**Team 6**

**Table Script**

/\*Run Separate and First\*/

/\*---------------------------------------------------------------------------------------------------------------------------\*/

/\* Function for assigning cost to rooms based on date \*/

create or replace function price\_from\_date (reservation\_date date, price\_date date, num\_days int)

return integer is price int;

begin

if price\_date between to\_date('04-30','mm-dd') and to\_date('09-01','mm-dd')

then

if reservation\_date <= add\_months(price\_date, -2) then

Price := 270 \* num\_days;

else

Price := 300 \* num\_days;

end if;

else

if reservation\_date <= add\_months(price\_date, -2) then

Price := 180 \* num\_days;

else

Price := 200 \* num\_days;

end if;

end if;

Return price;

end;

/\* Function for assigning cost of services based on type \*/

create or replace function price\_from\_name (s\_name varchar2)

return integer is price int;

begin

if s\_name = 'Restaurant service'

then Price := 20;

Else if s\_name = 'Pay-Per-View'

then Price := 5;

Else if s\_name = 'Laundry service'

then Price := 10;

else price := 0;

end if;

end if;

end if;

Return price;

end;

/\*---------------------------------------------------------------------------------------------------------------------------\*/

Drop table service;

Drop table booking;

Drop table customer;

Drop table room;

Drop table hotel;

Drop sequence service\_seq;

Drop sequence booking\_seq;

Drop sequence customer\_seq;

Drop sequence room\_seq;

Drop sequence hotel\_seq;

Create table hotel (

H\_ID int,

H\_name varchar (20),

H\_address varchar (50),

H\_city varchar (50),

H\_state varchar (50),

H\_phone\_number varchar (10),

H\_status varchar (10),

primary Key(H\_ID)

);

Create table customer (

C\_ID int,

C\_name varchar (20),

C\_Address varchar (50),

C\_Phone number (10),

C\_Creditcard varchar (20),

primary Key(C\_ID)

);

Create table room (

R\_ID int,

H\_ID int,

R\_type varchar(50),

primary key(R\_ID),

foreign key (H\_ID) references hotel (H\_ID)

);

Create table booking (

B\_ID int,

C\_ID int,

R\_ID int,

B\_date date,

Check\_in\_date date,

Check\_out\_date date,

Room\_cost int,

Num\_guests int,

has\_reservation varchar(10),

primary key (B\_ID),

foreign key (C\_ID) references customer (C\_ID),

foreign key (R\_ID) references room (R\_ID)

);

Create table service (

S\_ID int,

B\_ID int,

S\_Name varchar (25),

S\_Price number,

S\_date date,

primary key (S\_ID),

foreign key (B\_ID) references booking (B\_ID)

);

Create Sequence hotel\_seq

Start with 10

Increment by 1;

Create Sequence room\_seq

Start with 10

Increment by 1;

Create Sequence customer\_seq

Start with 10

Increment by 1;

Create Sequence booking\_seq

Start with 10

Increment by 1;

Create Sequence service\_seq

Start with 10

Increment by 1;

**Member Procedures**

***/\*Member 1 Ami Oza\*/***

/\*1. **Add a new hotel: Create a new hotel with appropriate information about the hotel as input parameters\*/**

/\*The following states the inputs\*/

CREATE OR REPLACE PROCEDURE addnewhotel (

new\_h\_id NUMBER,

new\_h\_name VARCHAR2,

new\_h\_address VARCHAR2,

new\_h\_city VARCHAR2,

new\_h\_state VARCHAR2,

new\_h\_phone\_number NUMBER,

new\_h\_status VARCHAR2

)

/\*The following states the the output that will be given when a new hotel is added along with hotel info\*/

AS

BEGIN

dbms\_output.put\_line('New Hotel '

|| new\_h\_name

|| ' added with ID '

|| new\_h\_id);

INSERT INTO hotel VALUES (

new\_h\_id,

new\_h\_name,

new\_h\_address,

new\_h\_city,

new\_h\_state,

new\_h\_phone\_number,

new\_h\_status

);

END;

**/\* 2. Find a hotel: Provide as input the address of the hotel and return its hotel ID\*/**

/\*The following states the inputs\*/

CREATE OR REPLACE PROCEDURE findhotel (

input\_h\_address VARCHAR2

) AS

/\*The following finds the hotels address by looking for the address that matches the input of hotel ID\*/

temp\_h\_id INT;

BEGIN

SELECT

h\_id

INTO

temp\_h\_id

FROM

hotel

WHERE

h\_address = input\_h\_address;

/\*The following states the output that will be triggered through this\*/

dbms\_output.put\_line('Hotel found at '

|| input\_h\_address

|| ' has the Hotel ID '

|| temp\_h\_id);

END;

**/\*3. Sell existing hotel: Sell a hotel by providing its hotel ID. Mark it as sold, do not delete the record.\*/**

/\*The following states the inputs\*/

CREATE OR REPLACE PROCEDURE sellhotel (

given\_h\_id NUMBER

)

IS

/\*The following finds the matching hotel ID and changes the status of the hotel to SOLD\*/

BEGIN

UPDATE hotel

SET

h\_status = 'SOLD'

WHERE

given\_h\_id = h\_id;

/\*The following states the output triggered\*/

dbms\_output.put\_line('Hotel ID '

|| given\_h\_id

|| ' was sold');

END;

**/\*4. Display hotel info: Given a hotel ID, display all information about that hotel\*/**

SET SERVEROUTPUT ON;

/\*The following states the inputs\*/

CREATE OR REPLACE PROCEDURE hotel\_id (

input\_h\_id IN NUMBER

) IS

/\*The following defines the search through the hotel table to find the matching ID to the input\*/

CURSOR idfinder\_cursor IS SELECT

\*

FROM

hotel

WHERE

h\_id = input\_h\_id;

id\_row idfinder\_cursor%rowtype;

/\*The following states the output\*/

BEGIN

OPEN idfinder\_cursor;

dbms\_output.put\_line('Information of Hotels');

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

;

LOOP

/\*The following searches through the table and finds all the hotel information to display\*/

FETCH idfinder\_cursor INTO id\_row;

EXIT WHEN idfinder\_cursor%notfound;

/\*If search is unsuccessful Loop ends\*/

dbms\_output.put\_line(id\_row.h\_id

|| ' '

|| id\_row.h\_name

|| ' '

|| id\_row.h\_address

|| ' '

|| id\_row.h\_city

|| ' '

|| id\_row.h\_state

|| ' '

|| id\_row.h\_phone\_number

|| ' '

|| id\_row.h\_status);

END LOOP;

CLOSE idfinder\_cursor;

END;

**/\* 5. Sold Hotels: Print all sold hotel information. Show hotel ID, location, etc.\*/**

/\*The following states the inputs\*/

CREATE OR REPLACE PROCEDURE showsold\_hotels AS

CURSOR sold\_cursor IS SELECT

\*

FROM

hotel

WHERE

h\_status = 'SOLD';

/\*The following states the outputs\*/

BEGIN

dbms\_output.put\_line('Hotel(s) that have been sold: ');

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

FOR soldinfo IN sold\_cursor LOOP

/\*The following states the outputs in a loop where it states all the hotels found meeting the sold status requirement\*/

dbms\_output.put\_line('H\_ID: '

|| soldinfo.h\_id);

dbms\_output.put\_line('H\_name: '

|| soldinfo.h\_name);

dbms\_output.put\_line('H\_state: '

|| soldinfo.h\_state);

dbms\_output.put\_line('H\_phone\_number: '

|| soldinfo.h\_phone\_number);

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

END LOOP;

END;

**/\* 6. Report Hotels In State: Given a state, display information of all hotels in that particular state.\*/**

/\*The following states the inputs\*/

CREATE OR REPLACE PROCEDURE hotel\_state (

input\_h\_state IN VARCHAR2

) AS

/\*The following defines a cursor to search for the input criteria and find the hotels that apply\*/

CURSOR state\_cursor IS SELECT

h\_id,

h\_name,

h\_state

FROM

hotel

WHERE

h\_state = input\_h\_state;

state\_row state\_cursor%rowtype;

BEGIN

/\*The following states the outputs that will state the first query result and loop til it runs out of query results that apply\*/

dbms\_output.put\_line('Hotels in state of ' || input\_h\_state);

dbms\_output.put\_line('h\_id'

|| ' '

|| 'h\_name');

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

FOR state\_row IN state\_cursor LOOP

dbms\_output.put\_line(state\_row.h\_id

|| ' '

|| state\_row.h\_name);

END LOOP;

/\*The following states if there are no queries that apply to the search\*/

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No hotels in '

|| input\_h\_state);

END;

***/\*Member 2 Greg Hotchkiss\*/***

**/\*7. Make a reservation: Input parameters: Hotel ID, guest’s name, start date, end date, room type, date of reservation, etc. Output: reservation ID (this is called confirmation code in real-life). NOTE: Only one guest per reservation. However, the same guest can make multiple reservations.\*/**

CREATE OR REPLACE PROCEDURE newbooking (

customerid INT,

roomid INT,

bookingdate DATE,

checkin DATE,

checkout DATE,

hasreservation VARCHAR

) AS

bookingid INT;

BEGIN

-- a savepoint in case there is an issue with the insertion

SAVEPOINT pre\_insert;

-- a new booking is added

INSERT INTO booking (

b\_id,

c\_id,

r\_id,

b\_date,

check\_in\_date,

check\_out\_date,

room\_cost,

num\_guests,

has\_reservation

) VALUES (

booking\_seq.NEXTVAL,

customerid,

roomid,

bookingdate,

checkin,

checkout,

price\_from\_date(bookingdate,checkout, to\_number(checkout - checkin)),

1,

hasreservation

);

-- changes are committed

COMMIT;

-- the output parameter is selected based on equality from all the input parameters and the most -- recent insertion

SELECT

b\_id

INTO

bookingid

FROM

booking

WHERE

c\_id = customerid

AND r\_id = roomid

AND b\_date = bookingdate

AND check\_in\_date = checkin

AND check\_out\_date = checkout

AND has\_reservation = hasreservation;

dbms\_output.put\_line('New reservation made for customer id '|| customerid);

dbms\_output.put\_line('Your booking ID is '

|| bookingid);

EXCEPTION

-- if the booking ID can't be found, the table reverts to the savepoint

WHEN no\_data\_found THEN

ROLLBACK TO SAVEPOINT pre\_insert;

dbms\_output.put\_line('The booking could not be completed. Please make sure you have filled

out all information.');

END;

**/\* 8: Find a reservation: Input is guest’s name and date, hotel ID. Output is reservation ID\*/**

CREATE OR REPLACE PROCEDURE findbooking (

cname IN VARCHAR,

bdate IN DATE,

hotelid IN INT

) AS

bookingid INT;

BEGIN

-- the booking ID is selected based on a match with the 3 input parameters

SELECT

booking.b\_id

INTO

bookingid

FROM

booking

INNER JOIN room ON booking.r\_id = room.r\_id

INNER JOIN customer ON booking.c\_id = customer.c\_id

WHERE

booking.b\_date = bdate

AND customer.c\_name = cname

AND room.h\_id = hotelid;

-- the matching booking ID is printed

dbms\_output.put\_line('The booking ID for ' || cName || ' for the reservation made on ' || bDate || ' is: '|| bookingID);

-- if no match is found, the user is told

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No reservation was found');

END;

**/\*9. Cancel a reservation: Input the reservationID and mark the reservation as cancelled (do NOT delete it)\*/**

CREATE OR REPLACE PROCEDURE cancelbooking (

booking\_id IN INT

) IS

resconfirmed booking.has\_reservation%TYPE;

BEGIN

SELECT

has\_reservation

INTO

resconfirmed

FROM

booking

WHERE

booking\_id = booking.b\_id;

UPDATE booking

-- has\_reservation field is changed to no when the input parameter matches a record in the

-- database

SET

has\_reservation = 'NO'

WHERE

booking\_id = booking.b\_id;

dbms\_output.put\_line('Booking ID '

|| booking\_id

|| ' has been cancelled');

EXCEPTION

-- if the input parameter doesn't match any record, the user is notified

WHEN no\_data\_found THEN

dbms\_output.put\_line('The booking id entered does not match any record in the database.');

END;

**/\*10. ShowCancelations: Print all canceled reservations in the hotel management system.**

**Show reservation ID, hotel name, location, guest name, room type, dates\*/**

create or replace procedure showCancellations

AS

CURSOR cancelled\_cursor IS

-- select statement gets all relevant data for cancelled reservations

select booking.b\_id, booking.b\_date, booking.check\_in\_date, booking.check\_out\_date,

customer.c\_name, room.r\_type, hotel.h\_name, hotel.h\_address, hotel.h\_city, hotel.h\_state

from booking

inner join customer ON booking.c\_id = customer.c\_id

inner join room ON booking.r\_id = room.r\_id

inner join hotel on room.h\_id = hotel.h\_id

where booking.has\_reservation = 'NO';

-- to hold the data

c\_b\_id booking.b\_id%type;

c\_b\_date booking.b\_date%type;

c\_check\_in booking.check\_in\_date%type;

c\_check\_out booking.check\_out\_date%type;

c\_c\_name customer.c\_name%type;

c\_r\_type room.r\_type%type;

c\_h\_name hotel.h\_name%type;

c\_h\_address hotel.h\_address%type;

c\_h\_city hotel.h\_city%type;

c\_h\_state hotel.h\_state%type;

-- user defined exception declared

null\_exception exception;

BEGIN

OPEN cancelled\_cursor;

LOOP

-- loop gets all relevant data

FETCH cancelled\_cursor INTO c\_b\_id, c\_b\_date, c\_check\_in, c\_check\_out, c\_c\_name,

c\_r\_type, c\_h\_name,

c\_h\_address, c\_h\_city, c\_h\_state;

-- loop ends when there are no more records of cancelled reservations

EXIT WHEN cancelled\_cursor%notfound;

-- exception raised when there are 0 cancellations

if c\_b\_id IS NULL then

raise null\_exception;

end if;

-- All cancelled reservations are printed if there are any

Dbms\_output.put\_line('Cancelled booking id: ' || c\_b\_id || ' Booking date: ' || c\_b\_date ||

'| Check in date: ' || c\_check\_in || '| Check out date: ' || c\_check\_out || '| Customer name: ' ||

c\_c\_name ||

'| Room type: ' || c\_r\_type || '| Hotel name: ' || c\_h\_name || '| Hotel address: ' || c\_h\_address || ', ' ||

c\_h\_city || ', ' || c\_h\_state);

END LOOP;

CLOSE cancelled\_cursor;

exception

when null\_exception then

-- If there are no cancelled reservations, the user is told

dbms\_output.put\_line('There are no cancelled records');

END;

***/\*Member 3 Caleb Massey\*/***

**/\* 11. Change a reservationDate: Input the reservation ID and change reservation start and end date, if there is availability in the same room type for the new date interval\*/**

/\*======================================================================\*/

/\* Checks if the room is available during the date \*/

CREATE OR REPLACE FUNCTION checkbookavailable (

temp\_b\_id INT,

new\_in\_date DATE,

new\_out\_date DATE

) RETURN VARCHAR2 AS

free CHAR;

CURSOR check\_cursor IS SELECT

b\_id,

check\_in\_date,

check\_out\_date

FROM

booking

WHERE

booking.r\_id = (

SELECT

r\_id

FROM

booking

WHERE

temp\_b\_id = b\_id

)

AND b\_id != temp\_b\_id;

check\_row check\_cursor%rowtype;

keep\_count INT := 0;

no\_other\_reservations EXCEPTION;

BEGIN

FOR check\_row IN check\_cursor LOOP

/\*Checks for conflicting date\*/

keep\_count := keep\_count + 1;

IF

new\_in\_date BETWEEN check\_row.check\_in\_date AND check\_row.check\_out\_date

THEN

RETURN TO\_CHAR('Occupied');

ELSE

/\*Checks for conflicting date\*/

IF

new\_out\_date BETWEEN check\_row.check\_in\_date AND check\_row.check\_out\_date

THEN

RETURN TO\_CHAR('Occupied');

ELSE

/\*Checks for conflicting date\*/

IF

check\_row.check\_in\_date > new\_in\_date AND new\_out\_date > check\_row.check\_in\_date

THEN

RETURN TO\_CHAR('Occupied');

ELSE

/\*Runs if there are no conflicting nights\*/

RETURN TO\_CHAR('Free');

END IF;

END IF;

END IF;

END LOOP;

IF

keep\_count = 0

THEN

RAISE no\_other\_reservations;

END IF;

EXCEPTION

WHEN no\_other\_reservations THEN

RETURN TO\_CHAR('Free');

END;

/\*---------------------------------------------------------------------------------------------------------------------------\*/

/\* temp\_id is a booking id \*/

CREATE OR REPLACE PROCEDURE update\_reservation\_date (

temp\_id NUMBER,

new\_in\_date DATE,

new\_out\_date DATE

) AS

temp\_b\_date DATE;

date\_occupied EXCEPTION;

BEGIN

IF

checkbookavailable(temp\_id,new\_in\_date,new\_out\_date) = 'Free'

THEN

SELECT

b\_date

INTO

temp\_b\_date

FROM

booking

WHERE

b\_id = temp\_id;

dbms\_output.put\_line('Updated '|| temp\_id);

dbms\_output.put\_line('Check In Date is now '

|| new\_in\_date);

dbms\_output.put\_line('Check Out Date is now '

|| new\_out\_date);

dbms\_output.put\_line('Booking Date is '

|| temp\_b\_date);

UPDATE booking

SET

check\_in\_date = new\_in\_date,

check\_out\_date = new\_out\_date,

room\_cost = price\_from\_date(temp\_b\_date,new\_out\_date, to\_number(check\_out\_date - check\_in\_date))

WHERE

b\_id = temp\_id;

ELSE

RAISE date\_occupied;

END IF;

EXCEPTION

WHEN date\_occupied THEN

dbms\_output.put\_line('New date is currently occupied');

END;

/\*======================================================================\*/

**/\* 12. Show single hotel reservations: Given a hotel ID, show all reservations for that hotel\*/**

SET SERVEROUTPUT ON;

/\*temp\_id is a Hotel ID\*/

CREATE OR REPLACE PROCEDURE show\_reservations (

temp\_id NUMBER

) AS

CURSOR reserve\_cursor IS SELECT

booking.b\_id,

booking.r\_id,

booking.b\_date,

booking.check\_in\_date,

booking.check\_out\_date,

booking.num\_guests,

booking.has\_reservation

FROM

booking,

room,

hotel

WHERE

hotel.h\_id = room.h\_id

AND room.r\_id = booking.r\_id

AND hotel.h\_id = temp\_id;

reserve\_row reserve\_cursor%rowtype;

keep\_count INT := 0;

no\_reservation EXCEPTION;

BEGIN

dbms\_output.put\_line('Reservations in Hotel ' || temp\_id);

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

FOR reserve\_row IN reserve\_cursor LOOP

IF

reserve\_row.b\_date IS NULL

THEN

dbms\_output.put\_line('Booking ID: '

|| reserve\_row.b\_id);

dbms\_output.put\_line('Room ID: '

|| reserve\_row.r\_id);

dbms\_output.put\_line('Booking Date: NULL');

dbms\_output.put\_line('Check In Date: '

|| reserve\_row.check\_in\_date);

dbms\_output.put\_line('Check Out Date: '

|| reserve\_row.check\_out\_date);

dbms\_output.put\_line('Number of Guests: '

|| reserve\_row.num\_guests);

dbms\_output.put\_line('Has Reservations: '

|| reserve\_row.has\_reservation);

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

ELSE

dbms\_output.put\_line('Booking ID: '

|| reserve\_row.b\_id);

dbms\_output.put\_line('Room ID: '

|| reserve\_row.r\_id);

dbms\_output.put\_line('Booking Date: '

|| reserve\_row.b\_date);

dbms\_output.put\_line('Check In Date: '

|| reserve\_row.check\_in\_date);

dbms\_output.put\_line('Check Out Date: '

|| reserve\_row.check\_out\_date);

dbms\_output.put\_line('Number of Guests: '

|| reserve\_row.num\_guests);

dbms\_output.put\_line('Has Reservations: '

|| reserve\_row.has\_reservation);

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

END IF;

keep\_count := keep\_count + 1;

END LOOP;

IF

keep\_count = 0

THEN

RAISE no\_reservation;

END IF;

EXCEPTION

WHEN no\_reservation THEN

dbms\_output.put\_line('No reservations for hotel id '

|| temp\_id);

END;

**/\* 13. Show single guest reservations: Given a guest name, find all reservations under that name\*/**

/\*======================================================================\*/

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE customer\_reservations (

search\_name VARCHAR2

) AS

CURSOR customer\_cursor IS SELECT

booking.b\_id,

booking.r\_id,

booking.b\_date,

booking.check\_in\_date,

booking.check\_out\_date,

booking.num\_guests,

booking.has\_reservation

FROM

customer,

booking

WHERE

customer.c\_id = booking.c\_id

AND customer.c\_name = search\_name;

customer\_row customer\_cursor%rowtype;

keep\_count INT := 0;

no\_reservation EXCEPTION;

BEGIN

dbms\_output.put\_line('Reservations for ' || search\_name || ' are:');

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

FOR customer\_row IN customer\_cursor LOOP

IF

customer\_row.b\_date IS NULL

THEN

dbms\_output.put\_line('Booking ID: '

|| customer\_row.b\_id);

dbms\_output.put\_line('Room ID: '

|| customer\_row.r\_id);

dbms\_output.put\_line('Booking Date: NULL');

dbms\_output.put\_line('Check In Date: '

|| customer\_row.check\_in\_date);

dbms\_output.put\_line('Check Out Date: '

|| customer\_row.check\_out\_date);

dbms\_output.put\_line('Number of Guests: '

|| customer\_row.num\_guests);

dbms\_output.put\_line('Has Reservations: '

|| customer\_row.has\_reservation);

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

ELSE

dbms\_output.put\_line('Booking ID: '

|| customer\_row.b\_id);

dbms\_output.put\_line('Room ID: '

|| customer\_row.r\_id);

dbms\_output.put\_line('Booking Date: '

|| customer\_row.b\_date);

dbms\_output.put\_line('Check In Date: '

|| customer\_row.check\_in\_date);

dbms\_output.put\_line('Check Out Date: '

|| customer\_row.check\_out\_date);

dbms\_output.put\_line('Number of Guests: '

|| customer\_row.num\_guests);

dbms\_output.put\_line('Has Reservations: '

|| customer\_row.has\_reservation);

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

END IF;

keep\_count := keep\_count + 1;

END LOOP;

IF

keep\_count = 0

THEN

RAISE no\_reservation;

END IF;

EXCEPTION

WHEN no\_reservation THEN

dbms\_output.put\_line('Customer '

|| search\_name

|| ' has no reservations');

END;

/\*======================================================================\*/

**/\* 14. Change a reservation RoomType: Input the reservation ID and change reservation room type if there is availability for that room type during the reservation’s date interval\*/**

/\* Checks if the room is available during the date \*/

CREATE OR REPLACE FUNCTION checkavailable (

temp\_h\_id INT,

temp\_r\_id INT,

in\_date DATE,

out\_date DATE,

new\_r\_type VARCHAR2

) RETURN VARCHAR2 AS

free CHAR;

CURSOR check\_cursor IS SELECT

booking.r\_id,

check\_in\_date,

check\_out\_date

FROM

room,

hotel,

booking

WHERE

hotel.h\_id = room.h\_id

AND booking.r\_id = room.r\_id

AND hotel.h\_id = temp\_h\_id

AND r\_type = new\_r\_type;

check\_row check\_cursor%rowtype;

BEGIN

FOR check\_row IN check\_cursor LOOP

/\*Checks for conflicting date\*/

IF

in\_date BETWEEN check\_row.check\_in\_date AND check\_row.check\_out\_date

THEN

RETURN TO\_CHAR('Occupied');

ELSE

/\*Checks for conflicting date\*/

IF

out\_date BETWEEN check\_row.check\_in\_date AND check\_row.check\_out\_date

THEN

RETURN TO\_CHAR('Occupied');

ELSE

/\*Checks for conflicting date\*/

IF

check\_row.check\_in\_date > in\_date AND out\_date > check\_row.check\_in\_date

THEN

RETURN TO\_CHAR('Occupied');

ELSE

/\*Runs if there are no conflicting nights\*/

RETURN TO\_CHAR('Free');

END IF;

END IF;

END IF;

END LOOP;

END;

/\*---------------------------------------------------------------------------------------------------------------------------\*/

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE updateroomtype (

temp\_b\_id NUMBER,

new\_r\_type VARCHAR2

) AS

temp\_r\_id INT; /\*Temporary track r\_id\*/

keep\_count INT := 0; /\*Keeps count of if booking has been updated yet\*/

occupied EXCEPTION; /\*exception for if no rooms available of type new\_r\_type\*/

row\_count INT := 0; /\*Counts the number of rows in date\_cursor, to know if empty\*/

/\* Finds the rooms that are in that hotel, with the r\_type being changed to\*/

CURSOR type\_cursor IS SELECT

hotel.h\_id,

room.r\_id,

r\_type

FROM

room,

hotel

WHERE

hotel.h\_id = room.h\_id

AND hotel.h\_id = (

SELECT

hotel.h\_id

FROM

hotel,

room,

booking

WHERE

hotel.h\_id = room.h\_id

AND booking.r\_id = room.r\_id

AND booking.b\_id = temp\_b\_id

)

AND r\_type = new\_r\_type;

type\_row type\_cursor%rowtype;

/\* Finds the rooms that are in that hotel, with the r\_type being changed to, and is booked at any time\*/

CURSOR date\_cursor IS SELECT

hotel.h\_id,

room.r\_id,

check\_in\_date,

check\_out\_date

FROM

room,

hotel,

booking

WHERE

booking.r\_id = room.r\_id

AND hotel.h\_id = room.h\_id

AND hotel.h\_id = (

SELECT

hotel.h\_id

FROM

hotel,

room,

booking

WHERE

hotel.h\_id = room.h\_id

AND booking.r\_id = room.r\_id

AND booking.b\_id = temp\_b\_id

)

AND r\_type = new\_r\_type;

date\_row date\_cursor%rowtype;

BEGIN

dbms\_output.put\_line('Reservation '|| temp\_b\_id || ' has been changed to room type ' || new\_r\_type);

SELECT

r\_id

INTO

temp\_r\_id

FROM

booking

WHERE

b\_id = temp\_b\_id;

FOR date\_row IN date\_cursor LOOP

row\_count := row\_count + 1;

END LOOP;

/\*Loops through all rooms that are in the hotel of the desired r\_type\*/

FOR type\_row IN type\_cursor LOOP

/\*Loops through all rooms that are in booking and hotel of the same r\_type\*/

FOR date\_row IN date\_cursor LOOP

/\*Checks if the room is in both cursors\*/

IF

type\_row.r\_id = date\_row.r\_id

THEN

/\*Checks if the room in booking is available during the date the original booking is set for \*/

IF

checkavailable(date\_row.h\_id,date\_row.r\_id,date\_row.check\_in\_date,date\_row.check\_out\_date,type\_row.r\_type) = 'Free'

THEN

/\*Checks if the new room has been updated yet\*/

IF

keep\_count = 0

THEN

keep\_count := keep\_count + 1;

/\*update booking set booking.r\_id = type\_row.r\_id where booking.b\_id = temp\_b\_id;\*/

dbms\_output.put\_line('by changing the reserved room from '

|| type\_row.r\_id

|| ' to '

|| temp\_r\_id);

END IF;

/\*Runs if the room was not available\*/

ELSE

IF

checkavailable(type\_row.h\_id,type\_row.r\_id,date\_row.check\_in\_date,date\_row.check\_out\_date,type\_row.r\_type) = 'Occupied'

THEN

dbms\_output.put\_line(type\_row.r\_id

|| ' and '

|| date\_row.r\_id

|| ' is Occupied');

END IF;

END IF;

/\*If this runs it means that room has not been booked yet and is therefor free \*/

ELSE

/\*Checks if the new room has been updated yet\*/

IF

keep\_count = 0

THEN

keep\_count := keep\_count + 1;

/\*update booking set booking.r\_id = type\_row.r\_id where booking.b\_id = temp\_b\_id;\*/

dbms\_output.put\_line('by changing the reserved room from '

|| type\_row.r\_id

|| ' to '

|| temp\_r\_id);

END IF;

END IF;

END LOOP;

IF

row\_count = 0

THEN

/\*Runs if there is no rooms in booking at all\*/

IF

keep\_count = 0

THEN

keep\_count := keep\_count + 1;

/\*update booking set booking.r\_id = type\_row.r\_id where booking.b\_id = temp\_b\_id;\*/

dbms\_output.put\_line('by changing the reserved room from '

|| type\_row.r\_id

|| ' to '

|| temp\_r\_id);

END IF;

END IF;

END LOOP;

/\*If nothing can be updated, runs exception, case is no rooms of this type in hotel or no room replacement available\*/

IF

keep\_count = 0

THEN

RAISE occupied;

END IF;

EXCEPTION

WHEN occupied THEN

dbms\_output.put\_line('No Rooms available of type '

|| new\_r\_type);

END;

/\*---------------------------------------------------------------------------------------------------------------------------\*/

**/\*15. Total Monthly Income Report: Calculate and display total income from all sources of all hotels. Totals must be printed by month, and for each month by room type, service type. Include discounts.\*/**

/\*======================================================================\*/

CREATE OR REPLACE PROCEDURE calculate\_income AS

total\_income INT; /\*Variable that holds the sum of service prices and room costs for the month\*/

sum\_price INT; /\*The sum of all the service prices for that month\*/

temp\_r\_cost INT; /\*Holds the sum of all the room costs for that month, discount included\*/

month\_date DATE; /\*Keeps track of the dates as mon-yyyy instead of dd-mon-yyyy\*/

/\*Splits into all months a room is being rented\*/

CURSOR income\_cursor IS SELECT DISTINCT

TO\_CHAR(booking.check\_out\_date,'MON-YYYY') AS month\_date

FROM

hotel,

room,

booking

WHERE

hotel.h\_id = room.h\_id

AND room.r\_id = booking.r\_id

ORDER BY

month\_date;

income\_row income\_cursor%rowtype;

BEGIN

FOR income\_row IN income\_cursor LOOP

/\*Finds the sum of service prices for each month, can be NULL\*/

SELECT

SUM(s\_price)

INTO

sum\_price

FROM

hotel,

room,

booking,

service

WHERE

hotel.h\_id = room.h\_id

AND room.r\_id = booking.r\_id

AND booking.b\_id = service.b\_id

AND TO\_CHAR(booking.check\_out\_date,'MON-YYYY') LIKE '%'

|| income\_row.month\_date

|| '%';

/\*Finds the sum of room costs for each month\*/

SELECT

SUM(room\_cost)

INTO

temp\_r\_cost

FROM

hotel,

room,

booking

WHERE

hotel.h\_id = room.h\_id

AND room.r\_id = booking.r\_id

AND TO\_CHAR(booking.check\_out\_date,'MON-YYYY') LIKE '%'

|| income\_row.month\_date

|| '%';

/\*Figures out if there were any services provided and adds up the total income\*/

IF

sum\_price IS NULL

THEN

total\_income := temp\_r\_cost;

ELSE

total\_income := temp\_r\_cost + sum\_price;

END IF;

/\*Prints out each months total income\*/

dbms\_output.put\_line('Income Total for '

|| income\_row.month\_date

|| ' is: $'

|| total\_income);

END LOOP;

END;

/\*---------------------------------------------------------------------------------------------------------------------------\*/

**/\*Member 4 Rohan Pandya\*/**

**/\*16. Add a service to a reservation: Input: ReservationID, specific service. Add the service to the reservation for a particular date. Multiple services are allowed on a reservation for the same date. (PROCEDURE)\*/**

set serveroutput on;

/\*----------------------------------------------------------------------\*/

/\* The Procedure takes in room ID, and Service Name and Service Dates, and inserts the service into the table\*/

/\*----------------------------------------------------------------------\*/

CREATE OR REPLACE PROCEDURE resservice (

booking\_id NUMBER,

service\_name VARCHAR2,

service\_date DATE

)

AS

/\*----------------------------------------------------------------------\*/

/\* The insert statement uses the auto sequence to add assign service\_id and gets the price based on the service name\*/

/\*----------------------------------------------------------------------\*/

BEGIN

INSERT INTO service (

s\_id,

b\_id,

s\_name,

s\_price,

s\_date

) VALUES (

service\_seq.NEXTVAL,

booking\_id,

service\_name,

price\_from\_name(service\_name),

service\_date

);

/\*----------------------------------------------------------------------\*/

/\*PRINTS A STATMENT ONCE SERVICE HAS BEEN ADDED TO THE RESERVATION \*/

/\*----------------------------------------------------------------------\*/

dbms\_output.put\_line(service\_name

|| ' has been charged to reservation '

|| booking\_id);

END;

**/\*17. Reservation Services Report: Input the reservation ID and display all services on this reservation. Print “no services for this reservation” if none exists.\*/**

SET SERVEROUTPUT ON;

/\*----------------------------------------------------------------------\*/

/\*THE PROCEDURE TAKES IN THE RESERVATION NUMBER AND FINDS WHAT SERVICES ARE USED BY THE RESERVATION\*/

/\*----------------------------------------------------------------------\*/

CREATE OR REPLACE PROCEDURE reservation\_service (book\_id IN NUMBER) IS

/\*----------------------------------------------------------------------\*/

/\*THE CURSOR STORES THE ID, NAME, AND THE PRICE OF THE SERVICES USED BY THE RESERVATION\*/

/\*----------------------------------------------------------------------\*/

CURSOR serv\_cursor IS

SELECT s\_id, s\_name, s\_price

FROM service, booking, customer

WHERE service.b\_id = booking.b\_id

AND booking.b\_id = book\_id

AND booking.c\_id = customer.c\_id;

res\_service\_row serv\_cursor%rowtype;

/\*----------------------------------------------------------------------\*/

/\*USER DEFINED EXCEPTION \*/

/\*----------------------------------------------------------------------\*/

no\_services EXCEPTION;

/\*----------------------------------------------------------------------\*/

/\*USER DEFINED PRICE AND COUNT TRACKER TO STORE VALUES\*/

/\*----------------------------------------------------------------------\*/

track\_price INT := 0;

track\_count INT := 0;

BEGIN

dbms\_output.put\_line('Reservation Services Report for RESERVATION ID: '|| BOOK\_ID);

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

FOR res\_service\_row IN serv\_cursor LOOP

dbms\_output.put\_line('Service ID: ' || res\_service\_row.s\_id);

dbms\_output.put\_line('SERVICE\_Name: ' || res\_service\_row.s\_name);

dbms\_output.put\_line('SERVICE\_PRICE: ' || res\_service\_row.s\_price);

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

track\_price := track\_price + res\_service\_row.s\_price;

track\_count := track\_count + 1;

END LOOP;

/\*----------------------------------------------------------------------\*/

/\*sums up the prices for all the services used by the reservation \*/

/\*----------------------------------------------------------------------\*/

IF

track\_count = 0

THEN

RAISE no\_services;

ELSE

dbms\_output.put\_line('The total cost of services for reservation ' || book\_id || ' is $' || track\_price);

END IF;

/\*----------------------------------------------------------------------\*/

/\*Throws exception where there is no reservation\*/

/\*----------------------------------------------------------------------\*/

EXCEPTION

WHEN no\_services THEN

dbms\_output.put\_line('no services for this reservation');

END;

/\*======================================================================\*/

**/\*18. Show Specific Service Report: Input the service name, and display information on all reservations that have this service in all hotels\*/**

/\*----------------------------------------------------------------------\*/

/\*The Function below takes in the name of the service Name and

counts how many times the service was used in the specific reservation\*/

/\*----------------------------------------------------------------------\*/

create or replace function serviceDetails(ser\_name in Varchar2)

Return varchar2 is

servCount number;

Begin

/\*----------------------------------------------------------------------\*/

/\*THe service is counted based and stored in the variable based on the user input of the servie name\*/

/\*----------------------------------------------------------------------\*/

select count(s\_id) into servCount

from booking b,service s

where b.b\_id = s.b\_id

and s.s\_name = ser\_name;

Return ('THE ' || ser\_name || ' is in ' || servCount|| ' reservation');

/\*----------------------------------------------------------------------\*/

/\*Exception is thrown when there is no service in the reservation\*/

/\*----------------------------------------------------------------------\*/

Exception

when

no\_data\_found then

return ('There is No service for this reservation');

End;

Set serveroutput on ;

/\*----------------------------------------------------------------------\*/

/\*Procedure below takes in the name, locates all the information about the service name by finding where the bookig\_Id in the service table is same as the Booking\_id in the Booking table\*/

/\*----------------------------------------------------------------------\*/

create or replace procedure serviceReport (ser\_name IN varchar2)is

cursor service\_detail is

select booking.B\_ID, S\_ID, s\_name

from service, booking

where s\_name = ser\_name

and service.b\_id = booking.b\_ID;

/\*----------------------------------------------------------------------\*/

/\*the information is that was retrived is stored in the cursor\*/

/\*----------------------------------------------------------------------\*/

service\_row service\_detail%rowtype;

begin

for service\_row in service\_detail

loop

dbms\_output.put\_line ('RESERVATION ID:' || service\_row.B\_ID);

dbms\_output.put\_line ('SERVICE\_ID: ' || service\_row.s\_id);

dbms\_output.put\_line ('Service Name: ' || service\_row.s\_name);

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

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

end loop;

/\*----------------------------------------------------------------------\*/

/\*The name is also passed to the function so that the function can run and find the data\*/

/\*----------------------------------------------------------------------\*/

dbms\_output.put\_line(serviceDetails(ser\_name));

end;

**/\*19. Total Services Income Report: Given a hotelID, calculate and display income from all services in all reservations in that hotel.\*/**

SET SERVEROUTPUT ON;

/\*----------------------------------------------------------------------\*/

/\* The procedure takes in the Hotel\_ID\*/

/\*----------------------------------------------------------------------\*/

CREATE OR REPLACE PROCEDURE hotel\_service (

hotel\_id\_in IN NUMBER

) IS

CURSOR hotel\_cursor IS SELECT

s\_ID, s\_name, s\_date, s\_price

FROM

service,

booking,

room,

Hotel

/\*----------------------------------------------------------------------\*/

/\*this statement finds where the Hotel ID in the Hotel table and matches it with the

Input value and finds where the input value has connections to other tables. \*/

/\*----------------------------------------------------------------------\*/

WHERE

hotel.h\_id = hotel\_id\_in

AND hotel.h\_id = room.h\_id

AND booking.r\_id = room.r\_id

AND service.b\_id = booking.b\_id;

/\*THE cursor stores the data\*/

hotel\_service\_row hotel\_cursor%rowtype;

total\_price INT := 0;

BEGIN

dbms\_output.put\_line('Hotel ID: '

|| hotel\_id\_in);

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

FOR hotel\_service\_row IN hotel\_cursor LOOP

dbms\_output.put\_line('SERVICE\_ID: '

|| hotel\_service\_row.s\_id);

dbms\_output.put\_line('SERVICE\_Name: '

|| hotel\_service\_row.s\_Name);

dbms\_output.put\_line('SERVICE\_Date: '

|| hotel\_service\_row.s\_Date);

dbms\_output.put\_line('SERVICE\_PRICE: '

|| hotel\_service\_row.s\_price);

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

/\*----------------------------------------------------------------------\*/

/\*The statemenet below calculates all the services used by the Hotel\*/

/\*----------------------------------------------------------------------\*/

total\_price := total\_price + hotel\_service\_row.s\_price;

END LOOP;

/\*----------------------------------------------------------------------\*/

/\*THE STATEMENT prints total Revenue Made from services\*/

/\*----------------------------------------------------------------------\*/

dbms\_output.put\_line('The total income from Srvervice for hotel '

|| hotel\_id\_in

|| ' is $'

|| total\_price);

END;

***/\*Member 5(Nandu):\*/***

**/\*20. Show reservations: display all rooms reserved in all hotels\*/**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE get\_reserved AS

nodata EXCEPTION;

b\_id booking.b\_id%TYPE;

r\_id room.r\_id%TYPE;

h\_id hotel.h\_id%TYPE;

CURSOR c1 IS SELECT

b\_id,

r\_id,

h\_id

FROM

booking

NATURAL JOIN room

NATURAL JOIN hotel

WHERE

has\_reservation = 'YES';

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO b\_id,r\_id,h\_id;

EXIT WHEN c1%notfound;

dbms\_output.put\_line('Booking ID:'

|| b\_id);

dbms\_output.put\_line('Room ID:'

|| r\_id);

dbms\_output.put\_line('Hotel ID:'

|| h\_id);

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

END LOOP;

CLOSE c1;

END;

**/\*21. Show available rooms by type: Given a hotel ID, display the count of all available rooms by room type.\*/**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE room\_type\_available2 (

hotel\_id IN NUMBER

) AS

CURSOR room\_type\_by\_hid IS SELECT

r\_type,

COUNT(r\_type)

FROM

room

WHERE

h\_id = hotel\_id

GROUP BY

r\_type,

h\_id;

rtype room.r\_type%TYPE;

rcount NUMBER;

BEGIN

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

dbms\_output.put\_line('Hotel ID:'

|| hotel\_id);

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

OPEN room\_type\_by\_hid;

LOOP

FETCH room\_type\_by\_hid INTO rtype,rcount;

EXIT WHEN room\_type\_by\_hid%notfound;

dbms\_output.put\_line(rtype

|| ': '

|| rcount);

END LOOP;

CLOSE room\_type\_by\_hid;

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

END;

**/\*22. Room Checkout Report: Input: ReservationID Output: Guest name, Room number, rate per day and possibly multiple rooms (if someone reserved several rooms), Services rendered per date, type, and amount, Discounts applied (if any), Total amount to be paid\*/**

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE checkout\_report (

booking\_id NUMBER

) AS

CURSOR cursor\_name IS SELECT

c\_name

FROM

customer

NATURAL JOIN booking

WHERE

b\_id = booking\_id;

CURSOR cursor\_check\_booking IS SELECT

COUNT(DISTINCT b\_id)

FROM

booking

WHERE

b\_id = booking\_id;

cnt NUMBER;

no\_data\_exp EXCEPTION;

guest\_name customer.c\_name%TYPE;

CURSOR cursor\_room\_rent IS SELECT

to\_number(check\_out\_date - check\_in\_date) day\_count,

booking.room\_cost,

r\_type,

check\_in\_date,

check\_out\_date

FROM

booking

NATURAL JOIN room

WHERE

b\_id = booking\_id;

total\_room\_rent NUMBER;

day\_count NUMBER;

room\_rent\_per\_day booking.room\_cost%TYPE;

room\_type room.r\_type%TYPE;

check\_in booking.check\_in\_date%TYPE;

check\_out booking.check\_out\_date%TYPE;

CURSOR cursor\_service\_detail IS SELECT

s\_price,

s\_name,

s\_date

FROM

booking

NATURAL JOIN service

WHERE

b\_id = booking\_id;

total\_service\_cost NUMBER;

service\_price NUMBER;

service\_name service.s\_name%TYPE;

service\_date service.s\_date%TYPE;

BEGIN

total\_room\_rent := 0;

total\_service\_cost := 0;

OPEN cursor\_check\_booking;

FETCH cursor\_check\_booking INTO cnt;

IF

cnt = 0

THEN

RAISE no\_data\_exp;

END IF;

OPEN cursor\_name;

FETCH cursor\_name INTO guest\_name;

--TODO: Exception when guest not found or something like that

dbms\_output.put\_line('Guest Name: '

|| guest\_name);

CLOSE cursor\_name;

dbms\_output.put\_line(chr(10) );

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

dbms\_output.put\_line('--=Room Rent Details=--');

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

OPEN cursor\_room\_rent;

LOOP

FETCH cursor\_room\_rent INTO day\_count,room\_rent\_per\_day,room\_type,check\_in,check\_out;

EXIT WHEN cursor\_room\_rent%notfound;

total\_room\_rent := total\_room\_rent + room\_rent\_per\_day;

dbms\_output.put\_line('Room Type: '

|| room\_type);

dbms\_output.put\_line('Check In: '

|| check\_in);

dbms\_output.put\_line('Check Out:'

|| check\_out);

dbms\_output.put\_line('Days: '

|| day\_count);

dbms\_output.put\_line('Rent: $'

|| room\_rent\_per\_day);

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

END LOOP;

CLOSE cursor\_room\_rent;

dbms\_output.put\_line('Total Rent: $'

|| total\_room\_rent);

dbms\_output.put\_line(chr(10) );

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

dbms\_output.put\_line('--=Service Details=--');

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

dbms\_output.put\_line('Type Cost Date');

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

OPEN cursor\_service\_detail;

LOOP

FETCH cursor\_service\_detail INTO service\_price,service\_name,service\_date;

EXIT WHEN cursor\_service\_detail%notfound;

total\_service\_cost := total\_service\_cost + service\_price;

dbms\_output.put\_line(service\_name

|| ' $'

|| service\_price

|| ' '

|| service\_date);

END LOOP;

CLOSE cursor\_service\_detail;

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

dbms\_output.put\_line('Total Service Cost: $'

|| total\_service\_cost);

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

dbms\_output.put\_line('Total Amount: $'

|| (total\_service\_cost + total\_room\_rent) );

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

EXCEPTION

WHEN no\_data\_exp THEN

dbms\_output.put\_line('NO DATA FOUND FOR GIVEN BOOKING ID '

|| booking\_id);

END;

**/\*23. Income By State Report: Input is state. Print total income from all sources of all hotels by room type and service type in the given state. Include discounts.\*/**

SET SERVEROUTPUT ON;

create or replace procedure get\_state\_report(state varchar2)

as

cursor cursor\_hotel is

SELECT room.r\_type, SUM(booking.room\_cost) TOTAL\_INCOME

FROM HOTEL INNER JOIN ROOM ON hotel.h\_id = room.h\_id inner join booking on booking.r\_id = room.r\_id

WHERE hotel.h\_state = UPPER(state)

GROUP BY room.r\_type, hotel.h\_state;

room\_type room.r\_type%type;

total\_room\_cost number;

cursor cursor\_service is

SELECT service.s\_name, SUM(service.s\_price) TOTAL\_SERVICE

FROM HOTEL INNER JOIN ROOM ON hotel.h\_id = room.h\_id inner join booking on booking.r\_id = room.r\_id inner join service on booking.b\_id = service.B\_id

WHERE hotel.h\_state = UPPER(state)

GROUP BY service.s\_name, hotel.h\_state;

service\_type service.s\_name%type;

service\_cost number;

total\_room number;

total\_service number;

begin

total\_room := 0;

total\_service := 0;

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

dbms\_output.put\_line('STATE: '||state);

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

open cursor\_hotel;

loop

fetch cursor\_hotel into room\_type, total\_room\_cost;

exit when

cursor\_hotel%notfound;

dbms\_output.put\_line(room\_type || ': $'|| total\_room\_cost);

total\_room := total\_room + total\_room\_cost;

end loop;

close cursor\_hotel;

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

dbms\_output.put\_line('Total Room: $'||total\_room);

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

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

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

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

open cursor\_service;

loop

fetch cursor\_service into service\_type, service\_cost;

exit when cursor\_service%notfound;

dbms\_output.put\_line(service\_type || ': $'|| service\_cost);

total\_service := total\_service + service\_cost;

end loop;

close cursor\_service;

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

dbms\_output.put\_line('Total Service: $'||total\_service);

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

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

dbms\_output.put\_line('Total: $'||(total\_room+total\_service));

end;

/\*---------------------------------------------------------------------------------------------------------------------------\*/

/\*Execute Procedures\*/

**/\*For member M1\*/**

**/\*1. Add a hotel:\*/**

**/\*1a. Add a new hotel called H1 in New York, NY\*/**

EXECUTE addNewHotel (hotel\_seq.nextval, 'H1', '123 Wall Street', 'New York', 'NY', 6178894573, 'UNSOLD');

Insert into room values (room\_seq.nextval, 10, 'single');

Insert into room values (room\_seq.nextval, 10, 'suite');

Insert into room values (room\_seq.nextval, 10, 'single');

Insert into room values (room\_seq.nextval, 10, 'conference');

Insert into room values (room\_seq.nextval, 10, 'double');

Insert into room values (room\_seq.nextval, 10, 'suite');

Insert into room values (room\_seq.nextval, 10, 'single');

Insert into room values (room\_seq.nextval, 10, 'suite');

Insert into room values (room\_seq.nextval, 10, 'double');

Insert into room values (room\_seq.nextval, 10, 'double');

Insert into room values (room\_seq.nextval, 10, 'single');

Insert into room values (room\_seq.nextval, 10, 'conference');

**/\*1b. Add a new hotel called H2 in Baltimore, MD\*/**

EXECUTE addNewHotel (hotel\_seq.nextval, 'H2', '51 Pratt St', 'Baltimore', 'MD', 4436251745, 'UNSOLD');

Insert into room values (room\_seq.nextval, 11, 'single');

Insert into room values (room\_seq.nextval, 11, 'single');

Insert into room values (room\_seq.nextval, 11, 'single');

Insert into room values (room\_seq.nextval, 11, 'single');

Insert into room values (room\_seq.nextval, 11, 'double');

Insert into room values (room\_seq.nextval, 11, 'double');

Insert into room values (room\_seq.nextval, 11, 'double');

Insert into room values (room\_seq.nextval, 11, 'double');

Insert into room values (room\_seq.nextval, 11, 'suite');

Insert into room values (room\_seq.nextval, 11, 'suite');

Insert into room values (room\_seq.nextval, 11, 'conference');

Insert into room values (room\_seq.nextval, 11, 'conference');

**/\*1c. Add a new hotel called H3 in San Francisco, CA\*/**

EXECUTE addNewHotel (hotel\_seq.nextval, 'H3', '83 Sunshine Ave', 'San Francisco', 'CA', 5137648943, 'UNSOLD');

Insert into room values (room\_seq.nextval, 12, 'double');

Insert into room values (room\_seq.nextval, 12, 'suite');

Insert into room values (room\_seq.nextval, 12, 'single');

Insert into room values (room\_seq.nextval, 12, 'conference');

Insert into room values (room\_seq.nextval, 12, 'double');

Insert into room values (room\_seq.nextval, 12, 'suite');

Insert into room values (room\_seq.nextval, 12, 'single');

Insert into room values (room\_seq.nextval, 12, 'conference');

**/\*2. Find hotel H2\*/**

EXECUTE findHotel('51 Pratt St');

**/\*3. Display hotel info\*/**

EXECUTE hotel\_id(11);

**/\*4. Sell hotel H3\*/**

EXECUTE sellHotel(12);

**/\*5. Display hotel info\*/**

EXECUTE hotel\_id(12);

**/\*6. Show sold hotels\*/**

EXECUTE showSold\_hotels;

**/\*7. Add a new hotel called H21 in Annapolis, MD\*/**

EXECUTE addNewHotel (hotel\_seq.nextval, 'H21', '54 Annapolis St', 'Annapolis', 'MD', 4438928364, 'UNSOLD');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'single');

Insert into room values (room\_seq.nextval, 13, 'double');

Insert into room values (room\_seq.nextval, 13, 'double');

Insert into room values (room\_seq.nextval, 13, 'double');

Insert into room values (room\_seq.nextval, 13, 'suite');

Insert into room values (room\_seq.nextval, 13, 'suite');

Insert into room values (room\_seq.nextval, 13, 'conference');

**/\*8. Show hotels in MD\*/**

Execute hotel\_state('MD');

**/\*For member M2\*/**

**/\*9. Jon Doe makes a reservation in H2 for 2 nights in August (choose the dates) for a suite\*/**

Insert into customer values (customer\_seq.nextval, 'Jon Doe', '16 Montrose Ct, Catonsville, MD', 2405816713, '6418-25761528-9481');

Execute newBooking(10, 26, to\_date('25-AUG-18','DD-MON-YY'), to\_date('27-AUG-18','DD-MON-YY'), to\_date('29-AUG-18','DD-MON-YY'), 'YES');

**/\*10. Jon Doe makes a reservation in H2 for 1 night in April (choose the dates) for a double\*/**

execute newBooking(10, 30, to\_date('09-APR-18','DD-MON-RR'),to\_date('10-APR-19','DD-MON-RR'), to\_date('11-APR-19','DD-MON-RR'), 'YES');

**/\*11. Susan Dawson reserves in H1 5 nights in August (choose the dates) for a single room\*/**

Insert into customer values (customer\_seq.nextval, 'Susan Dawson', '14014 Del Boca Vista, Tampa, FL', 4109978512, '6582-8711-6500-8902');

execute newBooking(11, 14, to\_date('09-AUG-18','DD-MON-RR'),to\_date('14-AUG-18','DD-MON-RR'), to\_date('16-AUG-18','DD-MON-RR'), 'YES');

**/\*12. Suki Mamuki reserves 2 nights in H1 for a double room for May 10-12\*/**

Insert into customer values (customer\_seq.nextval, 'Suki Mamuki', '700 Grove Park Drive, York, PA', 4439825195, '6548-0168-6577-3122');

execute newBooking(12, 14, to\_date('09-MAY-18','DD-MON-RR'),to\_date('11-MAY-18','DD-MON-RR'), to\_date('12-MAY-18','DD-MON-RR'), 'YES');

**/\*13. Find all reservations for Jon Doe\*/**

execute findBooking('Jon Doe', to\_date('25-AUG-18','DD-MON-RR'),11);

execute findBooking('Jon Doe', to\_date('09-APR-18','DD-MON-RR'), 11);

**/\*14. Cancel Susan’s reservation\*/**

execute cancelBooking(12);

**/\*15. Show cancelations\*/**

execute showCancellations();

**/\*For member M3\*/**

**/\*16. Change Jon Doe’s 2nd reservation from April to November 18, 2018\*/**

execute update\_reservation\_date(11, to\_date('18-NOV-18','dd-mon-yy'), to\_date('19-NOV-18','dd-mon-yy'));

**/\*17. Change Suki’s reservation from double to suite.\*/**

execute UpdateRoomType(13, 'suite');

**/\*18. Show all reservations for H2\*/**

Execute show\_reservations(11);

**/\*19. Show all reservations under the name of Jon Doe\*/**

execute customer\_reservations('Jon Doe');

**/\*20. Show all reservations under the name of Suki Mamuki\*/**

execute customer\_reservations('Suki Mamuki');

**/\*21. Show Total Monthly Income Report\*/**

execute calculate\_income;

**/\*For member M4\*/**

**/\*22. Add laundry to reservation in item 9 above\*/**

execute resService(10, 'Laundry service', to\_date('25-AUG-18','DD-MON-YY'));

**/\*23. Add restaurant to reservation in item 9 above\*/**

execute resService(10, 'Restaurant service', to\_date('25-AUG-18','DD-MON-YY'));

**/\*24. Add restaurant to reservation in item 9 above\*/**

execute resService(10, 'Restaurant service', to\_date('27-AUG-18','DD-MON-YY'));

**/\*25. Add a movie to reservation in item 9 above\*/**

execute resService(10, 'Pay-Per-View', to\_date('26-AUG-18','DD-MON-YY'));

**/\*26. Show Reservation Services report for the previous reservation\*/**

execute reservation\_service(10);

**/\*27. Show Specific Service Report for ‘restaurant’ service\*/**

Execute serviceReport('Restaurant service')

**/\*28. Show Total Services Income report for H2\*/**

execute hotel\_service(11);

**/\*29. Show Total Monthly Income Report\*/**

execute calculate\_income;

**/\*For member M5\*/**

**/\*30. Show reservations (all hotels)\*/**

execute get\_reserved;

**/\*31. Show available suites in H2\*/**

execute room\_type\_available2(11);

**/\*32. Show room check out report for Jon Doe\*/**

execute checkout\_report(10);

**/\*33. Show income by state for MD\*/**

execute get\_state\_report('MD');

**/\*34. Show income by state for NY\*/**

execute get\_state\_report('NY');