August 3, 2025

Initial Table Set Up

-- drops tables for testing, i had to reconnect to db to test

-- Unsure if we need this

drop table reservation\_room cascade constraints;

drop table service cascade constraints;

drop table reservation cascade constraints;

drop table guest cascade constraints;

drop table room cascade constraints;

drop table hotel cascade constraints;

-- hotel table (hid is pk)

create table hotel (

hid number primary key,

hname varchar2(50),

haddress varchar2(50),

hcity varchar2(50),

hstate varchar2(50),

hphone number

);

-- rid is pk, hid fk

create table room (

rid number primary key,

hid number,

rnumber number,

rtype varchar2(50),

r\_ratePerNight number,

r\_rateType number,

foreign key (hid) references hotel(hid)

);

-- gid is pk

create table guest (

gid number primary key,

gname varchar2(50),

gaddress varchar2(50),

gphone number,

g\_creditCard varchar2(50)

);

-- resid is pk

-- gid & hid are fk

create table reservation (

resid number primary key,

gid number,

hid number,

resdate varchar2(50),

res\_startDateg varchar2(50),

res\_endDate varchar2(50),

res\_discount number,

foreign key (gid) references guest(gid),

foreign key (hid) references hotel(hid)

);

-- res\_roomid is pk

-- resid & rid are fk

create table reservation\_room (

res\_roomid number primary key,

resid number,

rid number,

foreign key (resid) references reservation(resid),

foreign key (rid) references room(rid)

);

-- sid pk

-- resid fk

create table service (

sid number primary key,

resid number,

stype varchar2(50),

squantity number,

sdate varchar2(50),

sprice number,

foreign key (resid) references reservation(resid)

);

-- insert sample data

insert into hotel values (1, 'Carls Hilton', '1234 Main Street', 'Baltimore', 'MD', '4444444444');

insert into hotel values (2, 'Alexs Hilton', '0000 Nope Lane', 'Bel Air', 'MD', '5555555555');

insert into hotel values (3, 'Erzas Hilton', '63362 Creativity Drive', 'Laurel', 'MD', '666666666');

-- room

insert into room values (1, 1, 101, 'Single', 100, 1);

insert into room values (2, 1, 102, 'Double', 150, 2);

insert into room values (3, 2, 201, 'Conference', 300, 3);

-- guest

insert into guest values (1, 'Barack Obama', '1234 Example Street', '1111111111', '0000000000');

insert into guest values (2, 'Pikachu', '5678 Hillstop Circle', '2222222222', '1111111111');

insert into guest values (3, 'Michael Jordan', '90123 Catnip Lane', '3333333333', '222222222');

-- reservation

insert into reservation values (1, 1, 1, '2025-01-01', '2025-04-04', '2025-05-05', 5);

insert into reservation values (2, 2, 2, '2025-02-02', '2025-05-04', '2025-06-05', 10);

insert into reservation values (3, 3, 3, '2025-03-03', '2025-06-04', '2025-07-05', 15);

-- reservation room

insert into reservation\_room values (1, 1, 1);

insert into reservation\_room values (2, 2, 2);

insert into reservation\_room values (3, 3, 3);

-- service

insert into service values (1, 1, 'Restaurant', 1, '2025-04-06', 10);

insert into service values (2, 2, 'Laundry', 2, '2025-05-06', 20);

insert into service values (3, 3, 'Movie', 3, '2025-06-06', 30);

1. Add a new hotel: Create a new hotel with appropriate information about the hotel as input parameter

-- add a hotel entities (see d1)

create or replace procedure add\_new\_hotel(

hid in number,

hname in varchar2,

haddress in varchar2,

hcity in varchar2,

hstate in varchar2,

hphone in number

) as

-- starts

Begin

insert into hotel (hid, hname, haddress, hcity, hstate, hphone)

-- inserts values

values (hid, hname, haddress, hcity, hstate, hphone);

-- print if successful

dbms\_output.put\_line('Hotel added!');

exception

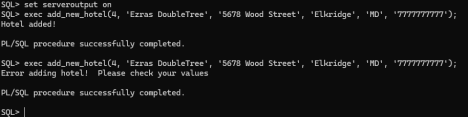
when others then

-- print a generic error on exception

dbms\_output.put\_line('Error adding hotel! Please check your values');

end;

/

**Screenshot**

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

-- Find address and return values to use

create or replace function find\_hotel\_by\_address(

addr in varchar2) return number as

hid number;

hname varchar2(50);

begin

select hid,

hname into hid,

hname from hotel

where haddress = addr;

-- print hotel name when found

dbms\_output.put\_line(hname);

-- return hid for usage later

return hid;

exception

when others then

dbms\_output.put\_line('Error finding hotel!');

-- return nothing when failed

return null;

end;

/

**Screenshot**

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

-- This adds a table which wasn't in D1

alter table hotel add (is\_sold number default 0);

-- Get hid for usage

create or replace procedure sell\_hotel(

hid in number

) as

begin

-- update is\_sold for hotel & print message

update hotel set is\_sold = 1 where hid = hid;

dbms\_output.put\_line('Hotel marked as sold!');

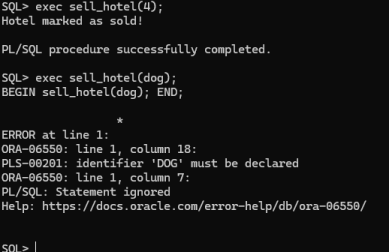
exception

when others then

dbms\_output.put\_line('Error marking hotel as sold!'); end;

/

**Screenshot**



**3. Testing**

-- THIS IS NEEDED TO PRINT EXCEPTIONS

set serveroutput on

-- Add hotel, this is just a random example

exec add\_new\_hotel(4, 'Ezras DoubleTree', '5678 Wood Street',

'Elkridge', 'MD', '7777777777');

-- run again to get an exception (duplicate)

exec add\_new\_hotel(4, 'Ezras DoubleTree', '5678 Wood Street',

'Elkridge', 'MD', '7777777777');

-- Find hotel by address

select find\_hotel\_by\_address('1234 Main Street') from dual;

-- Run with something that doesnt exist to get an exception

select find\_hotel\_by\_address('0000 Asdfjkl Lane') from dual;

-- Mark hotel as sold

exec sell\_hotel(4);

-- Not the exception exactly, but it'll fail on bad inputs

exec sell\_hotel(dog);

4. Make a reservation: Input parameters: Hotel, guest’s name, start date, end dates, 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 make\_reservation (

p\_hotel\_name in varchar2,

p\_guest\_name in varchar2,

p\_res\_date in date,

p\_start\_date in date,

p\_end\_date in date,

p\_room\_type in varchar2,

p\_discount in number,

p\_confirmation out number

)

as

v\_hid hotel.hid%type;

v\_gid guest.gid%type;

v\_rid room.rid%type;

v\_resid reservation.resid%type;

v\_res\_roomid reservation\_room.res\_roomid%type;

begin

-- find the hotel ID

select hid into v\_hid

from hotel

where hname = p\_hotel\_name;

-- find the guest ID

select gid into v\_gid

from guest

where gname = p\_guest\_name;

-- find an available room of the right type in that hotel

select rid into v\_rid

from room

where hid = v\_hid

and rtype = p\_room\_type

and rownum = 1; -- just grab one available room

-- get new reservation ID

select reservation\_seq.nextval into v\_resid from dual;

-- insert into reservation

insert into reservation (resid, gid, hid, resdate, res\_startDateg, res\_endDate, res\_discount)

values (v\_resid, v\_gid, v\_hid, p\_res\_date, p\_start\_date, p\_end\_date, p\_discount);

-- get new reservation\_room ID

select reservation\_room\_seq.nextval into v\_res\_roomid from dual;

-- link room to reservation

insert into reservation\_room (res\_roomid, resid, rid)

values (v\_res\_roomid, v\_resid, v\_rid);

-- return reservation ID as confirmation code

p\_confirmation := v\_resid;

exception

when no\_data\_found then

raise\_application\_error(-20001, 'Hotel, guest, or room type not found.');

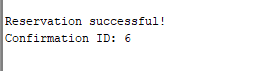
when others then

raise\_application\_error(-20002, 'Something went wrong: ' || sqlerrm);

end;

/

**Output:**

****

5. Find a reservation: Input is guest’s name and date, hotel ID. Output is reservation ID

set serveroutput on;

-- manually adding a new reservation to test the find\_reservation function

-- I'm using ID 4 since IDs 1-3 are already taken

insert into reservation values (4, 2, 2, '2025-07-01', '2025-07-10', '2025-07-15', 0);

insert into reservation\_room values (4, 4, 2);

-- turn on output so I can see the DBMS\_OUTPUT messages

set serveroutput on;

-- creating the function find\_reservation

-- this function checks if a guest has a reservation at a hotel on a specific date

-- input: guest name (p\_gname), date to check (p\_date), hotel id (p\_hid)

-- output: reservation id if found, otherwise null

create or replace function find\_reservation (

p\_gname in varchar2,

p\_date in varchar2,

p\_hid in number

) return number

as

v\_resid number; -- this will store the reservation id if we find one

begin

-- trying to find a reservation that matches the guest, hotel, and date

select r.resid into v\_resid

from reservation r

join guest g on r.gid = g.gid

where g.gname = p\_gname

and r.hid = p\_hid

and p\_date between r.res\_startDateg and r.res\_endDate;

return v\_resid; -- if we get here, the reservation was found

exception

when no\_data\_found then

-- if no reservation is found, this message shows up

dbms\_output.put\_line('no reservation found for guest ');

return null;

when others then

-- if there's some other error (like typo or table issue), this will catch it

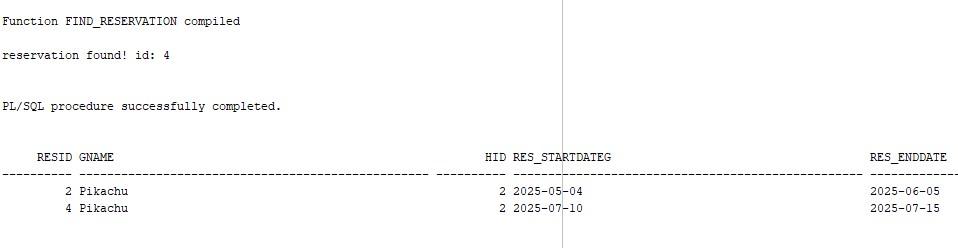
dbms\_output.put\_line('error finding reservation: ');

return null;

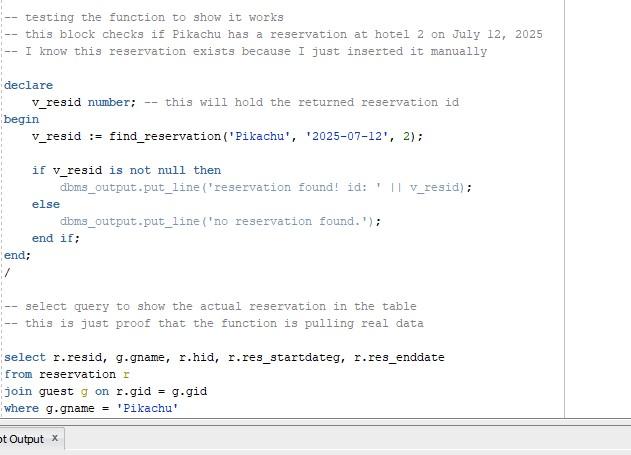
end;

/

**Output:**



**Calling Function:**

****

6. Add a service to a reservation: Input: ReservationID, specific service. Add it to the reservation for a particular date. Multiple services are allowed on a reservation for the same date.

CREATE OR REPLACE PROCEDURE AddServiceToReservation(

p\_reservation\_id IN NUMBER,

p\_service\_name IN VARCHAR2,

p\_service\_date IN VARCHAR2, -- stored as string in your table

p\_quantity IN NUMBER,

p\_price IN NUMBER

)

IS

v\_sid NUMBER;

BEGIN

-- Get next available sid

SELECT NVL(MAX(sid), 0) + 1 INTO v\_sid FROM service;

-- Insert into service table

INSERT INTO service (sid, resid, stype, squantity, sdate, sprice)

VALUES (v\_sid, p\_reservation\_id, p\_service\_name, p\_quantity, p\_service\_date, p\_price);

-- Confirm success

DBMS\_OUTPUT.PUT\_LINE('Service "' || p\_service\_name || '" added to reservation ID ' || p\_reservation\_id || ' on ' || p\_service\_date);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error adding service: ' || SQLERRM);

END;

/

--Callprocedure

BEGIN

AddServiceToReservation(1, 'Spa', '2025-08-04', 2, 60);

END;

/

--Query service table

SELECT \* FROM service WHERE resid = 1;

**Output:**



7. Show reservation details: Input the reservation ID and print all information about this Reservation

-- Create reservations table

CREATE TABLE reservations (

reservation\_id NUMBER PRIMARY KEY,

guest\_name VARCHAR2(100),

hotel\_id NUMBER,

room\_type VARCHAR2(50),

start\_date DATE,

end\_date DATE,

status VARCHAR2(20)

);

-- Create reservation\_services table

CREATE TABLE reservation\_services (

reservation\_id NUMBER,

service\_name VARCHAR2(100),

service\_date DATE

);

-- Insert sample reservation

INSERT INTO reservations (

reservation\_id, guest\_name, hotel\_id, room\_type, start\_date, end\_date, status

) VALUES (

1001, 'John Smith', 2, 'Suite',

TO\_DATE('2025-08-01', 'YYYY-MM-DD'),

TO\_DATE('2025-08-10', 'YYYY-MM-DD'),

'Active'

);

-- Insert related services

INSERT INTO reservation\_services (reservation\_id, service\_name, service\_date)

VALUES (1001, 'Restaurant', TO\_DATE('2025-08-01', 'YYYY-MM-DD'));

INSERT INTO reservation\_services (reservation\_id, service\_name, service\_date)

VALUES (1001, 'Spa', TO\_DATE('2025-08-03', 'YYYY-MM-DD'));

-- Commit changes

COMMIT;

-- Drop and recreate the procedure

BEGIN

EXECUTE IMMEDIATE 'DROP PROCEDURE ShowReservationDetails';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE OR REPLACE PROCEDURE ShowReservationDetails(

p\_reservation\_id IN NUMBER

)

IS

v\_guest\_name VARCHAR2(100);

v\_hotel\_id NUMBER;

v\_room\_type VARCHAR2(50);

v\_start\_date DATE;

v\_end\_date DATE;

v\_status VARCHAR2(20);

BEGIN

SELECT guest\_name, hotel\_id, room\_type, start\_date, end\_date, status

INTO v\_guest\_name, v\_hotel\_id, v\_room\_type, v\_start\_date, v\_end\_date, v\_status

FROM reservations

WHERE reservation\_id = p\_reservation\_id;

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

DBMS\_OUTPUT.PUT\_LINE('Reservation ID: ' || p\_reservation\_id);

DBMS\_OUTPUT.PUT\_LINE('Guest Name: ' || v\_guest\_name);

DBMS\_OUTPUT.PUT\_LINE('Hotel ID: ' || v\_hotel\_id);

DBMS\_OUTPUT.PUT\_LINE('Room Type: ' || v\_room\_type);

DBMS\_OUTPUT.PUT\_LINE('Start Date: ' || TO\_CHAR(v\_start\_date, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('End Date: ' || TO\_CHAR(v\_end\_date, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Status: ' || v\_status);

FOR rec IN (

SELECT service\_name, service\_date

FROM reservation\_services

WHERE reservation\_id = p\_reservation\_id

ORDER BY service\_date

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Service: ' || rec.service\_name || ' on ' || TO\_CHAR(rec.service\_date, 'YYYY-MM-DD'));

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(' No reservation found with ID: ' || p\_reservation\_id);

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(' Error retrieving reservation: ' || SQLERRM);

END;

/

-- Show details of the reservation you inserted

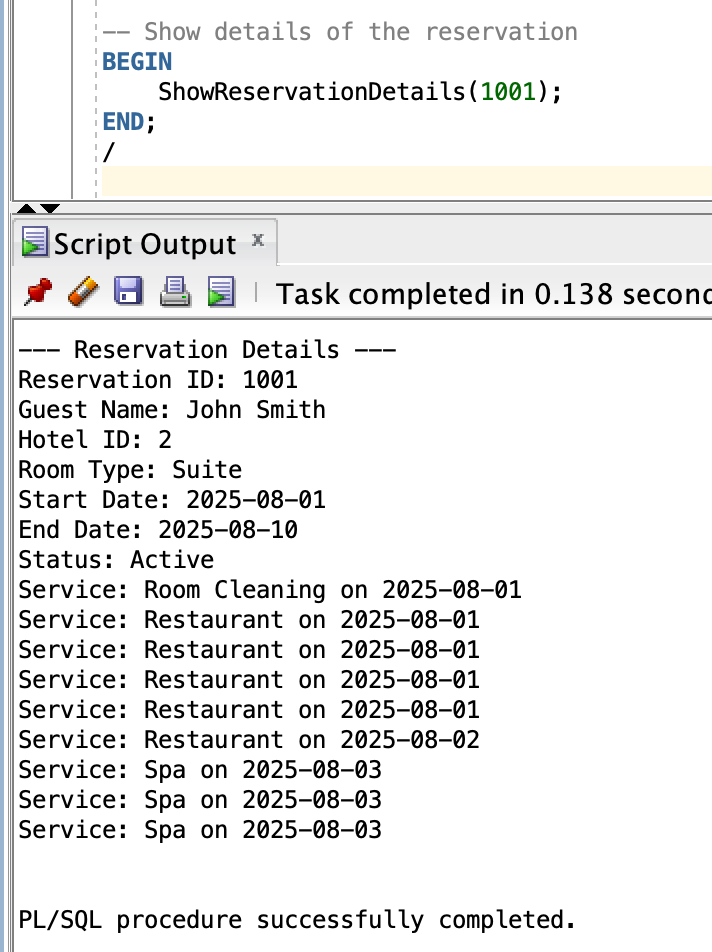
BEGIN

ShowReservationDetails(1001);

END;

/

**Output:**

****

8. Cancel a reservation: Input the reservationID and mark the reservation as cancelled (do NOT delete it)

ALTER TABLE reservation ADD status VARCHAR2(20);

CREATE OR REPLACE PROCEDURE CancelReservation (

p\_res\_id IN NUMBER

) AS

BEGIN

UPDATE reservation

SET status = 'CANCELLED'

WHERE resid = p\_res\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Reservation cancelled: ID ' || p\_res\_id);

END;

/

-- Test Case

BEGIN

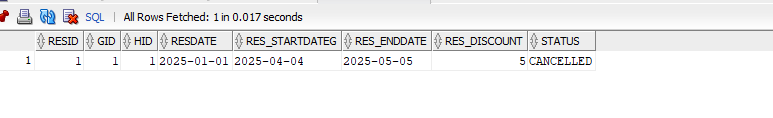
CancelReservation(1);

END;

-- Confirm cancellation

SELECT \* FROM reservation WHERE resid = 1;

**Output:**



9. 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 SQL Code

CREATE OR REPLACE PROCEDURE ChangeReservationDates (

p\_res\_id IN NUMBER,

p\_new\_start IN VARCHAR2,

p\_new\_end IN VARCHAR2

) AS

v\_hid NUMBER;

v\_rtype VARCHAR2(50);

v\_conflicts NUMBER;

BEGIN

SELECT r.hid, ro.rtype

INTO v\_hid, v\_rtype

FROM reservation r

JOIN reservation\_room rr ON r.resid = rr.resid

JOIN room ro ON rr.rid = ro.rid

WHERE r.resid = p\_res\_id

AND (r.status IS NULL OR r.status != 'CANCELLED');

SELECT COUNT(\*) INTO v\_conflicts

FROM reservation r2

JOIN reservation\_room rr2 ON r2.resid = rr2.resid

JOIN room ro2 ON rr2.rid = ro2.rid

WHERE r2.hid = v\_hid

AND ro2.rtype = v\_rtype

AND (r2.status IS NULL OR r2.status != 'CANCELLED')

AND r2.resid <> p\_res\_id

AND (

(p\_new\_start BETWEEN r2.res\_startDateg AND r2.res\_endDate) OR

(p\_new\_end BETWEEN r2.res\_startDateg AND r2.res\_endDate) OR

(r2.res\_startDateg BETWEEN p\_new\_start AND p\_new\_end) OR

(r2.res\_endDate BETWEEN p\_new\_start AND p\_new\_end)

);

IF v\_conflicts > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Cannot change dates. Conflict found for room type ' || v\_rtype);

ELSE

UPDATE reservation

SET res\_startDateg = p\_new\_start, res\_endDate = p\_new\_end

WHERE resid = p\_res\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Reservation dates updated for ID: ' || p\_res\_id);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Reservation ID not found or cancelled: ' || p\_res\_id);

END;

/

**Test Data:**

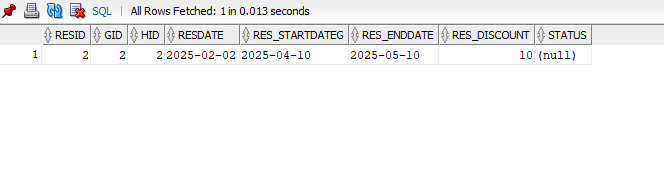
BEGIN

ChangeReservationDates(2, '2025-04-10', '2025-05-10');

END;

/

SELECT \* FROM reservation WHERE resid = 2;

**Output:**

10. Change a reservationRoomType: Input the reservation ID and change reservation room type if there is availability for that room type during the reservation’s date interval

-- Change the reservation type

CREATE OR REPLACE PROCEDURE change\_reservation\_room\_type(

p\_resid NUMBER,

p\_new\_room\_type VARCHAR2

) AS

v\_hid NUMBER;

v\_start DATE;

v\_end DATE;

v\_avail NUMBER;

BEGIN

SELECT hid, start\_date, end\_date INTO v\_hid, v\_start, v\_end

FROM reservation

WHERE resid = p\_resid;

v\_avail := available\_rooms\_during(v\_hid, p\_new\_room\_type, v\_start, v\_end);

IF v\_avail > 0 THEN

UPDATE reservation

SET room\_type = p\_new\_room\_type

WHERE resid = p\_resid;

DBMS\_OUTPUT.PUT\_LINE('Room type updated.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No available rooms of that type in the date range.');

END IF;

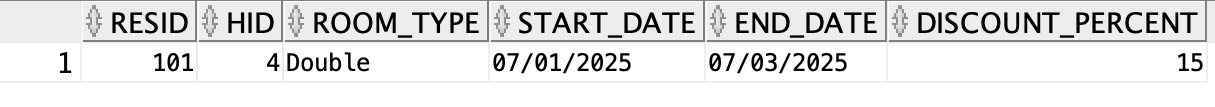
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error changing room type.');

END;

/

**Output:**

11. Available rooms at hotel: Input a specific hotel ID, a room type, and a date internal. Return the number of available rooms of that type during the interval

-- FUNCTION TO CHECK AVAILABLE ROOMS

CREATE OR REPLACE FUNCTION available\_rooms\_during(

p\_hid NUMBER,

p\_room\_type VARCHAR2,

p\_start\_date DATE,

p\_end\_date DATE

) RETURN NUMBER AS

total NUMBER;

reserved NUMBER;

BEGIN

SELECT total\_rooms INTO total

FROM room

WHERE hid = p\_hid AND room\_type = p\_room\_type;

SELECT COUNT(\*) INTO reserved

FROM reservation

WHERE hid = p\_hid AND room\_type = p\_room\_type

AND (start\_date <= p\_end\_date AND end\_date >= p\_start\_date);

RETURN total - reserved;

EXCEPTION

WHEN OTHERS THEN

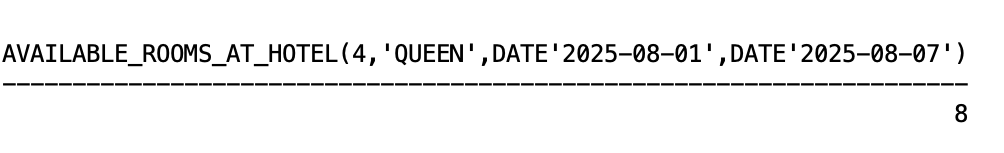
DBMS\_OUTPUT.PUT\_LINE('Error checking availability.');

RETURN -1;

END;

/

**Output:**

****

12. AvailabilityOfRoomHotelPerInterval: Input a room type, hotel ID, date interval and return the number of available rooms during that time interval

-- To find hotel availability

CREATE OR REPLACE FUNCTION available\_rooms\_at\_hotel(

p\_hid NUMBER,

p\_room\_type VARCHAR2,

p\_start\_date DATE,

p\_end\_date DATE

) RETURN NUMBER AS

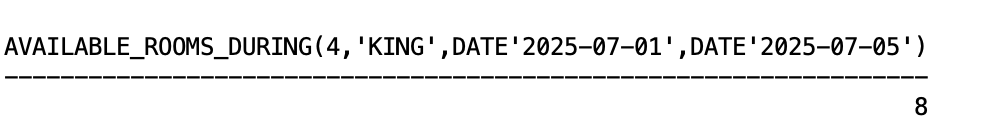
BEGIN

RETURN available\_rooms\_during(p\_hid, p\_room\_type, p\_start\_date, p\_end\_date);

END;

/

**Output:**

****

13. RoomCheckoutReceipt: Input: ReservationID Output: a. Guest name b. Room number, rate per day and possibly multiple rooms (if someone reserved several rooms) c. Services rendered per date, type, and amount d. Discounts applied (if any) e. Total amount to be paid

SELECT

r.resid AS reservation\_id,

g.gname AS guest\_name,

ro.rnumber AS room\_number,

ro.r\_ratePerNight AS rate\_per\_day,

TO\_DATE(r.res\_endDate, 'YYYY-MM-DD') - TO\_DATE(r.res\_startDateg, 'YYYY-MM-DD') AS num\_days,

(ro.r\_ratePerNight \* (TO\_DATE(r.res\_endDate, 'YYYY-MM-DD') - TO\_DATE(r.res\_startDateg, 'YYYY-MM-DD'))) AS room\_total,

s.sdate AS service\_date,

s.stype AS service\_type,

s.sprice AS service\_amount,

r.res\_discount AS discount\_percent,

ROUND(

(

(ro.r\_ratePerNight \* (TO\_DATE(r.res\_endDate, 'YYYY-MM-DD') - TO\_DATE(r.res\_startDateg, 'YYYY-MM-DD')))

+ NVL(s.sprice, 0)

) \* (1 - NVL(r.res\_discount, 0) / 100),

2

) AS total\_amount\_due

FROM reservation r

JOIN guest g ON r.gid = g.gid

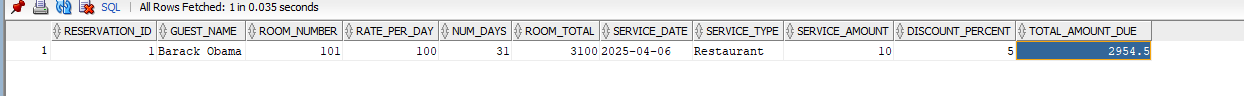
JOIN reservation\_room rr ON r.resid = rr.resid

JOIN room ro ON rr.rid = ro.rid

LEFT JOIN service s ON r.resid = s.resid

WHERE r.resid = 1

**Output:**



Explanation - In the example above I used a reservation id for “Barack Obama” which compiled the information for his reservation to give this result.

14. SoldHotels: Print all sold hotel information. Show ID, location, etc.

set serveroutput on

-- this was in my D2 but including it just incase alter table hotel add (is\_sold number default 0); -- set all hotels to unsold just in case

update hotel set is\_sold = 0;

commit;

-- set 2 hotels to sold. i modified the original SQL procedure a bit after this but it should be fine

exec sell\_hotel(2);

exec sell\_hotel(4);

-- commit to be sure

commit;

-- starts the procedure

create or replace procedure print\_sold\_hotels as

begin

-- loops & prints relevant data where is\_sold=1

for r in (

-- fields to print, any order is fine apparently

select hid, hcity, hname, hphone, hstate, haddress from hotel

where is\_sold = 1

) loop

dbms\_output.put\_line('Sold Hotel:');

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

dbms\_output.put\_line('ID: ' || r.hid);

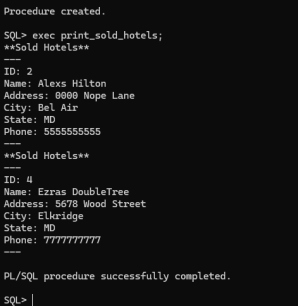
dbms\_output.put\_line('Name: ' || r.hname); dbms\_output.put\_line('Address: ' || r.haddress); dbms\_output.put\_line('City: ' || r.hcity); dbms\_output.put\_line('State: ' || r.hstate); dbms\_output.put\_line('Phone: ' || r.hphone); dbms\_output.put\_line('---');

end loop;

end;

/

**Screenshot**



**Testing**

-- run command

exec print\_sold\_hotels;\

15. ShowCancelations: Print all canceled reservations in the hotel management system.Show reservation ID, hotel name, location, guest name, room type, dates

ALTER TABLE reservation ADD res\_status VARCHAR2(20);

UPDATE reservation SET res\_status = 'CANCELED' WHERE resid = 2;

UPDATE reservation SET res\_status = 'ACTIVE' WHERE resid IN (1, 3);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Canceled Reservations');

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

FOR rec IN (

SELECT r.resid,

h.hname AS hotel\_name,

h.hcity || ', ' || h.hstate AS location,

g.gname AS guest\_name,

ro.rtype AS room\_type,

r.res\_startDateg AS start\_date,

r.res\_endDate AS end\_date

FROM reservation r

JOIN hotel h ON r.hid = h.hid

JOIN guest g ON r.gid = g.gid

JOIN reservation\_room rr ON r.resid = rr.resid

JOIN room ro ON rr.rid = ro.rid

WHERE r.res\_status = 'CANCELED'

) LOOP

DBMS\_OUTPUT.PUT\_LINE(

'Reservation ID: ' || rec.resid || ' | ' ||

'Hotel: ' || rec.hotel\_name || ' | ' ||

'Location: ' || rec.location || ' | ' ||

'Guest: ' || rec.guest\_name || ' | ' ||

'Room Type: ' || rec.room\_type || ' | ' ||

'Dates: ' || rec.start\_date || ' to ' || rec.end\_date

);

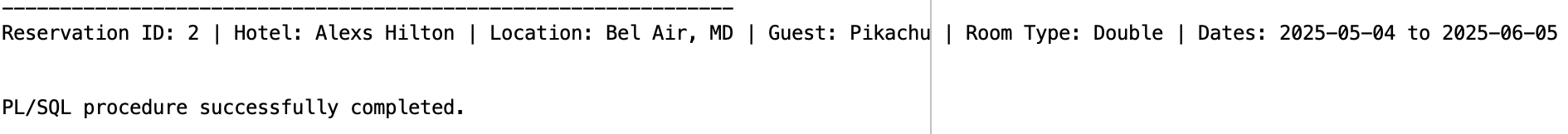
END LOOP;

END;

/

SET SERVEROUTPUT ON;

**Output:**

****

16. SpecificHotelReport: Input: hotelID, start-date, end-date. Print (for the given time interval): a. Income by room type b. Income of services, by service type c. Total income from all sources.

VARIABLE hotel\_id NUMBER;

VARIABLE start\_date VARCHAR2(20);

VARIABLE end\_date VARCHAR2(20);

EXEC :hotel\_id := 1;

EXEC :start\_date := '2024-01-01';

EXEC :end\_date := '2024-12-31';

SELECT

ct.ctname AS room\_type,

SUM(t.fare) AS income

FROM trip t

JOIN car c ON t.cid = c.cid

JOIN car\_type ct ON c.ctid = ct.ctid

WHERE ct.ctid = :hotel\_id

AND t.start\_time BETWEEN TO\_DATE(:start\_date, 'YYYY-MM-DD') AND TO\_DATE(:end\_date, 'YYYY-MM-DD') GROUP BY ct.ctname;

SELECT

'Trip Fare' AS service\_type,

SUM(t.fare) AS income

FROM trip t

JOIN car c ON t.cid = c.cid

WHERE c.ctid = :hotel\_id

AND t.start\_time BETWEEN TO\_DATE(:start\_date, 'YYYY-MM-DD') AND TO\_DATE(:end\_date, 'YYYY-MM-DD');

SELECT

SUM(t.fare) AS total\_income

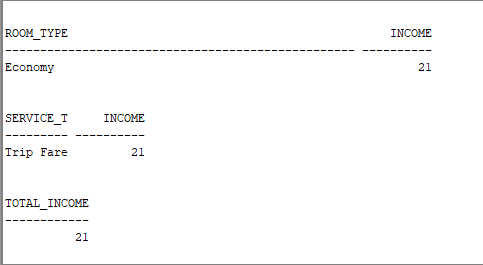
FROM trip t

JOIN car c ON t.cid = c.cid

WHERE c.ctid = :hotel\_id

AND t.start\_time BETWEEN TO\_DATE(:start\_date, 'YYYY-MM-DD') AND TO\_DATE(:end\_date, 'YYYY-MM-DD');

**Output:**



17. TotalHiltontMontlyReport: 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.

-- Monthly Report

CREATE OR REPLACE PROCEDURE total\_hilton\_monthly\_report AS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hilton Monthly Income Report (Room + Services)');

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

FOR rec IN (

SELECT

TO\_CHAR(r.start\_date, 'YYYY-MM') AS month,

r.room\_type,

NULL AS service\_type,

COUNT(\*) AS num\_res,

SUM(

(rr.price\_per\_night \* (r.end\_date - r.start\_date + 1)) \*

(1 - NVL(r.discount\_percent, 0) / 100)

) AS room\_income,

0 AS service\_income

FROM reservation r

JOIN room rr ON r.hid = rr.hid AND r.room\_type = rr.room\_type

WHERE r.hid = 4 -- Hilton

GROUP BY TO\_CHAR(r.start\_date, 'YYYY-MM'), r.room\_type

UNION ALL

SELECT

TO\_CHAR(r.start\_date, 'YYYY-MM') AS month,

r.room\_type,

s.service\_type,

COUNT(\*) AS num\_res,

0 AS room\_income,

SUM(s.base\_price \* rs.quantity) AS service\_income

FROM reservation r

JOIN reservation\_service rs ON r.resid = rs.resid

JOIN service s ON rs.service\_type = s.service\_type

WHERE r.hid = 4 -- Hilton

GROUP BY TO\_CHAR(r.start\_date, 'YYYY-MM'), r.room\_type, s.service\_type

)

LOOP

DBMS\_OUTPUT.PUT\_LINE('Month: ' || rec.month || ' | Room Type: ' || rec.room\_type ||

' | Service: ' || NVL(rec.service\_type, 'Room Charge') ||

' | Room Income: $' || rec.room\_income ||

' | Service Income: $' || rec.service\_income);

END LOOP;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error generating monthly report.');

END;

/

-- Runs Report

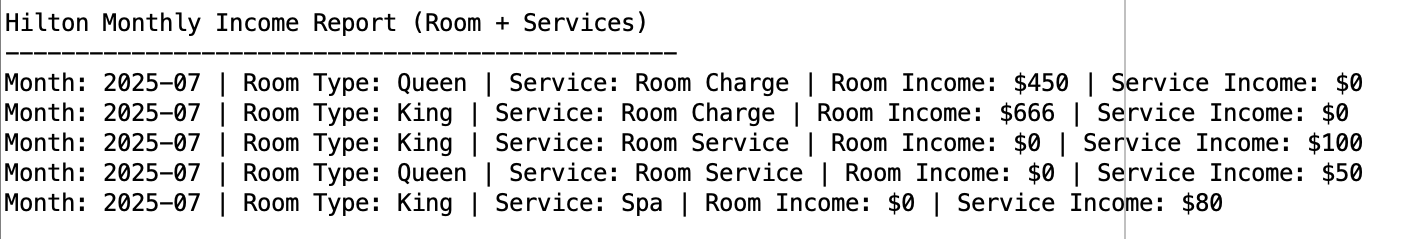
BEGIN

total\_hilton\_monthly\_report;

END;

/

**Output:**



18. TotalHiltontStateReport: Input is state. Print total income from all sources of all hotels by room type and service type in the given state. Include discounts

-- this procedure prints the total income from all sources of all hotels

create or replace procedure totalhiltonstatereport(p\_state in varchar2) as

-- grabs hotel + room + reservation + service data for a specific state

cursor state\_data is

select h.hname,

rm.rtype,

rm.r\_ratepernight,

s.stype,

s.sprice,

res.res\_discount

from hotel h

join room rm on h.hid = rm.hid

left join reservation\_room rr on rm.rid = rr.rid

left join reservation res on rr.resid = res.resid

left join service s on res.resid = s.resid

where h.hstate = p\_state;

v\_totalroomincome number := 0;

v\_totalserviceincome number := 0;

v\_nettotal number := 0;

v\_avg\_discount number := 0;

begin

dbms\_output.put\_line('--- total hilton state report for ' || p\_state || ' ---');

-- loop through each row

for rec in state\_data loop

if rec.r\_ratepernight is not null then

v\_totalroomincome := v\_totalroomincome + rec.r\_ratepernight;

end if;

if rec.sprice is not null then

v\_totalserviceincome := v\_totalserviceincome + rec.sprice;

end if;

end loop;

-- calculate average discount

select nvl(avg(res.res\_discount), 0)

into v\_avg\_discount

from reservation res

join hotel h on res.hid = h.hid

where h.hstate = p\_state;

-- apply the discount

v\_nettotal := (v\_totalroomincome + v\_totalserviceincome) \* (1 - v\_avg\_discount / 100);

-- output sum

dbms\_output.put\_line('total room income: $' || v\_totalroomincome);

dbms\_output.put\_line('total service income: $' || v\_totalserviceincome);

dbms\_output.put\_line('average discount applied: ' || to\_char(v\_avg\_discount, '999.99') || '%');

dbms\_output.put\_line('net total income: $' || to\_char(v\_nettotal, '9999.99'));

end;

/

-- test:

exec totalhiltonstatereport('MD');

**Output:**

