we need to check if the room is already reserved by another customer in that date dimension, and also check if the random generated book id is exist, if one of the previous conditions exist then abort the insertion.

**Code:**

create or replace trigger before\_insert\_book before insert on book for each row

declare

available int;

tempBookId int;

exist int;

room\_price float;

begin

select isAvailable(:new.room\_id, :new.hotel\_id, :new.check\_in, :new.check\_out) into available from dual;

if (available = 1) then

select generateBookId() into tempBookId from dual;

select isExist(tempBookId) into exist from dual;

if (exist = -1) then

:new.id := tempBookId;

select price into room\_price from room where id = :new.room\_id and hotel\_id = :new.hotel\_id;

:new.total\_days := :new.check\_out - :new.check\_in;

:new.total\_price := :new.total\_days \* room\_price;

else

Raise\_application\_error(-20007, 'There is a problem happend when generating Book id, so please try again');

end if;

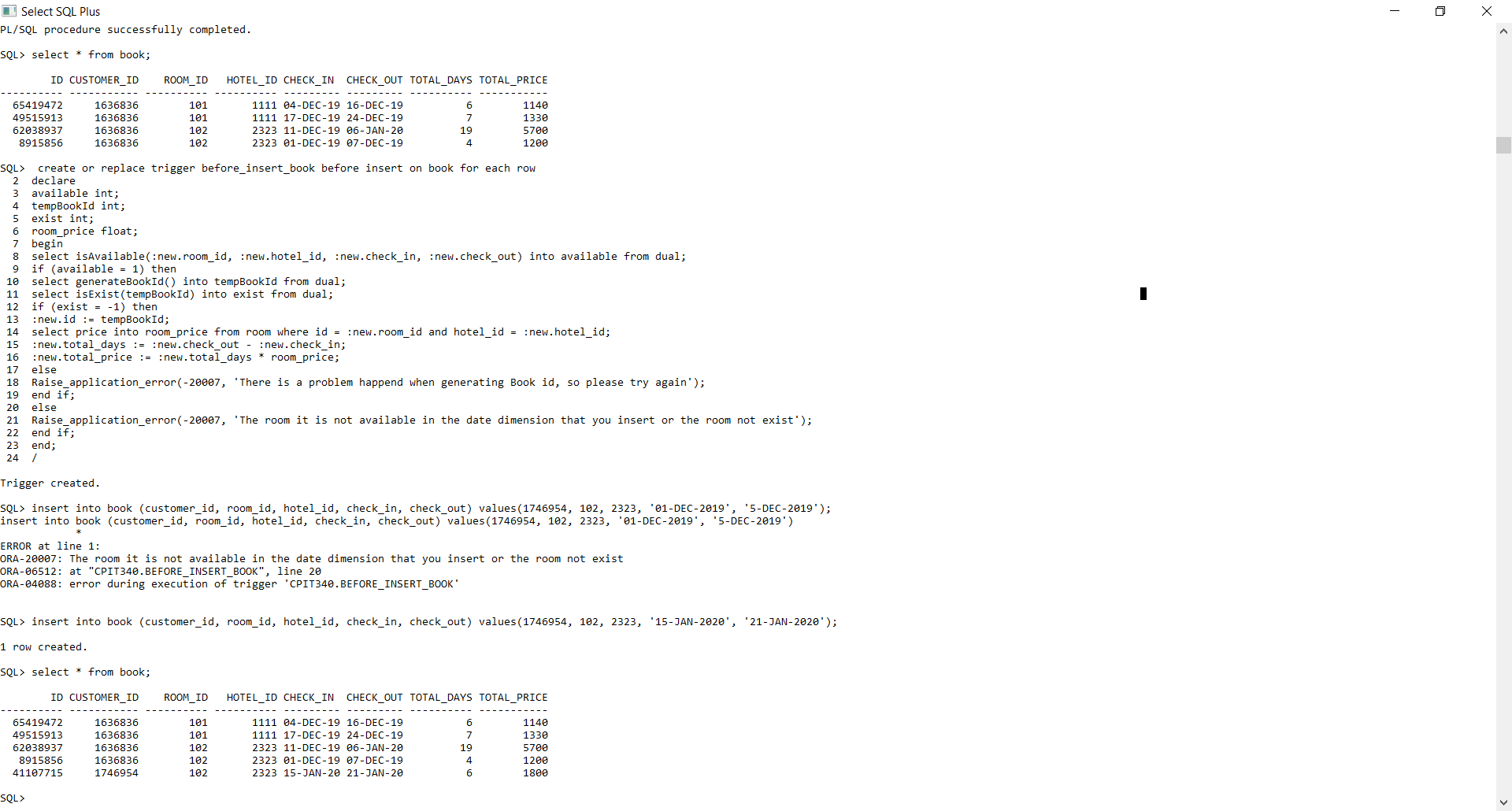
else

Raise\_application\_error(-20007, 'The room it is not available in the date dimension that you insert or the room not exist');

end if;

end;

**before\_insert\_book EVENT & ACTION**



**SQL/PL PROCEDURE**

**1-extendBook**(book id, new check-out date)

**Description:** extend the customer book if there is availability for that, we need to check if there is a customer booked the room on coming date that make conflicts.

**Code:**

create or replace procedure extendBook(bookId in int, newOutDate in date)

is

availability int;

oldOutDate date;

roomId int;

hotelId int;

exist int;

begin

select isExist(bookId) into exist from dual;

if (exist = 1) then

select check\_out, room\_id, hotel\_id into oldOutDate, roomId, hotelId from book where id = bookId;

select isAvailable(roomId, hotelId, oldOutDate+1, newOutDate) into availability from dual;

if (availability = 1) then

update book set check\_out = newOutDate where id = bookId;

dbms\_output.put\_line('The check-out date has been extended.');

else

dbms\_output.put\_line('The room is Not Available for extending.');

end if;

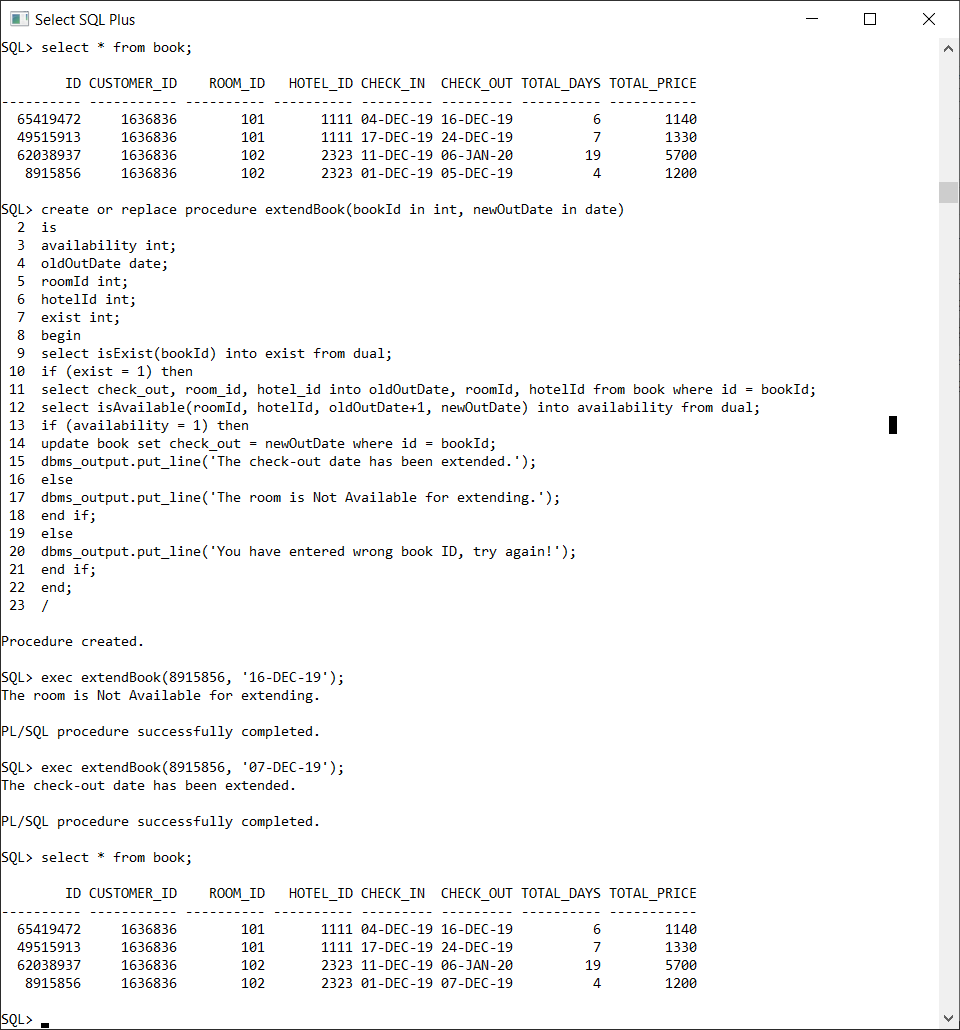
else

dbms\_output.put\_line('You have entered wrong book ID, try again!');

end if;

end;

**extendBook execution**



**SQL/PL FUNCTION**

1- **isAvailable**(roomId in int, hotelId in int, inDate in date, outdate in date)

**Description:** To check if the room is available for customer book or it’s already booked for another customer in the same specified date dimension.

**Result:**

* Available = 1
* Not Available = -1

**Code:**

create or replace function isAvailable(roomId in int, hotelId in int, inDate in date, outDate in date) return int

as

availability int := 1;

cursor RoomRow

is

select room\_id, hotel\_id from book where

check\_in between inDate and outDate

or

check\_out between inDate and outDate;

VarRoom RoomRow%RowType;

begin

open RoomRow;

loop

fetch RoomRow into VarRoom;

exit when(RoomRow%notfound);

if (VarRoom.ROOM\_ID = roomId and VarRoom.HOTEL\_ID = hotelId) then

availability := -1;

end if;

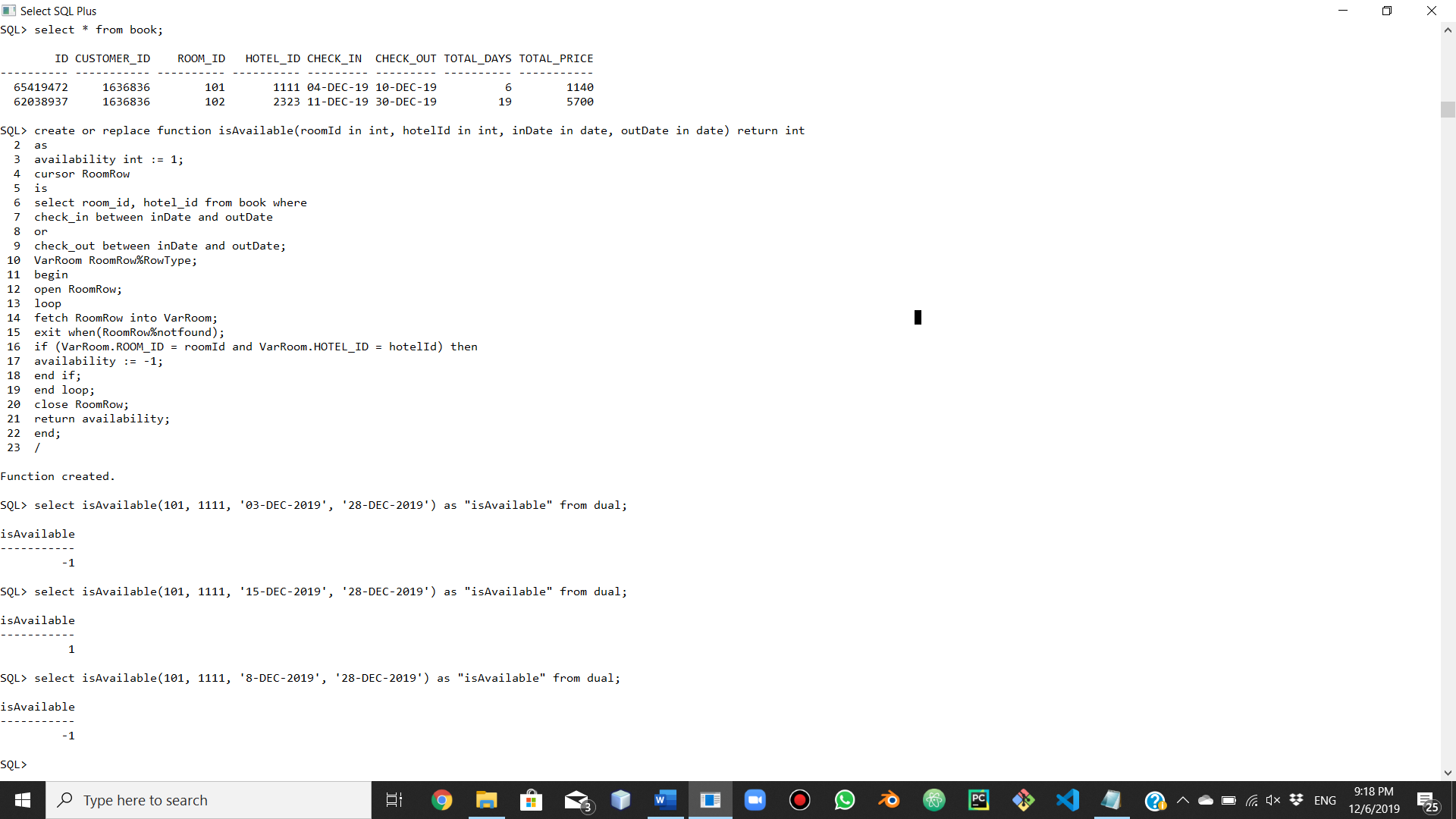
end loop;

close RoomRow;

return availability;

end;

**isAvailable Execution**



**2- displayRoomsAvailability**(hotel id, check-in date, check-out date)

**Description:** This function is to return table with additional column that show the rooms availability in a hotel, with specified date.

**Result:** return Table.

**Code:**

create or replace function displayRoomsAvailability(hotelId in int, inDate in date, outDate in date) return sys\_refcursor

as

result\_curs sys\_refcursor;

begin

open result\_curs for

select hotel\_id, id as room\_id, type, price, case isAvailable(id, hotelId, inDate, outDate)

when 1 then 'Available'

when -1 then 'Not Available'

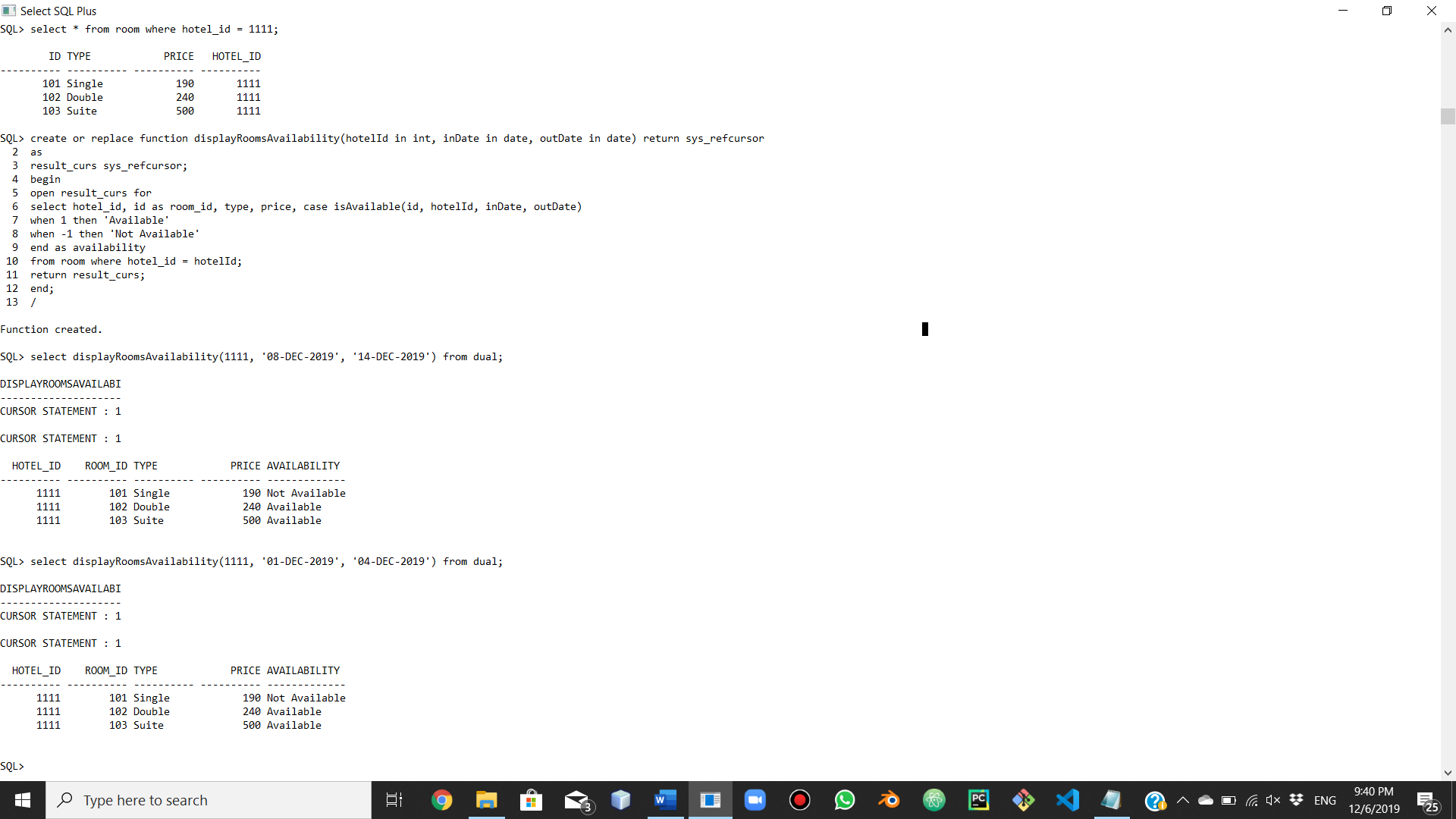
end as availability

from room where hotel\_id = hotelId;

return result\_curs;

end;

**displayRoomsAvailability Execution**



3- **generateBookId()**

**Description:** Generate random book id for customer book.

**Result:** random integer.

**Code:**

create or replace function generateBookId return int

as

randomNumber int;

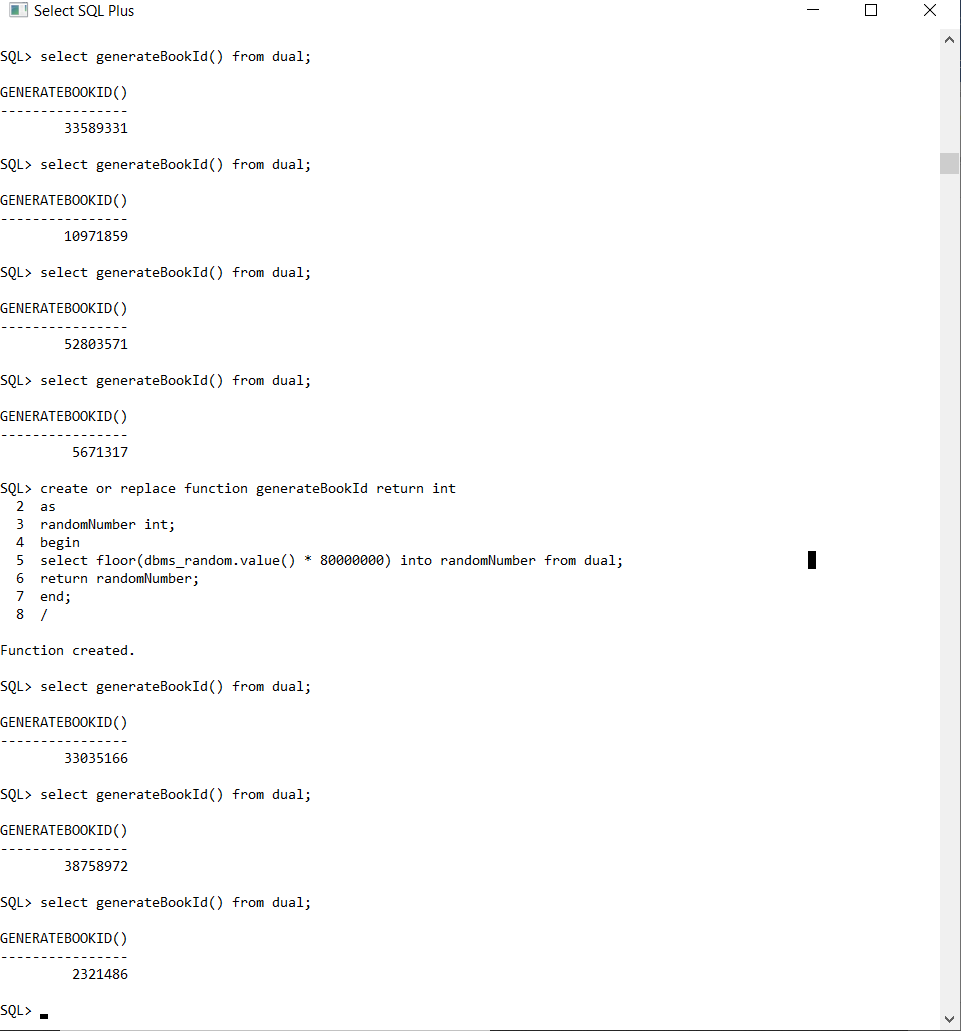
begin

select floor(dbms\_random.value() \* 80000000) into randomNumber from dual;

return randomNumber;

end;

**generateBookId execution**



**EXTRA TRIGGERS**

**AFTER INSERT ROOM**

create or replace trigger after\_insert\_room after insert on room for each row

begin

update hotel

set total\_rooms = total\_rooms + 1

where id = :new.hotel\_id;

end;

**BEFORE DELETE ROOM**

create or replace trigger before\_delete\_room before delete on room for each row

declare

booked int;

begin

select isThereFutureBook(:old.id) into booked from dual;

if(booked = -1) then

delete from book where room\_id = :old.id;

update hotel

set total\_rooms = total\_rooms - 1

where id = :old.hotel\_id;

else

raise\_application\_error(-20007, 'the room cannot be deleted there is depending book on it, you need to modify manually and then try again');

end if;

end;

create or replace trigger before\_delete\_useradmin before delete on users for each row

begin

if(:old.type = 'Administrator') then

raise\_application\_error(-20007, 'User Administrator Account cannot be deleted');

end if;

end;

**BEFORE DELETE CUSTOMER**

create or replace trigger before\_delete\_customer before delete on customer for each row

begin

delete from book where customer\_id = :old.id;

end;

**AFTER DELETE CUSTOMER**

create or replace trigger after\_delete\_customer after delete on customer for each row

begin

delete from users where id = :old.id and type = 'Customer';

end;

**BEFORE DELETE ADMIN**

create or replace trigger before\_delete\_admin before delete on administrator for each row

begin

raise\_application\_error(-20007, 'Administrator Account cannot be deleted, you can only modify it');

end;

**BEFORE UPDATE BOOK**

create or replace trigger before\_update\_book before update on book for each row

declare

roomPrice float;

begin

select price into roomPrice from room where id = :old.room\_id and hotel\_id = :old.hotel\_id;

:new.total\_days := :new.check\_out - :old.check\_in ;

:new.total\_price := roomPrice \* :new.total\_days;

end;

**EXTRA PROCEDURES**

**DISCOUNT ON ROOMS**

create or replace procedure discount(hotelId in int, percentage in float)

is

begin

update room

set

price = price - (price \* percentage)

where

hotel\_id = hotelId;

if (sql%found) then

dbms\_output.put\_line('The number of discounted rooms is: ' || sql%rowcount);

else

dbms\_output.put\_line('There is no rooms to perform discount');

end if;

end;

**EXTRA FUNCTIONS**

**NUMBER OF BOOKED ROOMS IN HOTEL WITH SPECIFIED DATE**

create or replace function getBookedRooms(hotelId in int, inDate in date, outDate in date) return int

as

counter int := 0;

begin

select count(room\_id) into counter from (select room\_id, hotel\_id from book where

check\_in between to\_date(inDate,'DD-MON-YY') and to\_date(outDate, 'DD-MON-YY')

or

check\_out between to\_date(inDate,'DD-MON-YY') and to\_date(outDate, 'DD-MON-YY') ) where hotel\_id = hotelId;

return counter;

end;

**CHECK IF ROOM HAVE FUTURED BOOK, USED IN BEFORE DELETE ROOM TRIGGER**

create or replace function isThereFutureBook(roomId in int) return int

as

booked int := -1;

tempDate date;

cursor books

is

select check\_in from book where room\_id = roomId;

begin

open books;

loop

fetch books into tempDate;

exit when(books%notfound);

if (tempDate > sysdate) then

booked := 1;

end if;

end loop;

close books;

return booked;

end;

**CHECK IF THE BOOK ID EXIST OR NOT, 1 FOR EXIST, -1 FOR NOT EXIST**

create or replace function isExist(bookId int) return int

as

exist int:= 1;

cursor book\_id

is

select id from book where id = bookId;

temp int;

begin

open book\_id;

fetch book\_id into temp;

if(book\_id%notfound) then

exist := -1;

end if;

return exist;

end;