**----------------------------------------------------------------------------------------------------------------**

**1.4 - Add Event room**

**----------------------------------------------------------------------------------------------------------------**

Set serveroutput on;

create or replace procedure add\_event\_room(

h\_id in number,

r\_type in varchar2

)

--Add Event Room: Given a hotel ID, add a new room for a specific event

--to that hotel. The room types are: Small Hall, Medium hall, Large hall.

--@param h\_id, the hotels id

--@param room\_type, the room\_type to add to the hotel

AS

hotel\_exists number;

BEGIN

--check that hotel exists

select count(\*)

into hotel\_exists

from hotel h

where h.hotel\_id = h\_id;

if(hotel\_exists = 1) then

if (r\_type = 'Small room') then

update hotel set max\_small\_room\_avail = max\_small\_room\_avail + 1 where hotel.hotel\_id = h\_id;

dbms\_output.put\_line('An additional small room is now available for booking for hotel with id ' || h\_id);

elsif (r\_type = 'Medium room') then

update hotel set max\_medium\_room\_avail = max\_medium\_room\_avail + 1 where hotel.hotel\_id = h\_id;

dbms\_output.put\_line('An additional medium room is now available for booking for hotel with id ' || h\_id);

elsif (r\_type = 'Large room') then

update hotel set max\_large\_room\_avail = max\_large\_room\_avail + 1 where hotel.hotel\_id = h\_id;

dbms\_output.put\_line('An additional large room is now available for booking for hotel with id ' || h\_id);

end if;

else

dbms\_output.put\_line('Hotel with id: ' || h\_id || ' does not exists' || chr(10) ||

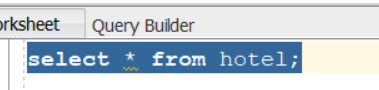
'No table was updated');

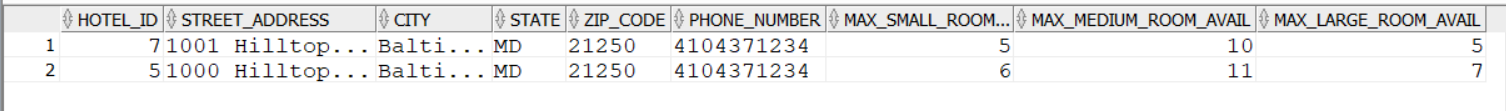
end if;

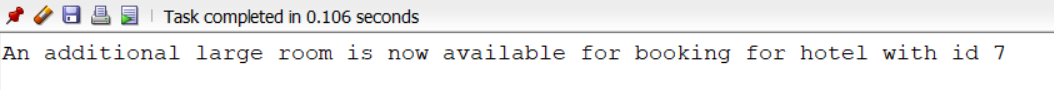
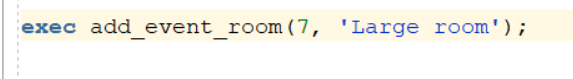
END;

**1.4 Output**

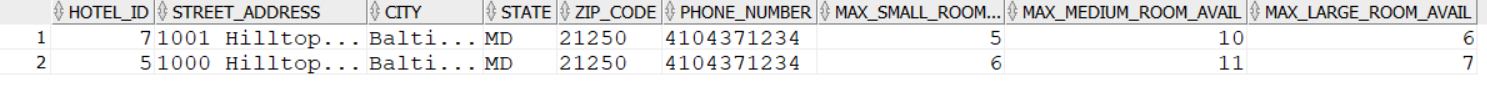
**--------------------------------------------------------------------------------------------------------------------------------**











**1.5 - display available rooms**

**--------------------------------------------------------------------------------------------------------------------------------**

create or replace procedure display\_available\_rooms(

h\_id in number

)

--Show available rooms by type: Given a hotel ID,

--display the count of all available rooms by room type.

--@param h\_id, the hotel id

AS

small\_room number;

medium\_room number;

large\_room number;

BEGIN

SELECT max\_small\_room\_avail, max\_medium\_room\_avail, max\_large\_room\_avail into small\_room, medium\_room, large\_room from hotel h where h.hotel\_id = h\_id;

--display all available rooms by type

dbms\_output.put\_line('Rooms vailable: ' || chr(10) ||

'---------------------------');

if(small\_room > 0) then dbms\_output.put\_line('Number of small rooms available: ' || small\_room); end if;

if(medium\_room > 0) then dbms\_output.put\_line('Number of medium rooms available: ' || medium\_room); end if;

if(large\_room > 0) then dbms\_output.put\_line('Number of large rooms available: ' || large\_room); end if;

EXCEPTION

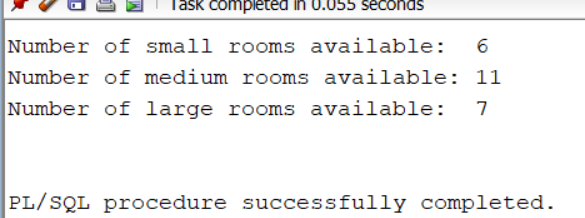
when no\_data\_found

then dbms\_output.put\_line('No hotel with id:' || h\_id || ' found.');

END;

**1.5 Output**

**--------------------------------------------------------------------------------------------------------------------------------**



**1.6 - Display hotel event rooms**

**--------------------------------------------------------------------------------------------------------------------------------**

create or replace procedure display\_hotel\_event\_rooms(

hotel\_state in varchar2

)

--Report Hotels and Event Rooms In State: Given a state,

--display event room information of all hotels in that particular state.

--Include total capacity per event room type per hotel.

--@param hotel\_state, a state in which hotels belong

AS

CURSOR hotel\_cursor IS

SELECT \* from hotel h where h.state = hotel\_state;

capacity\_small number;

capacity\_medium number;

capacity\_large number;

BEGIN

--grab capacity for small, medium and large room types

select room\_capacity into capacity\_small from room\_type where room\_type = 'Small room';

select room\_capacity into capacity\_medium from room\_type where room\_type = 'Medium room';

select room\_capacity into capacity\_large from room\_type where room\_type = 'Large room';

--iterate over all hotels with in state and display their information

dbms\_output.put\_line('Hotel(s) available in : ' || hotel\_state || chr(10) ||

'---------------------------'

);

FOR hotel in hotel\_cursor

LOOP

dbms\_output.put\_line(

'hotel id : ' || hotel.hotel\_id || ' (' || hotel.street\_address || ', ' || hotel.city || ' ' || hotel.state || ' ' || hotel.zip\_code || ')' || chr(10) ||

'Number of small rooms available : ' || hotel.max\_small\_room\_avail || '| with a total capacity of ' || hotel.max\_small\_room\_avail \* capacity\_small || '(' || capacity\_small || ' per room)' || chr(10) ||

'Number of medium rooms available: ' || hotel.max\_medium\_room\_avail || '| with a total capacity of ' || hotel.max\_medium\_room\_avail \* capacity\_medium || '(' || capacity\_medium || ' per room)' || chr(10) ||

'Number of large rooms available : ' || hotel.max\_large\_room\_avail || '| with a total capacity of ' || hotel.max\_large\_room\_avail \* capacity\_large || '(' || capacity\_large || ' per room)' || chr(10) ||

'---------------------------'

);

END LOOP;

EXCEPTION

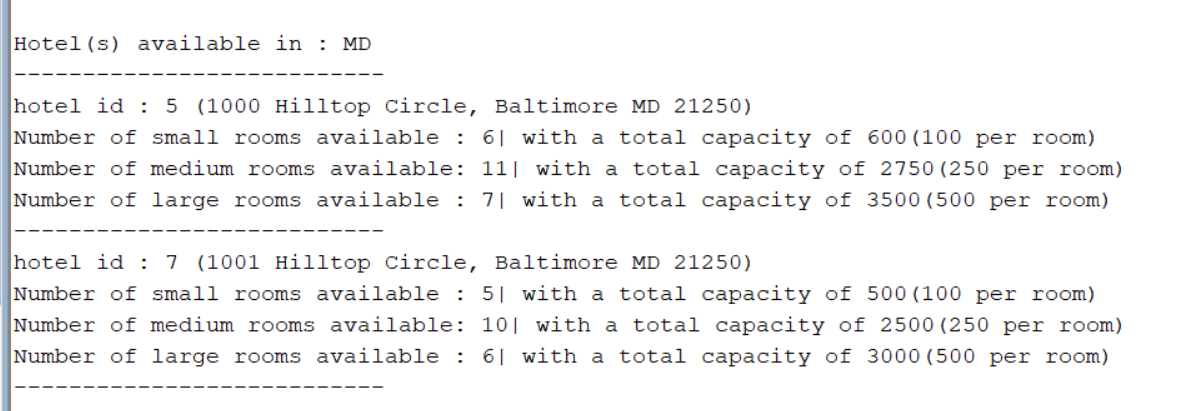
when no\_data\_found

then dbms\_output.put\_line('No hotels within state: ' || hotel\_state);

END;

**1.6 - Output**

**------------------------------------------****--------------------------------------------------------------------------------------**



**--------------------------------------------------------------------------------------------------------------**

**create or replace procedure**

**reserve\_for\_event**

**(**

**hotelid in hotel.hotel\_id%type,**

**hostname in hosts.host\_name%type,**

**startdate in event\_reservation.START\_DATE%TYPE,**

**enddate in event\_reservation.END\_DATE%TYPE,**

**dateofreservation in event\_reservation.DATE\_OF\_RESERVATION%TYPE,**

**numofsmallroom in event\_reservation.NUM\_SMALL\_ROOMS%TYPE,**

**numofmediumroom in event\_reservation.NUM\_MEDIUM\_ROOMS%TYPE,**

**numoflargeroom in event\_reservation.NUM\_LARGE\_ROOMS%TYPE,**

**dis in event\_reservation.discount%type,**

**numberofpeopleattending IN event\_reservation.NUM\_OF\_GUESTS%TYPE ,**

**eventname in event\_type.event\_name%type,**

**reservationstatus in event\_reservation.event\_reservation\_status%type,**

**eventreservationid out event\_reservation.event\_reservation\_id%type**

**)**

**As**

**Num\_small\_room\_avail number;**

**Num\_medium\_room\_avail number;**

**Num\_large\_room\_avail number;**

**Small\_capacity number;**

**Medium\_capacity number;**

**Large\_capacity number;**

**eventid event\_type.event\_id%type;**

**hostid hosts.host\_id%type;**

**begin**

**select event\_id into eventid from event\_type where event\_name = eventname;**

**select host\_id into hostid from hosts where host\_name = hostname;**

**--- Using the sequence to generate the reservation ID ---**

**eventreservationid := event\_reservation\_id\_seq.nextval;**

**-- Check that each number of rooms available before updating**

**SELECT max\_small\_room\_avail, max\_medium\_room\_avail, max\_large\_room\_avail**

**INTO num\_small\_room\_avail, num\_medium\_room\_avail, Num\_large\_room\_avail**

**FROM hotel h**

**WHERE h.hotel\_id = hotelid;**

**SELECT room\_capacity\*numofsmallroom**

**into small\_capacity**

**From room\_type**

**Where room\_type = 'Small room';**

**SELECT room\_capacity\*numofmediumroom**

**into Medium\_capacity**

**From room\_type**

**Where room\_type = 'Medium room';**

**SELECT room\_capacity\*numoflargeroom**

**into large\_capacity**

**From room\_type**

**Where room\_type = 'Large room';**

**--Check to see if all everyone will fit into the rooms you are trying to reserve**

**if (numberofpeopleattending > (small\_capacity+ medium\_capacity + large\_capacity)) then**

**dbms\_output.put\_line('YOU NEED MORE ROOMS FOR RESERVATION AS PROVIDED NUMBER OF PEOPLE WILL NOT FIT INTO GIVEN NUMBER OF ROOMS ');**

**-- now that we know they want to reserve enough rooms to fit everyone, we need to**

**-- check if we actually have those rooms available.**

**ELSE**

**IF (numofsmallroom <= num\_small\_room\_avail) then**

**if (numofmediumroom <= num\_medium\_room\_avail) then**

**if (numoflargeroom <= Num\_large\_room\_avail) then**

**insert into event\_reservation**

**values (eventreservationid, hotelid, hostid, startdate, enddate, dateofreservation, numofsmallroom, numofmediumroom , numoflargeroom, dis, numberofpeopleattending, eventid,reservationstatus);**

**End if;**

**End if;**

**End if;**

**End if;**

**end;**

**--- output of the reservation confirmation ---**

**Declare**

**res\_id number;**

**Begin**

**reserve\_for\_event(5,'Kajal Pansare', date '2006-08-10', date '2006-08-13', date '2006-08-10',4, 2, 6, 5, 2,'Birthday', 'RESERVERD',res\_id);**

**Dbms\_output.put\_line('Event Reservation Completed, Event Registraion id is : ' || res\_id);**

**End;**

**CREATE OR REPLACE PROCEDURE CANCEL\_AN\_EVENT\_RESERVATION (**

**eventreservationid IN event\_reservation.event\_reservation\_id%type**

**)**

**AS**

**BEGIN**

**UPDATE EVENT\_RESERVATION SET EVENT\_RESERVATION\_STATUS = 'CANCEL' WHERE EVENT\_RESERVATION\_ID = eventreservationid;**

**END;**

**Declare**

**Begin**

**CANCEL\_AN\_EVENT\_RESERVATION(43);**

**End;**

**--------------------------------------------**

**--ShowCancelations: Print all canceled events in the event management system. Show event reservation ID, hotel name, location, eventtype, room type, dates.**

**CREATE OR REPLACE PROCEDURE SHOW\_CANCELATION**

**AS**

**cursor c1 is select er.event\_reservation\_id,er.hotel\_id, h.street\_address as hotel\_name, h.city as location,er.event\_id, et.event\_name,ero.room\_type,er.start\_date, er.end\_date**

**from hotel h, event\_type et,event\_reservation er,event\_rooms ero**

**where er.hotel\_id = h.hotel\_id and er.event\_id = et.event\_id and ero.host\_id = er.host\_id and er.event\_reservation\_status = 'CANCEL';**

**eventreservationid event\_reservation.event\_reservation\_id%type;**

**hotelid hotel.hotel\_id%type;**

**streetaddress hotel.street\_address%type;**

**city hotel.city%type;**

**eventid event\_reservation.event\_id%type;**

**eventname event\_type.event\_name%type;**

**roomtype event\_rooms.room\_type%type;**

**startdate event\_reservation.start\_date%type;**

**enddate event\_reservation.end\_date%type;**

**BEGIN**

**open c1;**

**loop**

**fetch c1 into eventreservationid,hotelid,streetaddress,city,eventid,eventname,roomtype,startdate,enddate;**

**exit when c1%notfound;**

**dbms\_output.put\_line('Event Reservation ID is: ' || eventreservationid ||' Hotel ID :' || hotelid || ' Hotel Name : '|| streetaddress || ' Location: ' ||city|| ' Event ID: ' ||eventid ||' Event Name: '||eventname||' Room Type: '||roomtype||' Start Date: ' || startdate||' End Date: '||enddate);**

**end loop;**

**close c1;**

**exception**

**when no\_data\_found then**

**dbms\_output.put\_line('No canceled events');**

**end;**

**Begin**

**SHOW\_CANCELATION;**

**End;**

**create or replace procedure change\_an\_event\_date(**

**eventreservationid in event\_reservation.event\_reservation\_id%type,**

**startdate event\_reservation.start\_date%type,**

**enddate event\_reservation.end\_date%type**

**)**

**as**

**begin**

**update event\_reservation set start\_date = startdate, end\_date = enddate where event\_reservation\_id = eventreservationid;**

**dbms\_output.put\_line('Event start date is : '||startdate||' And Event end date is: ' || enddate);**

**end;**

**begin**

**change\_an\_event\_date(43,date '2006-09-10', date '2006-09-13');**

**end;**

**Priyanka Ranade ( Member 3)**

**----------------------------------------------------------------------------------------------------------------**

**NOTE: The remaining problem sets to complete were 3.1, 3.4, and 3.5, displayed below. Problems 3.2 and 3.3 were turned in for Deliverable 3.**

**Common Functions used in 3.4 and 3.5**

--1) get\_discount

--checks if discount applicable and applies 10%

create or replace function get\_discount ( v\_event\_reservation\_id in number)

return integer IS

v\_discount integer;

v\_duration integer;

begin

select ( start\_date - date\_of\_reservation)into v\_duration from event\_reservation where event\_reservation\_id = v\_event\_reservation\_id;

v\_discount :=0;

if v\_duration >= 59

then v\_discount := 10;

end if;

return v\_discount;

exception

when no\_data\_found then

dbms\_output.put\_line('no data');

return -1;

End;

--3. Get events function

--retrieves all events for a month (given the int form as input)

create or replace FUNCTION get\_events(month\_number in integer)

RETURN SYS\_REFCURSOR

AS

my\_cursor SYS\_REFCURSOR;

BEGIN

OPEN my\_cursor FOR select event\_reservation\_id from event\_reservation

where to\_number(to\_char(start\_date, 'mm') )= month\_number;

RETURN my\_cursor;

END;

--2. Service Charge

--totals all charges applicable to services purchased for an event

create or replace function service\_charge ( v\_num\_guests in number,v\_service\_id number )

return number IS

v\_service\_charge number;

v\_person\_or\_event varchar(50);

v\_service\_price number;

begin

select per\_person\_or\_event, service\_price into v\_person\_or\_event, v\_service\_price from services where service\_id = v\_service\_id;

if v\_person\_or\_event = 'person'

then

v\_service\_charge := v\_num\_guests \* v\_service\_price;

else

v\_service\_charge := v\_service\_price;

end if;

return v\_service\_charge;

exception

when no\_data\_found then

dbms\_output.put\_line('no data');

return -1;

end;

--Event Duration Function

-- Accepts event reservation id as input and calculates duration of event by subtracting start date from end date. Returns whole number.

create or replace function get\_event\_duration ( v\_event\_reservation\_id in number)

return integer IS

v\_duration varchar(50);

begin

select (end\_date - start\_date) into v\_duration from event\_reservation where event\_reservation\_id = v\_event\_reservation\_id;

if v\_duration is null

then dbms\_output.put\_line('invalid duration');

end if;

if v\_duration = 0

then v\_duration := 1;

end if;

return v\_duration;

exception

when no\_data\_found then

dbms\_output.put\_line('no event');

return -1;

end;

**Problem 3.1**

1. **Code**

--Author: Priyanka Ranade

-- called from procedure change\_event\_room\_type

--Updates type of room in a reservation to another type, depending on availability in hotels tabel.

--exec statement commented out at the end of file

create or replace procedure update\_hotel(res\_id in integer, v\_hotel\_id in integer,

orig\_rt in integer, new\_rt in integer, total\_rooms in integer)

as

v\_small integer;

v\_medium integer;

v\_large integer;

new\_value integer;

old\_value integer;

x integer;

y integer;

small\_room\_id integer;

medium\_room\_id integer;

large\_room\_id integer;

begin

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

small\_room\_id := 1;

medium\_room\_id := 2;

large\_room\_id := 3;

-- Do the query to get current values

select max\_small\_room\_avail, max\_medium\_room\_avail, max\_large\_room\_avail

into v\_small,v\_medium,v\_large

from hotel where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('S:' || v\_small || ' ,M:' || v\_medium || ', L:' || v\_large);

-- check if rooms are available

if (new\_rt = small\_room\_id) then

if ( v\_small >= total\_rooms) then

-- original small

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

-- update small = small - total\_romms

x := v\_small - total\_rooms;

update hotel set max\_small\_room\_avail = x

where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('Subtracting from small rooms:' || x );

else

dbms\_output.put\_line('No more small rooms available');

end if;

end if;

if (new\_rt = medium\_room\_id) then

if ( v\_medium >= total\_rooms) then

-- original small

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

-- update small = small - total\_romms

x := v\_medium - total\_rooms;

update hotel set max\_medium\_room\_avail = x

where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('Subtracting medium rooms:' || x );

else

dbms\_output.put\_line('No more small rooms available');

end if;

end if;

if (new\_rt = large\_room\_id) then

if ( v\_large >= total\_rooms) then

-- original small

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

-- update small = small - total\_romms

x := v\_large - total\_rooms;

update hotel set max\_large\_room\_avail = x

where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('Subtracting large rooms:' || x );

else

dbms\_output.put\_line('No more large rooms available');

end if;

end if;

-- original room ------------------------

if (orig\_rt = small\_room\_id) then

-- original small

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

-- update small = small - total\_romms

x := v\_small + total\_rooms;

update hotel set max\_small\_room\_avail = x

where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('Adding to small rooms to:' || x );

end if;

if (orig\_rt = medium\_room\_id) then

-- original small

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

-- update small = small - total\_romms

x := v\_medium + total\_rooms;

update hotel set max\_medium\_room\_avail = x

where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('Adding to medium rooms to:' || x );

end if;

if (orig\_rt = large\_room\_id) then

-- original small

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

-- update small = small - total\_romms

x := v\_large + total\_rooms;

update hotel set max\_large\_room\_avail = x

where hotel\_id = v\_hotel\_id;

dbms\_output.put\_line('Adding to large rooms :' || x );

end if;

--

-- original room ------------------------

if ( orig\_rt = small\_room\_id) then

dbms\_output.put\_line('Original room type small');

y :=v\_small + total\_rooms;

update hotel set max\_small\_room\_avail = y

where hotel\_id = v\_hotel\_id;

end if;

if ( orig\_rt = medium\_room\_id) then

dbms\_output.put\_line('Original room type medium');

y := v\_medium + total\_rooms;

update hotel set max\_medium\_room\_avail = y

where hotel\_id = v\_hotel\_id;

end if;

if ( orig\_rt = large\_room\_id) then

dbms\_output.put\_line('Original room type large');

y :=v\_large + total\_rooms;

update hotel set max\_large\_room\_avail = y

where hotel\_id = v\_hotel\_id;

end if;

end;

-- res hotel old new total

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

--Code for 3.1 part 2

--Author: Priyanka Ranade

--If availability condition is met, procedure subtracts changed room from hotels tabel, and adds it to event stable,

--while it adds original room type to hotel and subtracts from event

create or replace procedure change\_event\_room\_type( v\_hotel\_id in integer,

v\_reservation\_id in integer,

v\_orig\_room\_id\_type in integer,v\_update\_room\_id\_type in integer,

v\_rooms\_requested in integer, start\_date in date, end\_date in date)

is

v\_sm integer;

v\_md integer;

v\_lg integer;

begin

-- Small room

-- Medium room

-- Large room

--select event\_id,room\_type, room\_count from event\_rooms where

-- room\_type = 'Medium room' and event\_id = 2;

-- Get Room type string from event\_rooms

update\_hotel(v\_reservation\_id, v\_hotel\_id, v\_orig\_room\_id\_type, v\_update\_room\_id\_type , v\_rooms\_requested);

-- update\_hotel(v\_reservation\_id, Medium room, Small room ,3);

-- Update EVENT\_ROOMS TABLE

-- event\_id host\_id room\_type, room\_count

-- 2 2 Medium room 2 <----- before

-- 2 2 Small room 3 <----- after

update event\_rooms

set room\_type = get\_room\_type\_string(v\_update\_room\_id\_type),

room\_count = v\_rooms\_requested

where event\_id = v\_reservation\_id;

dbms\_output.put\_line('updating event\_rooms from room type ' || v\_update\_room\_id\_type

|| ' with ' || v\_rooms\_requested || ' Rooms');

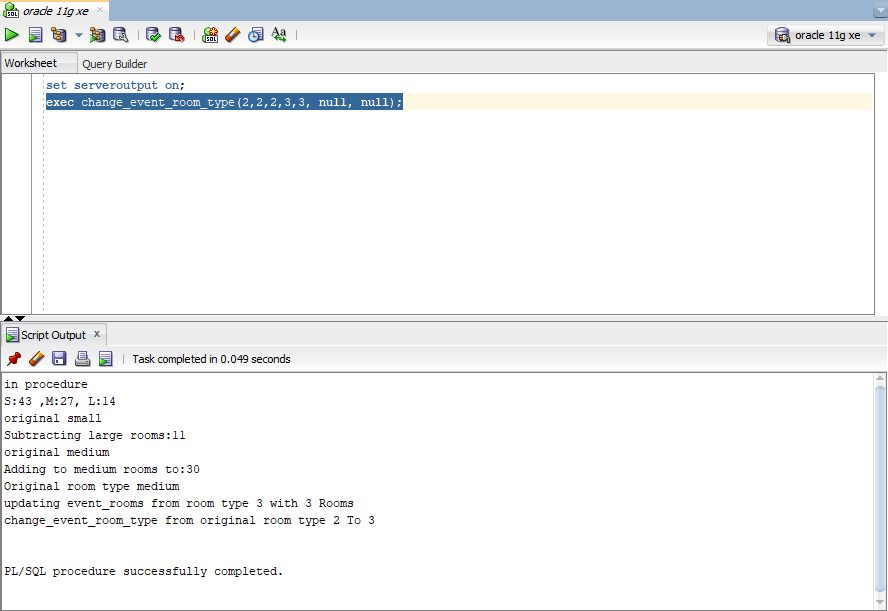
dbms\_output.put\_line('change\_event\_room\_type from ' || 'original room type ' || v\_orig\_room\_id\_type

|| ' To ' || v\_update\_room\_id\_type );

end;

--exec change\_event\_room\_type(2,2,2,3,3, null, null);

**2. Screenshot of Final Output (Combines both code snippets above)**

****

**3.4**

1. **Code**

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

--code 3.2

--Author: Priyanka Ranade

--Calculates and displays total income from all sources of all hotels given month (integer version) as input

--Totals printed by month, and for each month by event and service type, and if no events, shows appropriate output

--implicit cursor to execute commented out at the end

create or replace function monthly\_income\_report (v\_event\_id in integer)

return number is

v\_total\_return number;

begin

DECLARE

num\_guests integer;

num\_days integer;

sr integer;

mr integer;

lr integer;

room\_total number;

service\_id number;

service\_total number;

tmp number;

total\_discount\_percent number;

total\_discount\_amount number;

hostname varchar(50);

small\_room\_price number;

medium\_room\_price number;

large\_room\_price number;

sd date;

st varchar(50);

numitems number;

sp number;

tpd number;

total\_bill number;

total\_service number;

v\_event\_name varchar(50);

cursor c1 is

select service\_id from services\_for\_events where event\_reservation\_id = v\_event\_id;

-- cursor to find service total date,

cursor c2 is

select service\_date,service\_type, num\_event\_items,service\_price,

(num\_event\_items \* service\_price) as service\_total\_per\_day

from services\_for\_events, services

where services\_for\_events.service\_id = services.service\_id

and event\_reservation\_id = v\_event\_id;

begin

num\_days := get\_event\_duration (v\_event\_id);

dbms\_output.put\_line('Creating bill for eventid:' || v\_event\_id);

-- Get event name ----

select event\_name into v\_event\_name from event\_type, event\_reservation

where event\_reservation.event\_id=event\_type.event\_id and event\_reservation.event\_reservation\_id = v\_event\_id;

dbms\_output.put\_line('event type:' || v\_event\_name);

-- Event room number(s), rate per day and possibly multiple rooms

-- (if someone reserved several rooms)

select num\_small\_rooms, num\_medium\_rooms, num\_large\_rooms into sr, mr, lr from event\_reservation

where event\_reservation\_id = v\_event\_id ;

-- Get the rate per day for each room type

select room\_price into small\_room\_price from room\_type

where room\_id =1;

select room\_price into medium\_room\_price from room\_type

where room\_id =2;

select room\_price into large\_room\_price from room\_type

where room\_id =3;

-- find room total

room\_total :=0;

select num\_of\_guests, num\_small\_rooms, num\_medium\_rooms, num\_large\_rooms into num\_guests, sr, mr, lr from event\_reservation where event\_reservation\_id = get\_event\_duration (2);

-- dbms\_output.put\_line('small rooms ' || sr || ', medium ' || mr || ' large ' || lr);

room\_total := (room\_charge(num\_days,1) \* sr) + (room\_charge(num\_days,2) \* mr ) + (room\_charge(num\_days, 3) \* lr);

--dbms\_output.put\_line('room total:' || Room\_total);

-- Get service by day

-- Services rendered per date, type, and amount

total\_bill := 0;

total\_service := 0;

open c2;

loop

fetch c2 into sd, st, numitems, sp, tpd;

EXIT WHEN c2%notfound;

dbms\_output.put\_line('service date:' ||sd || ', Service Type:' || st || ', number of items:' || numitems || ' ,service price $' || sp || ', Total price per day:$' ||tpd );

total\_service:= total\_service + tpd;

end loop;

close c2;

service\_total :=0;

open c1;

-- get event duration

loop

fetch c1 into service\_id ;

EXIT WHEN c1%notfound;

-- dbms\_output.put\_line(' service id====:' || service\_id);

-- calculate service charge for service id

service\_total := service\_total + service\_charge(10 ,service\_id);

-- dbms\_output.put\_line('service total:' || service\_total);

end loop;

close c1;

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

total\_discount\_percent := get\_discount (v\_event\_id);

dbms\_output.put\_line('Discount applied:' || total\_discount\_percent || '%');

-- calculate discount amoutn

-- find out if need to give discount

total\_discount\_percent := get\_discount (v\_event\_id);

if total\_discount\_percent > 0

then

total\_discount\_amount:=room\_total \*(total\_discount\_percent/100);

room\_total:= room\_total - total\_discount\_amount;

dbms\_output.put\_line('Discount amount:' || total\_discount\_amount);

dbms\_output.put\_line('Room charge after discountt:' || room\_total);

end if;

total\_bill := room\_total + total\_service;

dbms\_output.put\_line('Bill After Discount:$' || total\_bill);

v\_total\_return :=0;

v\_total\_return:= total\_bill;

end;

return v\_total\_return;

end;

--declare

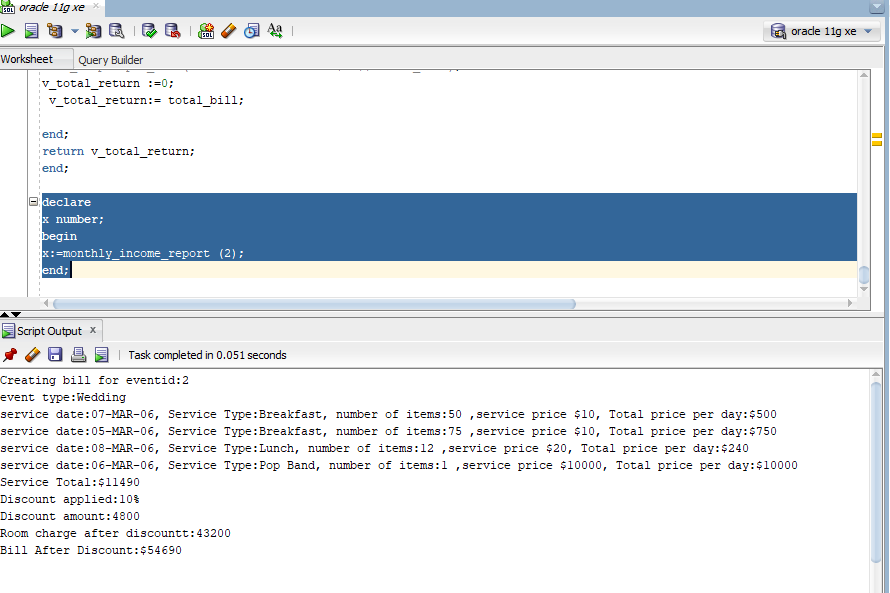
--x number;

--begin

--x:=monthly\_income\_report (2);

--end;

**2. Screenshot of Output**

****

**3.5**

1. **Code**

--loops through month and calls monthly\_bill\_for\_event

create or replace procedure monthly\_income ( v\_month in integer)

is

l\_cursor SYS\_REFCURSOR;

y integer;

x number;

monthname varchar(50);

begin

l\_cursor:=get\_events(v\_month);

SELECT TO\_CHAR(TO\_DATE(v\_month, 'MM'), 'MONTH') into monthname FROM DUAL;

loop

fetch l\_cursor into y;

IF l\_cursor%RowCount = 0 then

dbms\_output.put\_line('no data found for this month: ' || monthname);

end if;

EXIT WHEN l\_cursor%NOTFOUND;

-- get monthly bill

x:=0;

x := monthly\_bill\_for\_event(y);

dbms\_output.put\_line('Total bill:' || x);

end loop;

close l\_cursor;

end;

**--monthly\_bill\_for\_event**

--adds room prices and services for events in a month

create or replace function monthly\_bill\_for\_event (v\_event\_id in integer)

return number is

v\_total\_return number;

begin

DECLARE

num\_guests integer;

num\_days integer;

sr integer;

mr integer;

lr integer;

room\_total number;

service\_id number;

service\_total number;

tmp number;

total\_discount\_percent number;

total\_discount\_amount number;

hostname varchar(50);

small\_room\_price number;

medium\_room\_price number;

large\_room\_price number;

sd date;

st varchar(50);

numitems number;

sp number;

tpd number;

total\_bill number;

total\_service number;

v\_event\_name varchar(50);

cursor c1 is

select service\_id from services\_for\_events where event\_reservation\_id = v\_event\_id;

-- cursor to find service total date,

cursor c2 is

select service\_date,service\_type, num\_event\_items,service\_price,

(num\_event\_items \* service\_price) as service\_total\_per\_day

from services\_for\_events, services

where services\_for\_events.service\_id = services.service\_id

and event\_reservation\_id = v\_event\_id;

begin

num\_days := get\_event\_duration (v\_event\_id);

dbms\_output.put\_line('Creating bill for eventid:' || v\_event\_id);

-- Get event name ----

select event\_name into v\_event\_name from event\_type, event\_reservation

where event\_reservation.event\_id=event\_type.event\_id and event\_reservation.event\_reservation\_id = v\_event\_id;

dbms\_output.put\_line('event type:' || v\_event\_name);

-- Event room number(s), rate per day and possibly multiple rooms

-- (if someone reserved several rooms)

select num\_small\_rooms, num\_medium\_rooms, num\_large\_rooms into sr, mr, lr from event\_reservation

where event\_reservation\_id = v\_event\_id ;

-- Get the rate per day for each room type

select room\_price into small\_room\_price from room\_type

where room\_id =1;

select room\_price into medium\_room\_price from room\_type

where room\_id =2;

select room\_price into large\_room\_price from room\_type

where room\_id =3;

-- find room total

room\_total :=0;

select num\_of\_guests, num\_small\_rooms, num\_medium\_rooms, num\_large\_rooms into num\_guests, sr, mr, lr from event\_reservation where event\_reservation\_id = get\_event\_duration (2);

-- dbms\_output.put\_line('small rooms ' || sr || ', medium ' || mr || ' large ' || lr);

room\_total := (room\_charge(num\_days,1) \* sr) + (room\_charge(num\_days,2) \* mr ) + (room\_charge(num\_days, 3) \* lr);

--dbms\_output.put\_line('room total:' || Room\_total);

-- Get service by day

-- Services rendered per date, type, and amount

total\_bill := 0;

total\_service := 0;

open c2;

loop

fetch c2 into sd, st, numitems, sp, tpd;

EXIT WHEN c2%notfound;

dbms\_output.put\_line('service date:' ||sd || ', Service Type:' || st || ', number of items:' || numitems || ' ,service price $' || sp || ', Total price per day:$' ||tpd );

total\_service:= total\_service + tpd;

end loop;

close c2;

service\_total :=0;

open c1;

-- get event duration

loop

fetch c1 into service\_id ;

EXIT WHEN c1%notfound;

-- dbms\_output.put\_line(' service id====:' || service\_id);

-- calculate service charge for service id

service\_total := service\_total + service\_charge(10 ,service\_id);

-- dbms\_output.put\_line('service total:' || service\_total);

end loop;

close c1;

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

total\_discount\_percent := get\_discount (v\_event\_id);

dbms\_output.put\_line('Discount applied:' || total\_discount\_percent || '%');

-- calculate discount amoutn

-- find out if need to give discount

total\_discount\_percent := get\_discount (v\_event\_id);

if total\_discount\_percent > 0

then

total\_discount\_amount:=room\_total \*(total\_discount\_percent/100);

room\_total:= room\_total - total\_discount\_amount;

dbms\_output.put\_line('Discount amount:' || total\_discount\_amount);

dbms\_output.put\_line('Room charge after discountt:' || room\_total);

end if;

total\_bill := room\_total + total\_service;

dbms\_output.put\_line('Bill After Discount:$' || total\_bill);

v\_total\_return :=0;

v\_total\_return:= total\_bill;

end;

return v\_total\_return;

end;

--MAIN FILE--execute code above at once.

--loops through each month and calculates event expenses as expressed in code above.

--exec statement commented out at end of file

create or replace procedure execute\_monthly\_invoice as

monthname varchar(50);

begin

for mnt in 1..12 loop

SELECT TO\_CHAR(TO\_DATE(mnt, 'MM'), 'MONTH') into monthname FROM DUAL;

dbms\_output.put\_line('Creating monthly income for:' || monthname );

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

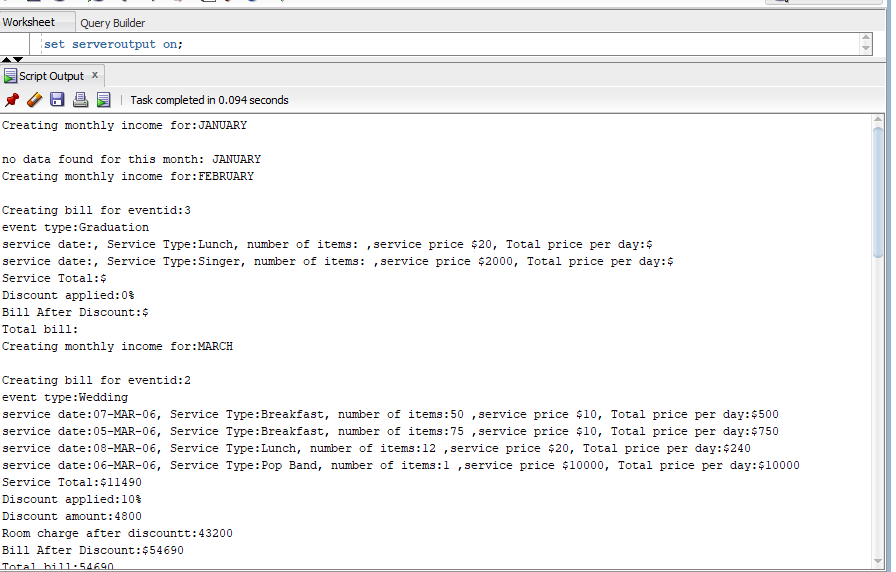
monthly\_income(mnt);

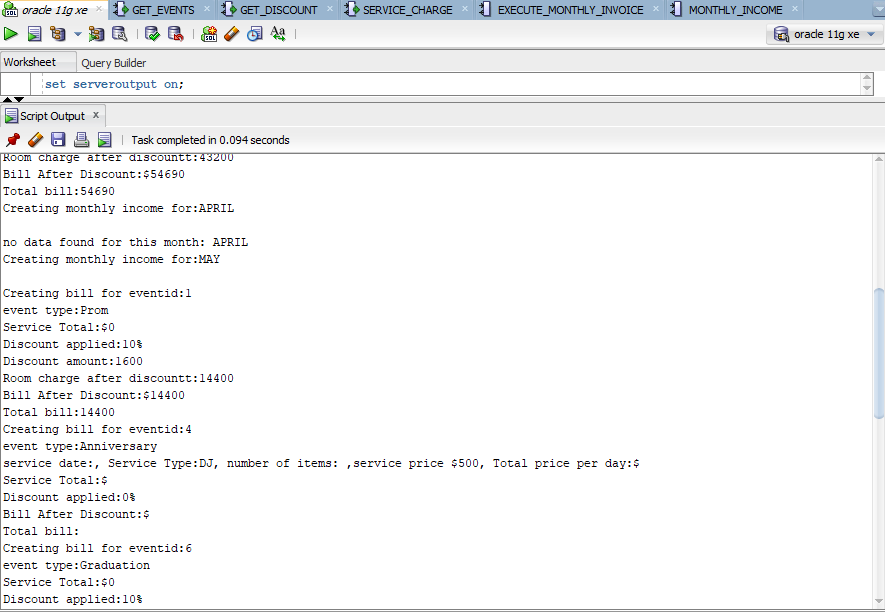
end loop;

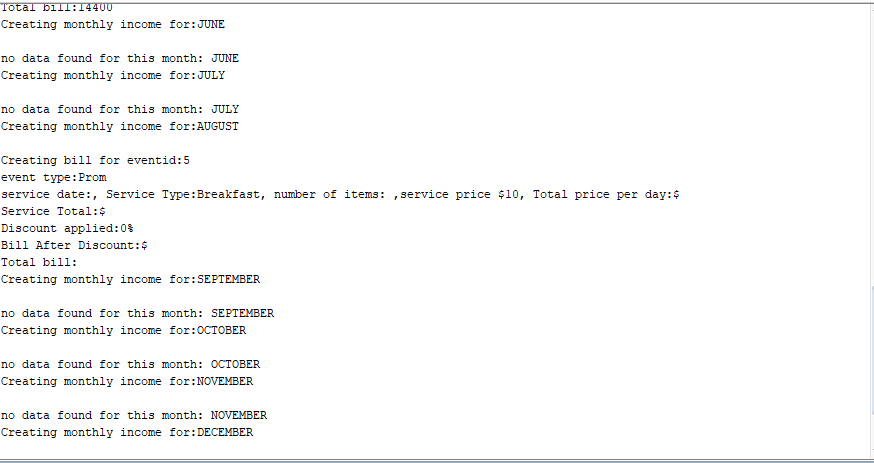
End;

--exec execute\_monthly\_invoice;

1. **Screenshots of Output**

****

****

****

**----------------------------------------------------------------------------------------------------------------**

Solution 17:

set serveroutput on;

create or replace procedure service\_to\_an\_event

(

eventreservationid in event\_reservation.event\_reservation\_id%type,

specificservice in services.service\_type%type,

serv\_date in date

)

is

var\_serv services%rowtype;

service\_type varchar(100);

tot\_amt number;

no\_of\_people int;

begin

select service\_id, service\_type, service\_price, per\_person\_or\_event

into var\_serv

from services

where service\_type = specificservice;

select num\_of\_guests

into no\_of\_people

from event\_reservation

where event\_reservation\_id=eventreservationid;

if(var\_serv.per\_person\_or\_event = 'person')then

tot\_amt := no\_of\_people \* var\_serv.service\_price;

elsif(var\_serv.per\_person\_or\_event = 'event')then

tot\_amt := var\_serv.service\_price;

else

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

end if;

if(specificservice = 'Breakfast') then

insert into services\_for\_events values (eventreservationid, var\_serv.service\_id, serv\_date, no\_of\_people, tot\_amt);

elsif (specificservice = 'Lunch') then

insert into services\_for\_events values (eventreservationid, var\_serv.service\_id, serv\_date, no\_of\_people, tot\_amt);

else

insert into services\_for\_events values (eventreservationid, var\_serv.service\_id, serv\_date, 1, tot\_amt);

end if;

end;

exec service\_to\_an\_event(1,’Breakfast’,to\_date('04-MAY-06','DD-MON-YY'));

Solution 20:

CREATE OR REPLACE PROCEDURE services\_income\_report(HID IN hotel.Hotel\_ID%TYPE)

IS

CURSOR c1 IS select sum(total) from services\_for\_events,event\_reservation,hotel

where services\_for\_events.event\_reservation\_ID = event\_reservation.event\_reservation\_ID

and hotel.hotel\_id=event\_reservation.hotel\_ID and hotel.Hotel\_ID=HID;

Services\_Amount float;

BEGIN

OPEN c1;

LOOP

FETCH c1 into Services\_Amount;

EXIT WHEN c1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('total amount for the services in the given hotel is :' ||Services\_Amount);

END LOOP;

close c1;

--Handling the exception ---

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Total services amount cannot be calculated for the given hotel !');

END;

exec services\_income\_report();

create or replace PROCEDURE services\_income\_report(HID IN hotel.Hotel\_ID%TYPE)

IS

CURSOR c1 IS select sum(total) from services\_for\_events,event\_reservation,hotel

where services\_for\_events.event\_reservation\_ID = event\_reservation.event\_reservation\_ID

and hotel.hotel\_id=event\_reservation.hotel\_ID and hotel.Hotel\_ID=HID;

Services\_Amount float;

BEGIN

select sum(total)

into Services\_Amount

from services\_for\_events,event\_reservation,hotel

where services\_for\_events.event\_reservation\_ID = event\_reservation.event\_reservation\_ID

and hotel.hotel\_id=event\_reservation.hotel\_ID and hotel.Hotel\_ID=HID;

DBMS\_OUTPUT.PUT\_LINE('total amount for the services in the given hotel is :' ||Services\_Amount);

--OPEN c1;

--LOOP

-- FETCH c1 into Services\_Amount;

-- EXIT WHEN c1%NOTFOUND;

-- DBMS\_OUTPUT.PUT\_LINE('total amount for the services in the given hotel is :' ||Services\_Amount);

--END LOOP;

--close c1;

--Handling the exception ---

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Total services amount cannot be calculated for the given hotel !');

END;

Solution 21:

create or replace FUNCTION get\_events\_in\_state(h\_ID in integer)

RETURN SYS\_REFCURSOR

AS

my\_cursor SYS\_REFCURSOR;

BEGIN

OPEN my\_cursor FOR select event\_reservation\_id from event\_reservation

where hotel\_ID= h\_ID;

RETURN my\_cursor;

END;

select get\_events\_in\_state (1)

from dual;

create or replace procedure income\_per\_state ( v\_h\_ID in integer)

is

l\_cursor SYS\_REFCURSOR;

y integer;

x number;

h\_ID varchar(50);

state\_name varchar(50);

begin

l\_cursor:=get\_events\_in\_state(v\_h\_ID);

SELECT v\_h\_ID into h\_ID FROM DUAL;

select state into state\_name from hotel , event\_reservation

where event\_reservation.hotel\_id = hotel.hotel\_ID and hotel.hotel\_ID=h\_ID;

loop

fetch l\_cursor into y;

IF l\_cursor%RowCount = 0 then

dbms\_output.put\_line('no data found for this state: ' || state\_name );

end if;

EXIT WHEN l\_cursor%NOTFOUND;

-- get monthly bill

x:=0;

x := monthly\_bill\_for\_event(y);

dbms\_output.put\_line('Total bill:' || x);

end loop;

close l\_cursor;

end;