Team 1 Deliverable 4

Nicholas Stommel
Graig McConnell
John Richter
Eldho Regi
Jesse Jang

The following script creates the necessary tables for the project and includes insertion statements for a shared data set:

```
--Drop tables in reverse order created.
DROP TABLE Service_Event;
DROP TABLE Room_Assignment;
DROP TABLE Room;
DROP TABLE Reservation;
DROP TABLE Hotel;
DROP TABLE Customer;

CREATE TABLE Customer (
	Customer_ID INT, --in Oracle, INT type is an alias for NUMBER(38)
	First_Name VARCHAR2(20),
	Last_Name VARCHAR2(20),
	Address_Street VARCHAR2(30),
	Address_City VARCHAR2(20),
	Address_State CHAR(2),
	Address_Zipcode CHAR(5),
	Phone_Number VARCHAR2(14),
	CC_Number VARCHAR2(16),
	Email_Address VARCHAR2(40),
	CONSTRAINT Customer_PK PRIMARY KEY (Customer_ID)
);

CREATE TABLE Hotel (
	Hotel_ID INT,
	Address_Street VARCHAR2(30),
	Address_City VARCHAR2(20),
	Address_State CHAR(2),
	Address_Zipcode CHAR(5),
	Phone_Number VARCHAR2(14),
	Is_Sold CHAR(1)
	--Use CHECK to restrict values of attribute to boolean values.
	CHECK (Is_Sold IN ('T', 'F')),
	CONSTRAINT Hotel_PK PRIMARY KEY (Hotel_ID)
);
```

```
CREATE TABLE Reservation (
      Reservation_ID INT,
      Hotel_ID INT,
      Customer_ID INT,
      Reservation_Date DATE,
      Earliest_Checkin_Date DATE,
      Actual_Checkin_Date DATE,
      Latest_Checkout_Date DATE,
      Actual_Checkout_Date DATE,
      Rate_Type NUMBER(1)
      CHECK (Rate_Type IN (1, 2, 3)), --Use CHECK to restrict Rate_Type values.
      Total_Charged NUMBER(38, 2),
      Is_Cancelled CHAR(1)
      --Use CHECK to restrict values of attribute to boolean values.
      CHECK (Is_Cancelled IN ('T', 'F')),
      CONSTRAINT Reservation_PK PRIMARY KEY (Reservation_ID),
      CONSTRAINT Reservation_Hotel_FK FOREIGN KEY (Hotel_ID)
      REFERENCES Hotel(Hotel_ID)
      --When Hotel row is deleted, delete Reservation child rows as well.
      --https://www.techonthenet.com/oracle/foreign_keys/foreign_delete.php
      ON DELETE CASCADE,
      CONSTRAINT Reservation_Customer_FK FOREIGN KEY (Customer_ID)
      REFERENCES Customer(Customer_ID)
      --When Customer row is deleted, delete Reservation child rows as well.
      ON DELETE CASCADE
);

CREATE TABLE Room (
      Room_Number INT,
      Hotel_ID INT,
      Room_Type VARCHAR2(30)
      --Use CHECK to restrict possible values of Room_Type.
      CHECK (Room_Type IN ('single-room', 'double-room',
            'suite', 'luxury-suite', 'conference-room')),
      Room_Base_Cost NUMBER (6, 2),
      CONSTRAINT Room_PK PRIMARY KEY (Room_Number, Hotel_ID),
      CONSTRAINT Room_Hotel_FK FOREIGN KEY (Hotel_ID)
      REFERENCES Hotel(Hotel_ID)
      --When Hotel row is deleted, delete Room child rows as well.
      ON DELETE CASCADE
);

CREATE TABLE Room_Assignment (
      Room_Number INT,
      Hotel_ID INT,
      Reservation_ID INT,
      CONSTRAINT RA_PK PRIMARY KEY (Room_Number, Hotel_ID, Reservation_ID),
      --RA_Room_FK is a composite foreign key that refers to a composite
```

```
	--primary key in the Room table.
	CONSTRAINT RA_Room_FK FOREIGN KEY (Room_Number, Hotel_ID)
	REFERENCES Room(Room_Number, Hotel_ID)
	--When Room row is deleted, delete Room_Assignments child rows as well.
	ON DELETE CASCADE,
	CONSTRAINT RA_Reservation_FK FOREIGN KEY (Reservation_ID)
	REFERENCES Reservation(Reservation_ID)
	--When Reservation row is deleted, delete Room_Assignment child rows as well.
	ON DELETE CASCADE
);

CREATE TABLE Service_Event (
	Service_Event_ID INT,
	Reservation_ID INT,
	Service_Type VARCHAR2(20)
	--Use CHECK to restrict possible values of Service_Event.
	CHECK (Service_Type IN ('restaurant meal', 'pay-per-view movie',
		'laundry')),
	Service_Cost NUMBER(6, 2),
	Service_Date DATE,
	CONSTRAINT SE_PK PRIMARY KEY (Service_Event_ID),
	CONSTRAINT SE_Reservation_FK FOREIGN KEY (Reservation_ID)
	REFERENCES Reservation(Reservation_ID)
	--When Reservation row is deleted, delete child Service_Event rows as well.
	ON DELETE CASCADE
);

DELETE FROM Service_Event;
DELETE FROM Room_Assignment;
DELETE FROM Room;
DELETE FROM Reservation;
DELETE FROM Hotel;
DELETE FROM Customer;
--Drop the following sequences if re-inserting data.
DROP SEQUENCE Hotel_PK_Seq;
DROP SEQUENCE Customer_PK_Seq;
DROP SEQUENCE Room_PK_Seq;
DROP SEQUENCE Reservation_PK_Seq;
DROP SEQUENCE Service_Event_PK_Seq;

CREATE SEQUENCE Hotel_PK_Seq
INCREMENT BY 1
START WITH 1;
--(Hotel_ID, Address_Street, Address_City, Address_State, Address_Zipcode,
--Phone_Number, Is_Sold)
INSERT INTO Hotel VALUES(Hotel_PK_Seq.NEXTVAL, '1739 W Nursery Rd',
	'Linthicum Heights', 'MD', '21090', '410-694-0808', 'F');
INSERT INTO Hotel VALUES(Hotel_PK_Seq.NEXTVAL, '401 W Pratt St', 'Baltimore',
       'MD', '21201', '443-573-8700', 'F');
INSERT INTO Hotel VALUES(Hotel_PK_Seq.NEXTVAL, '903 Dulaney Valley Rd',
       'Towson', 'MD', '21204', '410-321-7400', 'F');
INSERT INTO Hotel VALUES(Hotel_PK_Seq.NEXTVAL, '80 Compromise St' , 'Annapolis',
       'MD', '21401', '410-268-7555', 'F');
INSERT INTO Hotel VALUES(Hotel_PK_Seq.NEXTVAL, '750 Kearny St', 'San Francisco',
       'CA', '94108', '415-433-6600', 'F');


CREATE SEQUENCE Customer_PK_Seq
INCREMENT BY 1
START WITH 1;
--(Customer_ID, First_Name, Last_Name, Address_Street, Address_City,
--Address_State, (Address_Zipcode, Phone_Number, CC_Number, Email_Address)
INSERT INTO Customer VALUES(Customer_PK_Seq.NEXTVAL, 'John', 'Doe',
       '101 Cramer Terrace', 'Annandale', 'VA', '22000', '320-162-0093',
       '4184908623156854', 'JohnDoe@gmail.com');
INSERT INTO Customer VALUES(Customer_PK_Seq.NEXTVAL, 'James', 'Kim',
       '201 Dalaney Rd', 'Houston', 'TX', '77001', '281-123-6426',
       '4556366231237652', 'JamesKim@yahoo.com');
INSERT INTO Customer VALUES(Customer_PK_Seq.NEXTVAL, 'George', 'Smith',
       '301 Mermaid Ln', 'Los Angeles', 'CA', '90005', '213-174-0985',
       '5185892910234571', 'SmithGeorge@outlook.com');
INSERT INTO Customer VALUES(Customer_PK_Seq.NEXTVAL, 'Darren', 'Johnson',
       '401 Jackson St', 'Chicago', 'IL', '60007', '773-0572-0095',
       '4885729513641561', 'DarrenJohnson@gmail.com');
INSERT INTO Customer VALUES(Customer_PK_Seq.NEXTVAL, 'Sarah', 'Jones',
       '501 Cedar Blvd', 'Charlotte', 'NC', '28202', '704-0251-6926',
       '3786240682902934', 'SarahJones@yahoo.com');

CREATE SEQUENCE Room_PK_Seq
INCREMENT BY 1
START WITH 1;
--(Room_Number, Hotel_Id, Room_Type, Room_Base_Cost)
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 1, 'single-room', 102.69);
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 1, 'luxury-suite', 152.25);
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 1, 'single-room', 102.69);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY -2;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 2, 'double-room', 120.58);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY 1;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 2, 'single-room', 102.69);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY -1;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 3, 'double-room', 120.58);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY 1;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 3, 'luxury-suite', 200.12);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY -1;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 4, 'single-room', 90.58);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY 1;
```

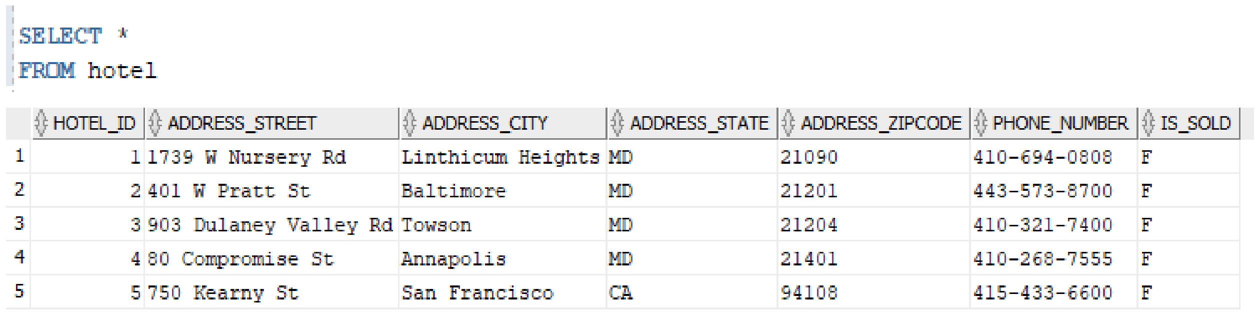
```
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 4, 'double-room', 120.58);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY -1;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 5, 'single-room', 70.58);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY 1;
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 5, 'double-room', 100.58);
INSERT INTO ROOM VALUES(Room_PK_Seq.NEXTVAL, 5, 'conference-room', 200.45);
ALTER SEQUENCE Room_PK_Seq INCREMENT BY -2;

--See room values.
SELECT * FROM room
ORDER BY hotel_id, room_number;

CREATE SEQUENCE Reservation_PK_Seq
INCREMENT BY 1
START WITH 1;
--(Reservation_ID, Hotel_ID, Customer_ID, Reservation_Date,
--Earliest_Checkin_Date, Actual_Checkin_Date, Latest_Checkout_Date,
--Actual_Checkout_Date, Rate_type, Total_Charged, Is_Cancelled)
INSERT INTO Reservation VALUES(Reservation_PK_Seq.NEXTVAL, 1, 2,
      DATE '2016-11-11', DATE '2017-01-02', DATE '2017-01-02',
      DATE '2017-01-05', DATE '2017-01-05', 3, 295.26, 'F');
INSERT INTO Reservation VALUES(Reservation_PK_Seq.NEXTVAL, 4, 1,
      DATE '2018-03-01', DATE '2018-03-10', DATE '2018-03-10', DATE '2018-03-17',
      DATE '2018-03-20', 1, 1090.22, 'F');
INSERT INTO Reservation VALUES(Reservation_PK_Seq.NEXTVAL, 3, 3,
      DATE '2019-08-02', DATE '2019-09-06', DATE '2019-09-08', DATE '2019-09-15',
      DATE '2019-09-15', 2, 1021.96, 'F');
INSERT INTO Reservation VALUES(Reservation_PK_Seq.NEXTVAL, 5, 4,
      DATE '2020-05-10', DATE '2020-05-23', DATE '2020-05-20', DATE '2020-05-27',
      DATE '2020-05-29', 2, 0, 'T');
INSERT INTO Reservation VALUES(Reservation_PK_Seq.NEXTVAL, 2, 5,
      DATE '2021-06-02', DATE '2021-06-02', DATE '2021-06-02', DATE '2021-06-05',
      DATE '2021-06-05', 3, 376.74, 'F');
INSERT INTO Reservation VALUES(Reservation_PK_Seq.NEXTVAL, 1, 3,
      DATE '2018-01-08', DATE '2018-03-09', DATE '2018-03-10', DATE '2018-03-14',
      DATE '2018-03-14', 1, 514.85, 'F');

--(Room_Number, Hotel_ID, Reservation_ID)
INSERT INTO Room_Assignment VALUES(1, 1, 1);
INSERT INTO Room_Assignment VALUES(2, 1, 1);
INSERT INTO Room_Assignment VALUES(2, 4, 2);
INSERT INTO Room_Assignment VALUES(1, 3, 3);
INSERT INTO Room_Assignment VALUES(2, 5, 4);
INSERT INTO Room_Assignment VALUES(1, 2, 5);
INSERT INTO Room_Assignment VALUES(3, 1, 6);

CREATE SEQUENCE Service_Event_PK_Seq
INCREMENT BY 1
```

```
START WITH 1;
--(Service_Event_ID, Reservation_ID, Service_Type, Service_Cost, Service_Date)
--Types: 'restaurant meal': $20, 'pay-per-view movie': $5, 'laundry': $10
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 1,
       'restaurant meal', 20, DATE '2017-01-03');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 1,
       'laundry', 10, DATE '2017-01-03');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 1,
       'laundry', 10, DATE '2017-01-04');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 2,
       'pay-per-view movie', 5, DATE '2018-03-15');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 2,
       'laundry', 10, DATE '2019-09-10');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 3,
       'pay-per-view movie', 5, DATE '2021-06-02');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 4,
       'laundry', 10, DATE '2021-06-03');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 6,
       'restaurant meal', 20, DATE '2018-03-11');
INSERT INTO Service_Event VALUES(Service_Event_PK_Seq.NEXTVAL, 6,
       'pay-per-view movie', 5, DATE '2018-03-12');
```

The operations implemented by each group member are given below, ordered by number the same way as those in the Project Operations file. For each procedure given, there are examples calling that procedure and screenshots showing correct output based on the shared data set declared above.

1. Graig [*] Add a new hotel: Create a new hotel with appropriate information about the hotel as input parameters

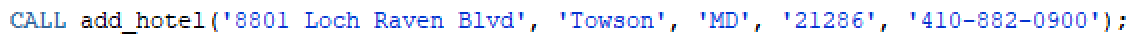
```
CREATE OR REPLACE PROCEDURE add_hotel
(
-- The procedure accepts this information as inputs
       address_street_param hotel.address_street%TYPE,
       address_city_param hotel.address_city%TYPE,
       address_state_param hotel.address_state%TYPE,
       address_zipcode_param hotel.address_zipcode%TYPE,
       phone_number_param hotel.phone_number%TYPE
)
AS
-- The procedure generates the hotel automatically and sets sold status to False.
       hotel_id_var hotel.hotel_id%TYPE;
       is_sold_var hotel.is_sold%TYPE DEFAULT 'F';
BEGIN
       SELECT hotel_pk_seq.NEXTVAL INTO hotel_id_var FROM dual;
       INSERT INTO hotel
       VALUES (hotel_id_var,
                address_street_param, address_city_param, address_state_param,
                address_zipcode_param, phone_number_param, is_sold_var);
        COMMIT;
EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
                DBMS_OUTPUT.PUT_LINE('You attempted to insert a duplicate value.');
        WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('An exception occurred.');
               ROLLBACK;
END;
/
```



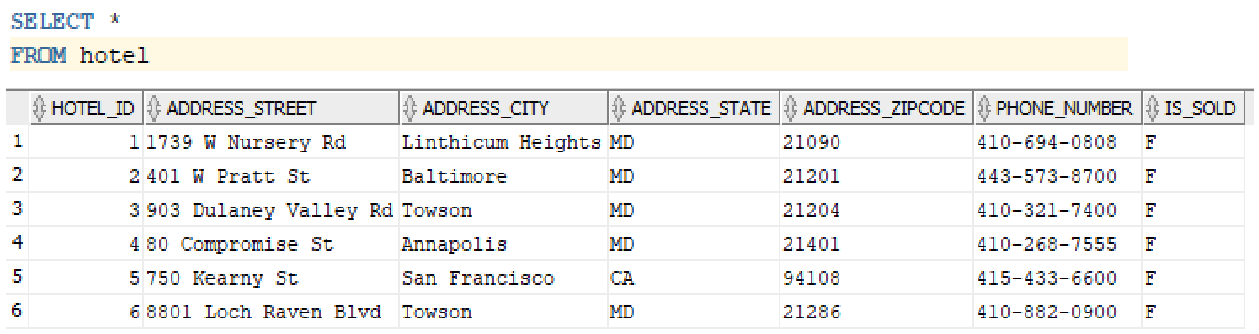
```
SELECT *
FROM hotel
```

	HOTEL_ID	ADDRESS_STREET	ADDRESS_CITY	ADDRESS_STATE	ADDRESS_ZIPCODE	PHONE_NUMBER	IS_SOLD
1	1	1739 W Nursery Rd	Linthicum Heights	MD	21090	410-694-0808	F
2	2	401 W Pratt St	Baltimore	MD	21201	443-573-8700	F
3	3	903 Dulaney Valley Rd	Towson	MD	21204	410-321-7400	F
4	4	80 Compromise St	Annapolis	MD	21401	410-268-7555	F
5	5	750 Kearny St	San Francisco	CA	94108	415-433-6600	F

Call to procedure is made and the following select statement confirms that a new hotel has been added.



```
CALL add_hotel('8801 Loch Raven Blvd', 'Towson', 'MD', '21286', '410-882-0900');
```



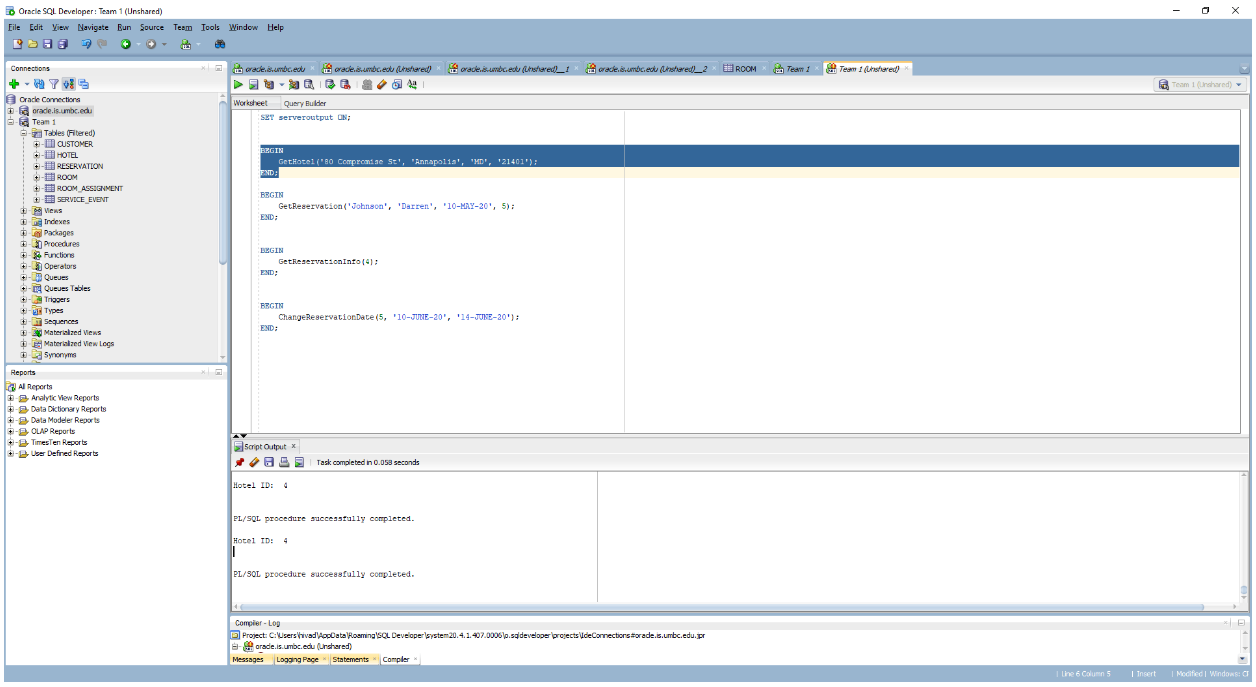
```
SELECT *
FROM hotel
```

	HOTEL_ID	ADDRESS_STREET	ADDRESS_CITY	ADDRESS_STATE	ADDRESS_ZIPCODE	PHONE_NUMBER	IS_SOLD
1	1	1739 W Nursery Rd	Linthicum Heights	MD	21090	410-694-0808	F
2	2	401 W Pratt St	Baltimore	MD	21201	443-573-8700	F
3	3	903 Dulaney Valley Rd	Towson	MD	21204	410-321-7400	F
4	4	80 Compromise St	Annapolis	MD	21401	410-268-7555	F
5	5	750 Kearny St	San Francisco	CA	94108	415-433-6600	F
6	6	8801 Loch Raven Blvd	Towson	MD	21286	410-882-0900	F

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

```
create or replace PROCEDURE GetHotel(street_in IN varchar2, city_in IN varchar2,
        state_in IN char, zip_in IN char)
--This procedure take and address as input and outputs the hotel ID
--street_in is street name, city_in is city name, state_in is state name,
--zip_in is zip code
IS
        hotel NUMBER;
BEGIN
-- Selecting the hotel ID based on paramters
```

```
        SELECT hotel_id  INTO hotel FROM hotel
        WHERE address_street = street_in
        AND address_city = city_in
        AND address_state = state_in;

-- Output the hotel ID
        DBMS_OUTPUT.PUT_LINE('Hotel ID:  ' || hotel );

EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Hotel does not exist.');
        WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Too many rows.');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('A different exception occurred.');
END;

BEGIN
        GetHotel('80 Compromise St', 'Annapolis', 'MD', '21401');
END;
```



```
SET serveroutput ON;


BEGIN
    GetHotel('80 Compromise St', 'Annapolis', 'MD', '21401');
END;

BEGIN
    GetReservation('Johnson', 'Darren', '10-MAY-20', 5);
END;


BEGIN
    GetReservationInfo(4);
END;


BEGIN
    ChangeReservationDate(5, '10-JUNE-20', '14-JUNE-20');
END;
```

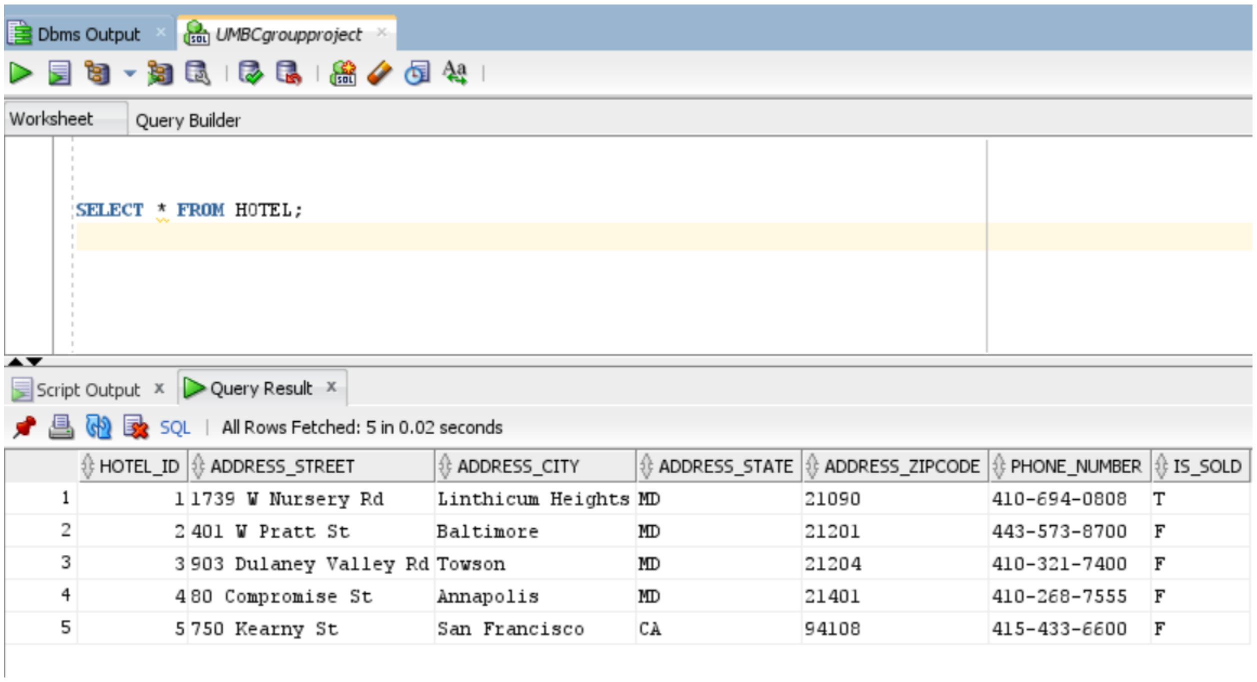
Task completed in 0.058 seconds

```
Hotel ID:  4


PL/SQL procedure successfully completed.

Hotel ID:  4


PL/SQL procedure successfully completed.
```

```
Project: C:\Users\hivad\AppData\Roaming\SQL Developer\system20.4.1.407.0006\o.sqldeveloper\projects\IdeConnections#oracle.is.umbc.edu.jpr
oracle.is.umbc.edu (Unshared)
```

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

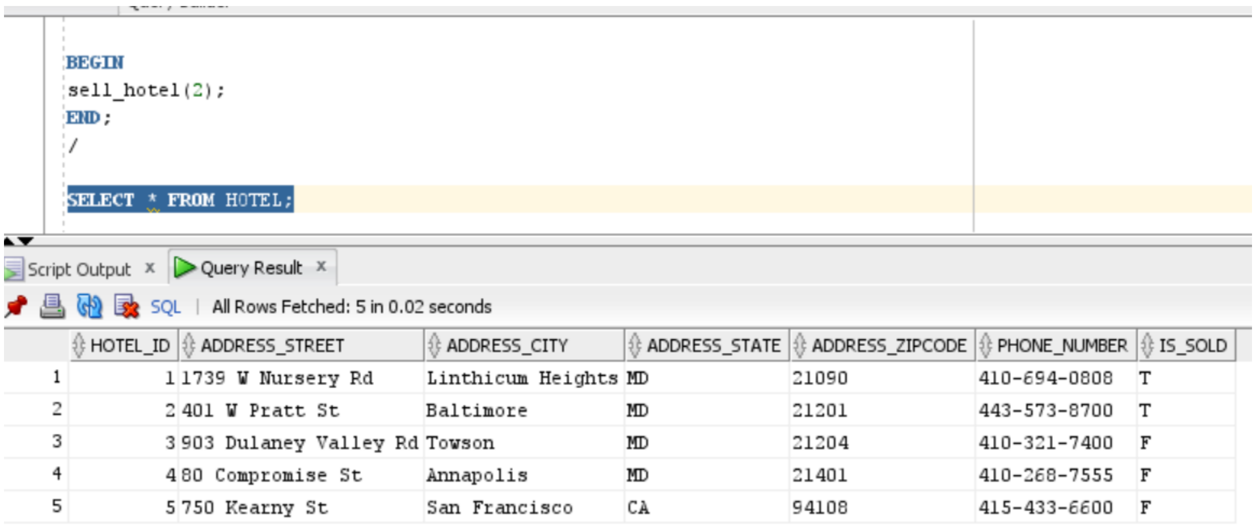
```
CREATE OR REPLACE PROCEDURE sell_hotel(Hotel_ID_Param  IN HOTEL.HOTEL_ID%TYPE) AS
BEGIN
    UPDATE HOTEL h
    SET h.is_sold ='T'
    where h.hotel_id = Hotel_ID_Param;
    DBMS_OUTPUT.PUT_LINE('The hotel with hotel id ' || Hotel_ID_Param || ' is sold');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Hotel does not exist.');
    WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Too many rows.');
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('A different exception occurred.');
END;
/

BEGIN
sell_hotel(2);
END;
/
```



```
SELECT * FROM HOTEL;
```

All Rows Fetched: 5 in 0.02 seconds

	HOTEL_ID	ADDRESS_STREET	ADDRESS_CITY	ADDRESS_STATE	ADDRESS_ZIPCODE	PHONE_NUMBER	IS_SOLD
1	1	1739 W Nursery Rd	Linthicum Heights	MD	21090	410-694-0808	T
2	2	401 W Pratt St	Baltimore	MD	21201	443-573-8700	F
3	3	903 Dulaney Valley Rd	Towson	MD	21204	410-321-7400	F
4	4	80 Compromise St	Annapolis	MD	21401	410-268-7555	F
5	5	750 Kearny St	San Francisco	CA	94108	415-433-6600	F



```
BEGIN
sell_hotel(2);
END;
/

SELECT * FROM HOTEL;
```

Script Output × Query Result ×

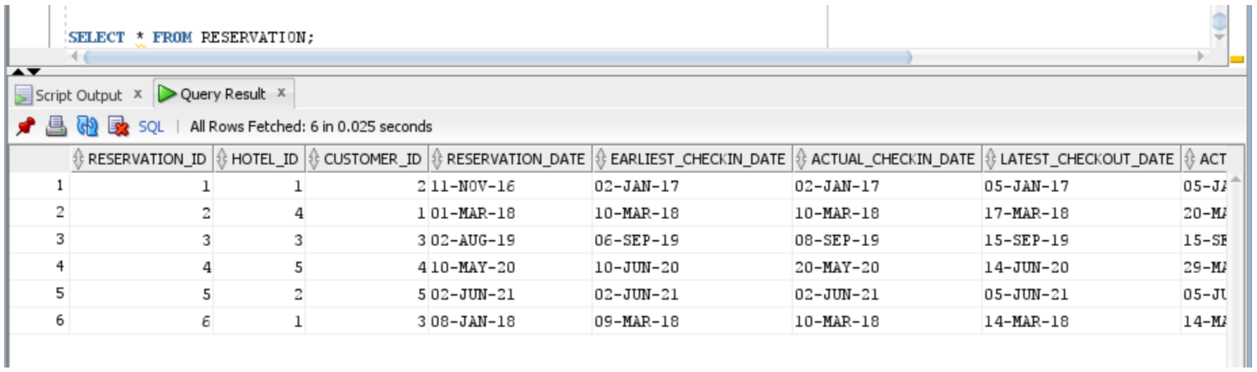
SQL | All Rows Fetched: 5 in 0.02 seconds

	HOTEL_ID	ADDRESS_STREET	ADDRESS_CITY	ADDRESS_STATE	ADDRESS_ZIPCODE	PHONE_NUMBER	IS_SOLD
1	1	1739 W Nursery Rd	Linthicum Heights	MD	21090	410-694-0808	T
2	2	401 W Pratt St	Baltimore	MD	21201	443-573-8700	T
3	3	903 Dulaney Valley Rd	Towson	MD	21204	410-321-7400	F
4	4	80 Compromise St	Annapolis	MD	21401	410-268-7555	F
5	5	750 Kearny St	San Francisco	CA	94108	415-433-6600	F

4. Eldho[**] 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( HOTEL_ID IN NUMBER, CUSTOMER_ID IN
NUMBER, reservation_date IN DATE, CHECK_IN_DATE IN DATE, CHECK_OUT_DATE IN DATE)
AS
BEGIN
INSERT INTO RESERVATION VALUES(Reservation_PK_Seq.NEXTVAL, HOTEL_ID,CUSTOMER_ID,
reservation_date, CHECK_IN_DATE, NULL, CHECK_OUT_DATE, NULL, NULL, 0, 'F');
END;
/

CREATE OR REPLACE PROCEDURE ADD_ROOM(HOTEL_ID_PARAM IN NUMBER , ROOM_TYPE_PARAM IN
VARCHAR) AS

ROOM_NUMBER_PARAM NUMBER;
Start_Date_Param DATE;
End_Date_Param DATE;
RESERVATION_ID_PARAM NUMBER;

BEGIN

RESERVATION_ID_PARAM := Reservation_PK_Seq.CURRVAL;
SELECT EARLIEST_CHECKIN_DATE, latest_checkout_date INTO
Start_Date_Param,End_Date_Param FROM RESERVATION WHERE RESERVATION_ID =
Reservation_ID_Param;

SELECT Room_Number INTO ROOM_NUMBER_PARAM
```

```
    from ROOM
    WHERE (Hotel_ID , Room_Number) NOT IN
        (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
        FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
        WHERE RE.Reservation_ID = RA.Reservation_ID
        AND RO.Room_Number = RA.Room_Number
        AND RO.Hotel_ID = RA.Hotel_ID
        AND H.Hotel_ID = RO.Hotel_ID
        AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < latest_checkout_date
        AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= earliest_checkin_date
        AND Is_Sold = 'F'
        AND Is_Cancelled = 'F')
    AND Room_Type = ROOM_TYPE_PARAM
    AND HOTEL_ID = HOTEL_ID_PARAM
    AND rownum < 2;

INSERT INTO ROOM_ASSIGNMENT VALUES(ROOM_NUMBER_PARAM , HOTEL_ID_PARAM,
RESERVATION_ID_PARAM);

END;
/


BEGIN

MAKE_RESERVATION(5 ,5  ,TO_DATE('2020-10-20','YYYY-MM-DD'),
TO_DATE('2020-10-23','YYYY-MM-DD') , TO_DATE('2020-10-25','YYYY-MM-DD'));
ADD_ROOM(5,'single-room');

END;
/
```

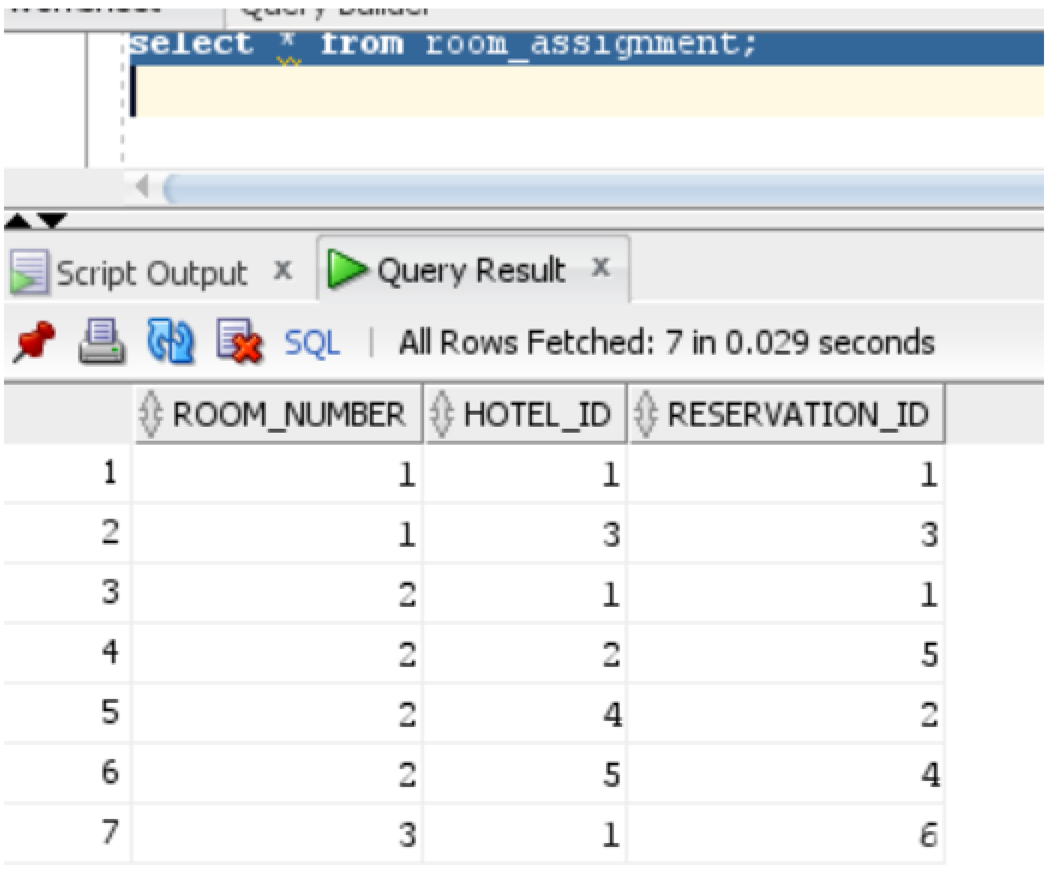


```
SELECT * FROM RESERVATION;
```

Script Output | Query Result

All Rows Fetched: 6 in 0.025 seconds

	RESERVATION_ID	HOTEL_ID	CUSTOMER_ID	RESERVATION_DATE	EARLIEST_CHECKIN_DATE	ACTUAL_CHECKIN_DATE	LATEST_CHECKOUT_DATE	ACT
1	1	1	2	11-NOV-16	02-JAN-17	02-JAN-17	05-JAN-17	05-J
2	2	4	1	01-MAR-18	10-MAR-18	10-MAR-18	17-MAR-18	20-M
3	3	3	3	02-AUG-19	06-SEP-19	08-SEP-19	15-SEP-19	15-S
4	4	5	4	10-MAY-20	10-JUN-20	20-MAY-20	14-JUN-20	29-M
5	5	2	5	02-JUN-21	02-JUN-21	02-JUN-21	05-JUN-21	05-J
6	6	1	3	08-JAN-18	09-MAR-18	10-MAR-18	14-MAR-18	14-M



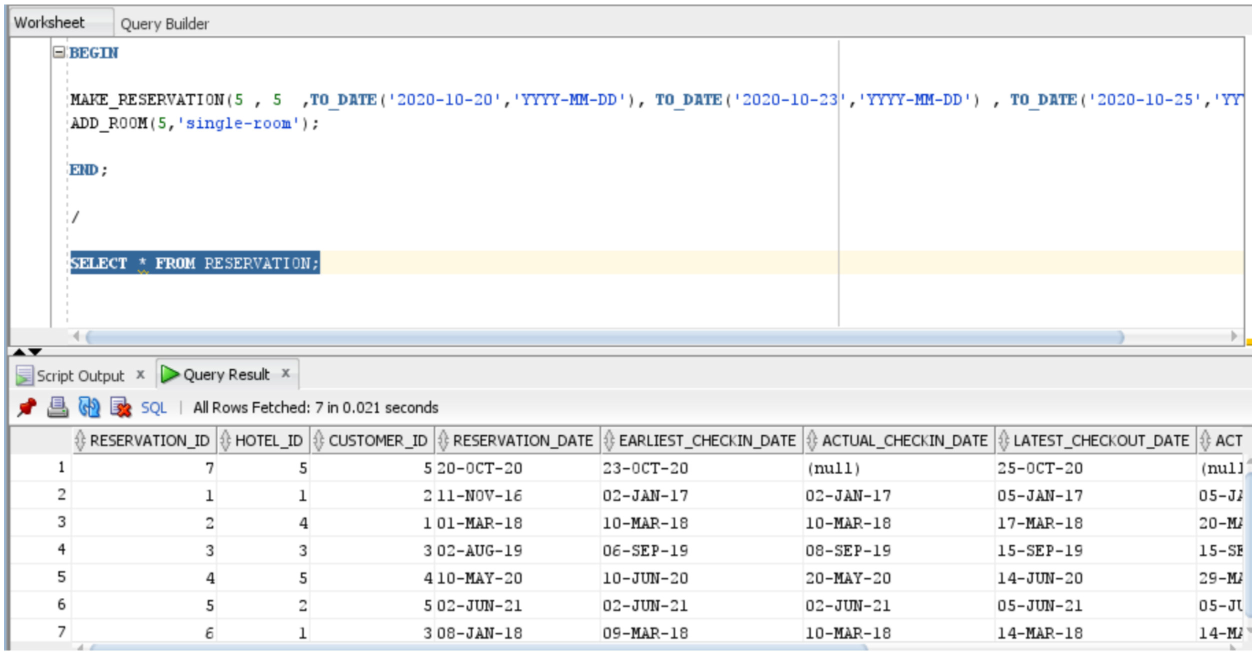
```
select * from room_assignment;
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.029 seconds

	ROOM_NUMBER	HOTEL_ID	RESERVATION_ID
1	1	1	1
2	1	3	3
3	2	1	1
4	2	2	5
5	2	4	2
6	2	5	4
7	3	1	6

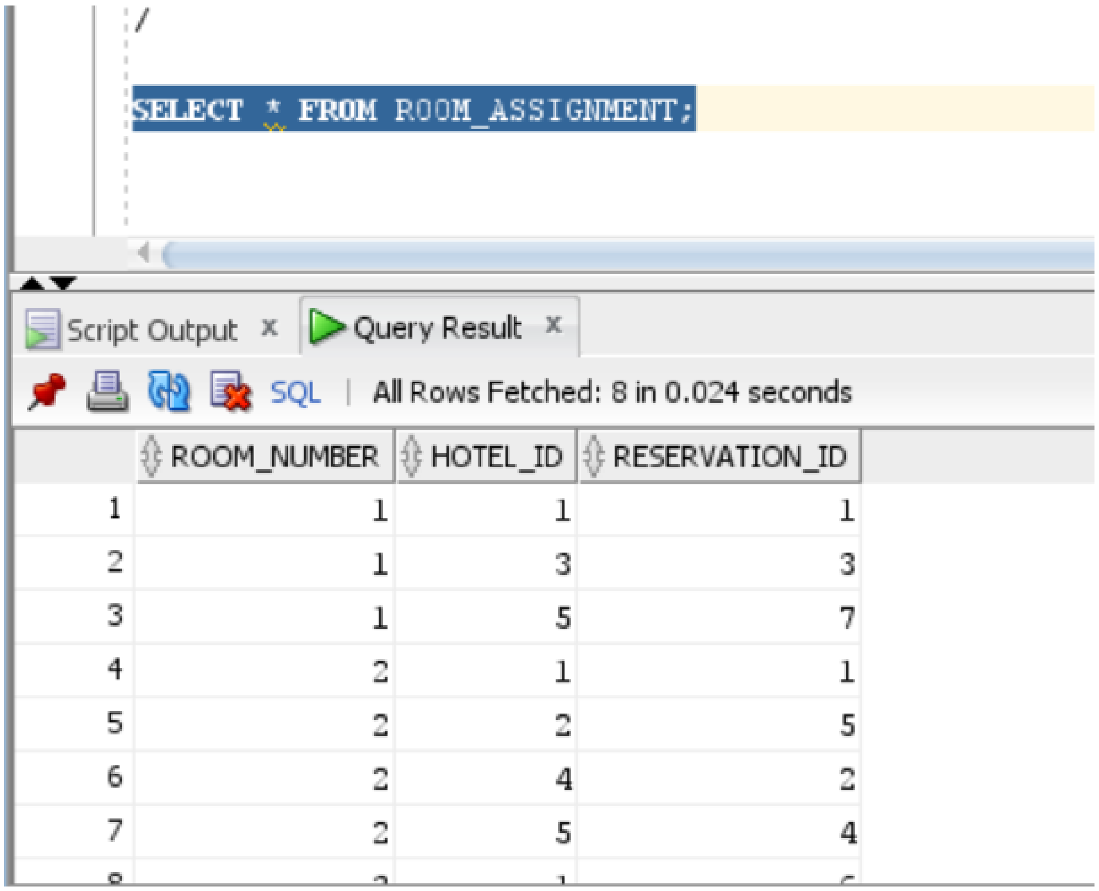
After making the procedure call we can see that a row is inserted in the reservation table with the input parameters. Also, a row was inserted in the room_assignment table.



```
BEGIN

MAKE_RESERVATION(5 , 5  ,TO_DATE('2020-10-20','YYYY-MM-DD'), TO_DATE('2020-10-23','YYYY-MM-DD') , TO_DATE('2020-10-25','YY
ADD_ROOM(5,'single-room');

END;

/

SELECT * FROM RESERVATION;
```

All Rows Fetched: 7 in 0.021 seconds

	RESERVATION_ID	HOTEL_ID	CUSTOMER_ID	RESERVATION_DATE	EARLIEST_CHECKIN_DATE	ACTUAL_CHECKIN_DATE	LATEST_CHECKOUT_DATE	ACT
1	7	5	5	20-OCT-20	23-OCT-20	(null)	25-OCT-20	(null
2	1	1	2	11-NOV-16	02-JAN-17	02-JAN-17	05-JAN-17	05-JA
3	2	4	1	01-MAR-18	10-MAR-18	10-MAR-18	17-MAR-18	20-MA
4	3	3	3	02-AUG-19	06-SEP-19	08-SEP-19	15-SEP-19	15-SE
5	4	5	4	10-MAY-20	10-JUN-20	20-MAY-20	14-JUN-20	29-MA
6	5	2	5	02-JUN-21	02-JUN-21	02-JUN-21	05-JUN-21	05-JU
7	6	1	3	08-JAN-18	09-MAR-18	10-MAR-18	14-MAR-18	14-MA



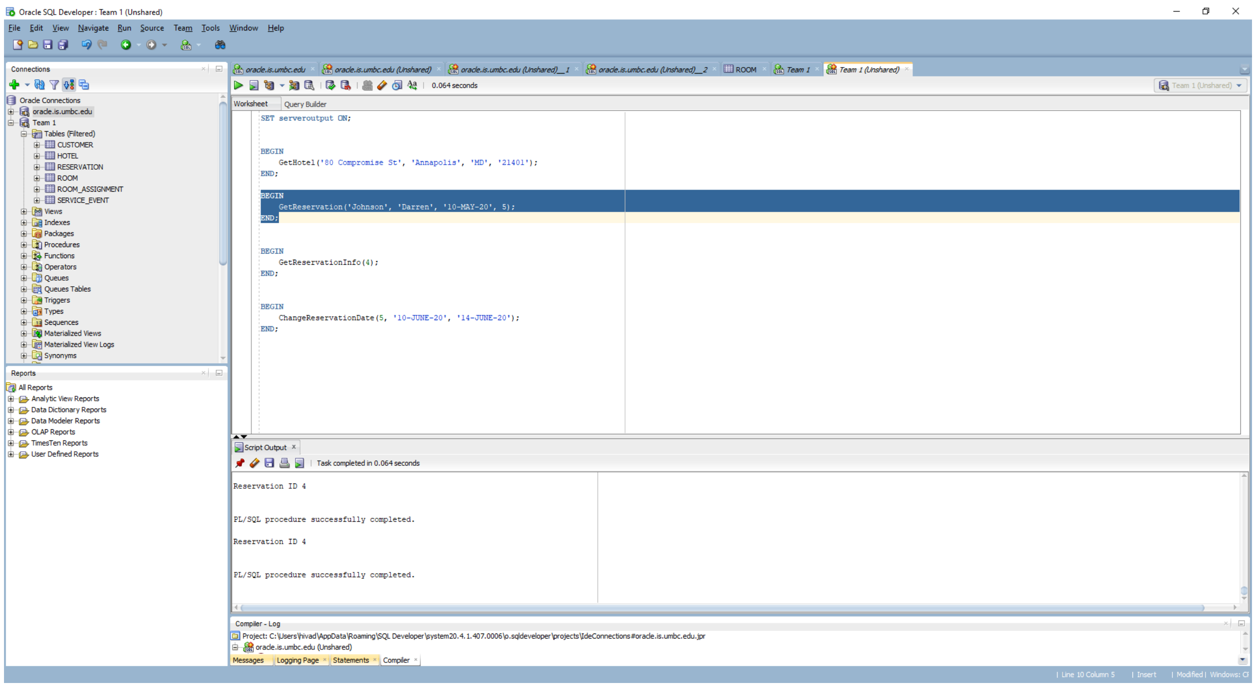
```
/

SELECT * FROM ROOM_ASSIGNMENT;
```

All Rows Fetched: 8 in 0.024 seconds

	ROOM_NUMBER	HOTEL_ID	RESERVATION_ID
1	1	1	1
2	1	3	3
3	1	5	7
4	2	1	1
5	2	2	5
6	2	4	2
7	2	5	4
8	2	1	6

5. John[*] Find a reservation: Input is guest's name and date, hotel ID. Output is reservation ID

```
create or replace PROCEDURE GetReservation(last_in IN varchar2, first_in IN varchar2,
date_in IN DATE, hotel_in IN NUMBER)
--this procedure takes input as last and first name, date, and hotel ID and
--outputs the reservation ID last_in is last_name, first_in is first_name, date_in
--is the date of the reservation, and hotel_in is the hotel_id
IS
    reservation_id NUMBER;
BEGIN
-- selects the reservation ID
    SELECT reservation_id INTO reservation_id FROM reservation, customer
    WHERE last_name = last_in
    AND first_name = first_in
    AND reservation.customer_id = customer.customer_id
    AND reservation_date = date_in
    AND hotel_id = hotel_in;

-- outputs the reservation ID
    DBMS_OUTPUT.PUT_LINE('Reservation ID ' || reservation_id );

    EXCEPTION
          WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE('Reservation does not exist.');
          WHEN TOO_MANY_ROWS THEN
          DBMS_OUTPUT.PUT_LINE('Too many rows.');
          WHEN OTHERS THEN
          DBMS_OUTPUT.PUT_LINE('A different exception occurred.');

END;

BEGIN
    GetReservation('Johnson', 'Darren', '10-MAY-20', 5);
END;
```



```
SET serveroutput ON;


BEGIN
    GetHotel('80 Compromise St', 'Annapolis', 'MD', '21401');
END;

BEGIN
    GetReservation('Johnson', 'Darren', '10-MAY-20', 5);
END;


BEGIN
    GetReservationInfo(4);
END;


BEGIN
    ChangeReservationDate(5, '10-JUNE-20', '14-JUNE-20');
END;
```

Task completed in 0.064 seconds

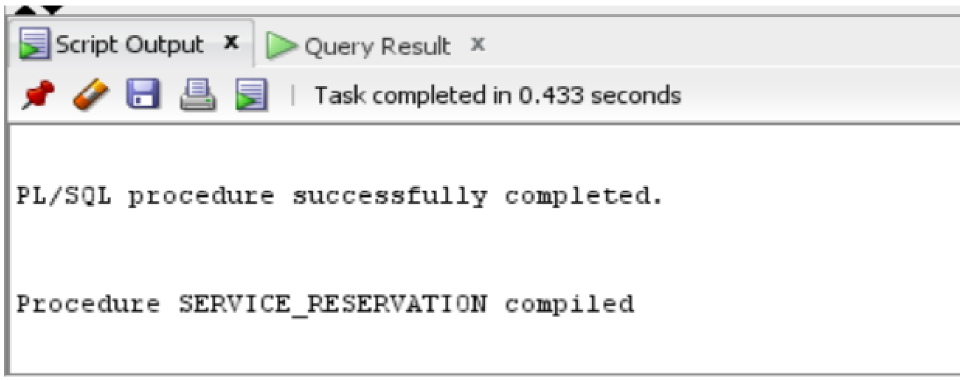
```
Reservation ID 4


PL/SQL procedure successfully completed.

Reservation ID 4


PL/SQL procedure successfully completed.
```

Project: C:\Users\hivad\AppData\Roaming\SQL Developer\system20.4.1.407.0006\o.sqldeveloper\projects\IdeConnections#oracle.is.umbc.edu.jpr

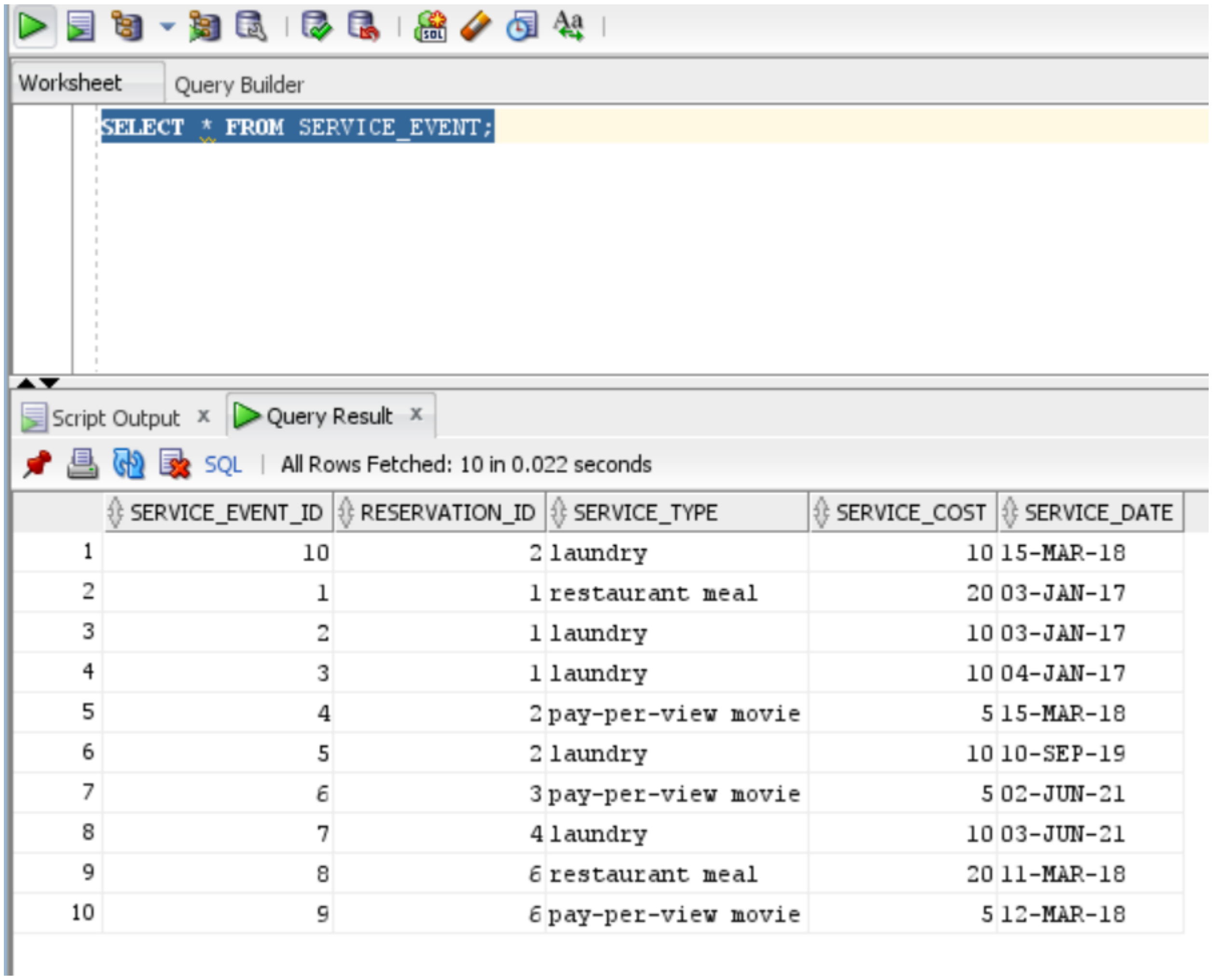
6. Eldho[*] Add a service to a reservation: Input: ReservationID, specific service. Add it to the reservation for a particular date. Multiple services are allowed on a reservation for the same date.

```
CREATE OR REPLACE PROCEDURE Service_Reservation(reservation_id_param IN NUMBER,
service_type IN VARCHAR2, service_cost IN NUMBER, service_date IN DATE) AS
BEGIN
 INSERT INTO SERVICE_EVENT  VALUES(Service_Event_PK_Seq.NEXTVAL,
 reservation_id_param, service_type, service_cost, service_date);

 DBMS_OUTPUT.PUT_LINE('The service event is added to the reservation id' ||
 reservation_id_param);

EXCEPTION
 WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('Reservation does not exist.');
 WHEN TOO_MANY_ROWS THEN
  DBMS_OUTPUT.PUT_LINE('Too many rows.');
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('A different exception occurred.');

END;
/

BEGIN
 service_reservation(2, 'laundry' , 10 , DATE '2018-03-15');
END;
```

/



```
PL/SQL procedure successfully completed.


Procedure SERVICE_RESERVATION compiled
```



```
SELECT * FROM SERVICE_EVENT;
```

All Rows Fetched: 10 in 0.022 seconds

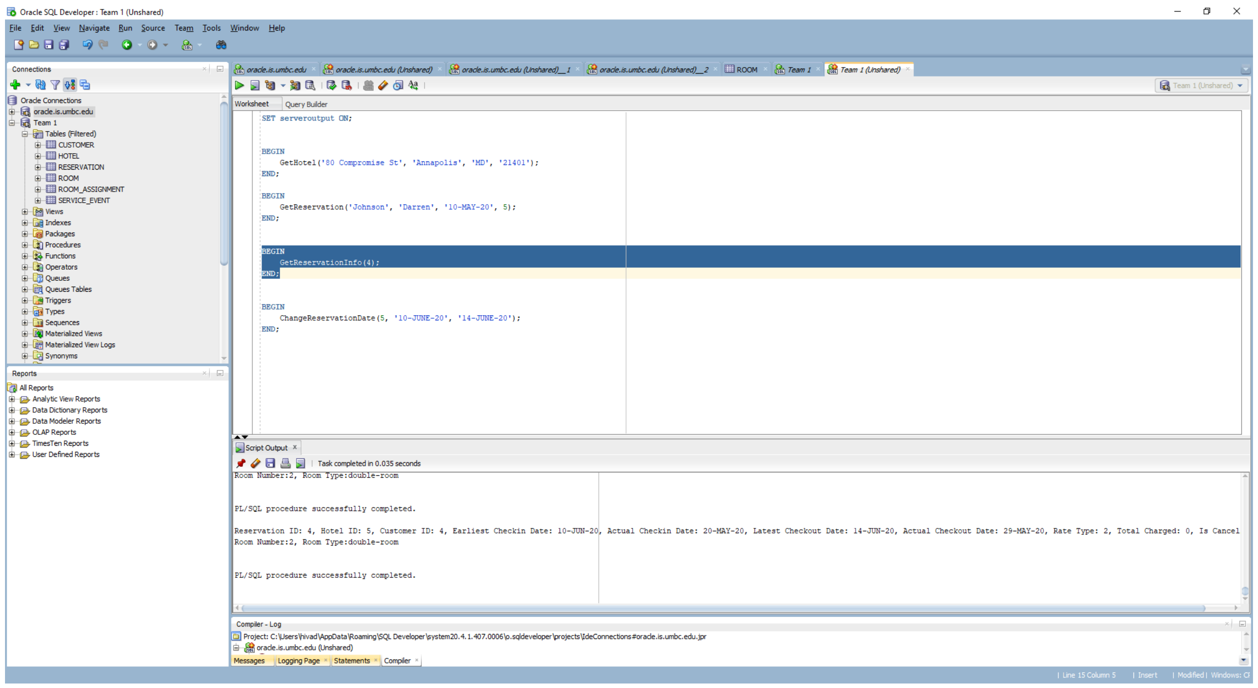
	SERVICE_EVENT_ID	RESERVATION_ID	SERVICE_TYPE	SERVICE_COST	SERVICE_DATE
1	10	2	laundry	10	15-MAR-18
2	1	1	restaurant meal	20	03-JAN-17
3	2	1	laundry	10	03-JAN-17
4	3	1	laundry	10	04-JAN-17
5	4	2	pay-per-view movie	5	15-MAR-18
6	5	2	laundry	10	10-SEP-19
7	6	3	pay-per-view movie	5	02-JUN-21
8	7	4	laundry	10	03-JUN-21
9	8	6	restaurant meal	20	11-MAR-18
10	9	6	pay-per-view movie	5	12-MAR-18

7. John[*] Show reservation details: Input the reservation ID and print all information about this reservation

```
CREATE OR REPLACE PROCEDURE GetReservationInfo(res_id IN INT) IS
--this procedure takes the reservation id and outputs all the information about the
--reservation
--res_id is reservation_id
```

```
       hotel_id_var NUMBER;
       room_number_var NUMBER;
       room_type_var VARCHAR2(30);
--creating the cursor
       CURSOR C IS
        SELECT * FROM reservation
        WHERE reservation_id = res_id;
       Row_Loc C%ROWTYPE;
BEGIN
-- displaying the contents of the cursor
       OPEN C;
       LOOP
       FETCH C INTO Row_Loc;
       EXIT WHEN C%NOTFOUND;
       DBMS_OUTPUT.PUT_LINE('Reservation ID: ' || Row_Loc.reservation_ID ||
       ', Hotel ID: ' || Row_Loc.hotel_id || ', Customer ID: ' ||
       Row_Loc.customer_id || ', Earliest Checkin Date: ' ||
       Row_Loc.earliest_checkin_date || ', Actual Checkin Date: ' ||
       Row_Loc.actual_checkin_date || ', Latest Checkout Date: ' ||
       Row_Loc.latest_checkout_date || ', Actual Checkout Date: ' ||
       Row_Loc.actual_checkout_date || ', Rate Type: ' || Row_Loc.rate_type
       || ', Total Charged: ' || Row_Loc.total_charged || ', Is Cancelled: '
       || Row_Loc.is_cancelled);
       END LOOP;
       CLOSE C;

--These next few select statements are to gather the required information to
--grab the room number and type
       SELECT hotel_id INTO hotel_id_var FROM room_assignment
       WHERE reservation_id = res_id;

       SELECT room_number INTO room_number_var FROM room_assignment
       WHERE reservation_id = res_id;

       SELECT room_type INTO room_type_var FROM ROOM
       WHERE hotel_id = hotel_id_var
       AND room_number = room_number_var;

       DBMS_OUTPUT.PUT_LINE('Room Number:' || room_number_var ||
       ', Room Type:' || room_type_var);

EXCEPTION
       WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;


BEGIN
```

```
        GetReservationInfo(4);
END;
```



```
SET serveroutput ON;


BEGIN
    GetHotel('80 Compromise St', 'Annapolis', 'MD', '21401');
END;

BEGIN
    GetReservation('Johnson', 'Darren', '10-MAY-20', 5);
END;


BEGIN
    GetReservationInfo(4);
END;


BEGIN
    ChangeReservationDate(5, '10-JUNE-20', '14-JUNE-20');
END;
```

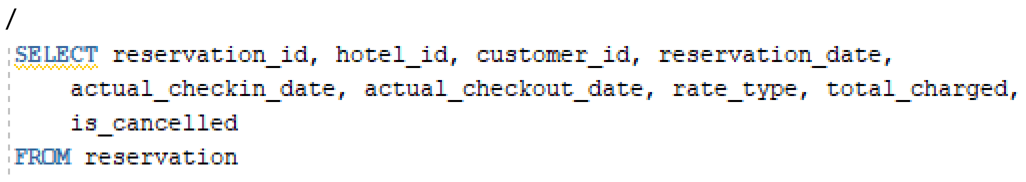
Task completed in 0.035 seconds

```
Room Number:2, Room Type:double-room


PL/SQL procedure successfully completed.

Reservation ID: 4, Hotel ID: 5, Customer ID: 4, Earliest Checkin Date: 10-JUN-20, Actual Checkin Date: 20-MAY-20, Latest Checkout Date: 14-JUN-20, Actual Checkout Date: 29-MAY-20, Rate Type: 2, Total Charged: 0, Is Cancel
Room Number:2, Room Type:double-room


PL/SQL procedure successfully completed.
```

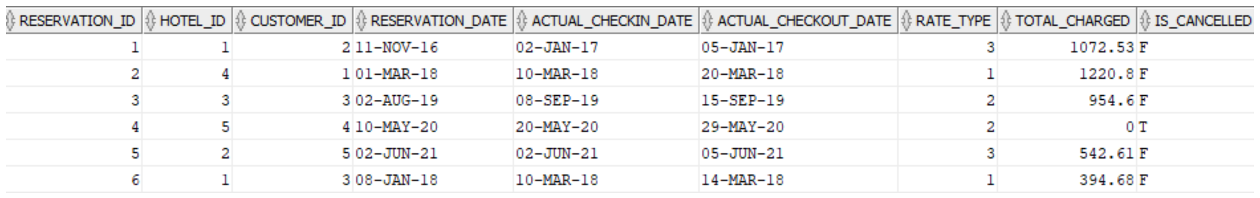
```
Project: C:\Users\hivad\AppData\Roaming\SQL Developer\system20.4.1.407.0006\o.sqldeveloper\projects\IdeConnections#oracle.is.umbc.edu.jpr
oracle.is.umbc.edu (Unshared)
```

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

```
-- This procedure sets a reservation to cancelled
CREATE OR REPLACE PROCEDURE cancel_reservation
(
-- Reservation ID number is accepted as input
        reservation_id_param number
)
AS
BEGIN
-- The procedure updates the selected reservation to canceled and sets the total to
-- charged to $0
UPDATE reservation
SET is_cancelled = 'T',
total_charged = 0
WHERE reservation_id = reservation_id_param;
COMMIT;
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No reservation was found');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception occurred.');
ROLLBACK;
END;
```

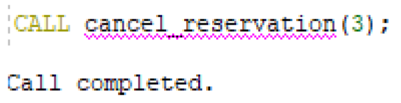


```
/
SELECT reservation_id, hotel_id, customer_id, reservation_date,
    actual_checkin_date, actual_checkout_date, rate_type, total_charged,
    is_cancelled
FROM reservation
```

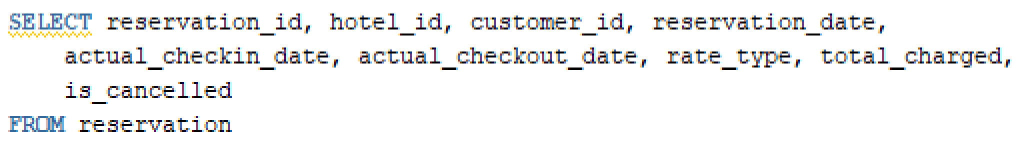


RESERVATION_ID	HOTEL_ID	CUSTOMER_ID	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RATE_TYPE	TOTAL_CHARGED	IS_CANCELLED
1	1	2	11-NOV-16	02-JAN-17	05-JAN-17	3	1072.53	F
2	4	1	01-MAR-18	10-MAR-18	20-MAR-18	1	1220.8	F
3	3	3	02-AUG-19	08-SEP-19	15-SEP-19	2	954.6	F
4	5	4	10-MAY-20	20-MAY-20	29-MAY-20	2	0	T
5	2	5	02-JUN-21	02-JUN-21	05-JUN-21	3	542.61	F
6	1	3	08-JAN-18	10-MAR-18	14-MAR-18	1	394.68	F

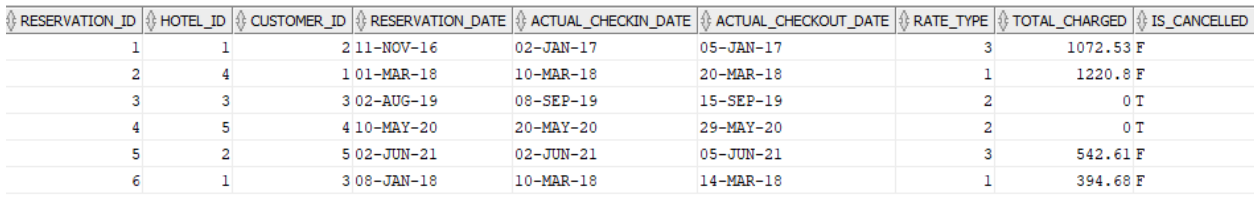
The procedure is called and the corresponding reservation is set to cancelled and the total to be charged is set to $0.



```
CALL cancel_reservation(3);

Call completed.
```



```
SELECT reservation_id, hotel_id, customer_id, reservation_date,
    actual_checkin_date, actual_checkout_date, rate_type, total_charged,
    is_cancelled
FROM reservation
```

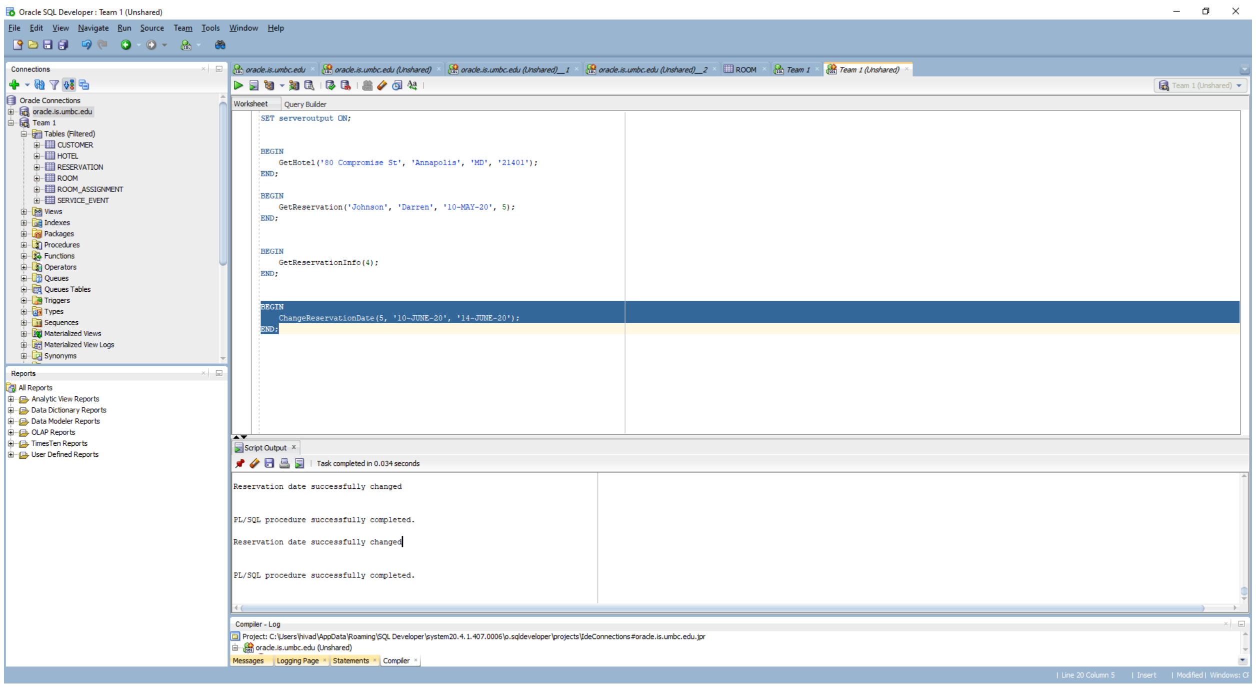


RESERVATION_ID	HOTEL_ID	CUSTOMER_ID	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RATE_TYPE	TOTAL_CHARGED	IS_CANCELLED
1	1	2	11-NOV-16	02-JAN-17	05-JAN-17	3	1072.53	F
2	4	1	01-MAR-18	10-MAR-18	20-MAR-18	1	1220.8	F
3	3	3	02-AUG-19	08-SEP-19	15-SEP-19	2	0	T
4	5	4	10-MAY-20	20-MAY-20	29-MAY-20	2	0	T
5	2	5	02-JUN-21	02-JUN-21	05-JUN-21	3	542.61	F
6	1	3	08-JAN-18	10-MAR-18	14-MAR-18	1	394.68	F

9. John[**] 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

```
create or replace procedure changeReservationDate(res_id IN NUMBER, res_start_date IN
VARCHAR, res_end_date IN VARCHAR) IS
--This procedure takes a reservation id and the dates of the desired new reservation
--date and changed the reservation date for that reservation
--res_id is reservation_id, res_start_date is earliest_checkin_date and res_end_date
--is latest_checkout_date
    room_type_var VARCHAR2(30);
    hotel_id_var NUMBER;
    Num_Available_Rooms_Loc NUMBER;
BEGIN

--Selecting the room type
    select room_type INTO room_type_var
    from reservation, room_assignment, room
    where reservation.reservation_id = res_id
    and reservation.reservation_id = room_assignment.reservation_id
    and room_assignment.hotel_id = room.hotel_id
    and room_assignment.room_number = room.room_number;

--selecting the hotel_id
    select reservation.hotel_id INTO hotel_id_var
    from reservation, room_assignment
    where reservation.reservation_id = res_id
    and reservation.reservation_id = room_assignment.reservation_id;

--Selecting the number of available rooms
    SELECT COUNT(Room_Type) INTO Num_Available_Rooms_Loc
       FROM Room
       WHERE (Hotel_ID, Room_Number) NOT IN
           (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
           FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
           WHERE RE.Reservation_ID = RA.Reservation_ID
           AND RO.Room_Number = RA.Room_Number
           AND RO.Hotel_ID = RA.Hotel_ID
           AND H.Hotel_ID = RO.Hotel_ID
           AND TO_DATE(res_start_date, 'YYYY-MM-DD') < latest_checkout_date
           AND TO_DATE(res_end_date, 'YYYY-MM-DD') >= earliest_checkin_date
           AND Is_Sold = 'F'
           AND Is_Cancelled = 'F')
       AND Hotel_ID = hotel_id_var
       AND Room_Type = room_type_var
       GROUP BY Room_Type;

--updating the reservation
    UPDATE reservation
        SET earliest_checkin_date = res_start_date,
            latest_checkout_date = res_end_date
        WHERE reservation_id = res_id
        and Num_Available_Rooms_Loc > 0;
    COMMIT;

    IF Num_Available_Rooms_Loc > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Reservation date successfully changed');
    END IF;

EXCEPTION
       WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('No rooms available of type ' || room_type_var ||
       ' in hotel #' || hotel_id_var || ' from dates:');
       DBMS_OUTPUT.PUT_LINE(res_start_date || ' to ' || res_end_date);
       WHEN TOO_MANY_ROWS THEN
       DBMS_OUTPUT.PUT_LINE('Something went wrong, too many rows.');
       WHEN OTHERS THEN
             ROLLBACK;
END;
```

```
BEGIN
    ChangeReservationDate(5, '10-JUNE-20', '14-JUNE-20');
END;
```



```
SET serveroutput ON;


BEGIN
    GetHotel('80 Compromise St', 'Annapolis', 'MD', '21401');
END;

BEGIN
    GetReservation('Johnson', 'Darren', '10-MAY-20', 5);
END;


BEGIN
    GetReservationInfo(4);
END;


BEGIN
    ChangeReservationDate(5, '10-JUNE-20', '14-JUNE-20');
END;
```

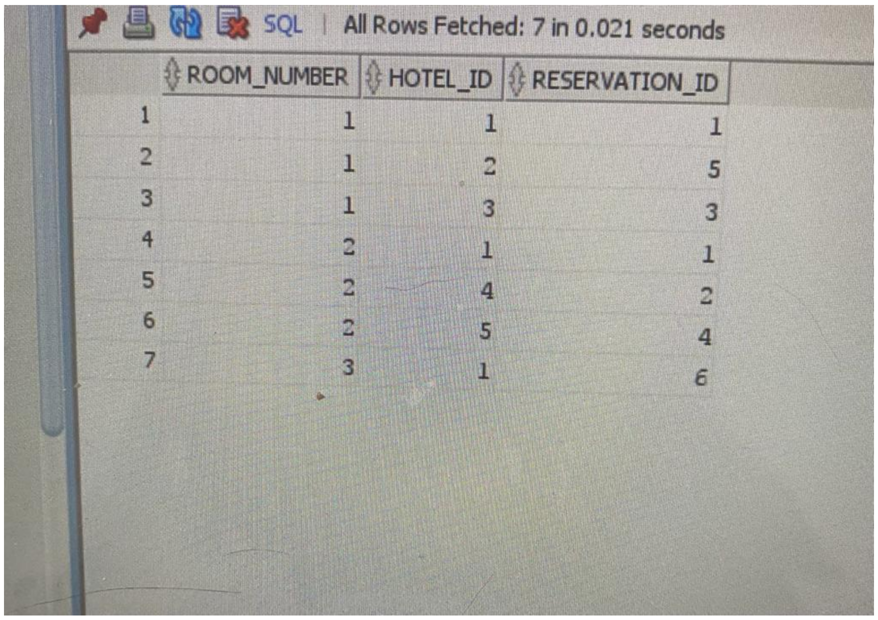
Task completed in 0.034 seconds

```
Reservation date successfully changed


PL/SQL procedure successfully completed.

Reservation date successfully changed


PL/SQL procedure successfully completed.
```

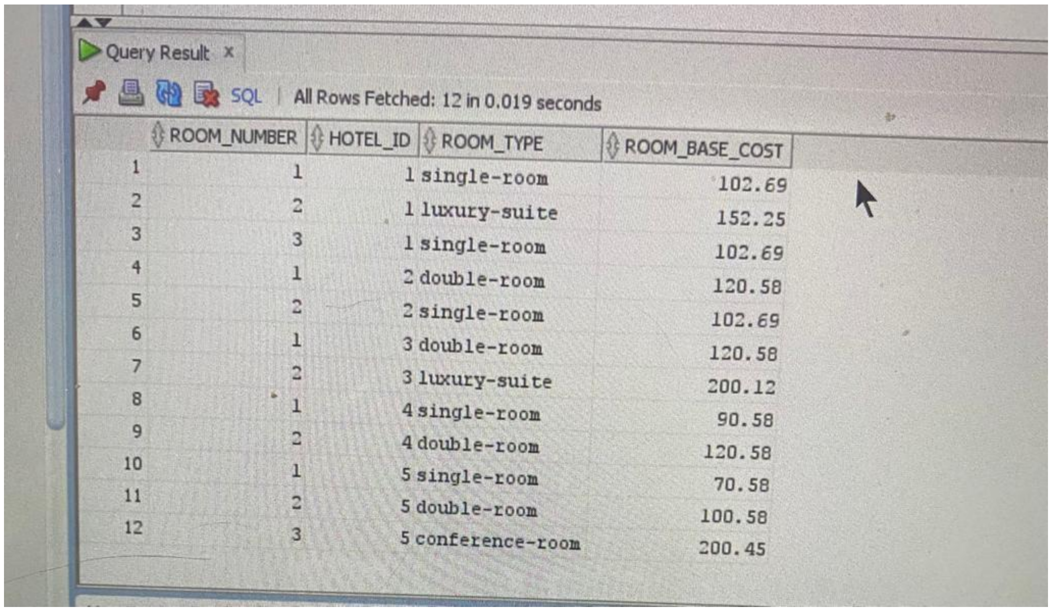
```
Project: C:\Users\hivad\AppData\Roaming\SQL Developer\system20.4.1.407.0006\o.sqldeveloper\projects\IdeConnections#oracle.is.umbc.edu.jpr
oracle.is.umbc.edu (Unshared)
```

10. Eldho[**] 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

```
CREATE OR REPLACE PROCEDURE change_roomtype(Reservation_ID_Param IN NUMBER,
Room_Type_Param IN VARCHAR) AS

HOTEL_ID_PARAM NUMBER;
ROOM_NUMBER_PARAM NUMBER;
Start_Date_Param DATE;
End_Date_Param DATE;

BEGIN
    SELECT EARLIEST_CHECKIN_DATE, latest_checkout_date INTO
    Start_Date_Param,End_Date_Param FROM RESERVATION WHERE RESERVATION_ID =
    Reservation_ID_Param;
    SELECT HOTEL_ID INTO HOTEL_ID_PARAM FROM RESERVATION WHERE RESERVATION_ID =
    Reservation_ID_Param;
    DELETE FROM ROOM_ASSIGNMENT WHERE RESERVATION_ID = Reservation_ID_Param;
    SELECT Room_Number INTO ROOM_NUMBER_PARAM
    from ROOM
    WHERE (Hotel_ID , Room_Number) NOT IN
        (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
        FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
        WHERE RE.Reservation_ID = RA.Reservation_ID
        AND RO.Room_Number = RA.Room_Number
        AND RO.Hotel_ID = RA.Hotel_ID
        AND H.Hotel_ID = RO.Hotel_ID
        AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < latest_checkout_date
        AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= earliest_checkin_date
        AND Is_Sold = 'F'
        AND Is_Cancelled = 'F')
    AND Room_Type = Room_Type_Param
    AND HOTEL_ID = HOTEL_ID_PARAM
    AND rownum < 2;
INSERT INTO ROOM_ASSIGNMENT VALUES(ROOM_NUMBER_PARAM, HOTEL_ID_PARAM,
Reservation_ID_Param);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No rooms available of type ' || Room_Type_Param ||
        ' in hotel #' || Hotel_ID_Param || ' from dates:');
        DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' to ' || End_Date_Param);
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Something went wrong, too many rows.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('A different exception occurred.');
END;
/
```



	ROOM_NUMBER	HOTEL_ID	RESERVATION_ID
1	1	1	1
2	1	2	5
3	1	3	3
4	2	1	1
5	2	4	2
6	2	5	4
7	3	1	6

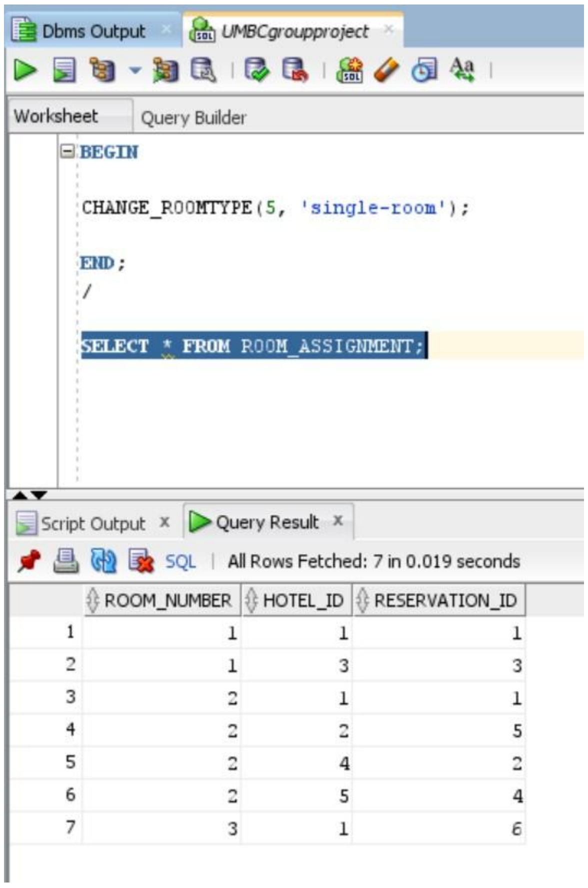

Query Result

All Rows Fetched: 12 in 0.019 seconds

	ROOM_NUMBER	HOTEL_ID	ROOM_TYPE	ROOM_BASE_COST
1	1	1	single-room	102.69
2	2	1	luxury-suite	152.25
3	3	1	single-room	102.69
4	1	2	double-room	120.58
5	2	2	single-room	102.69
6	1	3	double-room	120.58
7	2	3	luxury-suite	200.12
8	1	4	single-room	90.58
9	2	4	double-room	120.58
10	1	5	single-room	70.58
11	2	5	double-room	100.58
12	3	5	conference-room	200.45

```
BEGIN
CHANGE_ROOMTYPE(5, 'single-room');
END;
/
```

After making the procedure call with the above input, running the select statement gives the below output. We can see that, for reservation_id = 5 that a different room of type single-room was assigned.



```
BEGIN

CHANGE_ROOMTYPE(5, 'single-room');

END;
/

SELECT * FROM ROOM_ASSIGNMENT;
```

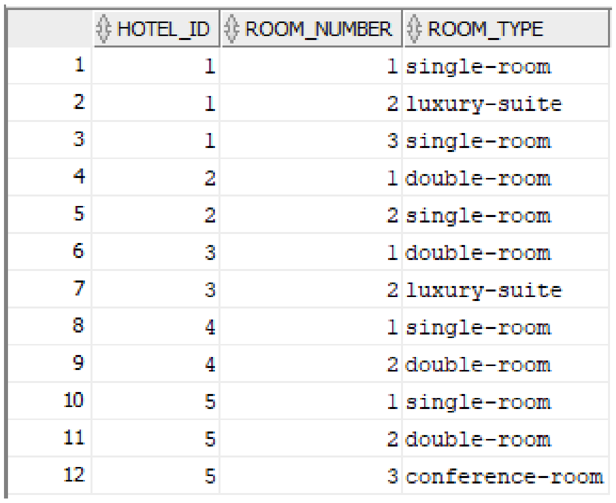
All Rows Fetched: 7 in 0.019 seconds

	ROOM_NUMBER	HOTEL_ID	RESERVATION_ID
1	1	1	1
2	1	3	3
3	2	1	1
4	2	2	5
5	2	4	2
6	2	5	4
7	3	1	6

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

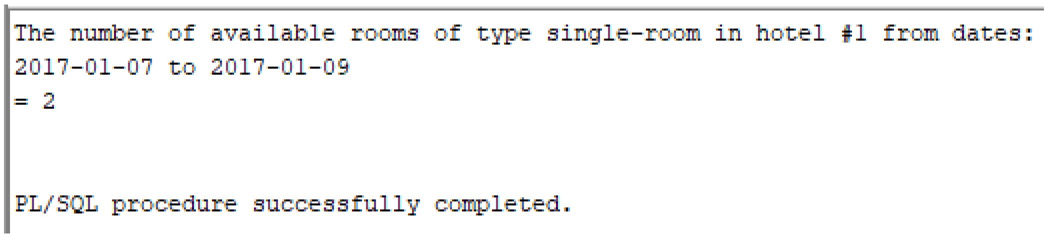
```
CREATE OR REPLACE PROCEDURE Num_Available_Rooms_Of_Type(Hotel_ID_Param
       IN NUMBER, Room_Type_Param IN VARCHAR, Start_Date_Param IN VARCHAR,
       End_Date_Param IN VARCHAR) IS
       Num_Available_Rooms_Loc NUMBER;
BEGIN
       SELECT COUNT(Room_Type) INTO Num_Available_Rooms_Loc
       FROM Room
       WHERE (Hotel_ID, Room_Number) NOT IN
       (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
       FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
       WHERE RE.Reservation_ID = RA.Reservation_ID
       AND RO.Room_Number = RA.Room_Number
       AND RO.Hotel_ID = RA.Hotel_ID
       AND H.Hotel_ID = RO.Hotel_ID
       AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < latest_checkout_date
       AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= earliest_checkin_date
       AND Is_Sold = 'F'
       AND Is_Cancelled = 'F')
       AND Hotel_ID = Hotel_ID_Param
```

```
	AND Room_Type = Room_Type_Param
	GROUP BY Room_Type;

	DBMS_OUTPUT.PUT_LINE('The number of available rooms of type ' ||
	Room_Type_Param || ' in hotel #' || Hotel_ID_Param || ' from dates:');
	DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' to ' || End_Date_Param);
	DBMS_OUTPUT.PUT_LINE('= ' || Num_Available_Rooms_Loc);
EXCEPTION
	WHEN NO_DATA_FOUND THEN
	    DBMS_OUTPUT.PUT_LINE('No rooms available of type ' || Room_Type_Param ||
	    ' in hotel #' || Hotel_ID_Param || ' from dates:');
	    DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' to ' || End_Date_Param);
	WHEN TOO_MANY_ROWS THEN
	    DBMS_OUTPUT.PUT_LINE('Something went wrong, too many rows.');
	WHEN OTHERS THEN
	    DBMS_OUTPUT.PUT_LINE('A different exception occurred.');
END;
/

--Here are all available rooms between the date range given in the procedure call
--below of all types in all hotels obtained from the following select query:
SELECT Hotel_ID, Room_Number, Room_Type
FROM Room
WHERE (Hotel_ID, Room_Number) NOT IN
	(SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
	FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
	WHERE RE.Reservation_ID = RA.Reservation_ID
	AND RO.Room_Number = RA.Room_Number
	AND RO.Hotel_ID = RA.Hotel_ID
	AND H.Hotel_ID = RO.Hotel_ID
	AND DATE '2017-01-07' < latest_checkout_date
	AND DATE '2017-01-09' >= earliest_checkin_date
	AND Is_Sold = 'F'
	AND Is_Cancelled = 'F')
ORDER BY Hotel_ID, Room_Number;
```



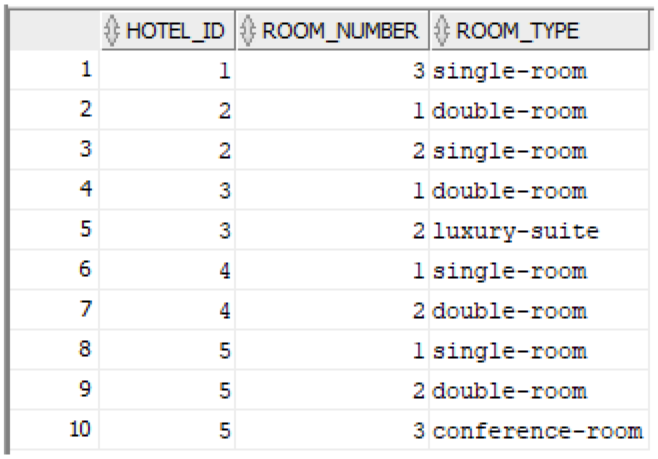
	HOTEL_ID	ROOM_NUMBER	ROOM_TYPE
1	1	1	single-room
2	1	2	luxury-suite
3	1	3	single-room
4	2	1	double-room
5	2	2	single-room
6	3	1	double-room
7	3	2	luxury-suite
8	4	1	single-room
9	4	2	double-room
10	5	1	single-room
11	5	2	double-room
12	5	3	conference-room

```
--The following command produces the following number of
--available single rooms in hotel 1, which is 2.
EXEC Num_Available_Rooms_Of_Type(1, 'single-room', '2017-01-07', '2017-01-09');
```



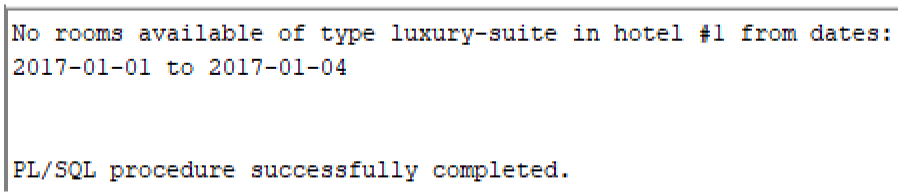
```
The number of available rooms of type single-room in hotel #1 from dates:
2017-01-07 to 2017-01-09
= 2


PL/SQL procedure successfully completed.
```

```
--Here are all available rooms between the date range given in the procedure call
--below of all types in all hotels obtained from the following select query:
SELECT Hotel_ID, Room_Number, Room_Type
FROM Room
WHERE (Hotel_ID, Room_Number) NOT IN
       (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
       FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
       WHERE RE.Reservation_ID = RA.Reservation_ID
       AND RO.Room_Number = RA.Room_Number
       AND RO.Hotel_ID = RA.Hotel_ID
       AND H.Hotel_ID = RO.Hotel_ID
       AND DATE '2017-01-01' < latest_checkout_date
       AND DATE '2017-01-04' >= earliest_checkin_date
       AND Is_Sold = 'F'
       AND Is_Cancelled = 'F')
ORDER BY Hotel_ID, Room_Number;
```



	HOTEL_ID	ROOM_NUMBER	ROOM_TYPE
1	1	3	single-room
2	2	1	double-room
3	2	2	single-room
4	3	1	double-room
5	3	2	luxury-suite
6	4	1	single-room
7	4	2	double-room
8	5	1	single-room
9	5	2	double-room
10	5	3	conference-room

```
--The following procedure is called with a room type that is not available
--in the specified date range, so the NO_DATA_FOUND exception is thrown and
--a meaningful error message is printed:
--No double rooms are available within the below date range, so
--NO_DATA_FOUND exception is thrown.
EXEC Num_Available_Rooms_Of_Type(1, 'luxury-suite', '2017-01-01', '2017-01-04');
```

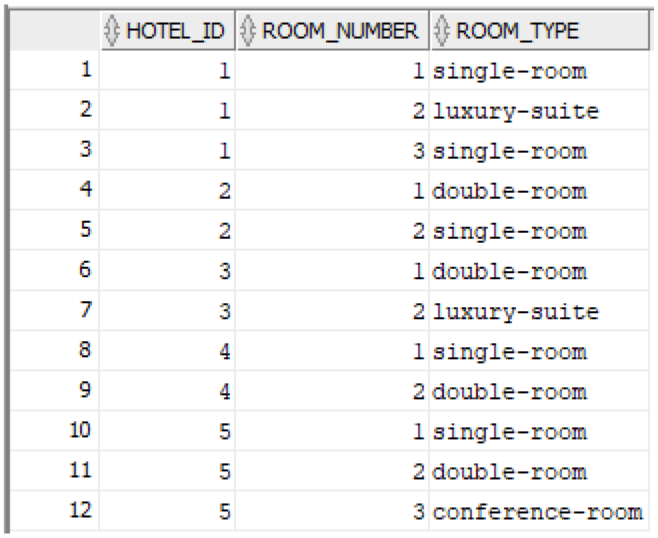


```
No rooms available of type luxury-suite in hotel #1 from dates:
2017-01-01 to 2017-01-04


PL/SQL procedure successfully completed.
```

12. Nicholas[*] AvailabilityOfRoomHotelPerInterval: Input a hotel ID, date interval and return the number of available rooms during that time interval
Reports

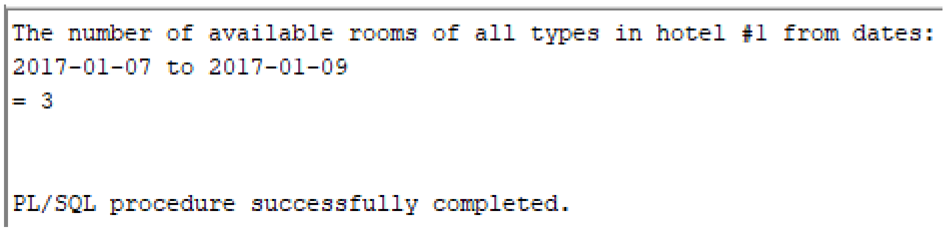
```
CREATE OR REPLACE PROCEDURE Num_Available_Rooms(Hotel_ID_Param
       IN NUMBER, Start_Date_Param IN VARCHAR, End_Date_Param IN VARCHAR) IS
       Num_Available_Rooms_Loc NUMBER;
BEGIN
       SELECT COUNT(Room_Number) INTO Num_Available_Rooms_Loc
       FROM Room
       WHERE (Hotel_ID, Room_Number) NOT IN
       (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
       FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
       WHERE RE.Reservation_ID = RA.Reservation_ID
       AND RO.Room_Number = RA.Room_Number
       AND RO.Hotel_ID = RA.Hotel_ID
       AND H.Hotel_ID = RO.Hotel_ID
       AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < latest_checkout_date
       AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= earliest_checkin_date
       AND Is_Sold = 'F'
```

```
        AND Is_Cancelled = 'F')
        AND Hotel_ID = Hotel_ID_Param;
        DBMS_OUTPUT.PUT_LINE('The number of available rooms of all types in hotel #'
        || Hotel_ID_Param || ' from dates:');
        DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' to ' || End_Date_Param);
        DBMS_OUTPUT.PUT_LINE('= ' || Num_Available_Rooms_Loc);
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Something went wrong, no data found.');
        WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Something went wrong, too many rows.');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('A different exception occurred.');
END;
/

--Here are all available rooms between the date range given in the procedure call
--below of all types in all hotels obtained from the following select query:
SELECT Hotel_ID, Room_Number, Room_Type
FROM Room
WHERE (Hotel_ID, Room_Number) NOT IN
        (SELECT DISTINCT RA.Hotel_ID, RA.Room_Number
        FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
        WHERE RE.Reservation_ID = RA.Reservation_ID
        AND RO.Room_Number = RA.Room_Number
        AND RO.Hotel_ID = RA.Hotel_ID
        AND H.Hotel_ID = RO.Hotel_ID
        AND DATE '2017-01-07' < latest_checkout_date
        AND DATE '2017-01-09' >= earliest_checkin_date
        AND Is_Sold = 'F'
        AND Is_Cancelled = 'F')
ORDER BY Hotel_ID, Room_Number;
```



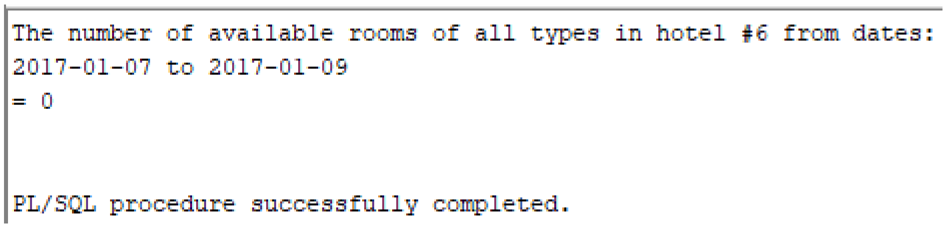
	HOTEL_ID	ROOM_NUMBER	ROOM_TYPE
1	1	1	single-room
2	1	2	luxury-suite
3	1	3	single-room
4	2	1	double-room
5	2	2	single-room
6	3	1	double-room
7	3	2	luxury-suite
8	4	1	single-room
9	4	2	double-room
10	5	1	single-room
11	5	2	double-room
12	5	3	conference-room

```
--The following command calls the procedure and outputs the number of
--available rooms of all types in just Hotel 1. The results are found below,
--which can be verified by counting the number of available rooms in hotel 1 (3)
--in the above query results:
EXEC Num_Available_Rooms(1, '2017-01-07', '2017-01-09');
```



```
The number of available rooms of all types in hotel #1 from dates:
2017-01-07 to 2017-01-09
= 3


PL/SQL procedure successfully completed.
```

```
--The following command calls the function for a non-existent hotel,
--correctly returning 0 rooms:
EXEC Num_Available_Rooms(6, '2017-01-07', '2017-01-09');
```



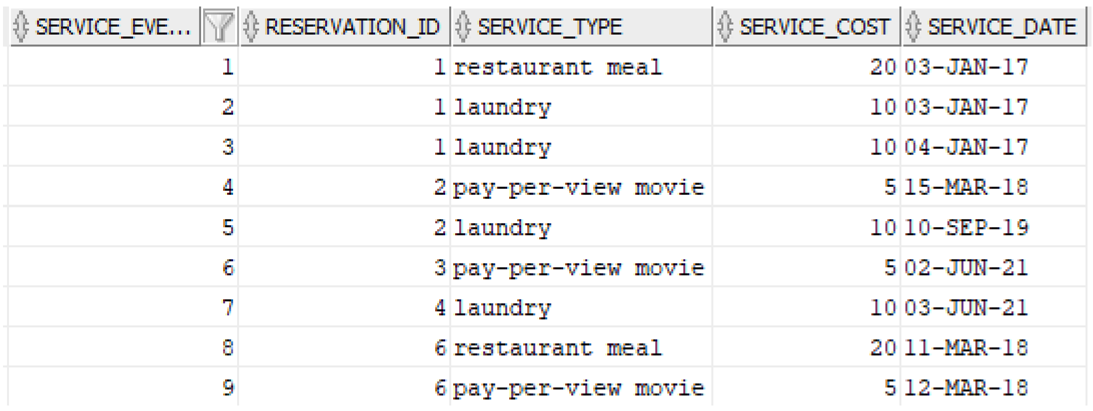
```
The number of available rooms of all types in hotel #6 from dates:
2017-01-07 to 2017-01-09
= 0


PL/SQL procedure successfully completed.
```

13. Graig[***] RoomCheckoutReceipt: Input: ReservationID Output:

    1. Guest name
    2. Room number, rate per day and possibly multiple rooms (if someone reserved several rooms)
    3. Services rendered per date, type, and amount
    4. Discounts applied (if any)
    5. Total amount to be paid

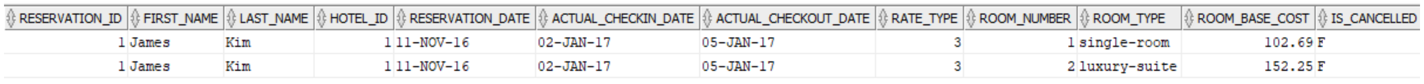
```
CREATE OR REPLACE PROCEDURE RoomCheckoutReceipt
(
    reservation_id_param   reservation.reservation_id%TYPE
)
AS
    timediff_var      NUMBER       := 0;
    staylength_var    NUMBER       := 0;
    total_var         NUMBER(6, 2) := 0;
    service_total_var NUMBER(6, 2) := 0;
    rooms_total_var   NUMBER(6, 2) := 0;
    discount_var      NUMBER       := 0;
    cancelled_var     reservation.is_cancelled%TYPE;
    customer_name_var VARCHAR2(30);
    hotel_id_var      hotel.hotel_id%TYPE;

    -- cursor used to allow for multiple rooms for a reservation
    CURSOR R IS
    SELECT re.reservation_id, re.hotel_id, re.rate_type, ra.room_number,
        ro.room_type, ro.room_base_cost
    FROM reservation re
        JOIN room_assignment ra
            ON (ra.reservation_id = reservation_id_param)
            AND (re.hotel_id = ra.hotel_id)
        JOIN room ro
            ON (ra.room_number = ro.room_number) AND (ra.hotel_id = ro.hotel_id)
    WHERE re.reservation_id = reservation_id_param
    ORDER BY ra.room_number;
    Row_Loc R%ROWTYPE;

    -- cursor used to allow for multiple service events for a reservation
    CURSOR S IS
    SELECT se.reservation_id, se.service_date, se.service_type, se.service_cost
    FROM service_event se
        RIGHT JOIN reservation re
            ON re.reservation_id = reservation_id_param
    WHERE se.reservation_id = reservation_id_param
    ORDER BY se.service_date, se.service_type;
    Row_Loc2 S%ROWTYPE;

BEGIN
-- if reservation is cancelled does not generate a receipt
    SELECT is_cancelled
    INTO cancelled_var
    FROM reservation
    WHERE reservation_id = reservation_id_param;
    IF cancelled_var = 'T' THEN
        DBMS_OUTPUT.PUT_LINE('Reservation ' || reservation_id_param ||
        ' is cancelled');
    ELSE
        SELECT first_name || ' ' || last_name
        INTO customer_name_var
        FROM customer c
            JOIN reservation re
                ON c.customer_id = re.customer_id
                AND re.reservation_id = reservation_id_param;
        DBMS_OUTPUT.PUT_LINE('Customer: ' || customer_name_var);

        SELECT hotel_id
        INTO hotel_id_var
        FROM reservation
        WHERE reservation_id = reservation_id_param;
        DBMS_OUTPUT.PUT_LINE('Hotel: ' || hotel_id_var);

-- accounts for multiple rooms adding to room total based on rate type
        open R;
        LOOP
            FETCH R INTO Row_Loc;
            EXIT WHEN R%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Room Number: ' || Row_Loc.room_number ||
                ' Room Type: ' || Row_Loc.room_type || ' Room base cost: ' ||
                Row_Loc.room_base_cost);
            -- total cost for room adjusted for rate_type
            IF ROW_LOC.rate_type = 1 THEN
                rooms_total_var := (ROW_LOC.room_base_cost * 1) +
                    rooms_total_var;
            ELSIF ROW_LOC.rate_type = 2 THEN
                rooms_total_var := (ROW_LOC.room_base_cost * 1.25) +
                    rooms_total_var;
            ELSIF ROW_LOC.rate_type = 3 THEN
                rooms_total_var := (ROW_LOC.room_base_cost * 1.5) +
                    rooms_total_var;
            ELSIF ROW_LOC.rate_type = 4 THEN
                rooms_total_var := (ROW_LOC.room_base_cost * 1.75) +
                    rooms_total_var;
            ELSE
                rooms_total_var := (ROW_LOC.room_base_cost * 2) +
                    rooms_total_var;
            END IF;
        END LOOP;
        close R;

-- accounts for multiple service events adding to service total
        open S;
        LOOP
            FETCH S INTO Row_Loc2;
            EXIT WHEN S%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE('Service date: ' || Row_Loc2.service_date ||
                ' Service type: ' || Row_Loc2.service_type ||
                ' Service amount: ' || Row_Loc2.service_cost);
            IF ROW_LOC2.service_cost > 0 THEN
            -- total cost for service events
                service_total_var := ROW_LOC2.service_cost + service_total_var;
            END IF;
        END LOOP;
        close S;

        -- calculates length of hotel stay
        SELECT (reservation.actual_checkout_date -
            reservation.actual_checkin_date)
        INTO staylength_var
        FROM reservation
        WHERE reservation.reservation_id = reservation_id_param;

        DBMS_OUTPUT.PUT_LINE('Length of stay: ' || staylength_var || ' days');

        -- determines if reservation is eligible for a discount
        SELECT (reservation.actual_checkin_date - reservation.reservation_date)
        INTO timediff_var
        FROM reservation
        WHERE reservation.reservation_id = reservation_id_param;

        -- assigns discount multiplier
        IF timediff_var >= 62 THEN
            discount_var := .9;
            DBMS_OUTPUT.PUT_LINE('10% discount applies');
        ELSIF timediff_var >= 31 THEN
            discount_var := .95;
            DBMS_OUTPUT.PUT_LINE('5% discount applies');
        ELSE
            discount_var := 1;
            DBMS_OUTPUT.PUT_LINE('0% discount applies');
        END IF;

        -- Total cost of stay adjusted by length and discount if eligible
        total_var := (rooms_total_var * staylength_var) * discount_var +
            service_total_var;
        DBMS_OUTPUT.PUT_LINE('Total: $' || total_var);
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('An error was made and no data has been found');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/
```

```
SELECT * FROM service_event;
```



SERVICE_EVE...	RESERVATION_ID	SERVICE_TYPE	SERVICE_COST	SERVICE_DATE
1	1	restaurant meal	20	03-JAN-17
2	1	laundry	10	03-JAN-17
3	1	laundry	10	04-JAN-17
4	2	pay-per-view movie	5	15-MAR-18
5	2	laundry	10	10-SEP-19
6	3	pay-per-view movie	5	02-JUN-21
7	4	laundry	10	03-JUN-21
8	6	restaurant meal	20	11-MAR-18
9	6	pay-per-view movie	5	12-MAR-18

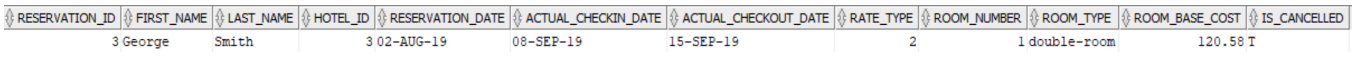
```
-- Same select will be called for each reservation with WHERE changed
SELECT re.reservation_id, c.first_name, c.last_name, re.hotel_id,
   re.reservation_date, re.actual_checkin_date, re.actual_checkout_date,
   re.rate_type, ra.room_number, ro.room_type, ro.room_base_cost,
   re.is_cancelled
   FROM reservation re
       JOIN customer c
           ON c.customer_id = re.customer_id
       JOIN room_assignment ra
           ON (ra.reservation_id = re.reservation_id)
           AND (re.hotel_id = ra.hotel_id)
       JOIN room ro
           ON (ra.room_number = ro.room_number) AND (ra.hotel_id = ro.hotel_id)
   WHERE re.reservation_id = 1
   ORDER BY ra.room_number;
```



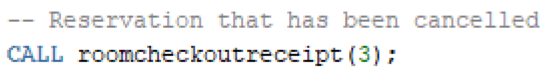
RESERVATION_ID	FIRST_NAME	LAST_NAME	HOTEL_ID	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RATE_TYPE	ROOM_NUMBER	ROOM_TYPE	ROOM_BASE_COST	IS_CANCELLED
1	James	Kim	1	11-NOV-16	02-JAN-17	05-JAN-17	3	1	single-room	102.69	F
1	James	Kim	1	11-NOV-16	02-JAN-17	05-JAN-17	3	2	luxury-suite	152.25	F

```
-- Reservation with multiple rooms and services
CALL roomcheckoutreceipt(1);
```

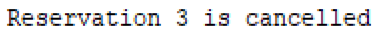
```
Customer: James Kim
Hotel: 1
Room Number: 1 Room Type: single-room Room base cost: 102.69
Room Number: 2 Room Type: luxury-suite Room base cost: 152.25
Service date: 03-JAN-17 Service type: laundry Service amount: 10
Service date: 03-JAN-17 Service type: restaurant meal Service amount: 20
Service date: 04-JAN-17 Service type: laundry Service amount: 10
Length of stay: 3 days
10% discount applies
Total: $1072.53
```



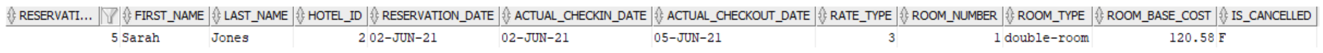
RESERVATION_ID	FIRST_NAME	LAST_NAME	HOTEL_ID	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RATE_TYPE	ROOM_NUMBER	ROOM_TYPE	ROOM_BASE_COST	IS_CANCELLED
3	George	Smith	3	02-AUG-19	08-SEP-19	15-SEP-19	2	1	double-room	120.58	T



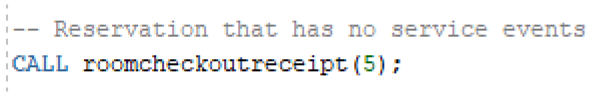
```
-- Reservation that has been cancelled
CALL roomcheckoutreceipt(3);
```



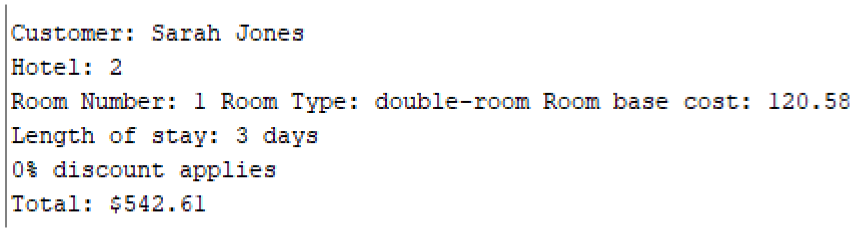
```
Reservation 3 is cancelled
```



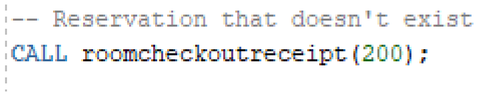
RESERVATI...	FIRST_NAME	LAST_NAME	HOTEL_ID	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RATE_TYPE	ROOM_NUMBER	ROOM_TYPE	ROOM_BASE_COST	IS_CANCELLED
5	Sarah	Jones	2	02-JUN-21	02-JUN-21	05-JUN-21	3	1	double-room	120.58	F



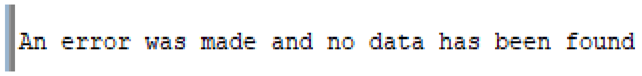
```
-- Reservation that has no service events
CALL roomcheckoutreceipt(5);
```



```
Customer: Sarah Jones
Hotel: 2
Room Number: 1 Room Type: double-room Room base cost: 120.58
Length of stay: 3 days
0% discount applies
Total: $542.61
```



```
-- Reservation that doesn't exist
CALL roomcheckoutreceipt(200);
```

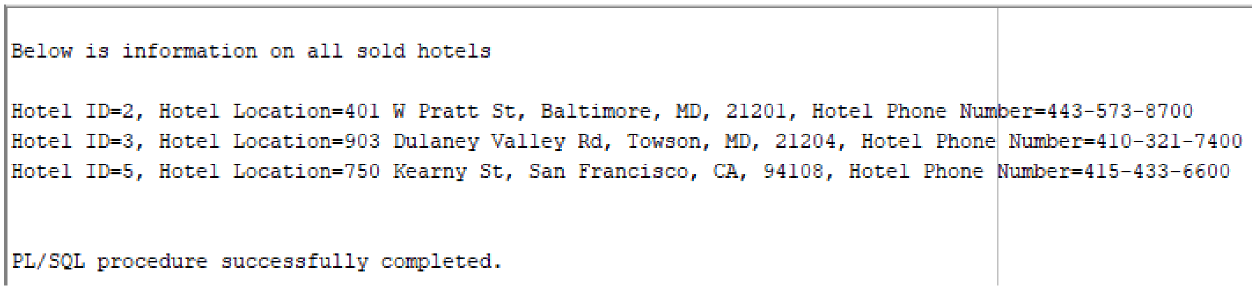


```
An error was made and no data has been found
```

14. Jesse[*] SoldHotels: Print all sold hotel information. Show ID, location, etc.

```
CREATE OR REPLACE PROCEDURE Sold_Hotels IS
       CURSOR C IS
       SELECT h.hotel_id,
         h.address_street || ', ' || h.address_city || ', ' ||
         h.address_state || ', ' || h.address_zipcode as hotel_location,
       h.phone_number
       FROM Hotel h
       WHERE Is_Sold='T'
       ORDER BY hotel_id;
       Row_Loc C%ROWTYPE;

BEGIN
       DBMS_OUTPUT.PUT_LINE('Below is information on all sold hotels');
       DBMS_OUTPUT.NEW_LINE;
       OPEN C;
       LOOP
       FETCH C INTO Row_Loc;
       EXIT WHEN C%NOTFOUND;
       DBMS_OUTPUT.PUT_LINE('Hotel ID=' || Row_Loc.hotel_id ||
             ', Hotel Location=' || Row_Loc.Hotel_Location ||
             ', Hotel Phone Number=' || Row_Loc.phone_number);
       END LOOP;
       CLOSE C;
```

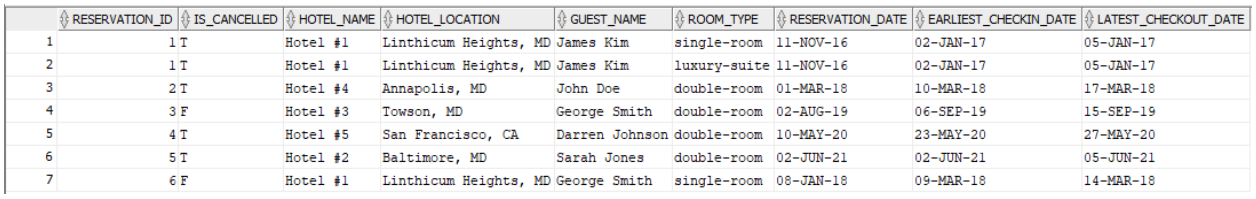
```
EXCEPTION
	WHEN OTHERS THEN
	DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The update command below sets three hotels to sold status.
UPDATE Hotel
SET Is_Sold = 'T'
WHERE Hotel_ID = 2
OR Hotel_ID = 3
OR Hotel_ID = 5;

--Here is the output of the procedure with this data.
EXEC Sold_Hotels;
```



```
Below is information on all sold hotels

Hotel ID=2, Hotel Location=401 W Pratt St, Baltimore, MD, 21201, Hotel Phone Number=443-573-8700
Hotel ID=3, Hotel Location=903 Dulaney Valley Rd, Towson, MD, 21204, Hotel Phone Number=410-321-7400
Hotel ID=5, Hotel Location=750 Kearny St, San Francisco, CA, 94108, Hotel Phone Number=415-433-6600


PL/SQL procedure successfully completed.
```

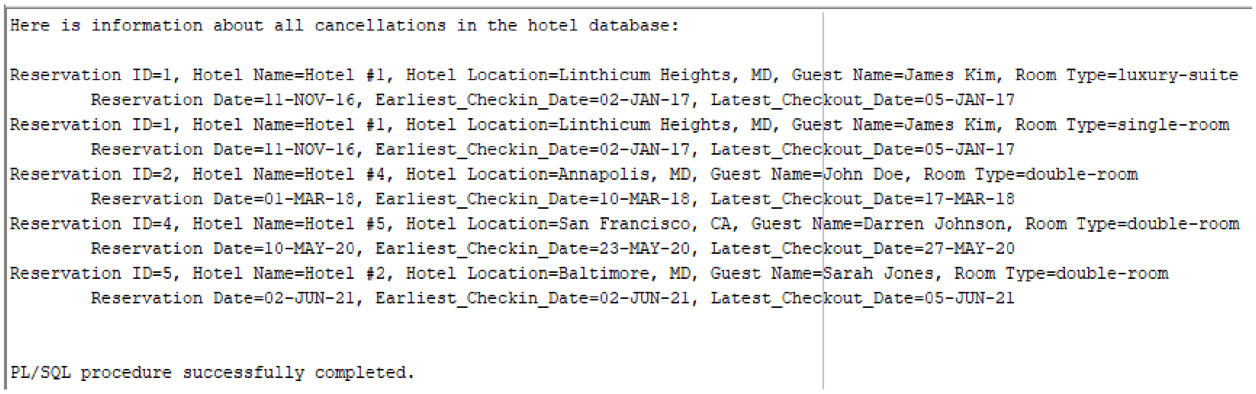
15. Nicholas[*] ShowCancelations: Print all canceled reservations in the hotel management system. Show reservation ID, hotel name, location, guest name, room type, dates

```
CREATE OR REPLACE PROCEDURE Show_Cancellations IS
	CURSOR C IS
	SELECT DISTINCT RE.Reservation_ID, 'Hotel #' || RE.Hotel_ID AS Hotel_Name,
	H.Address_City || ', ' || H.Address_State AS Hotel_Location,
	First_Name || ' ' || Last_Name AS Guest_Name, Room_Type,
	Reservation_Date, Earliest_Checkin_Date, Latest_Checkout_Date
	FROM Reservation RE, Room_Assignment RA, Room RO, Customer C, Hotel H
	WHERE RE.Reservation_ID = RA.Reservation_ID
	AND RO.Hotel_ID = RA.Hotel_ID
	AND RO.Room_Number = RA.Room_Number
	AND H.Hotel_ID = RE.Hotel_ID
	AND C.Customer_ID = RE.Customer_ID
	AND Is_Cancelled = 'T'
	ORDER BY Reservation_ID;
	Row_Loc C%ROWTYPE;
BEGIN
	DBMS_OUTPUT.PUT_LINE('Here is information about all cancellations in the' ||
		' hotel database:');
	DBMS_OUTPUT.NEW_LINE;
	OPEN C;
	LOOP
	FETCH C INTO Row_Loc;
	EXIT WHEN C%NOTFOUND;
	DBMS_OUTPUT.PUT_LINE('Reservation ID=' || Row_Loc.Reservation_ID ||
		', Hotel Name=' || Row_Loc.Hotel_Name || ', Hotel Location=' ||
		Row_Loc.Hotel_Location || ', Guest Name=' || Row_Loc.Guest_Name ||
		', Room Type=' || Row_Loc.Room_Type);
	--Output is broken up into two lines, indicated by 8 spaces in front of
	--each continued line.
	DBMS_OUTPUT.PUT_LINE('	Reservation Date=' ||
		Row_Loc.Reservation_Date || ', Earliest_Checkin_Date=' ||
		Row_Loc.Earliest_Checkin_Date || ', Latest_Checkout_Date=' ||
		Row_Loc.Latest_Checkout_Date);
	END LOOP;
	CLOSE C;
EXCEPTION
	WHEN OTHERS THEN
	DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The following update query is issued to change some reservations to cancelled
--in order to demonstrate functionality:
UPDATE Reservation
SET Is_Cancelled = 'T'
WHERE Reservation_ID = 1
OR Reservation_ID = 2
OR Reservation_ID = 4
OR Reservation_ID = 5;

--Using the following select query with sample data, the following information
--can be shown about all rooms that are found in all reservations:
SELECT RE.Reservation_ID, Is_Cancelled, 'Hotel #' || RE.Hotel_ID AS Hotel_Name,
	H.Address_City || ', ' || H.Address_State AS Hotel_Location,
	First_Name || ' ' || Last_Name AS Guest_Name, Room_Type, Reservation_Date,
	Earliest_Checkin_Date, Latest_Checkout_Date
FROM Reservation RE, Room_Assignment RA, Room RO, Customer C, Hotel H
WHERE RE.Reservation_ID = RA.Reservation_ID
AND RO.Hotel_ID = RA.Hotel_ID
AND RO.Room_Number = RA.Room_Number
AND H.Hotel_ID = RE.Hotel_ID
AND C.Customer_ID = RE.Customer_ID
ORDER BY Reservation_ID;
```



	RESERVATION_ID	IS_CANCELLED	HOTEL_NAME	HOTEL_LOCATION	GUEST_NAME	ROOM_TYPE	RESERVATION_DATE	EARLIEST_CHECKIN_DATE	LATEST_CHECKOUT_DATE
1	1	T	Hotel #1	Linthicum Heights, MD	James Kim	single-room	11-NOV-16	02-JAN-17	05-JAN-17
2	1	T	Hotel #1	Linthicum Heights, MD	James Kim	luxury-suite	11-NOV-16	02-JAN-17	05-JAN-17
3	2	T	Hotel #4	Annapolis, MD	John Doe	double-room	01-MAR-18	10-MAR-18	17-MAR-18
4	3	F	Hotel #3	Towson, MD	George Smith	double-room	02-AUG-19	06-SEP-19	15-SEP-19
5	4	T	Hotel #5	San Francisco, CA	Darren Johnson	double-room	10-MAY-20	23-MAY-20	27-MAY-20
6	5	T	Hotel #2	Baltimore, MD	Sarah Jones	double-room	02-JUN-21	02-JUN-21	05-JUN-21
7	6	F	Hotel #1	Linthicum Heights, MD	George Smith	single-room	08-JAN-18	09-MAR-18	14-MAR-18

```
--Calling the procedure Show_Cancellations gives the following
--output only for cancelled reservations, note that for reservations that have
--multiple room types, there are additional rows to show that the room type
--belongs to the reservation. Rows with duplicate room types (that is, if
--multiple rooms of the same type are reserved in the same reservation) are
--suppressed by using DISTINCT in the procedure. As you can see, reservation
--details with associated room types are printed from reservations where
--Is_Cancelled is true ('T').
EXEC Show_Cancellations;

--(Rows are separated/broken up into two lines each because they are too
--long otherwise.)
```



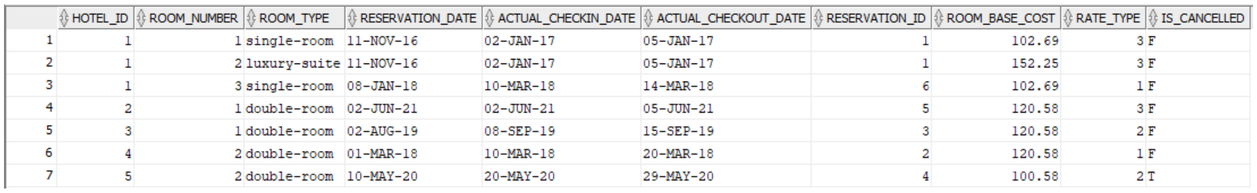
```
Here is information about all cancellations in the hotel database:

Reservation ID=1, Hotel Name=Hotel #1, Hotel Location=Linthicum Heights, MD, Guest Name=James Kim, Room Type=luxury-suite
        Reservation Date=11-NOV-16, Earliest_Checkin_Date=02-JAN-17, Latest_Checkout_Date=05-JAN-17
Reservation ID=1, Hotel Name=Hotel #1, Hotel Location=Linthicum Heights, MD, Guest Name=James Kim, Room Type=single-room
        Reservation Date=11-NOV-16, Earliest_Checkin_Date=02-JAN-17, Latest_Checkout_Date=05-JAN-17
Reservation ID=2, Hotel Name=Hotel #4, Hotel Location=Annapolis, MD, Guest Name=John Doe, Room Type=double-room
        Reservation Date=01-MAR-18, Earliest_Checkin_Date=10-MAR-18, Latest_Checkout_Date=17-MAR-18
Reservation ID=4, Hotel Name=Hotel #5, Hotel Location=San Francisco, CA, Guest Name=Darren Johnson, Room Type=double-room
        Reservation Date=10-MAY-20, Earliest_Checkin_Date=23-MAY-20, Latest_Checkout_Date=27-MAY-20
Reservation ID=5, Hotel Name=Hotel #2, Hotel Location=Baltimore, MD, Guest Name=Sarah Jones, Room Type=double-room
        Reservation Date=02-JUN-21, Earliest_Checkin_Date=02-JUN-21, Latest_Checkout_Date=05-JUN-21


PL/SQL procedure successfully completed.
```

16. Nicholas[**] SpecificHotelReport: Input: hotelID, start-date, end-date. Print (for the given time interval):

    1. Income by room type

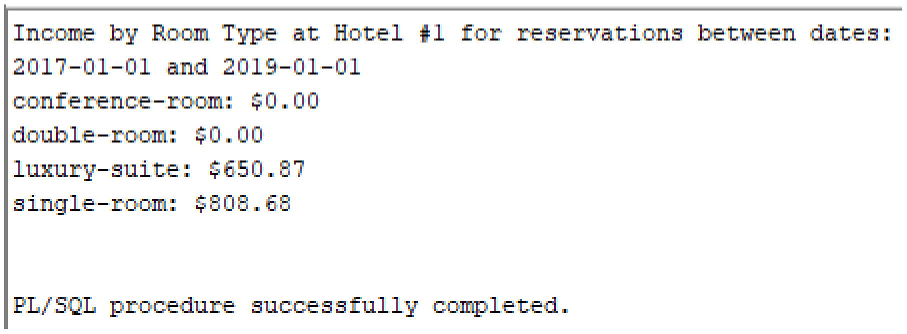
```
CREATE OR REPLACE PROCEDURE Income_By_Room_Type(Hotel_ID_Param IN NUMBER,
        Start_Date_Param IN VARCHAR, End_Date_Param IN VARCHAR) IS
        Room_Type_Loc Room.Room_Type%TYPE;
        Income_By_Room_Type_Loc NUMBER;
        CURSOR C IS
        --First find room types that are not found within the main query, then
        --union the results of this operation with the results from the main
        --query into one table containing all the incomes by room type.
        SELECT Room_Type, 0 AS Income_By_Room_Type FROM
                (SELECT DISTINCT Room_Type
       FROM Room
       WHERE Room_Type NOT IN
       (SELECT DISTINCT Room_Type
       FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
       WHERE RE.Reservation_ID = RA.Reservation_ID
       AND RO.Room_Number = RA.Room_Number
       AND RO.Hotel_ID = RA.Hotel_ID
       AND H.Hotel_ID = RO.Hotel_ID
       AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < Actual_Checkout_Date
       AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= Actual_Checkin_Date
       AND H.Hotel_ID = Hotel_ID_Param
       AND Is_Cancelled = 'F'))
--Union together income by existing room types and non-existing room types.
UNION
--Find the total income from each type of room with SUM and group by
 --room type.
SELECT Room_Type, SUM(Room_Cost_Final) AS Income_By_Room_Type FROM
--Apply discount of 5% or 10% depending on how far back the reservation was
 --made to get the final cost for each room.
(SELECT Room_Type,
       CASE
       WHEN MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) >= 1 AND
             MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) < 2
             THEN Room_Cost_Rated * 0.95
       WHEN MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) >= 2
             THEN Room_Cost_Rated * 0.90
       ELSE
             Room_Cost_Rated * 1
       END AS Room_Cost_Final
FROM
--Then, increase the price of each room depending on the rate type of the
--reservation (25% more for rate type 2, 50% more for rate type 3).
(SELECT Room_Type, Reservation_Date, Actual_Checkin_Date,
        Actual_Checkout_Date,
       CASE Rate_Type
             WHEN 1 THEN Room_Total_Cost * 1
             WHEN 2 THEN Room_Total_Cost * 1.25
             WHEN 3 THEN Room_Total_Cost * 1.5
       END AS Room_Cost_Rated
       FROM
--First, find the aggregate base cost of rooms by multiplying their base cost
--with the number of nights stayed at the hotel.
(SELECT Room_Type, Reservation_Date, Actual_Checkin_Date,
       Actual_Checkout_Date, Rate_Type,
       TO_NUMBER(Actual_Checkout_Date - Actual_Checkin_Date) * Room_Base_Cost
       AS Room_Total_Cost
FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
WHERE RE.Reservation_ID = RA.Reservation_ID
```

```
        AND RO.Room_Number = RA.Room_Number
        AND RO.Hotel_ID = RA.Hotel_ID
        AND H.Hotel_ID = RO.Hotel_ID
        AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < Actual_Checkout_Date
        AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= Actual_Checkin_Date
        AND H.Hotel_ID = Hotel_ID_Param
        AND Is_Cancelled = 'F')))
        --Group main query results by Room_Type
        GROUP BY Room_Type;
BEGIN
        DBMS_OUTPUT.PUT_LINE('Income by Room Type at Hotel #' || Hotel_ID_Param
        || ' for reservations between dates:');
        DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' and ' || End_Date_Param);
        OPEN C;
        LOOP
        FETCH C INTO Room_Type_Loc, Income_By_Room_Type_Loc;
        EXIT WHEN C%NOTFOUND;
        IF Income_By_Room_Type_Loc = 0 THEN
                DBMS_OUTPUT.PUT_LINE(Room_Type_Loc || ': $0.00');
        ELSE
                DBMS_OUTPUT.PUT_LINE(Room_Type_Loc || ': ' ||
                LTRIM(TO_CHAR(ROUND(Income_By_Room_Type_Loc, 2), '$9999999.99')));
        END IF;
        END LOOP;
        CLOSE C;
EXCEPTION
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The following table produced from the SQL query below shows details of all
--rooms found in all reservations in all hotels in a set of sample data. The base
--cost of each room is given along with the date range occupied so it can be
--manually compared to the results of the PL/SQL block below:
SELECT H.Hotel_ID, RO.Room_Number, Room_Type, Reservation_Date, Actual_Checkin_Date,
        Actual_Checkout_Date, RE.Reservation_ID, Room_Base_Cost, Rate_Type,
Is_Cancelled
FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
WHERE RE.Reservation_ID = RA.Reservation_ID
AND RO.Room_Number = RA.Room_Number
AND RO.Hotel_ID = RA.Hotel_ID
AND H.Hotel_ID = RO.Hotel_ID;
```



	HOTEL_ID	ROOM_NUMBER	ROOM_TYPE	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RESERVATION_ID	ROOM_BASE_COST	RATE_TYPE	IS_CANCELLED
1	1	1	single-room	11-NOV-16	02-JAN-17	05-JAN-17	1	102.69	3	F
2	1	2	luxury-suite	11-NOV-16	02-JAN-17	05-JAN-17	1	152.25	3	F
3	1	3	single-room	08-JAN-18	10-MAR-18	14-MAR-18	6	102.69	1	F
4	2	1	double-room	02-JUN-21	02-JUN-21	05-JUN-21	5	120.58	3	F
5	3	1	double-room	02-AUG-19	08-SEP-19	15-SEP-19	3	120.58	2	F
6	4	2	double-room	01-MAR-18	10-MAR-18	20-MAR-18	2	120.58	1	F
7	5	2	double-room	10-MAY-20	20-MAY-20	29-MAY-20	4	100.58	2	T

```
--Using the procedure defined above, we find the income by room type in the date
--range specified for only Hotel 1 in command below by factoring in
--number of days stayed, discounts, and rate types to get the following output.
--Note that room types that were not occupied during the date range give an
--income value of $0.
--Charges for each room in the range at hotel 1 can be calculated manually and
--match the values given when the procedure is called below (add the two costs
--of single-rooms together):
--Base_Cost*Num_Days*Rate_Type_Increase*Discount=Final_Room_Cost
--res1,room1,hotel1,single-room 102.69*3*1.5*0.95=438.9998
--res1,room2,hotel1,luxury-suite 152.25*3*1.5*0.95=650.8688
--res6,room3,hotel1,single-room 102.69*4*1*0.90=369.684
--(for single rooms) 438.9998+369.684=$808.67375

EXEC Income_By_Room_Type(1, '2017-01-01', '2019-01-01');
```



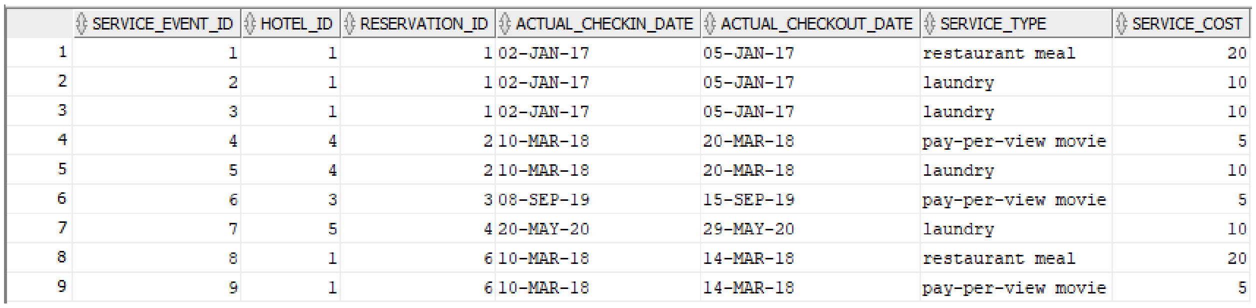
```
Income by Room Type at Hotel #1 for reservations between dates:
2017-01-01 and 2019-01-01
conference-room: $0.00
double-room: $0.00
luxury-suite: $650.87
single-room: $808.68


PL/SQL procedure successfully completed.
```

2. Income of services, by service type

```
CREATE OR REPLACE PROCEDURE Services_Income_By_Type(Hotel_ID_Param IN NUMBER,
Start_Date_Param IN VARCHAR, End_Date_Param IN VARCHAR) IS
        Service_Type_Loc Service_Event.Service_Type%TYPE;
        Service_Cost_Loc NUMBER;
        CURSOR C IS
        SELECT Service_Type, 0 AS Service_Cost FROM
        (SELECT DISTINCT Service_Type
        FROM Service_Event
        WHERE Service_Type NOT IN
                (SELECT Service_Type
                FROM Reservation R, Service_Event S
                WHERE R.Reservation_ID = S.Reservation_ID
```

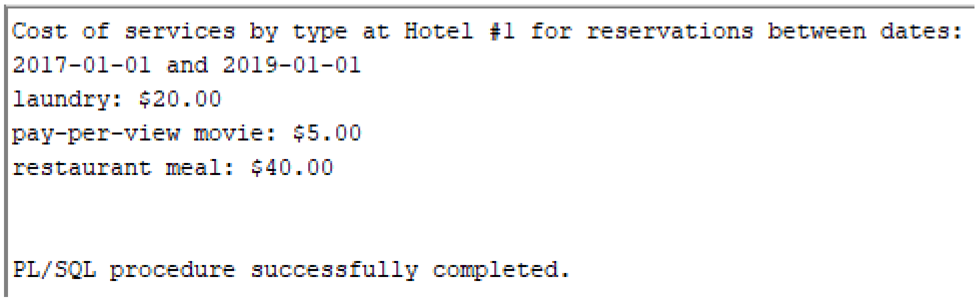
```
            AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < Actual_Checkout_Date
            AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= Actual_Checkin_Date
            AND Hotel_ID = Hotel_ID_Param))
        --Join together the services not found as $0 with the sum of income from
        --other services.
        UNION
        SELECT Service_Type, SUM(CASE Service_Type
                                    WHEN 'restaurant meal' THEN 20
                                   WHEN 'pay-per-view movie' THEN 5
                                   WHEN 'laundry' THEN 10
                                END) AS Service_Cost
        FROM Reservation R, Service_Event S
        WHERE R.Reservation_ID = S.Reservation_ID
        AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < Actual_Checkout_Date
        AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= Actual_Checkin_Date
        AND Hotel_ID = Hotel_ID_Param
        GROUP BY Service_Type;
BEGIN
        OPEN C;
        DBMS_OUTPUT.PUT_LINE('Cost of services by type at Hotel #' || Hotel_ID_Param
        || ' for reservations between dates:');
        DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' and ' || End_Date_Param);
        LOOP
            FETCH C INTO Service_Type_Loc, Service_Cost_Loc;
            EXIT WHEN C%NOTFOUND;
            IF Service_Cost_Loc = 0 THEN
            DBMS_OUTPUT.PUT_LINE(Service_Type_Loc || ': $0.00');
            ELSE
            DBMS_OUTPUT.PUT_LINE(Service_Type_Loc || ': ' ||
                 LTRIM(TO_CHAR(ROUND(Service_Cost_Loc, 2), '$9999999.99')));
            END IF;
        END LOOP;
        CLOSE C;
EXCEPTION
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The following table created using the SQL query provided below shows
--information for all service events in all reservations in all hotels in all
--date ranges:
SELECT Service_Event_ID, Hotel_ID, R.Reservation_ID, Actual_Checkin_Date,
        Actual_Checkout_Date, Service_Type,
        CASE Service_Type
        WHEN 'restaurant meal' THEN 20
        WHEN 'pay-per-view movie' THEN 5
        WHEN 'laundry' THEN 10
```

```
      END AS Service_Cost
FROM Reservation R, Service_Event S
WHERE R.Reservation_ID = S.Reservation_ID
ORDER BY Service_Event_ID;
```



	SERVICE_EVENT_ID	HOTEL_ID	RESERVATION_ID	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	SERVICE_TYPE	SERVICE_COST
1	1	1	1	02-JAN-17	05-JAN-17	restaurant meal	20
2	2	1	1	02-JAN-17	05-JAN-17	laundry	10
3	3	1	1	02-JAN-17	05-JAN-17	laundry	10
4	4	4	2	10-MAR-18	20-MAR-18	pay-per-view movie	5
5	5	4	2	10-MAR-18	20-MAR-18	laundry	10
6	6	3	3	08-SEP-19	15-SEP-19	pay-per-view movie	5
7	7	5	4	20-MAY-20	29-MAY-20	laundry	10
8	8	1	6	10-MAR-18	14-MAR-18	restaurant meal	20
9	9	1	6	10-MAR-18	14-MAR-18	pay-per-view movie	5

```
--The below procedure call finds the total income from all services of each type
--in the specified date range in Hotel 1, which can be confirmed by looking at
--the table above, calculated manually as follows.
--laundry (SE_ID2,Res1),(SE_ID3,Res1) 10+10=$20
--pay-per-view movie (SE_ID9,Res6) $5
--restaurant meal (SE_ID1,Res1),(SE_ID8,Res6) 20+20=40

EXEC Services_Income_By_Type(1, '2017-01-01', '2019-01-01');
```

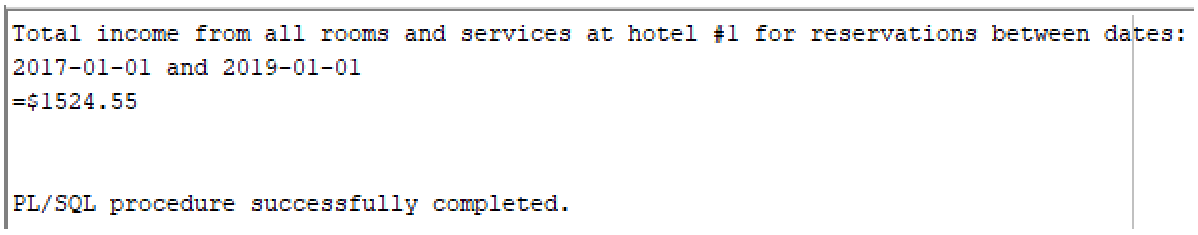


```
Cost of services by type at Hotel #1 for reservations between dates:
2017-01-01 and 2019-01-01
laundry: $20.00
pay-per-view movie: $5.00
restaurant meal: $40.00


PL/SQL procedure successfully completed.
```

3. Total income from all sources.

```
CREATE OR REPLACE PROCEDURE Total_Income(Hotel_ID_Param IN NUMBER,
      Start_Date_Param IN VARCHAR, End_Date_Param IN VARCHAR) IS
      Total_Income_Loc NUMBER;
BEGIN
      --Total income is summed with the number 0 by union and SUM function so that
      --if no reservations fall within the date range, 0 is printed instead of
      --NULL. Within, the total amount from services and the total amount from
      --rooms is also summed together using a UNION and SUM function.
      SELECT SUM(Income_Nonzero) INTO Total_Income_Loc FROM
      ((SELECT SUM(Income) AS Income_Nonzero FROM
      (SELECT
      SUM(CASE
        WHEN MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) >= 1 AND
        MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) < 2
        THEN Room_Cost_Rated * 0.95
        WHEN MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) >= 2
        THEN Room_Cost_Rated * 0.90
        ELSE
        Room_Cost_Rated * 1
END) AS Income
FROM
(SELECT Reservation_Date, Actual_Checkin_Date, Actual_Checkout_Date,
CASE Rate_Type
        WHEN 1 THEN Room_Total_Cost * 1
        WHEN 2 THEN Room_Total_Cost * 1.25
        WHEN 3 THEN Room_Total_Cost * 1.5
END AS Room_Cost_Rated
FROM
(SELECT Reservation_Date, Actual_Checkin_Date,
Actual_Checkout_Date, Rate_Type,
TO_NUMBER(Actual_Checkout_Date - Actual_Checkin_Date) * Room_Base_Cost
AS Room_Total_Cost
FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
WHERE RE.Reservation_ID = RA.Reservation_ID
AND RO.Room_Number = RA.Room_Number
AND RO.Hotel_ID = RA.Hotel_ID
AND H.Hotel_ID = RO.Hotel_ID
AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < Actual_Checkout_Date
AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= Actual_Checkin_Date
AND H.Hotel_ID = Hotel_ID_Param))
--Union together income from services and rooms so it can be summed together
UNION
SELECT SUM(CASE Service_Type
                WHEN 'restaurant meal' THEN 20
                WHEN 'pay-per-view movie' THEN 5
                WHEN 'laundry' THEN 10
        END) AS Total_Service_Cost
FROM Reservation R, Service_Event S
WHERE R.Reservation_ID = S.Reservation_ID
AND TO_DATE(Start_Date_Param, 'YYYY-MM-DD') < Actual_Checkout_Date
AND TO_DATE(End_Date_Param, 'YYYY-MM-DD') >= Actual_Checkin_Date
AND R.Hotel_ID = Hotel_ID_Param
AND Is_Cancelled = 'F'))
UNION
(SELECT 0 FROM DUAL));

DBMS_OUTPUT.PUT_LINE('Total income from all rooms and services at hotel #'
        || Hotel_ID_Param || ' for reservations between dates:');
DBMS_OUTPUT.PUT_LINE(Start_Date_Param || ' and ' || End_Date_Param);
IF Total_Income_Loc = 0 THEN
```

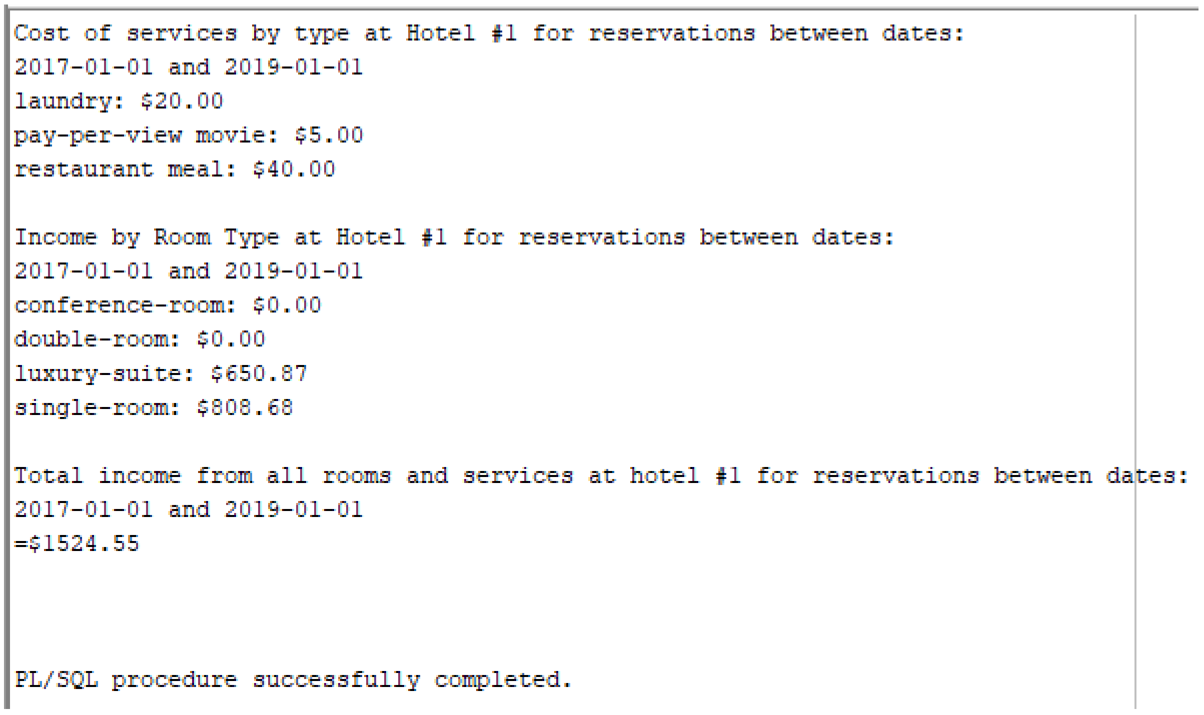
```
        DBMS_OUTPUT.PUT_LINE('= $0.00');
        ELSE
        DBMS_OUTPUT.PUT_LINE('=' || LTRIM(TO_CHAR(ROUND(Total_Income_Loc, 2),
               '$9999999.99')));
        END IF;
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Something went wrong, too few rows.');
        WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Something went wrong, too many rows.');
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('A different exception occurred.');
END;
/

--The income from all sources in Hotel 1 in the specified date range provided in
--the below PL/SQL block is the sum of all income from rooms and from service
--events. The results are shown after the first block. The total income in this
--sample range can be verified by summing the results of part 1 and part 2 above.
--res1,room1,hotel1,single-room 102.69*3*1.5*0.95=438.9998
--res1,room2,hotel1,luxury-suite 152.25*3*1.5*0.95=650.8688
--res6,room3,hotel1,single-room 102.69*4*1*0.90=369.684
--+
--laundry (SE_ID2,Res1),(SE_ID3,Res1) 10+10=$20
--pay-per-view movie (SE_ID9,Res6) $5
--restaurant meal (SE_ID1,Res1),(SE_ID8,Res6) 20+20=40
--438.9998 + 650.8688 + 369.684 + 20 + 5 + 40 = $1524.5526

EXEC Total_Income(1, '2017-01-01', '2019-01-01');
```



```
Total income from all rooms and services at hotel #1 for reservations between dates:
2017-01-01 and 2019-01-01
=$1524.55


PL/SQL procedure successfully completed.
```

4. Parent report function

```
--All three functions above are combined into a single function for the purpose
--of providing all the information for a single hotel and date range in one
--procedure call. The final procedure calls all three sub-functions.
CREATE OR REPLACE PROCEDURE Specific_Hotel_Report(Hotel_ID_Param IN NUMBER,
        Start_Date_Param IN VARCHAR, End_Date_Param IN VARCHAR) IS
BEGIN
        Services_Income_By_Type(Hotel_ID_Param, Start_Date_Param, End_Date_Param);
        DBMS_OUTPUT.NEW_LINE;
        Income_By_Room_Type(Hotel_ID_Param, Start_Date_Param, End_Date_Param);
        DBMS_OUTPUT.NEW_LINE;
        Total_Income(Hotel_ID_Param, Start_Date_Param, End_Date_Param);
        DBMS_OUTPUT.NEW_LINE;
EXCEPTION
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The following command finds all information for Hotel 1 in the given
--sample data necessary for the report by calling the parent function
--Specific_Hotel_Report.
EXEC Specific_Hotel_Report(1, '2017-01-01', '2019-01-01');
```

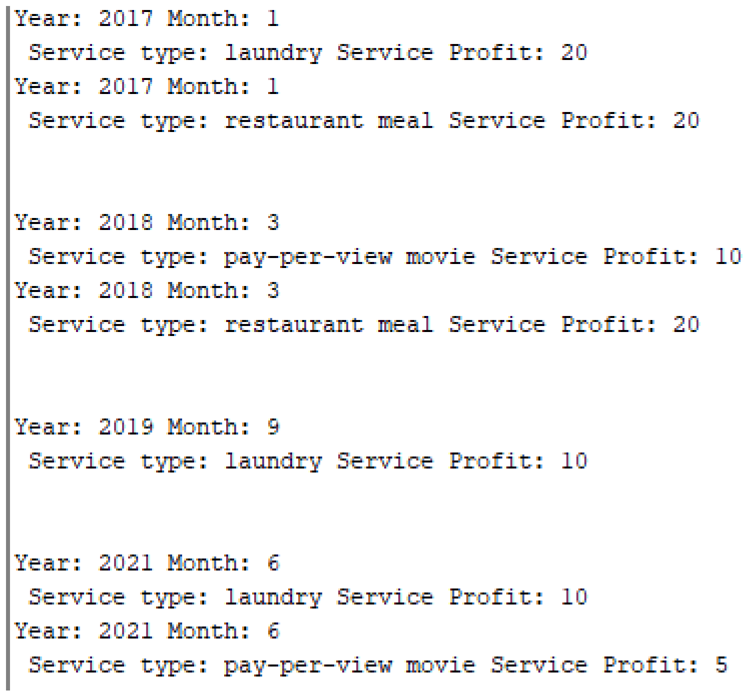


```
Cost of services by type at Hotel #1 for reservations between dates:
2017-01-01 and 2019-01-01
laundry: $20.00
pay-per-view movie: $5.00
restaurant meal: $40.00

Income by Room Type at Hotel #1 for reservations between dates:
2017-01-01 and 2019-01-01
conference-room: $0.00
double-room: $0.00
luxury-suite: $650.87
single-room: $808.68

Total income from all rooms and services at hotel #1 for reservations between dates:
2017-01-01 and 2019-01-01
=$1524.55



PL/SQL procedure successfully completed.
```

17. Jesse[**] (Coded by Graig) TotalHiltonMonthlyReport: 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. (Note that Jesse did not attempt this operation, it was attempted but not completed by Graig. It is incomplete because the room discount is calculated incorrectly and income by room type is not printed.)

```
CREATE OR REPLACE PROCEDURE TotalHiltonMonthlyReport
AS
    timediff_var NUMBER := 0;
    staylength_var NUMBER := 0;
    total_var NUMBER(6, 2) := 0;
    service_total_var NUMBER(6, 2) := 0;
    rooms_total_var NUMBER(6, 2) := 0;
    discount_var NUMBER := 0;
    cancelled_var reservation.is_cancelled%TYPE;
    hotel_id_var hotel.hotel_id%TYPE;
    actual_checkin_date_var reservation.actual_checkin_date%TYPE;
    temp_year_var number     :=0;
    temp_month_var number          := 0;
    temp_room_var NUMBER(10, 2) := 0;
    room_total_var NUMBER(10, 2) := 0;
    room_type_temp_var  VARCHAR2(20) := '';

    CURSOR R IS
    SELECT EXTRACT(year FROM re.actual_checkin_date) AS Year,
        EXTRACT(month FROM re.actual_checkin_date) AS Month, re.rate_type,
        ro.room_type, ro.room_base_cost,
        (re.actual_checkout_date - re.actual_checkin_date) AS stay_length,
        Re.is_cancelled
    FROM reservation re
        JOIN room_assignment ra
            ON (ra.reservation_id = ra.reservation_id)
            AND (re.hotel_id = ra.hotel_id)
        JOIN room ro
            ON (ra.room_number = ro.room_number) AND (ra.hotel_id = ro.hotel_id)
        JOIN hotel h
            ON (h.hotel_id = re.hotel_id)
     GROUP BY EXTRACT(year FROM re.actual_checkin_date), re.actual_checkin_date,
        EXTRACT(month FROM re.actual_checkin_date), ro.room_type, re.rate_type,
        re.is_cancelled, ro.room_base_cost
   ORDER BY year, month;

    Row_Loc R%ROWTYPE;
 -- cursor used to allow for multiple service events for a reservation
    CURSOR S IS
       SELECT EXTRACT(year FROM service_date) AS year,
        EXTRACT(month FROM service_date) AS month, service_type,
        SUM(service_cost) As month_service_cost
    FROM reservation re
        JOIN service_event se
            ON re.reservation_id = se.reservation_id
    GROUP BY EXTRACT(year FROM service_date), EXTRACT(month FROM service_date),
        Service_type
    ORDER BY year, month;
Row_Loc2 S%ROWTYPE;

BEGIN
    -- accounts for multiple rooms adding to room total based on rate type
  /*  open R;
```

```
    LOOP
        FETCH R INTO Row_Loc;
        EXIT WHEN R%NOTFOUND;
        IF cancelled_var = 'F' THEN
            room_type_temp_var := Row_Loc.room_type;
            IF (temp_year_var != Row_Loc.year)
            AND (temp_year_var != Row_Loc.month) AND (temp_year_var != 0) THEN
                DBMS_OUTPUT.PUT_LINE(chr(10));
                DBMS_OUTPUT.PUT_LINE('Room type total: ' || rooms_total_var);
            ELSE
                rooms_total_var := temp.room.var + rooms_total_var;
            END IF;
            DBMS_OUTPUT.PUT_LINE(' Room Type: ' || Row_Loc.room_type);
            -- total cost for room adjusted for rate_type
            IF ROW_LOC.rate_type = 1 THEN
                ROW_LOC.room_base_cost := (ROW_LOC.room_base_cost * 1) *
                    ROW_LOC.stay_length;
            ELSIF ROW_LOC.rate_type = 2 THEN
                ROW_LOC.room_base_cost := (ROW_LOC.room_base_cost * 1.25) *
                    ROW_LOC.stay_length;
            ELSIF ROW_LOC.rate_type = 3 THEN
                ROW_LOC.room_base_cost := (ROW_LOC.room_base_cost * 1.5) *
                    ROW_LOC.stay_length;
            ELSIF ROW_LOC.rate_type = 4 THEN
                ROW_LOC.room_base_cost := (ROW_LOC.room_base_cost * 1.75) *
                    ROW_LOC.stay_length;
            ELSE
                ROW_LOC.room_base_cost := (ROW_LOC.room_base_cost * 2) *
                    ROW_LOC.stay_length;
            END IF;
     --discount
           IF ROW_Loc.stay_length >= 62 THEN
             ROW_LOC.room_base_cost := ROW_LOC.room_base_cost * .9;
            ELSIF ROW_Loc.stay_length >= 31 THEN
             ROW_LOC.room_base_cost := ROW_LOC.room_base_cost * .95;
            END IF;
        END IF;
        END LOOP;
        close R; */
open S;
        temp_year_var := 0;
        LOOP
            FETCH S INTO Row_Loc2;
            EXIT WHEN S%NOTFOUND;
                IF (temp_year_var != Row_Loc2.year) AND (temp_year_var !=
                    Row_Loc2.month) AND (temp_year_var != 0) THEN
                    DBMS_OUTPUT.PUT_LINE(chr(10));
                END IF;
                temp_year_var := Row_Loc2.year;
                temp_year_var := Row_Loc2.month;
                DBMS_OUTPUT.PUT_LINE('Year: ' || Row_Loc2.year || ' Month: ' ||
                    Row_Loc2.month);
                DBMS_OUTPUT.PUT_LINE(' Service type: ' || Row_Loc2.service_type
                    || ' Service Profit: ' || Row_Loc2.month_service_cost);
            END LOOP;
        close S;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('An error was made and no data has been found');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/
```



```
Year: 2017 Month: 1
 Service type: laundry Service Profit: 20
Year: 2017 Month: 1
 Service type: restaurant meal Service Profit: 20


Year: 2018 Month: 3
 Service type: pay-per-view movie Service Profit: 10
Year: 2018 Month: 3
 Service type: restaurant meal Service Profit: 20


Year: 2019 Month: 9
 Service type: laundry Service Profit: 10


Year: 2021 Month: 6
 Service type: laundry Service Profit: 10
Year: 2021 Month: 6
 Service type: pay-per-view movie Service Profit: 5
```

18. Jesse[**] (Coded by Nicholas) 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. (Note that this operation was not attempted by Jesse, but was completed by Nicholas.)

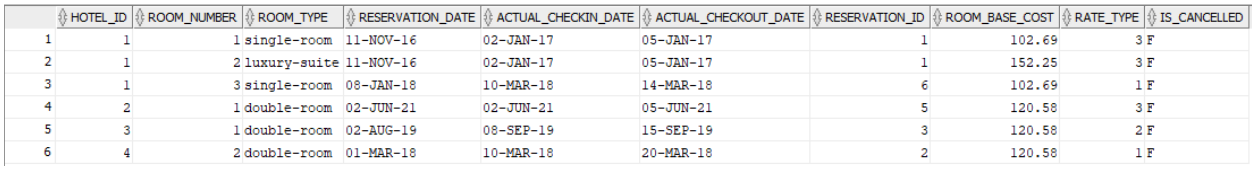
```
--There are two sub-procedures that are called by the parent function for
--this operation. The first procedure finds income for a state for all hotels
--for all dates.
CREATE OR REPLACE PROCEDURE Room_Type_Income_By_State(Hotel_State_Param IN
      Hotel.Address_State%TYPE) IS
      Room_Type_Loc Room.Room_Type%TYPE;
      Income_By_Room_Type_Loc NUMBER;
      CURSOR C IS
```

```
--First find room types that are not found within the main query, then
--union the results of this operation with the results from the main
--query into one table containing all the incomes by room type.
SELECT Room_Type, 0 AS Income_By_Room_Type FROM
       (SELECT DISTINCT Room_Type
       FROM Room
       WHERE Room_Type NOT IN
       (SELECT DISTINCT Room_Type
       FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
       WHERE RE.Reservation_ID = RA.Reservation_ID
       AND RO.Room_Number = RA.Room_Number
       AND RO.Hotel_ID = RA.Hotel_ID
       AND H.Hotel_ID = RO.Hotel_ID
       AND H.Address_State = Hotel_State_Param
       AND Is_Cancelled = 'F'))
--Union together income by existing room types and non-existing room types.
UNION
--Find the total income from each type of room with SUM and group by
 --room type.
SELECT Room_Type, SUM(Room_Cost_Final) AS Income_By_Room_Type FROM
--Apply discount of 5% or 10% depending on how far back the reservation was
 --made to get the final cost for each room.
(SELECT Room_Type,
       CASE
       WHEN MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) >= 1 AND
             MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) < 2
             THEN Room_Cost_Rated * 0.95
       WHEN MONTHS_BETWEEN(Actual_Checkin_Date, Reservation_Date) >= 2
             THEN Room_Cost_Rated * 0.90
       ELSE
             Room_Cost_Rated * 1
       END AS Room_Cost_Final
FROM
--Then, increase the price of each room depending on the rate type of the
--reservation (25% more for rate type 2, 50% more for rate type 3).
(SELECT Room_Type, Reservation_Date, Actual_Checkin_Date,
        Actual_Checkout_Date,
       CASE Rate_Type
             WHEN 1 THEN Room_Total_Cost * 1
             WHEN 2 THEN Room_Total_Cost * 1.25
             WHEN 3 THEN Room_Total_Cost * 1.5
       END AS Room_Cost_Rated
       FROM
--First, find the aggregate base cost of rooms by multiplying their base cost
--with the number of nights stayed at the hotel.
(SELECT Room_Type, Reservation_Date, Actual_Checkin_Date,
       Actual_Checkout_Date, Rate_Type,
       TO_NUMBER(Actual_Checkout_Date - Actual_Checkin_Date) * Room_Base_Cost
```

```
				AS Room_Total_Cost
		FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
		WHERE RE.Reservation_ID = RA.Reservation_ID
		AND RO.Room_Number = RA.Room_Number
		AND RO.Hotel_ID = RA.Hotel_ID
		AND H.Hotel_ID = RO.Hotel_ID
		AND H.Address_State = Hotel_State_Param
		AND Is_Cancelled = 'F')))
		--Group main query results by Room_Type
		GROUP BY Room_Type;
BEGIN
		DBMS_OUTPUT.PUT_LINE('Income by Room Type at all hotels in state '
		|| Hotel_State_Param);
		OPEN C;
		LOOP
		FETCH C INTO Room_Type_Loc, Income_By_Room_Type_Loc;
		EXIT WHEN C%NOTFOUND;
		IF Income_By_Room_Type_Loc = 0 THEN
				DBMS_OUTPUT.PUT_LINE(Room_Type_Loc || ': $0.00');
		ELSE
				DBMS_OUTPUT.PUT_LINE(Room_Type_Loc || ': ' ||
				LTRIM(TO_CHAR(ROUND(Income_By_Room_Type_Loc, 2), '$9999999.99')));
		END IF;
		END LOOP;
		CLOSE C;
EXCEPTION
		WHEN OTHERS THEN
		DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The second sub-procedure calculates income for a state for all hotels for
--all dates.
CREATE OR REPLACE PROCEDURE Services_Income_By_State(Hotel_State_Param IN
		Hotel.Address_State%TYPE) IS
		Service_Type_Loc Service_Event.Service_Type%TYPE;
		Service_Cost_Loc NUMBER;
		CURSOR C IS
		SELECT Service_Type, 0 AS Service_Cost FROM
		(SELECT DISTINCT Service_Type
		FROM Service_Event
		WHERE Service_Type NOT IN
				(SELECT Service_Type
				FROM Reservation R, Service_Event S, Hotel H
				WHERE R.Reservation_ID = S.Reservation_ID
				AND H.Hotel_ID = R.Hotel_ID
				AND H.Address_State = Hotel_State_Param))
		--Join together the services not found as $0 with the sum of income from
```

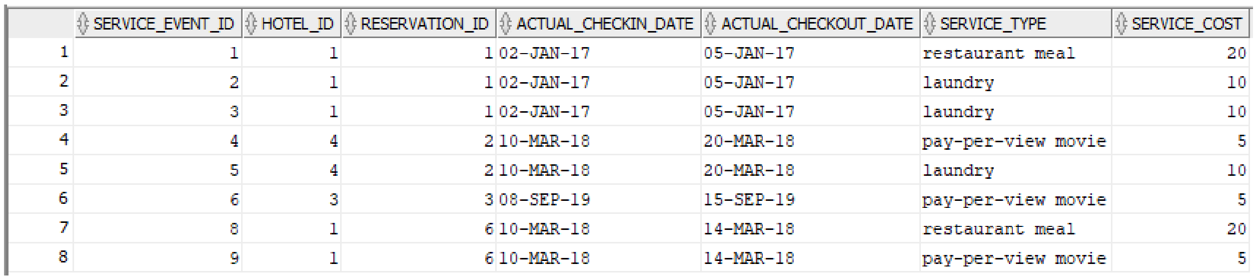
```
        --other services.
        UNION
        SELECT Service_Type, SUM(CASE Service_Type
                                  WHEN 'restaurant meal' THEN 20
                                  WHEN 'pay-per-view movie' THEN 5
                                  WHEN 'laundry' THEN 10
                                  END) AS Service_Cost
        FROM Reservation R, Service_Event S, Hotel H
        WHERE R.Reservation_ID = S.Reservation_ID
        AND H.Hotel_ID = R.Hotel_ID
        AND H.Address_State = Hotel_State_Param
        GROUP BY Service_Type;
BEGIN
        OPEN C;
        DBMS_OUTPUT.PUT_LINE('Cost of services by type at all hotels in state '
            || Hotel_State_Param);
        LOOP
              FETCH C INTO Service_Type_Loc, Service_Cost_Loc;
              EXIT WHEN C%NOTFOUND;
              IF Service_Cost_Loc = 0 THEN
              DBMS_OUTPUT.PUT_LINE(Service_Type_Loc || ': $0.00');
              ELSE
              DBMS_OUTPUT.PUT_LINE(Service_Type_Loc || ': ' ||
                    LTRIM(TO_CHAR(ROUND(Service_Cost_Loc, 2), '$9999999.99')));
              END IF;
        END LOOP;
        CLOSE C;
EXCEPTION
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An exception occurred.');
END;
/

--The parent procedure calls the two sub-procedures defined above.
CREATE OR REPLACE PROCEDURE TotalHiltonStateReport(
        State_Param IN Hotel.Address_State%TYPE) IS
BEGIN
        Room_Type_Income_By_State(State_Param);
        DBMS_OUTPUT.NEW_LINE;
        Services_Income_By_State(State_Param);
        DBMS_OUTPUT.NEW_LINE;
END;
/

--For verifying the data below for state 'MD', use the following select statements:

--This select statement shows details of all rooms found in all reservations for all
--hotels in the state of 'MD'.
```

```
SELECT H.Hotel_ID, RO.Room_Number, Room_Type, Reservation_Date, Actual_Checkin_Date,
       Actual_Checkout_Date, RE.Reservation_ID, Room_Base_Cost, Rate_Type,
       Is_Cancelled
FROM Room_Assignment RA, Reservation RE, Room RO, Hotel H
WHERE RE.Reservation_ID = RA.Reservation_ID
AND RO.Room_Number = RA.Room_Number
AND RO.Hotel_ID = RA.Hotel_ID
AND H.Hotel_ID = RO.Hotel_ID
AND H.Address_State = 'MD'
ORDER BY RO.Hotel_ID, RO.Room_Number;
```



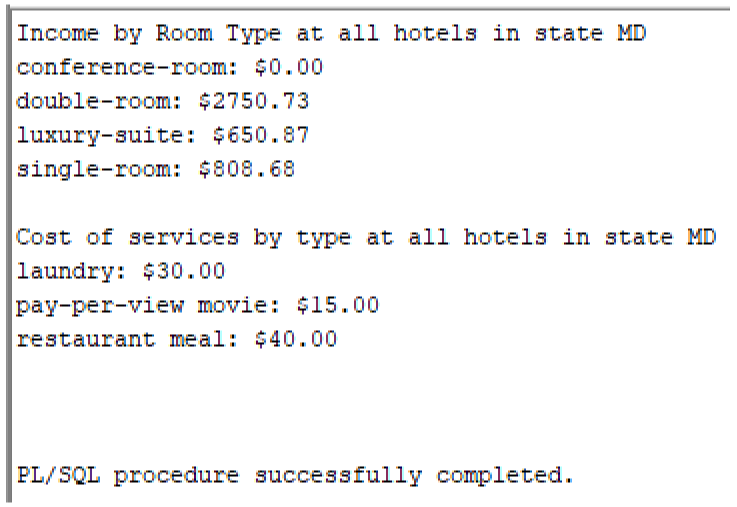
	HOTEL_ID	ROOM_NUMBER	ROOM_TYPE	RESERVATION_DATE	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	RESERVATION_ID	ROOM_BASE_COST	RATE_TYPE	IS_CANCELLED
1	1	1	single-room	11-NOV-16	02-JAN-17	05-JAN-17	1	102.69	3	F
2	1	2	luxury-suite	11-NOV-16	02-JAN-17	05-JAN-17	1	152.25	3	F
3	1	3	single-room	08-JAN-18	10-MAR-18	14-MAR-18	6	102.69	1	F
4	2	1	double-room	02-JUN-21	02-JUN-21	05-JUN-21	5	120.58	3	F
5	3	1	double-room	02-AUG-19	08-SEP-19	15-SEP-19	3	120.58	2	F
6	4	2	double-room	01-MAR-18	10-MAR-18	20-MAR-18	2	120.58	1	F

```
--This select statement shows all service events in all reservations for all hotels
--in the state of 'MD'.
SELECT Service_Event_ID, H.Hotel_ID, R.Reservation_ID, Actual_Checkin_Date,
      Actual_Checkout_Date, Service_Type,
      CASE Service_Type
      WHEN 'restaurant meal' THEN 20
      WHEN 'pay-per-view movie' THEN 5
      WHEN 'laundry' THEN 10
      END AS Service_Cost
FROM Reservation R, Service_Event S, Hotel H
WHERE R.Reservation_ID = S.Reservation_ID
AND H.Hotel_ID = R.Hotel_ID
AND H.Address_State = 'MD'
ORDER BY Service_Event_ID;
```

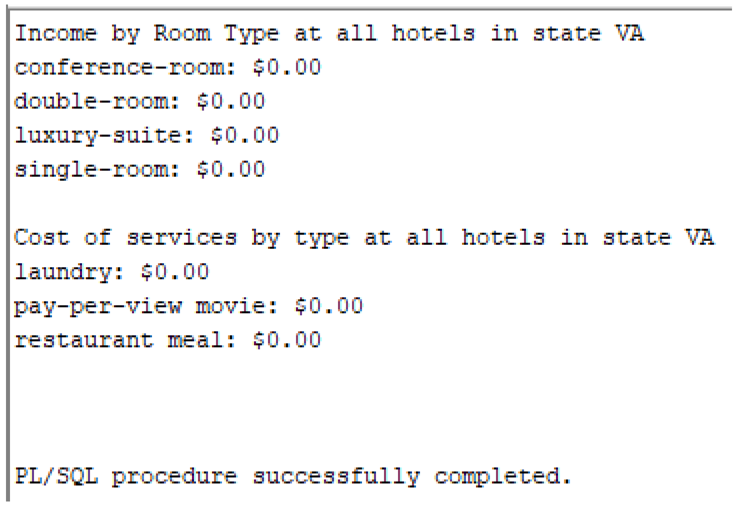


	SERVICE_EVENT_ID	HOTEL_ID	RESERVATION_ID	ACTUAL_CHECKIN_DATE	ACTUAL_CHECKOUT_DATE	SERVICE_TYPE	SERVICE_COST
1	1	1	1	02-JAN-17	05-JAN-17	restaurant meal	20
2	2	1	1	02-JAN-17	05-JAN-17	laundry	10
3	3	1	1	02-JAN-17	05-JAN-17	laundry	10
4	4	4	2	10-MAR-18	20-MAR-18	pay-per-view movie	5
5	5	4	2	10-MAR-18	20-MAR-18	laundry	10
6	6	3	3	08-SEP-19	15-SEP-19	pay-per-view movie	5
7	8	1	6	10-MAR-18	14-MAR-18	restaurant meal	20
8	9	1	6	10-MAR-18	14-MAR-18	pay-per-view movie	5

```
--The following procedure call prints the operation report showing income by room
--type and service type for the state MD. Output is shown below. Income for a
--non-reserved room type is correctly given as 0 (see conference room income in
--output). Results can be verified by looking at the two select statement
--screenshots above.
EXEC TotalHiltonStateReport('MD');
```


```
Income by Room Type at all hotels in state MD
conference-room: $0.00
double-room: $2750.73
luxury-suite: $650.87
single-room: $808.68

Cost of services by type at all hotels in state MD
laundry: $30.00
pay-per-view movie: $15.00
restaurant meal: $40.00



PL/SQL procedure successfully completed.
```

```
--The following procedure call tests an edge case showing income for a state not
--found in the group data set. $0 is correctly printed for all sections. Output
--is shown below:
EXEC TotalHiltonStateReport('VA');
```


```
Income by Room Type at all hotels in state VA
conference-room: $0.00
double-room: $0.00
luxury-suite: $0.00
single-room: $0.00

Cost of services by type at all hotels in state VA
laundry: $0.00
pay-per-view movie: $0.00
restaurant meal: $0.00



PL/SQL procedure successfully completed.
```