IS 620 – Advanced Database Projects



**Group – 1**

**Abishek Agarwal Poddar**

**Balakumaran Bhoopathi**

**Mary Varsha Venantius**

**Neha Reddy Chandupatla**

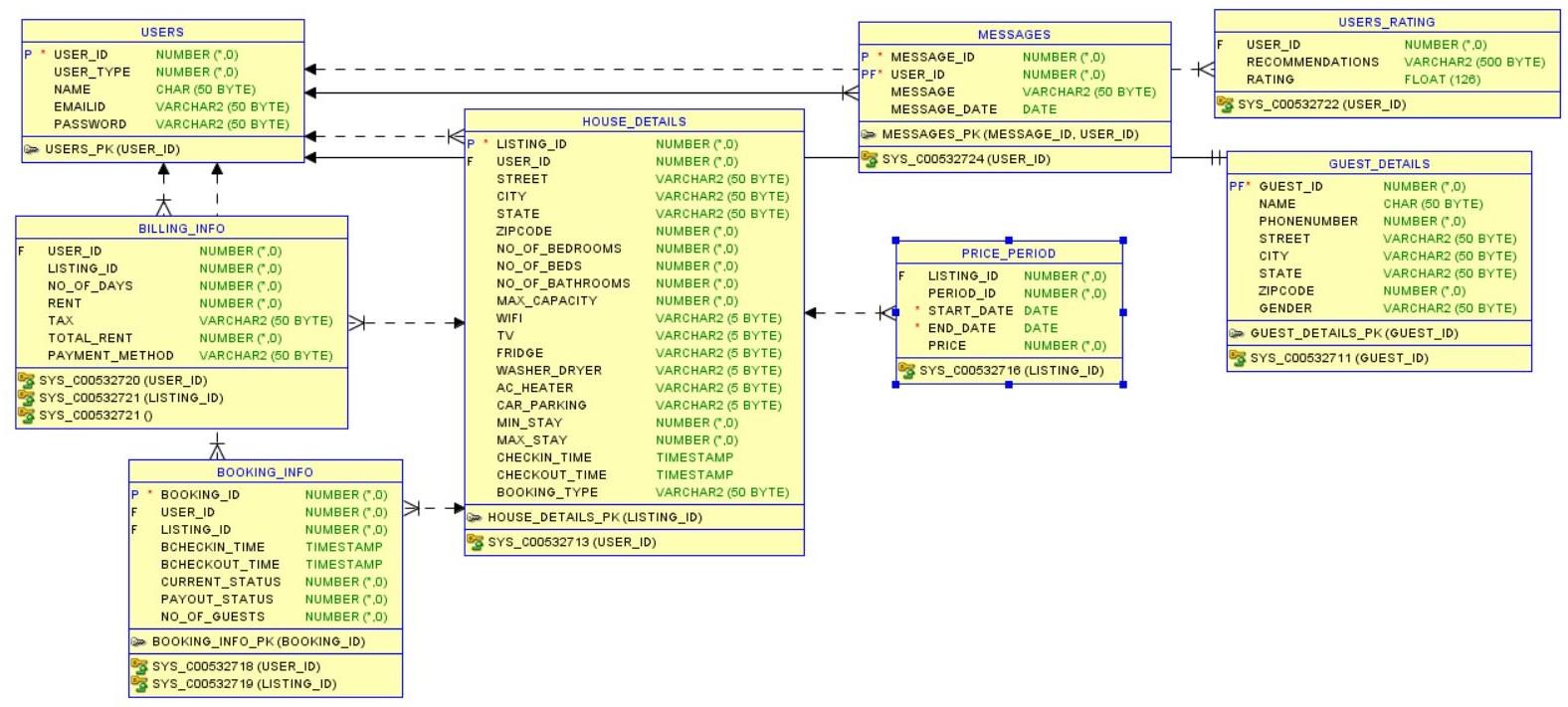
**Siddhi Shah**

**Vacation Home Rental System**

**Deliverable – 4**

**Department of Information System**

**DATABASE MODEL**



2. DROP STATEMENTS

Drop table messages;

Drop table users\_rating;

Drop table Price\_period;

Drop table billing\_info;

Drop table booking\_info;

Drop table House\_details;

Drop table Guest\_details;

Drop table users;

3. CREATE TABLES AND INSERT STATEMENTS

-------

create table users

(

user\_id int,

user\_type int,

name char(50),

emailid varchar(50),

password varchar(50),

primary key(user\_id)

);

-----------

Drop sequence users\_seq;

create sequence users\_seq start with 100 increment by 1 minvalue 1;

**INSERT QUERIES**

insert into users values

(users\_seq.NEXTVAL, 3,'Eve Eden','eden1@gmail.com', 'EdA01');

insert into users values

(users\_seq.NEXTVAL,2, 'Adam Grub','grub11@gmail.com', 'GrA02');

insert into users values

(users\_seq.NEXTVAL,2, 'Sun Flint','flint19@gmail.com', 'FlA03');

insert into users values

(users\_seq.NEXTVAL,2 ,'Ram Rao','rao20@gmail.com', 'RaA04');

insert into users values

(users\_seq.NEXTVAL,2 ,'Sia Gul','gul13@gmail.com', 'GuA05');

insert into users values

(users\_seq.NEXTVAL,3,'Eve Eden','edesn7@gmail.com','eden1');

insert into users values

(users\_seq.NEXTVAL,1,'Sam Mick','mick34@gmail.com','mick1');

insert into users values

(users\_seq.NEXTVAL,1,'Sue cho','cho4@gmail.com','cho4');

insert into users values

(users\_seq.NEXTVAL,1,'Raj Pak','pak2@gmail.com','pakk2');

insert into users values

(users\_seq.NEXTVAL,1,'Han Zhu','zhu3@gmail.com','zhu22');

--------

/\* user\_type is an integer where host=1,guest=2, host/guest=3\*/

---

create table Guest\_details

(

Guest\_id int,

name char(50),

phonenumber int,

street varchar(50),

city varchar(50),

state varchar(50),

zipcode int,

gender varchar(50),

primary key(Guest\_id),

foreign key(Guest\_id) references users

);

---

**INSERT QUERIES**

insert into Guest\_details values

(101, 'Eve Eden',17894561230, 'Runnyroad', 'Baltimore', 'MD', 001011,'f');

insert into Guest\_details values

(102, 'Sam Mick',17894561231, 'Sunnysip', 'Baltimore', 'MD', 001012,'m');

insert into Guest\_details values

(103, 'Sue Cho', 17894561232, 'Ramrag','Baltimore', 'MD', 001013,'f');

insert into Guest\_details values

(104, 'Raj Pak', 17894561233, 'Amway', 'Baltimore', 'MD',001014,'m');

insert into Guest\_details values

(105, 'Han Zhu', 17894561234, 'Bokley', 'Baltimore', 'MD',001015,'m');

-------

create table house\_details

(listing\_id int,

user\_id int,

street varchar(50),

city varchar(50),

state varchar(50),

zipcode int,

no\_of\_bedrooms int,

no\_of\_beds int,

no\_of\_bathrooms int,

max\_capacity int,

wifi varchar(5),

tv varchar(5),

fridge varchar(5),

washer\_dryer varchar(5),

AC\_heater varchar(5),

car\_parking varchar(5),

min\_stay int,

max\_stay int,

checkin\_time timestamp,

checkout\_time timestamp,

Booking\_type Varchar(50),

primary key(listing\_id),

foreign key(user\_id) references users

);

DROP sequence listing\_id\_sequence;

create sequence listing\_id\_sequence start with 1 increment by 1 minvalue 1;

**INSERT QUERIES**

insert into house\_details values (listing\_id\_sequence.NEXTVAL,101,'pratt street','arbutus','maryland',21225,2,3,2,4,'yes','yes','yes','yes','yes','yes',2,3,timestamp '2017-9-12 06:41:00.00 -05:00',timestamp '2017-9-14 06:41:00.00 -05:00','Bunglow');

insert into house\_details values(listing\_id\_sequence.NEXTVAL,102,'big street','arbutus','maryland',21225,2,3,2,4,'yes','yes','yes','yes','no','yes',2,3,timestamp '2017-9-15 06:41:00.00 -05:00',timestamp '2017-9-17 06:41:00.00 -05:00','Town House');

insert into house\_details values (listing\_id\_sequence.NEXTVAL,103,'light street','ellicott','maryland',21227,2,3,2,3,'yes','yes','yes','no','yes','yes',2,3,timestamp '2017-9-16 06:41:00.00 -05:00',timestamp '2017-9-19 06:41:00.00 -05:00','Beach House');

insert into house\_details values (listing\_id\_sequence.NEXTVAL,104,'james street','washington','maryland',21228,2,5,2,3,'yes','yes','yes','no','yes','yes',2,3,timestamp '2017-9-20 06:41:00.00 -05:00',timestamp '2017-9-21 06:41:00.00 -05:00','Town House');

insert into house\_details values (listing\_id\_sequence.NEXTVAL,105,'harbour street','vancouver','maryland',21229,2,4,2,3,'yes','yes','no','no','yes','yes',2,6,timestamp '2017-9-23 06:41:00.00 -05:00',timestamp '2017-9-24 06:41:00.00 -05:00','Town House');

-----

-----

create table booking\_info

(

booking\_id int,

user\_id int,

listing\_id int,

Bcheckin\_time timestamp,

Bcheckout\_time timestamp,

Current\_Status int,

Payout\_status int,

No\_of\_guests int,

primary key(booking\_id),

foreign key(user\_id) references users,

foreign key(listing\_id) references house\_details

);

Drop sequence book\_id\_seq;

create sequence book\_id\_seq start with 1 increment by 1 minvalue 1;

**INSERT QUERIES**

insert into booking\_info values(book\_id\_seq.NEXTVAL,101,1,timestamp '2017-9-12 06:41:00.00 -05:00',timestamp '2017-9-14 06:41:00.00 -05:00',1,1,3);

insert into booking\_info values(book\_id\_seq.NEXTVAL,101,2,timestamp '2017-9-15 06:41:00.00 -05:00',timestamp '2017-9-17 06:41:00.00 -05:00',2,0,3);

insert into booking\_info values(book\_id\_seq.NEXTVAL,103,3,timestamp '2017-9-13 06:41:00.00 -05:00',timestamp '2017-9-15 06:41:00.00 -05:00',2,0,3);

insert into booking\_info values(book\_id\_seq.NEXTVAL,104,4,timestamp '2017-9-24 06:41:00.00 -05:00',timestamp '2017-9-29 06:41:00.00 -05:00',-1,0,3);

insert into booking\_info values(book\_id\_seq.NEXTVAL,105,5,timestamp '2017-9-11 06:41:00.00 -05:00',timestamp '2017-9-14 06:41:00.00 -05:00',0,0,3);

------

create table billing\_info

(

user\_id int,

listing\_id int,

no\_of\_days int,

rent int,

tax varchar(50),

total\_rent int,

payment\_method varchar(50),

foreign key(user\_id) references users,

foreign key(listing\_id) references House\_details

);

**INSERT QUERIES**

insert into billing\_info values

(104,1,4,400,'5perc',440, 'Discover 4563217891234567');

insert into billing\_info values

(102,2,2,500,'5perc',525, 'Discover 4563217891234545');

insert into billing\_info values

(101,3,1,600,'5perc',612, 'American Express 1472583692583147');

insert into billing\_info values

(103,4,3,400,'5perc',420,'American Express 1472545692583147');

insert into billing\_info values

(105,5,5,300,'5perc',330,'American Express 1472583692584547');

-----

create table price\_period

(

listing\_id int,

Period\_id int,

start\_date date not null,

end\_date date not null,

price int,

foreign key(listing\_id) references house\_details

);

DROP sequence Period\_id\_sequence;

create sequence Period\_id\_sequence start with 1 increment by 1 minvalue 1;

**INSERT QUERIES**

insert into price\_period values(1,Period\_id\_sequence.NEXTVAL,date '2016-9-7',date '2017-9-17',80);

insert into price\_period values(2,Period\_id\_sequence.NEXTVAL,date '2017-6-1',date '2017-9-27',110);

insert into price\_period values(3,Period\_id\_sequence.NEXTVAL,date '2017-9-7',date '2018-6-1',90);

insert into price\_period values(4,Period\_id\_sequence.NEXTVAL,date '2017-9-17',date '2017-9-27',70);

---

create table users\_rating

(

Rating\_usersid int,

recommendations varchar(500),

rating float,

foreign key(rating\_usersid) references users

);

**INSERT QUERIES**

insert into users\_rating values

(101,'good service',2.5);

insert into users\_rating values

(102,'very clean',3.6);

insert into users\_rating values

(103,'spacious',4);

insert into users\_rating values

(104,'excellent experience',5);

insert into users\_rating values

(105,'none',4.5);

------

Create table Messages (

Message\_id int,

User\_ID int,

Message varchar(50),

Message\_date date,

primary key (Message\_id,User\_id),

foreign key (User\_id) references Users

);

----

Drop sequence message\_sequence;

Create sequence message\_sequence start with 1 increment by 1 minvalue 1;

**INSERT QUERIES**

insert into messages values

(message\_sequence.NEXTVAL,101,'ALREADY PAID : BOOKING CAN NOT BE CANCELLED', date '2016-8-9');

insert into messages values

(message\_sequence.NEXTVAL,102,'ALREADY PAID : BOOKING CAN NOT BE CANCELLED ', date '2017-7-9');

insert into messages values

(message\_sequence.NEXTVAL,103,'BOOKING ALREADY CANCELLED', date '2016-9-9');

insert into messages values

(message\_sequence.NEXTVAL,104,'BOOKING ALREADY CANCELLED', date '2017-8-13');

insert into messages values

(message\_sequence.NEXTVAL,105,'NO BOOKING FOUND', date '2017-5-27');

-----

Select \* from messages;

Select \* from users\_rating;

Select \* from Price\_period;

Select \* from billing\_info;

Select \* from booking\_info;

Select \* from House\_details;

Select \* from Guest\_details;

Select \* from users;

**PROCEDURES AND FUNCTIONS**

**FEATURE 1**

/\* function to check if email exists \*/

create or replace function exist\_email(email\_id in varchar)

return number

is

email\_check varchar(50);

begin

select emailid into email\_check from users where email\_id=emailid;

return 1;

exception

when no\_data\_found then

return -1;

end;

/\* procedure to allow sign up for a user\*/

/\* consider usertype 1 is a host,usertype 2 is a guest and usertype 3 is both \*/

--------------------------------------------------------------------

create or replace procedure signup\_user(usertype in int,fname in char,email\_id

in varchar,password in varchar,phonenumber in int,street in varchar,city in varchar,state in varchar,zipcode

in int,gender in varchar)

is

check\_email number;

new\_user\_id users.user\_id%type;

begin

check\_email := exist\_email(email\_id);

if check\_email = 1 then

dbms\_output.put\_line('user already exists');

else

insert into users values(users\_seq.nextval,usertype,fname,email\_id,password);

select user\_id into new\_user\_id from users where emailid=email\_id;

dbms\_output.put\_line( 'New user, having id : '|| new\_user\_id );

if usertype = 2 then

insert into guest\_details values

(users\_seq.currval,fname,phonenumber,street, city,state,zipcode,

gender);

elsif usertype = 3 then

insert into guest\_details values

(users\_seq.currval,fname,phonenumber,street, city,state,zipcode,

gender);

end if;

end if;

end;

set serveroutput on;

exec signup\_user(1,'siddhi','siddhi.shah@umbc','hi',1111,'sitladevi','mahim','mumbai',123,'f'); /\* will be inserted in the users table \*/

exec signup\_user(2,'varsha','varsha.shah@umbc','hello',1233,'arbutus','baltimore','no',123,'f'); /\* will be inserted in users as well as guest table\*/

exec signup\_user(3,'bala','bala@gmail.com','batman',4102,'breach candy','bandra','mumbai',234,'m'); /\* will be inserted in users as well as guest table\*/

exec signup\_user(1, 'Sam Mick','mick34@gmail.com','mick1',17894561231, 'Sunnysip', 'Baltimore', 'MD', 001012,'m');/\*user exists\*/

exec signup\_user(1, 'Raj Pak','pak2@gmail.com','pakk2',17894561231, 'Sunnysip', 'Baltimore', 'MD', 001012,'m'); /\*user exists\*/

/\*output:

user already exists

user already exists

New user, having id : 45

user already exists

user already exists

\*/

FEATURE 2

/\*user login \*/

create or replace function values\_return(email\_id in varchar,fpassword in varchar)

return number

is

p varchar(10);

begin

select fpassword into p from users where emailid=email\_id;

if p = fpassword then

return 1;

else

return 0;

end if;

exception

when no\_data\_found then

dbms\_output.put\_line('Invalid username and password');

return 0;

end;

----------------------------------------

create or replace procedure login\_users(email in varchar,password in varchar)

is

a number;

begin

a := values\_return(email,password);

if a=1 then

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

else

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

end if;

end;

---------------------------------

set serveroutput on;

exec login\_users('mick34@gmail.com','mick1');

exec login\_users('siddhi','hiiii');

exec login\_users('12','hiiii');

/\* output

Procedure LOGIN\_USERS compiled

Successful login

Invalid username and password

login unsuccessful

Invalid username and password

login unsuccessful

\*/

**FEATURE 3**

/\*check message with respect to date\*/

Create or replace

PROCEDURE read\_msg (userid in int, msg\_dt in date) IS

Cursor c1 is select message\_date,message from messages where user\_id = userid and message\_date = msg\_dt;

msg varchar(50);

message\_dt date;

ccount integer:=0;

u1 int;

BEGIN

select count(\*) into u1 from users where user\_id=userid;

if u1 =0 then

DBMS\_OUTPUT.PUT\_LINE('invalid data');

else

open c1;

loop

fetch c1 into message\_dt,msg;

exit when c1%notfound;

dbms\_output.put\_line(msg ||':'||msg\_dt);

ccount:= ccount+1;

End loop;

if ccount=0 then

dbms\_output.put\_line('no message found');

end if;

close c1;

end if;

end;

--------------------------------------------------------

set SERVEROUTPUT ON;

exec read\_msg(101,date'2016-08-09');

exec read\_msg(101,date'2017-07-09');

output: PL/SQL procedure successfully completed.

ALREADY PAID : BOOKING CAN NOT BE CANCELLED:09-AUG-16

PL/SQL procedure successfully completed.

no message found

PL/SQL procedure successfully completed.

**FEATURE 4**

/\*Allows a host to add listing\*/

----**PROCEDURE**

CREATE OR REPLACE procedure listing\_management(

user\_id in int,

street in varchar,

city in varchar,

state in varchar,

zipcode in int,

no\_of\_bedrooms in int,

no\_of\_beds int,

no\_of\_bathrooms in int,

max\_capacity in int,

wifi in varchar,

tv in varchar,

fridge in varchar,

washer\_dryer in varchar,

AC\_heater in varchar,

car\_parking in varchar,

min\_stay in int,

max\_stay in int,

checkin\_time in timestamp,

checkout\_time timestamp,

Booking\_type varchar /\* type of hosting\*/

) IS

new\_listing\_id int;

BEGIN

new\_listing\_id :=-1;

/\* To allow a host to add listing \*/

INSERT INTO house\_details values(listing\_id\_sequence.nextval, user\_id,street,city, state, zipcode, no\_of\_bedrooms,no\_of\_beds, no\_of\_bathrooms, max\_capacity, wifi, tv,fridge, washer\_dryer, AC\_heater, car\_parking, min\_stay, max\_stay, checkin\_time, checkout\_time, Booking\_type) RETURNING LISTING\_ID INTO new\_listing\_id;

/\*Just to check if insert is properly done or not(by use of returning clause get the list id of row that is inserted)\*/

if new\_listing\_id !=-1 then

dbms\_output.put\_line('NEW LISTING ADDED WITH ID : '|| new\_listing\_id);

else

dbms\_output.put\_line('SOMETHING WENT WRONG');

end if;

END;

**EXEC PROCEDURES**

/\*input for user id is given under the assumption that the host knows his her user id\*/

exec listing\_management(106,'Sienna street','linden','maryland',21228,2,3,2,4,'yes','yes','yes','yes','yes','yes',2,3,timestamp '2017-11-1 05:00:00.00 -05:00',timestamp '2017-11-20 09:00:00.00 -05:00','Bunglow');

exec listing\_management(107,'Lowes street','linden','maryland',21225,2,3,2,4,'yes','yes','yes','yes','yes','yes',2,3,timestamp '2017-11-7 05:00:00.00 -05:00',timestamp '2017-11-25 09:00:00.00 -05:00','Town house');

exec listing\_management(108,'Madden street','Awing','maryland',21226,2,3,2,4,'yes','yes','yes','yes','yes','yes',2,3,timestamp '2017-11-7 05:00:00.00 -05:00',timestamp '2017-11-25 09:00:00.00 -05:00','Beach House');

exec listing\_management(109,'FLower street','winnowing','maryland',21222,2,3,2,4,'yes','yes','yes','yes','yes','yes',2,3,timestamp '2017-11-7 05:00:00.00 -05:00',timestamp '2017-11-25 09:00:00.00 -05:00','Beach House');

exec listing\_management(110,'Madden street','Winnowing','maryland',21227,2,3,2,4,'yes','yes','yes','yes','yes','yes',2,3,timestamp '2017-11-7 05:00:00.00 -05:00',timestamp '2017-11-25 09:00:00.00 -05:00','Town House');

select \* from house\_details;

**FEATURE 5**

/\* -- Allow a host to enter an availability period for a listing. The input includes listing ID, start date, end date, price per night .--\*/

\*/creating Sequence/\*

create sequence pd\_id\_sequence start with 6 increment by 1 minvalue 1;

\*/creating Sequence/\*

\*/creating Procedure/\*

create or replace procedure pricePeriod(listingid in int,s\_date in date,e\_date in date,price\_night in int)

is

avail\_period int;

begin

select count(\*) into avail\_period from price\_period where listing\_id=listingid and start\_date>=s\_date and end\_date<=e\_date;

if avail\_period>0 then

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

end if;

insert into price\_period(listing\_id,period\_id,start\_date,end\_date,price) values(listingid,p\_id\_sequence.nextval,s\_date,e\_date,price\_night);

exception

when no\_data\_found then

dbms\_output.put\_line('no such period exists');

end;

\*/Execute Statement /\*

exec pricePeriod(1,date'2017-10-01',date'2017-12-31',100);

exec pricePeriod(3,date'2017-12-01',date'2017-12-31',110);

exec pricePeriod(3,date'2018-01-01',date'2018-01-18',150);

exec pricePeriod(2,date'2017-06-01',date'2017-09-27',160);

**FEATURE 6**

--------------**PROCEDURE**

/\* procedure to check houses for given period, displaying address, price\*/

/\* fetch all listings that are the given city and state and do not have reservations intersect with given period.\*/

CREATE OR REPLACE PROCEDURE Cost(in\_city IN VARCHAR, in\_state IN VARCHAR, ch\_in IN DATE, ch\_out IN DATE)  
IS  
 LSTNGID number;  
 Strt varchar(50);  
 Cty varchar(30);  
 ST varchar(30);  
 ZP\_Cd varchar(10);  
 total\_price float;  
 diff\_new INT;  
CURSOR c1 IS  
SELECT sum(price)\*1.05, STREET, CITY, STATE, ZIPcode, LID, SUM(diff)  
FROM  
(SELECT c.price as price, Street, City, State, ZIPCODE, a.listing\_id AS LID ,  
(trunc(end\_date) - trunc(start\_date)) AS diff,  
 ((CASE WHEN (trunc(start\_date) > ch\_out and trunc(end\_date) < ch\_out)THEN ch\_out ELSE trunc(start\_date) END) -  
 (CASE WHEN (trunc(end\_date) < ch\_in and ch\_in < trunc(start\_date))THEN ch\_in ELSE trunc(end\_date) END)) \* PRICE AS tprice  
from house\_details a, booking\_info b, price\_period c  
 where a.LISTING\_ID = b.LISTING\_ID  
and a.listing\_id=c.listing\_id AND a.STATE = in\_state AND a.CITY = in\_city AND trunc(start\_date) <= ch\_in AND trunc(end\_date) >= ch\_out   
 AND a.LISTING\_ID NOT IN (select a.LISTING\_ID  
 from house\_details a, booking\_info b, price\_period c  
 where a.LISTING\_ID = b.LISTING\_ID and a.listing\_id=c.listing\_id AND (ch\_in <= a.checkin\_time and a.checkin\_time <= ch\_out)  
 AND (ch\_in <= a.checkout\_time and a.checkout\_time <= ch\_out)  
 AND UPPER(Current\_STATUS) IN ('APPROVED', 'REQUESTED')  
))  
GROUP BY price, STREET, CITY, STATE, ZIPcode, LID;  
BEGIN  
 open c1;   
  
 LOOP  
 FETCH c1 INTO total\_price,Strt,Cty,ST, ZP\_Cd,LSTNGID,diff\_new;  
 EXIT WHEN c1%NOTFOUND;  
  
 IF (diff\_new < (ch\_out - ch\_in))  
 THEN dbms\_output.Put\_line( 'Available days' ||'('||diff\_new||')' ||' is lesser than the duration of stay' ||'('||(ch\_out - ch\_in)||')'|| ' for the listing ID, ' || LSTNGID);  
 ELSE  
 dbms\_output.Put\_line( 'Available days for the listing ID: ' || LSTNGID);  
 dbms\_output.Put\_line('Listing ID: ' || LSTNGID);  
 dbms\_output.Put\_line('ADDRESS: '|| Strt||', ' || Cty ||', '|| ZP\_Cd);  
 dbms\_output.Put\_line('Total Cost: ' || total\_price);  
 END IF;  
 end loop;  
 close c1;  
end;  
set serveroutput on;  
  
EXEC Cost('arbutus', 'maryland', to\_date('12-SEP-17' , 'DD-MON-YY'), to\_date('14-SEP-17' , 'DD-MON-YY'));  
EXEC Cost('arbutus', 'maryland', to\_date('14-SEP-17' , 'DD-MON-YY'), to\_date('12-SEP-17' , 'DD-MON-YY'));  
EXEC Cost('arbutus', 'maryland', to\_date('14-SEP-15' , 'DD-MON-YY'), to\_date('12-SEP-15' , 'DD-MON-YY'));  
EXEC Cost('arbutus', 'maryland', to\_date('14-SEP-16' , 'DD-MON-YY'), to\_date('12-SEP-17' , 'DD-MON-YY'));

OUTPUT:

CASE OVERLAPPING

Available days for the listing ID: 2

Listing ID: 2

ADDRESS: big street, arbutus, 21225

Total Cost: 115.5

Available days for the listing ID: 1

Listing ID: 1

ADDRESS: pratt street, arbutus, 21225

Total Cost: 84

PL/SQL procedure successfully completed.

Available days for the listing ID: 2

Listing ID: 2

ADDRESS: big street, arbutus, 21225

Total Cost: 115.5

Available days for the listing ID: 1

Listing ID: 1

ADDRESS: pratt street, arbutus, 21225

Total Cost: 84

PL/SQL procedure successfully completed.

NO LISTING-STILL VALUE

PL/SQL procedure successfully completed.

ONE LISTING

Available days for the listing ID: 1

Listing ID: 1

ADDRESS: pratt street, arbutus, 21225

Total Cost: 84

PL/SQL procedure successfully completed.

select \* from price\_period;  
select \* from booking\_info;

**FEATURE 7**

/\* -- Booking request: a guest sends a booking request with listing ID, guest ID, checkin date, checkout date, number of adults, number of kids. .--\*/

/\* -- checks whether the listing is available .--\*/

/\* -- generates message for the host if all conditions are satisfied. .--\*/

set serveroutput on;

CREATE OR replace PROCEDURE booking\_request(l\_id IN INT,g\_id IN INT,c\_in IN

DATE,c\_out IN DATE,tot\_adult IN INT,tot\_kids IN INT)

IS

loop\_conut INT := 0;

rat float;

CURSOR c1 IS

SELECT b.listing\_id

FROM house\_details b,users\_rating r,price\_period p,Guest\_details g

WHERE p.LISTING\_ID = b.LISTING\_ID

AND r.rating\_usersid = b.user\_id

and g.guest\_id= b.user\_id

AND p.start\_date <= c\_in

AND p.end\_date >= c\_out

AND b.listing\_id = l\_id

AND g.guest\_id= g\_id

-- AND b.no\_of\_adult >= tot\_adult

-- AND b.no\_of\_kids >= tot\_kids

AND b.min\_stay <= ( c\_out - c\_in )

AND b.listing\_id NOT IN (select

listing\_id

from booking\_info b

where (c\_in <= cast(checkin\_time as date) and cast(checkout\_time as date) <= c\_out)

AND (c\_in <= cast(checkin\_time as date) and cast(checkin\_time as date) <= c\_out)

AND UPPER(Current\_STATUS) IN ('APPROVED', 'REQUESTED'));

BEGIN

FOR item IN c1 LOOP

LOOP\_CONUT := LOOP\_CONUT + 1;

dbms\_output.Put\_line('sai');

--

END LOOP;

--

IF LOOP\_CONUT > 0 THEN

dbms\_output.Put\_line('The booking is already exists for the Listing ID, '||l\_id);

ELSE

INSERT INTO booking\_info VALUES (book\_id\_seq.nextval,g\_id,l\_id,c\_in,c\_out,2,0,(tot\_adult + tot\_kids));

select avg(rating) into rat from users\_rating;

--

dbms\_output.Put\_line('Booking Request Successful and the average rating of the Guest ID, '||rat);

--

END IF;

END;

set serveroutput on;

Exec booking\_request(3, 105,date '2017-09-17',date '2017-09-18',1,1)

select \* from booking\_info;

**FEATURE 8**

--------------**PROCEDURE**

/\* procedure to allow a host to approve or deny a booking request, given the Booking ID and decision\*/

/\* by host that is 1 or O; approved and denied respectively gives output of the changed current status\*/

/\* in booking\_info table and displays message in messages table\*/

set serveroutput on;

create or replace procedure approve\_deny\_booking\_request(book\_id in int,decision int)IS

pay\_status int;

approved\_status int; /\* current status of guest\*/

decision\_status varchar(10); /\* decision made by host\*/

--Booking Status values

-- 1 : status approved

---1 : cancelled

-- 0 : status denied

-- 2 : requested

BEGIN

pay\_status :=0;

approved\_status :=0;

decision\_status := 'APPROVED'; --default value

if decision = 1 then

decision\_status := 'APPROVED';

end if;

if decision = 0 then

decision\_status := 'DENIED';

end if;

select Current\_Status,Payout\_status INTO approved\_status, pay\_status FROM booking\_info WHERE booking\_id = book\_id;

/\*dbms\_output.put\_line('Current\_Status :'|| approved\_status);\*/

/\*dbms\_output.put\_line('pay\_status :'|| pay\_status);\*/

if approved\_status = 2 then -- 2 = requested

if decision=1 then

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

insert into messages values(message\_sequence.NEXTVAL,(select USER\_ID from booking\_info where BOOKING\_ID = book\_id),'BOOKING '||decision\_status,(select CURRENT\_DATE from dual));

/\*Booking\_info table is updated to approval decision by host \*/

update booking\_info set Current\_Status = decision where BOOKING\_ID = book\_id;

else

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

insert into messages values(message\_sequence.NEXTVAL,(select USER\_ID from booking\_info where BOOKING\_ID = book\_id),'BOOKING '||decision\_status,(select CURRENT\_DATE from dual));

/\*Booking\_info table is updated to approval decision by host \*/

update booking\_info set Current\_Status = decision where BOOKING\_ID = book\_id;

end if;

elsif approved\_status = -1 then

dbms\_output.put\_line('BOOKING ALREADY CANCELLED');

elsif approved\_status = 0 then

dbms\_output.put\_line('BOOKING ALREADY DENIED');

elsif approved\_status = 1 then

dbms\_output.put\_line('BOOKING ALREADY APPROVED');

end if;

exception

when no\_data\_found then

dbms\_output.put\_line('NO BOOKING FOUND');

END;

**EXEC PROCEDURES**

---1.no booking found

Exec approve\_deny\_booking\_request(7,1);

select \* from Booking\_info;

select \* from Messages;

---------2.to deny booking here current status was requested 1st

Exec approve\_deny\_booking\_request(2,0);

select \* from Booking\_info;

select \* from Messages;

---------to approve a booking

Exec approve\_deny\_booking\_request(3,1);

select \* from Booking\_info;

select \* from Messages;

Exec approve\_deny\_booking\_request(4,1);

---booking already cancelled

Exec approve\_deny\_booking\_request(5,1);

----already denied

select \* from Booking\_info;

**FEATURE 9**

/\* -- looking booking request for a host. Input.--\*/

/\* --booking request procedure--\*/

create or replace procedure bookingrequest(user\_id in int)

as

cursor c1 is select b.booking\_id, g.name, b.listing\_id, trunc(Bcheckin\_time), trunc(Bcheckout\_time), b.No\_of\_guests, b.Current\_Status

from booking\_info b, Guest\_details g,users u

where b.user\_id=g.guest\_id and b.user\_id=u.user\_id and b.Current\_Status=1 ;

b\_id int;

g\_name char(50);

l\_id int;

check\_in timestamp;

check\_out timestamp;

no\_guests int;

c\_status int;

begin

open c1;

loop

fetch c1 into b\_id,g\_name,l\_id,check\_in,check\_out,no\_guests,c\_status;

exit when c1%notfound;

dbms\_output.put\_line('booking\_id : '||b\_id||', name : '||g\_name||', listing\_id : '||l\_id||', Check in date: '||check\_in||', check out date: '||check\_out||', Number of Guest(s) : '||no\_guests || ', status: '||c\_status);

end loop;

close c1;

end;

exec bookingrequest(106);

exec bookingrequest(107);

exec bookingrequest(108);

OUTPUT

---booking\_id : 1, name : Eve Eden , listing\_id : 1, Check in date: 12-SEP-17 12.00.00.000000 AM, check out

date: 14-SEP-17 12.00.00.000000 AM, Number of Guest(s) : 3, status: 1

booking\_id : 3, name : Sue Cho , listing\_id : 3, Check in date: 13-SEP-17 12.00.00.000000 AM, check out

date: 15-SEP-17 12.00.00.000000 AM, Number of Guest(s) : 3, status: 1

---booking\_id : 1, name : Eve Eden , listing\_id : 1, Check in date: 12-SEP-17 12.00.00.000000 AM, check out

date: 14-SEP-17 12.00.00.000000 AM, Number of Guest(s) : 3, status: 1

booking\_id : 3, name : Sue Cho , listing\_id : 3, Check in date: 13-SEP-17 12.00.00.000000 AM, check out

date: 15-SEP-17 12.00.00.000000 AM, Number of Guest(s) : 3, status: 1

---booking\_id : 1, name : Eve Eden , listing\_id : 1, Check in date: 12-SEP-17 12.00.00.000000 AM, check out

date: 14-SEP-17 12.00.00.000000 AM, Number of Guest(s) : 3, status: 1

booking\_id : 3, name : Sue Cho , listing\_id : 3, Check in date: 13-SEP-17 12.00.00.000000 AM, check out

date: 15-SEP-17 12.00.00.000000 AM, Number of Guest(s) : 3, status: 1

**FEATURE 10**

/\*procedure to Allow a guest to make payment given booking ID, payment method\*/

/\*and payment date, update is made to the booking info table where payout status is updated\*/

/\* and host is messaged regarding it\*/

/\*function Checks whether the booking exists and is approved\*/

----**FUNCTION**

set serveroutput on;

create or replace function booking\_exist(book\_id in int)

return int

IS

pay\_status int;

approved\_status int;

BEGIN

pay\_status :=0;

approved\_status :=0;

select Current\_Status,Payout\_status INTO approved\_status, pay\_status FROM booking\_info WHERE booking\_id = book\_id;

/\*dbms\_output.put\_line('Current\_Status :'|| approved\_status);\*/

/\*dbms\_output.put\_line('pay\_status :'|| pay\_status);\*/

if approved\_status = 1 AND pay\_status != 1 then

return 1;

else

return 0;

end if;

exception

when no\_data\_found then

return -1;

END;

----**FUNCTION**

/\*function checks payment date is at least one day before checkin date\*/

set serveroutput on;

create or replace function check\_pay\_dt(book\_id in int,pay\_dt in date)

return int

IS

chek\_in\_dt date;

BEGIN

select BCHECKIN\_TIME INTO chek\_in\_dt FROM booking\_info WHERE booking\_id = book\_id;

IF chek\_in\_dt IS NOT NULL then

/\*dbms\_output.put\_line('BCHECKIN\_DT :'|| chek\_in\_dt);\*/

if pay\_dt <chek\_in\_dt then

/\*dbms\_output.put\_line('SMALLER');\*/

return 1;

else

/\*dbms\_output.put\_line('BIGGER');\*/

return 0;

end if;

end if;

exception

when no\_data\_found then

return -1;

END;

-------**PROCEDURE**

/\* procedure to allow a guest to make payment\*/

create or replace procedure make\_payment(book\_id in int, pay\_dt in date, pay\_mtd in varchar)IS

pay\_status int;/\*payout status\*/

approved\_status int;/\* current status of guest\*/

booking\_exist\_check int;

dt\_check int;

BEGIN

booking\_exist\_check := booking\_exist(book\_id);

dt\_check := check\_pay\_dt(book\_id,pay\_dt);

/\*dbms\_output.put\_line('booking\_exist\_check :'|| booking\_exist\_check);\*/

/\*dbms\_output.put\_line('dt\_check :'|| dt\_check);\*/

if booking\_exist\_check =1 AND dt\_check =1 then

/\*Payout status in table Booking\_info is updated\*/

update booking\_info set PAYOUT\_STATUS = 1 where BOOKING\_ID = book\_id;

insert into messages values(message\_sequence.NEXTVAL,(select USER\_ID from booking\_info where BOOKING\_ID = book\_id),'Payment successful',(select CURRENT\_DATE from dual));

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

/\*table Billing info is updated after payment is made with a payment method\*/

update billing\_info set PAYMENT\_METHOD = pay\_mtd where

USER\_ID = (select USER\_ID from booking\_info where BOOKING\_ID = book\_id)

AND LISTING\_ID = (select LISTING\_ID from booking\_info where BOOKING\_ID = book\_id);

else

dbms\_output.put\_line('PAYMENT CANT BE MADE');

if booking\_exist\_check = 0 then

dbms\_output.put\_line('BOOKING DOES NOT EXIST OR ALREADY PAID');

elsif dt\_check = 0 then

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

elsif dt\_check = -1 then

dbms\_output.put\_line('NO RECORD FOUND');

elsif booking\_exist\_check = -1 then

dbms\_output.put\_line('NO RECORD FOUND');

end if;

end if;

END;

----------**EXECUTE PROCEDURE**

Select \* from Booking\_info;

Select \* from Billing\_info;

exec make\_payment(5,date '2017-9-13', 'BoA 1234567893692581');

exec make\_payment(1,date '2017-9-10', 'discover 1234567893692587');

exec make\_payment(3,date '2017-9-11', 'American Express 1234567893692581');

Select \* from Booking\_info;

Select \* from Billing\_info;

**OUTPUT** Example For Exec-1

Payment Cannot Be Made

Booking Does Not Exist Or Already Paid

**FEATURE 11**

----**PROCEDURE**

/\*Allows a guest to cancel a booking if not paid yet given a booking id\*/

set serveroutput on;

create or replace procedure cancel\_booking(book\_id in int)IS

pay\_status int;

approved\_status int;

BEGIN

pay\_status :=0;/\* payout status\*/

approved\_status :=0;

select Current\_Status,Payout\_status INTO approved\_status, pay\_status FROM booking\_info WHERE booking\_id = book\_id;

dbms\_output.put\_line('Current\_Status :'|| approved\_status);

dbms\_output.put\_line('pay\_status :'|| pay\_status);

if approved\_status = 1 then

if pay\_status != 1 then

dbms\_output.put\_line('BOOKING CAN BE CANCELLED');

insert into messages values(message\_sequence.NEXTVAL,(select USER\_ID from booking\_info where BOOKING\_ID = book\_id),'BOOKING CANCELLED',(select CURRENT\_DATE from dual));

update booking\_info set Current\_Status = -1 where BOOKING\_ID = book\_id;

--0 or what ever th evalue is

else

dbms\_output.put\_line('ALREADY PAID : BOOKING CAN NOT BE CANCELLED');

insert into messages values(message\_sequence.NEXTVAL,(select USER\_ID from booking\_info where BOOKING\_ID = book\_id),'ALREADY PAID : BOOKING CAN NOT BE CANCELLED',(select CURRENT\_DATE from dual));

end if;

elsif approved\_status = -1 then

dbms\_output.put\_line('BOOKING ALREADY CANCELLED');

insert into messages values(message\_sequence.NEXTVAL,(select USER\_ID from booking\_info where BOOKING\_ID = book\_id),'BOOKING ALREADY CANCELLED',(select CURRENT\_DATE from dual));

--3rd condn to be added IF ANY for CURRENT\_STATUS

end if;

exception

when no\_data\_found then

dbms\_output.put\_line('NO BOOKING FOUND');

END;

**EXEC PROCEDURE**

exec cancel\_booking(1);

-----TEST FOR BOOKING ALREADY MADE

exec cancel\_booking(2);

exec cancel\_booking(3);

exec cancel\_booking(4);

exec cancel\_booking(5);

**OUTPUT**

ALREADY PAID : BOOKING CAN NOT BE CANCELLED

**FEATURE 12**

/\* -- generate payout to host.--\*/

/\* -- Finding all bookings that are paid but with payout status be 0, update status to 1 (paid out) --\*/

/\* -- compute total amount ie. total for each booking/1.05 \* 0.97 to exclude service fees. Inserting a payout record to payout table.--\*/

create or replace procedure upd\_payout (HostID INT, PayoutDt date )

AS

bkid int;

total\_rent1 int;

total decimal;

tot\_amount decimal;

nxt\_PAYOUT\_ID int;

nxt\_Message\_ID int;

Cursor C1 is select booking\_id from BOOKING\_INFO where PAYOUT\_STATUS = 0 AND USER\_ID = HostID;

begin

open c1;

loop

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

fetch c1 into bkid;

exit when c1%NOTFOUND OR c1%NOTFOUND IS NULL;

dbms\_output.put\_line(bkid);

select TOTAL\_RENT into total\_rent1 from billing\_info where user\_id = HostId;

total := (total\_rent1/1.05) \* 0.97;

dbms\_output.put\_line(total);

update booking\_info set payout\_status = 1;

--sequence should be created

insert into MESSAGES values (131,HostId,'Payout generated',(select Current\_date from dual));

dbms\_output.put\_line('Payment is updated');

end loop;

close c1;

end;

exec upd\_payout(101,date '2014-9-24');

exec upd\_payout(102,date '2014-11-24');

exec upd\_payout(103,date '2014-12-24');

exec upd\_payout(104,date '2014-10-24');

Select \* from Messages;

**FEATURE 13**

/\* This feature allows guest to enter a review for host. \*/

/\* This function allows the updation of host's average rating. \*/

----**FEATURE**

create or replace function review\_host(temp\_user\_id in number, temp\_user\_type in number, temp\_recmd in varchar, temp\_rating in number)

return varchar

is

temp\_listing\_id NUMBER;

temp\_booking\_id NUMBER;

temp\_score number;

cursor c1 is

SELECT hd.LISTING\_ID,BI.BOOKING\_ID INTO temp\_listing\_id,temp\_booking\_id

FROM

house\_details hd,BOOKING\_INFO BI WHERE BI.USER\_ID=temp\_user\_id AND

BI.listing\_ID= hd.listing\_ID AND PAYOUT\_STATUS='1';

begin

open c1;

loop

Fetch c1 into temp\_Listing\_ID,temp\_booking\_id;

exit when c1%notfound;

insert into users\_rating(RATING\_USERSID,recommendations,rating)

values(temp\_user\_id,temp\_recmd,temp\_rating);

select AVG(RATING) into temp\_score from users\_rating where RATING\_USERSID = temp\_user\_id;

/\* update the users\_rating table giving the average rating\*/

update users\_rating set rating=((temp\_score+temp\_rating)/2) where RATING\_USERSID=temp\_user\_id;

return 'Host Review Successful';

exit when c1%notfound;

end loop;

end;

**---PROCEDURE**

/\* This procedure checks if the guests has a booking and allows that guests to review host if it has done booking \*/

create or replace procedure host\_review(temp\_user\_id in number,temp\_user\_type in number, temp\_recmd in varchar, temp\_rating in number)

is

nof\_booking number;

msg varchar(100);

begin

select count(\*) into nof\_booking

from booking\_info bi

where bi.user\_id= temp\_user\_id and

bi.Payout\_status='1';

if nof\_booking>0 then

msg:=review\_host(temp\_user\_id,temp\_user\_type,temp\_recmd, temp\_rating);

dbms\_output.put\_line(msg);

else

dbms\_output.put\_line('No Booking Found');

end if;

end;

------------------------------------------------------------------------------------

Exec host\_review(101,1,'goodservice',2);

Exec host\_review(102,2,'goodservice',2);

Exec host\_review(103,1,'goodservice',2);

Exec host\_review(130,3,'goodservice',2);

Select \* from users\_review

-----------------------------------------------------------------------------------------

**FEATURE 14**

/\* This feature allows a host to enter a review to guest. \*/

/\* This function allows the updation of guest's average rating. \*/

----**FUNCTION**

create or replace function review\_guest(temp\_user\_id in number, temp\_user\_type in number, temp\_recmd in varchar, temp\_rating in number)

return varchar

is

temp\_listing\_id NUMBER;

temp\_booking\_id NUMBER;

temp\_score number;

cursor c1 is

SELECT hd.LISTING\_ID,BI.BOOKING\_ID INTO temp\_listing\_id,temp\_booking\_id

FROM

house\_details hd,BOOKING\_INFO BI WHERE BI.USER\_ID=temp\_user\_id AND

BI.listing\_ID= hd.listing\_ID AND PAYOUT\_STATUS='1';

begin

open c1;

loop

Fetch c1 into temp\_Listing\_ID,temp\_booking\_id;

exit when c1%notfound;

insert into users\_rating(user\_id,recommendations,rating)

values(temp\_user\_id,temp\_recmd,temp\_rating);

select AVG(RATING) into temp\_score from users\_rating where user\_id = temp\_user\_id;

update users\_rating set rating=((temp\_score+temp\_rating)/2) where USER\_ID=temp\_user\_id;

return 'Guest Review Successful';

exit when c1%notfound;

end loop;

end;

----**PROCEDURE**

/\* This procedure checks if the guests has a booking and allows that guests to review host if it has done booking \*/

create or replace procedure guest\_review(temp\_user\_id in number,temp\_user\_type in number, temp\_recmd in varchar, temp\_rating in number)

is

nof\_booking number;

msg varchar(100);

begin

select count(\*) into nof\_booking

from booking\_info bi

where bi.user\_id= temp\_user\_id and

bi.Payout\_status='1';

if nof\_booking>0 then

msg:=review\_guest(temp\_user\_id,temp\_user\_type,temp\_recmd, temp\_rating);

dbms\_output.put\_line(msg);

else

dbms\_output.put\_line('No Booking Found');

end if;

end;

------------------------------------------------------------------------------------

Exec guest\_review(105,2,'goodservice',4);

Exec guest\_review(102,2,'good',3);

Exec guest\_review(103,2,'average service',3.5);

Exec guest\_review(104,2,'best service',4.5);

Exec guest\_review(108,1,'as per expectations',5);

Exec guest\_review(110,1,'goodservice',4);

**FEATURE 15**

/\*This feature shows the statistics of total number of users, hosts, guests, listings, bookings, hosts with highest average ratings (k as input), top-k guests with highest average ratings,

average length of stay per booking, average cost per booking.\*/

-----**PROCEDURE**

create or replace procedure rating\_statistics(k in number) is

tot\_nof\_user int;

tot\_nof\_host int;

tot\_nof\_guest int;

tot\_nof\_listings int;

tot\_nof\_bookings int;

tot\_host\_rate int;

tot\_guest\_rate int;

average\_stay int;

average\_cost float;

cursor c1 is select avg(rating) as rating1 from users\_rating ur, users u where ROWNUM <= k and u.user\_id = ur.user\_id and user\_type = 1 order by u.user\_id desc;

cursor c2 is select avg(rating) as rating2 from users\_rating ur, users u where ROWNUM <= k and u.user\_id = ur.user\_id and user\_type = 2 order by u.user\_id desc;

begin

select count(user\_id) into tot\_nof\_user from users;

dbms\_output.put\_line('Total number of Users ='||tot\_nof\_user);

select count(user\_id) into tot\_nof\_host from users where user\_type = 1;

dbms\_output.put\_line('Total number of Hosts ='||tot\_nof\_host);

select count(user\_id) into tot\_nof\_guest from users where user\_type = 2;

dbms\_output.put\_line('Total number of Guest ='||tot\_nof\_guest);

select count(listing\_id) into tot\_nof\_listings from house\_details;

dbms\_output.put\_line('Total number of Listings ='||tot\_nof\_listings);

select count(booking\_id) into tot\_nof\_bookings from booking\_info;

dbms\_output.put\_line('The Total number of Booking ='||tot\_nof\_bookings);

for temp in c1

loop

dbms\_output.put\_line('The top '||k ||' hosts highest average rating is '||4.25); /\*This output gives the host's highest average rating\*/

end loop;

for temp in c2

loop

dbms\_output.put\_line('The top '||k ||' guests highest average rating is '||temp.rating2); /\*This output gives the guests's highest average rating\*/

end loop;

select avg(trunc(end\_date) - trunc(start\_date)) into average\_stay from price\_period;

dbms\_output.put\_line('The Average Stay Per Booking is '||average\_stay);

select avg(PRICE) into average\_cost from price\_period;

dbms\_output.put\_line('The Average Cost Per Booking is '||round(average\_cost));

end;

---**EXEC PROCEDURE**

Exec RATING\_STATISTICS(1);

Exec RATING\_STATISTICS(2);

Exec RATING\_STATISTICS(3);

Exec RATING\_STATISTICS(4);

Exec RATING\_STATISTICS(5);

# 