CSCE – 5350 – Fundamentals of Database System

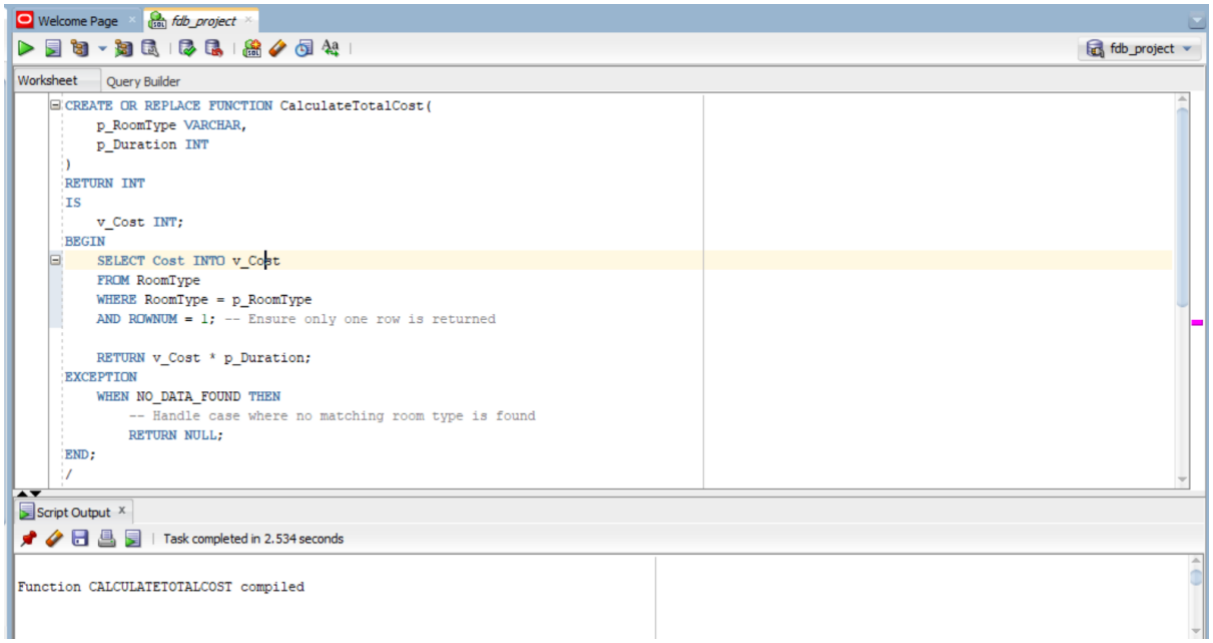
(PL/SQL Code)

Group - 18

STORED FUNCTIONS:

1. To find the total cost based on room type and duration of stay

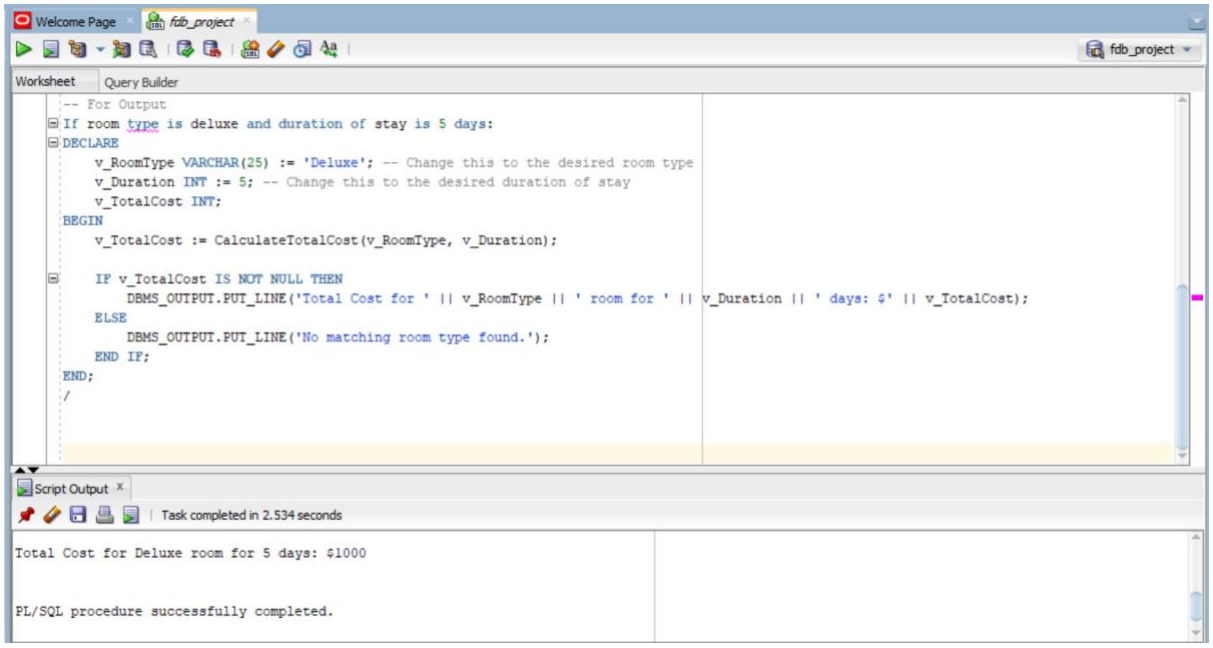
```
CREATE OR REPLACE FUNCTION CalculateTotalCost(
   p_RoomType VARCHAR,
   p_Duration INT
)
RETURN INT
IS
   v_Cost INT;
BEGIN
   SELECT Cost INTO v_Cost
   FROM RoomType
   WHERE RoomType = p_RoomType
   AND ROWNUM = 1; -- Ensure only one row is returned

   RETURN v_Cost * p_Duration;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      -- Handle case where no matching room type is found
      RETURN NULL;
END;
/
```

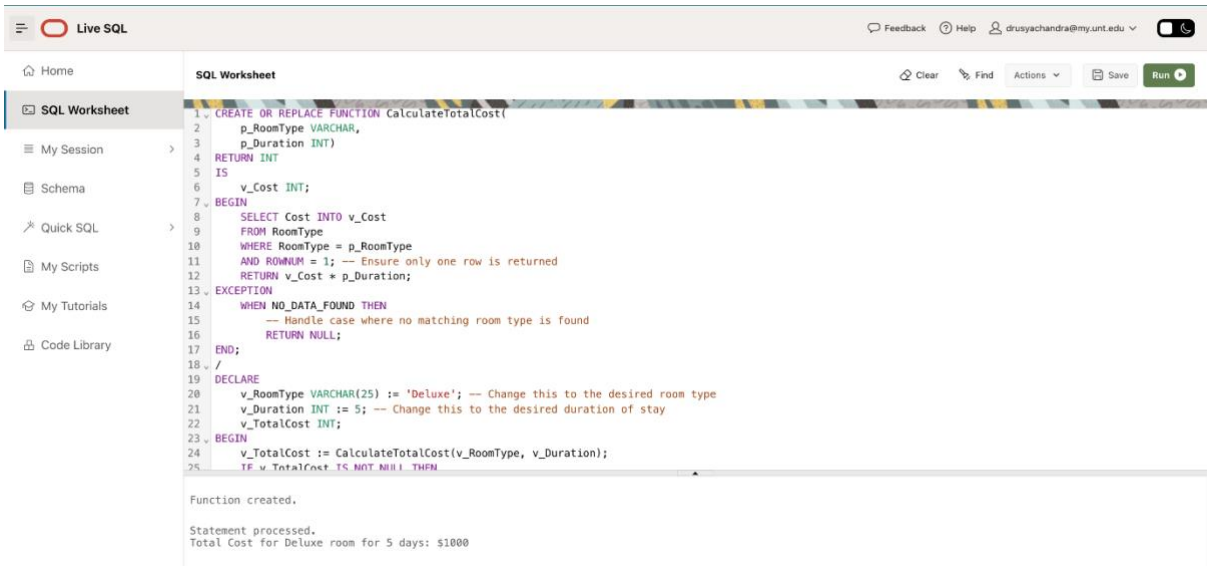


-- For Output

If room type is deluxe and duration of stay is 5 days:

```
DECLARE
   v_RoomType VARCHAR(25) := 'Deluxe'; -- Change this to the desired room type
   v_Duration INT := 5; -- Change this to the desired duration of stay
   v_TotalCost INT;
BEGIN
   v_TotalCost := CalculateTotalCost(v_RoomType, v_Duration);


   IF v_TotalCost IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE('Total Cost for ' || v_RoomType || ' room for ' || v_Duration || ' days: $' || v_TotalCost);
   ELSE
      DBMS_OUTPUT.PUT_LINE('No matching room type found.');
   END IF;
END;
/
```

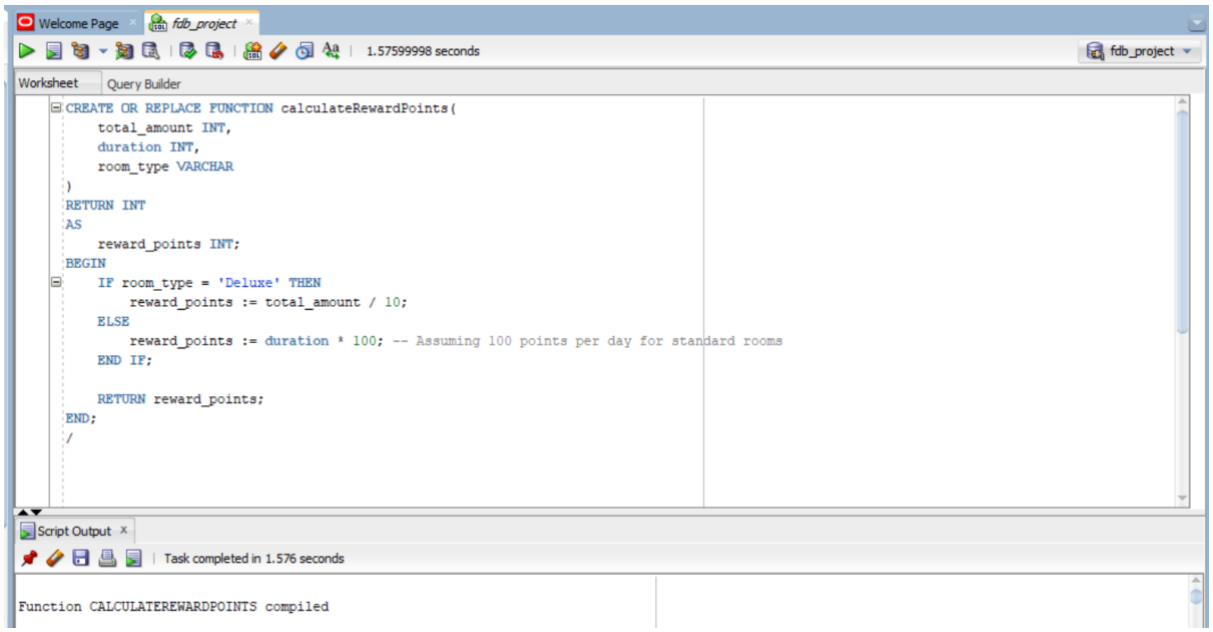




2. function to determine a guest's reward point total based on the amount they have booked. Depending on the hotel's policy, the computation may differ. For example, one point may be earned for every dollar spent, or points may be earned based on the length of stay and kind of accommodation.

```
CREATE OR REPLACE FUNCTION calculateRewardPoints(
    total_amount INT,
    duration INT,
```

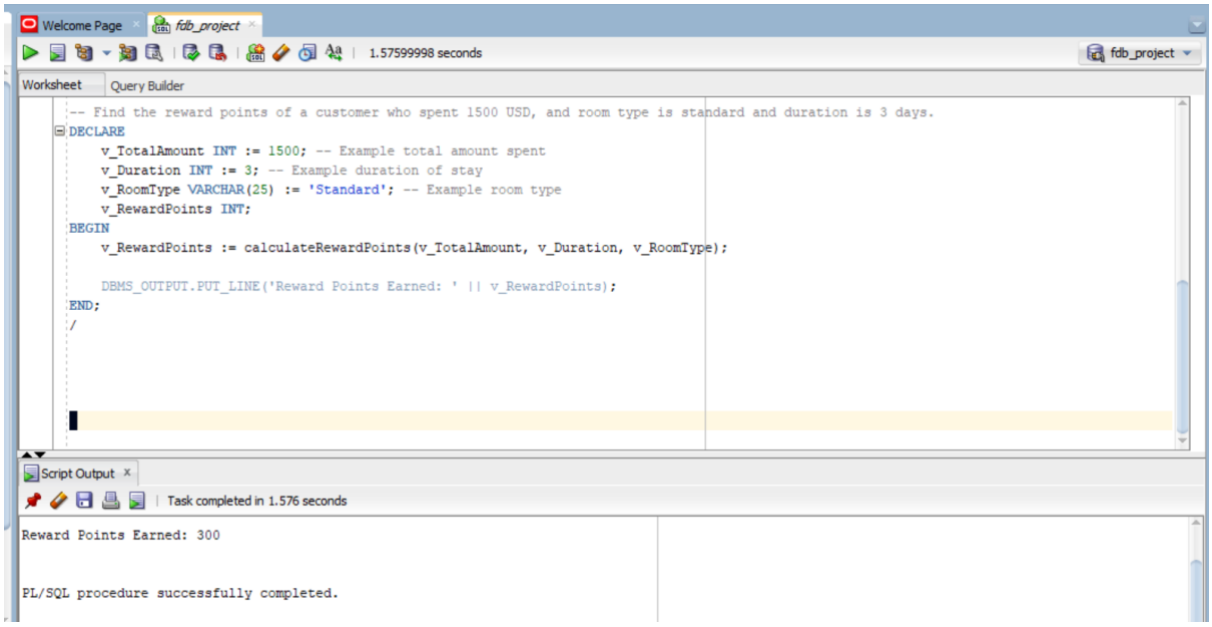
```
    room_type VARCHAR
)
RETURN INT
AS
    reward_points INT;
BEGIN
    IF room_type = 'Deluxe' THEN
        reward_points := total_amount / 10;
    ELSE
        reward_points := duration * 100; -- Assuming 100 points per day for standard rooms
    END IF;

    RETURN reward_points;
END;
/
```

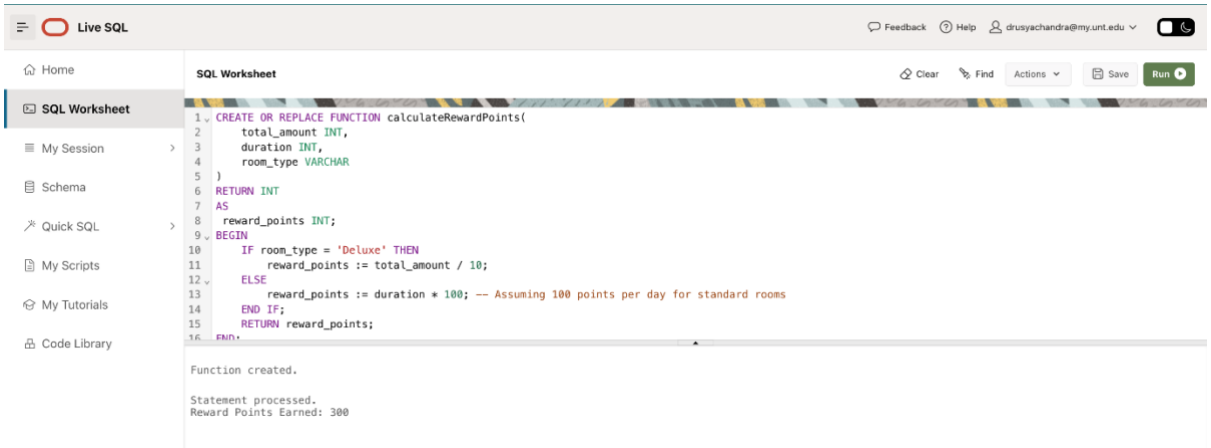


-- Find the reward points of a customer who spent 1500 USD, and room type is standard and duration is 3 days.

```
DECLARE
    v_TotalAmount INT := 1500; -- Example total amount spent
    v_Duration INT := 3; -- Example duration of stay
```

```
    v_RoomType VARCHAR(25) := 'Standard'; -- Example room type
    v_RewardPoints INT;
BEGIN
    v_RewardPoints := calculateRewardPoints(v_TotalAmount, v_Duration, v_RoomType);


    DBMS_OUTPUT.PUT_LINE('Reward Points Earned: ' || v_RewardPoints);
END;
/
```



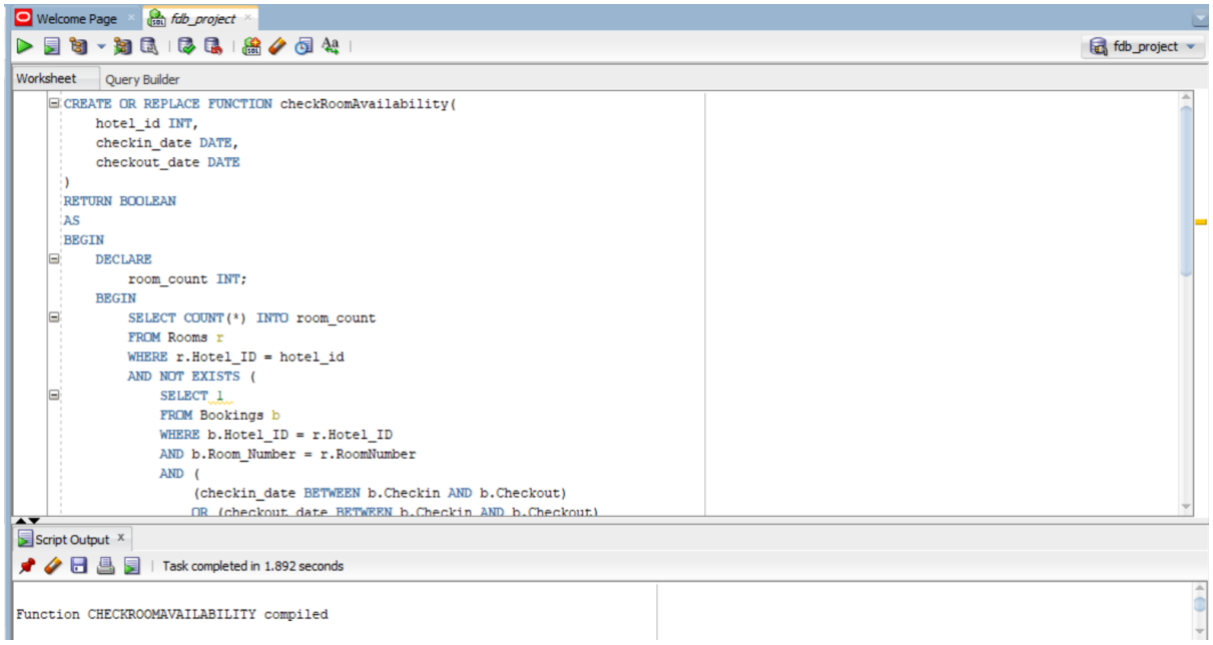


3. checkRoomAvailability: This function checks the availability of rooms in a hotel for a given check-in and check-out date range.

```
CREATE OR REPLACE FUNCTION checkRoomAvailability(
    hotel_id INT,
```

```
    checkin_date DATE,
    checkout_date DATE
)
RETURN BOOLEAN
AS
BEGIN
    DECLARE
        room_count INT;
    BEGIN
        SELECT COUNT(*) INTO room_count
        FROM Rooms r
        WHERE r.Hotel_ID = hotel_id
        AND NOT EXISTS (
            SELECT 1
            FROM Bookings b
            WHERE b.Hotel_ID = r.Hotel_ID
            AND b.Room_Number = r.RoomNumber
            AND (
                (checkin_date BETWEEN b.Checkin AND b.Checkout)
                OR (checkout_date BETWEEN b.Checkin AND b.Checkout)
                OR (b.Checkin BETWEEN checkin_date AND checkout_date)
            )
        );

        RETURN room_count > 0; -- Return TRUE if room_count > 0, FALSE otherwise
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN TRUE; -- No bookings found, all rooms are available
    END;
END;
```

/



-- call the function

DECLARE

v_HotelID INT := 1234; -- Example hotel ID

v_CheckinDate DATE := TO_DATE('2024-04-01', 'YYYY-MM-DD'); -- Example check-in date

v_CheckoutDate DATE := TO_DATE('2024-04-05', 'YYYY-MM-DD'); -- Example checkout date

v_RoomsAvailable BOOLEAN;

BEGIN

v_RoomsAvailable := checkRoomAvailability(v_HotelID, v_CheckinDate, v_CheckoutDate);

IF v_RoomsAvailable THEN

DBMS_OUTPUT.PUT_LINE('Rooms are available for the specified dates.');

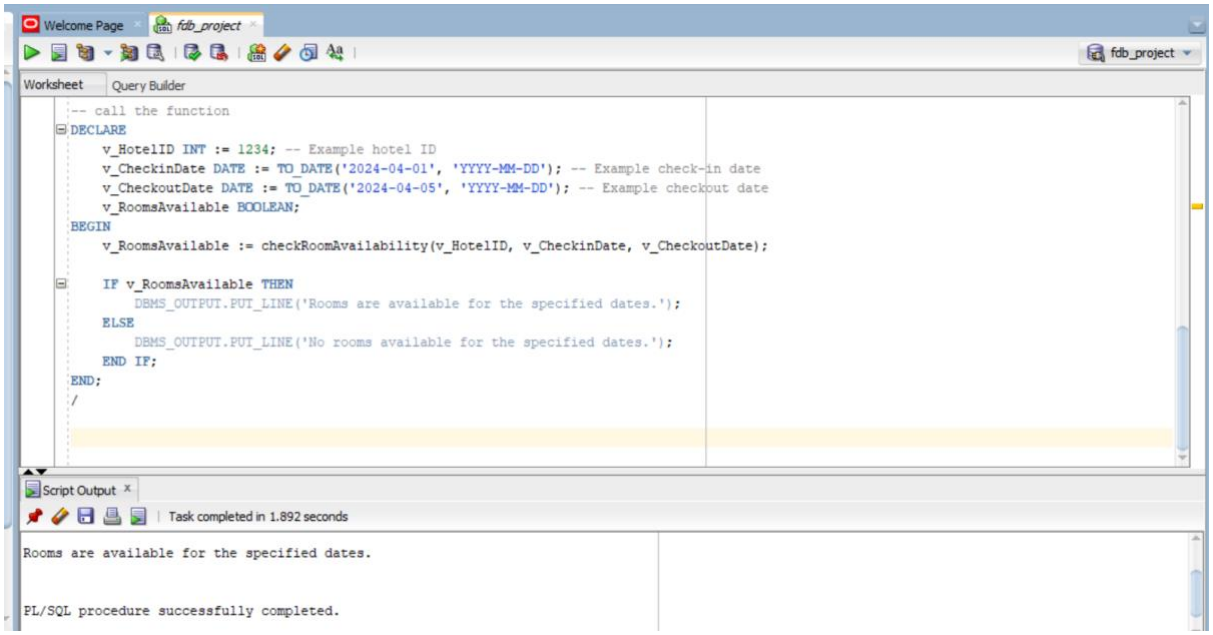
ELSE

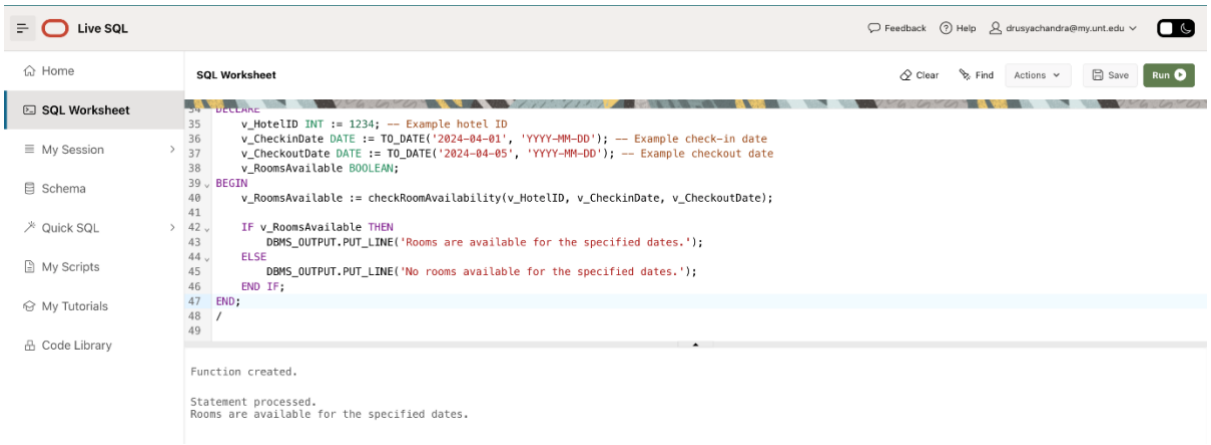
DBMS_OUTPUT.PUT_LINE('No rooms available for the specified dates.');

END IF;

END;

/

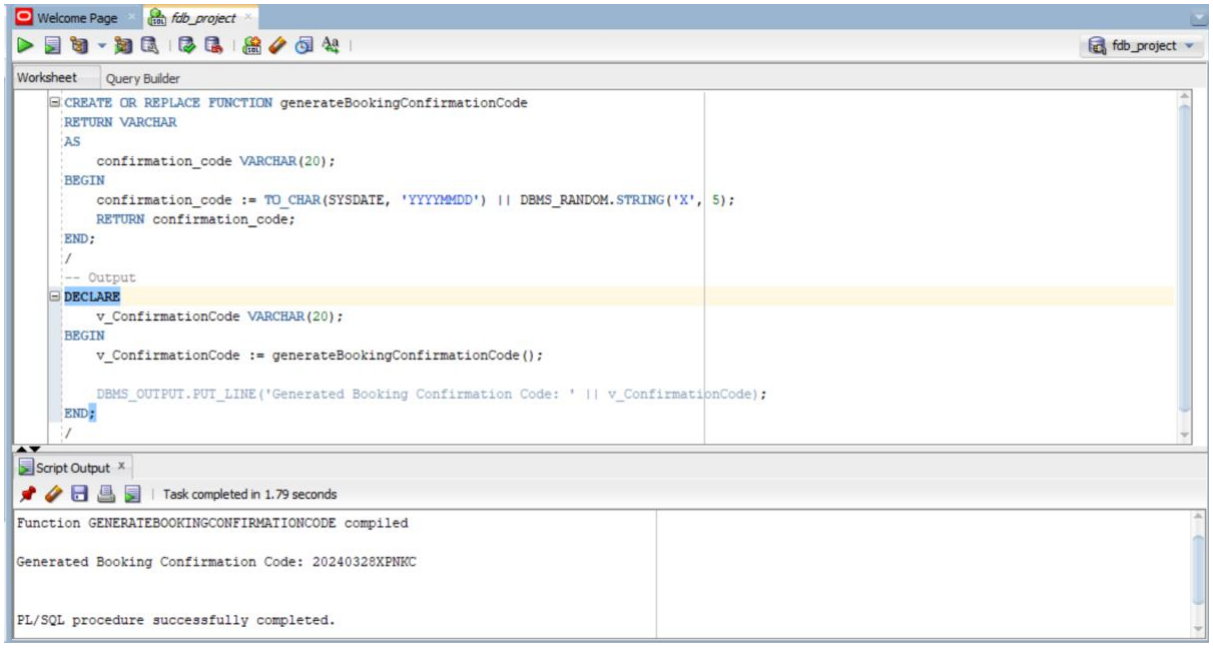


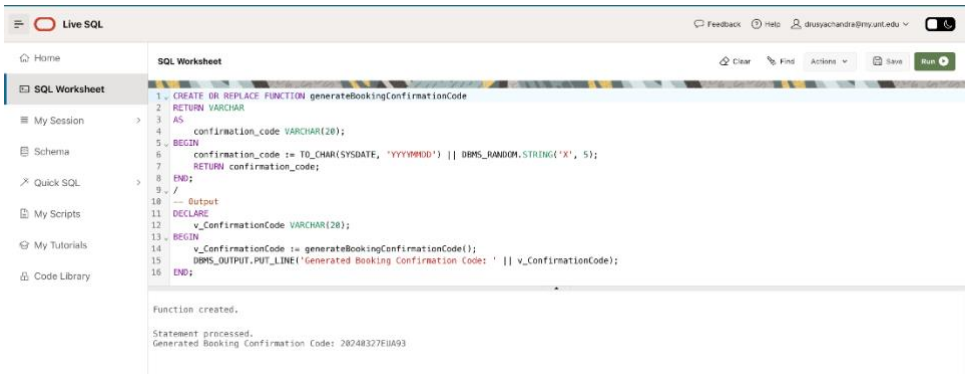


4. Function to generate a unique booking confirmation code for each booking made by concatenating the current date with a randomly generated alphanumeric string.

```
CREATE OR REPLACE FUNCTION generateBookingConfirmationCode

RETURN VARCHAR

AS

   confirmation_code VARCHAR(20);

BEGIN

   confirmation_code := TO_CHAR(SYSDATE, 'YYYYMMDD') ||
DBMS_RANDOM.STRING('X', 5);

   RETURN confirmation_code;

END;

/
```

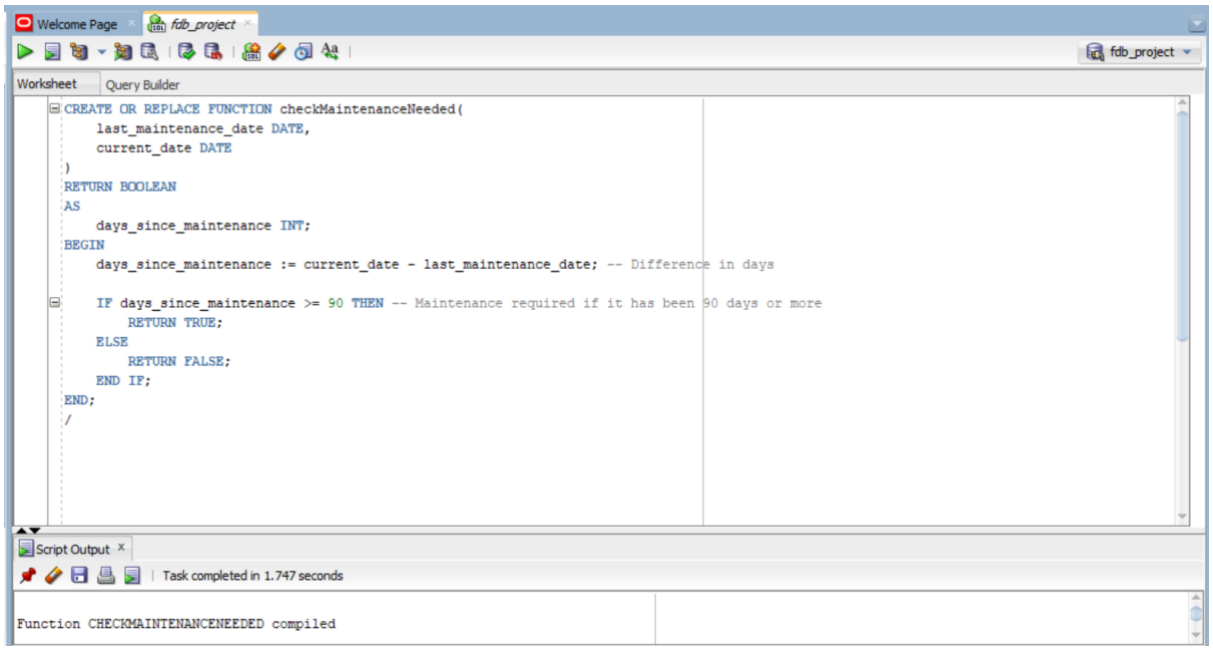
```
-- Output
DECLARE
    v_ConfirmationCode VARCHAR(20);
BEGIN
    v_ConfirmationCode := generateBookingConfirmationCode();


    DBMS_OUTPUT.PUT_LINE('Generated Booking Confirmation Code: ' ||
v_ConfirmationCode);
END;
/
```





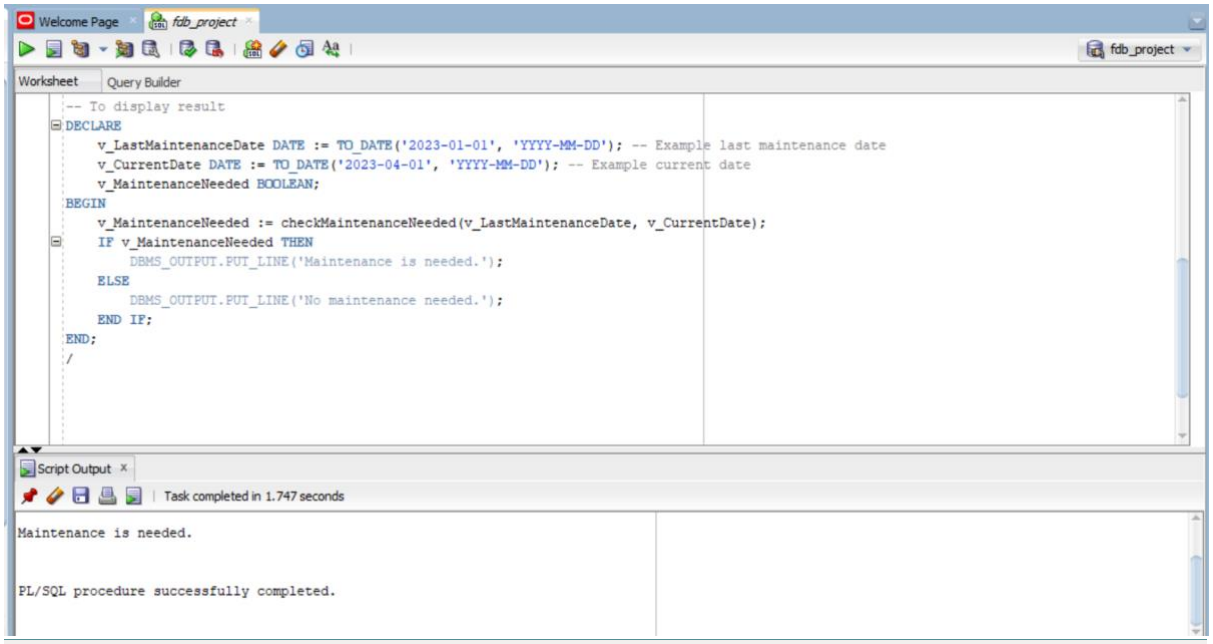
5. checkMaintenanceNeeded: Function to check whether maintenance is needed for a specific room in a hotel based on the duration since its last maintenance.

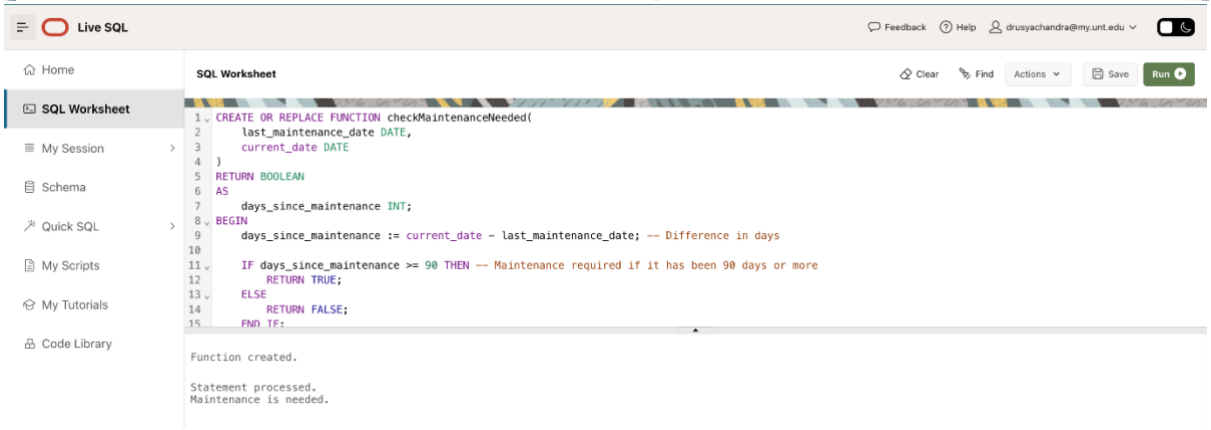
```
CREATE OR REPLACE FUNCTION checkMaintenanceNeeded(
   last_maintenance_date DATE,
   current_date DATE
)
RETURN BOOLEAN
AS
   days_since_maintenance INT;
BEGIN
   days_since_maintenance := current_date - last_maintenance_date; -- Difference in days

   IF days_since_maintenance >= 90 THEN -- Maintenance required if it has been 90 days or
more
      RETURN TRUE;
   ELSE
      RETURN FALSE;
   END IF;
END;
/
```




```
-- To display result
DECLARE
   v_LastMaintenanceDate DATE := TO_DATE('2023-01-01', 'YYYY-MM-DD'); --
Example last maintenance date
   v_CurrentDate DATE := TO_DATE('2023-04-01', 'YYYY-MM-DD'); -- Example current
date
   v_MaintenanceNeeded BOOLEAN;
BEGIN
   v_MaintenanceNeeded := checkMaintenanceNeeded(v_LastMaintenanceDate,
v_CurrentDate);
   IF v_MaintenanceNeeded THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Maintenance is needed.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No maintenance needed.');
    END IF;
END;
/
```



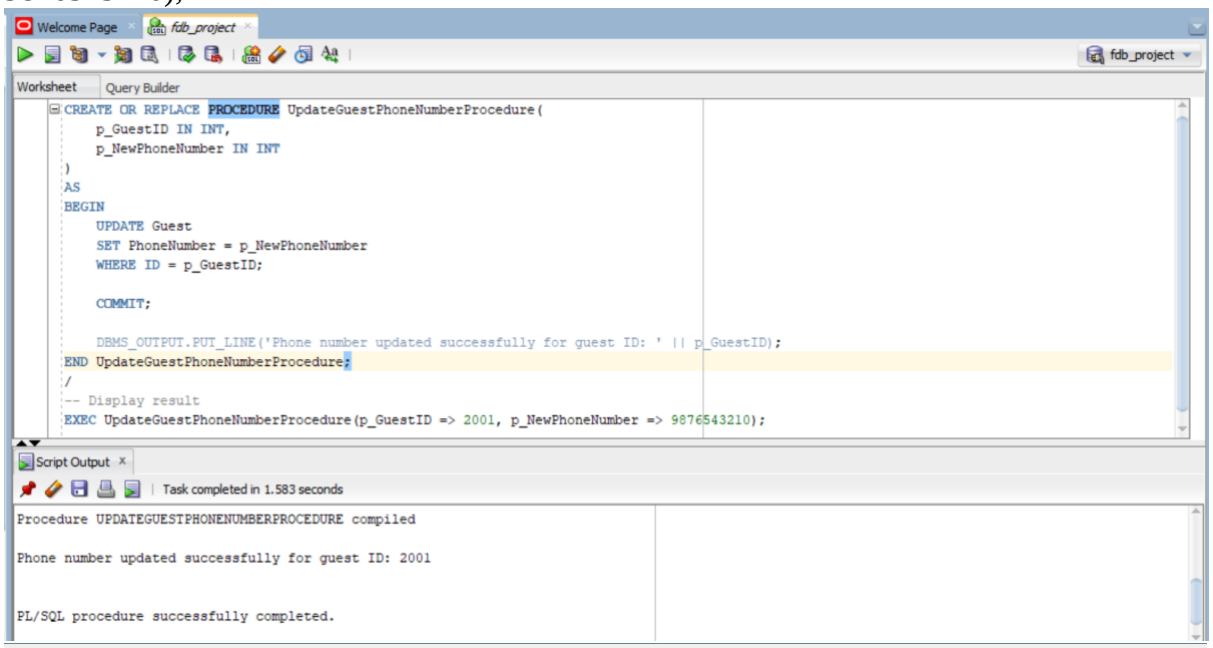


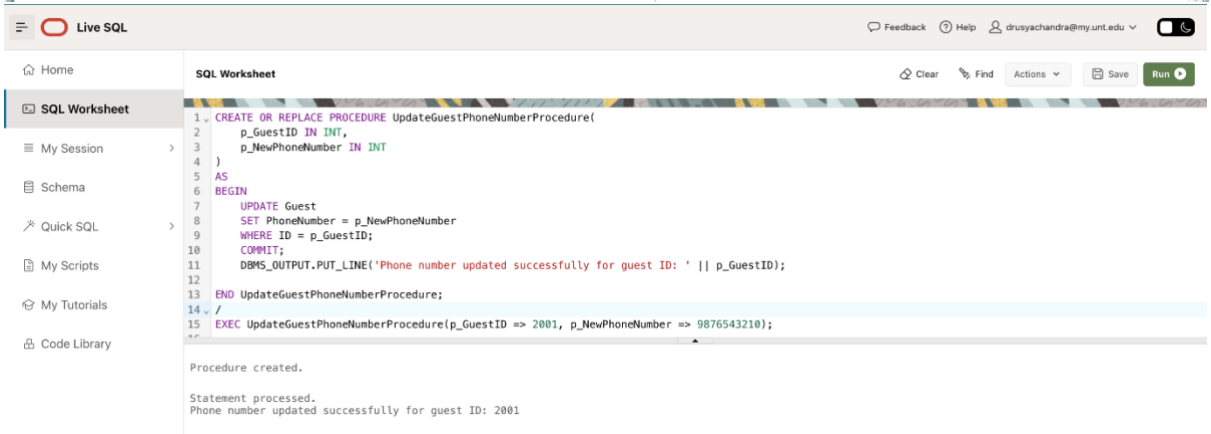
STORED PROCEDURES:

1. **UpdateGuestPhoneNumberProcedure: This stored procedure updates the phone number of a guest based on the provided guest ID.**

CREATE OR REPLACE PROCEDURE UpdateGuestPhoneNumberProcedure(

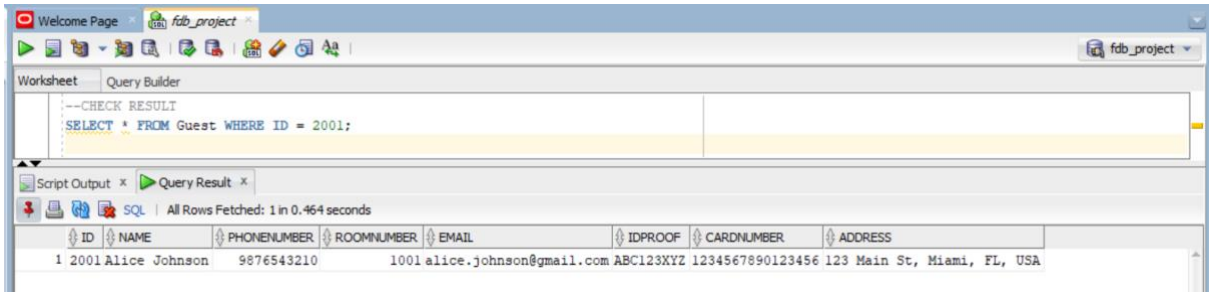
```
    p_GuestID IN INT,
    p_NewPhoneNumber IN INT
)
AS
BEGIN
    UPDATE Guest
    SET PhoneNumber = p_NewPhoneNumber
    WHERE ID = p_GuestID;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Phone number updated successfully for guest ID: ' ||
p_GuestID);
END UpdateGuestPhoneNumberProcedure;
/
-- Display result
EXEC UpdateGuestPhoneNumberProcedure(p_GuestID =>2001, p_NewPhoneNumber =>
9876543210);
```

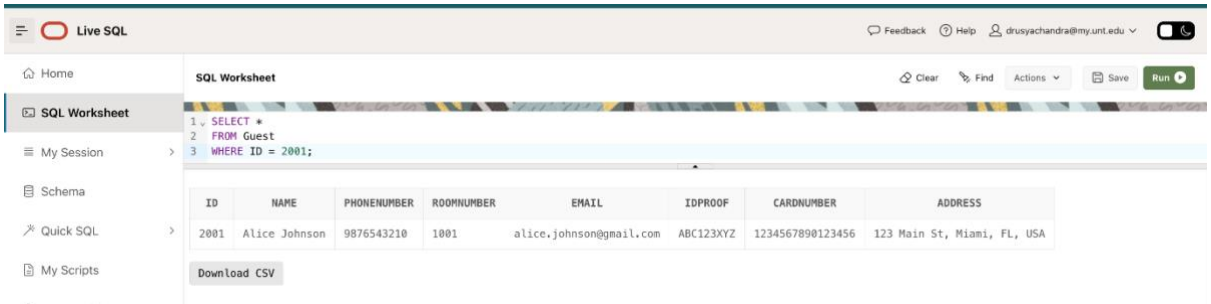





```
--CHECK RESULT
SELECT *
```

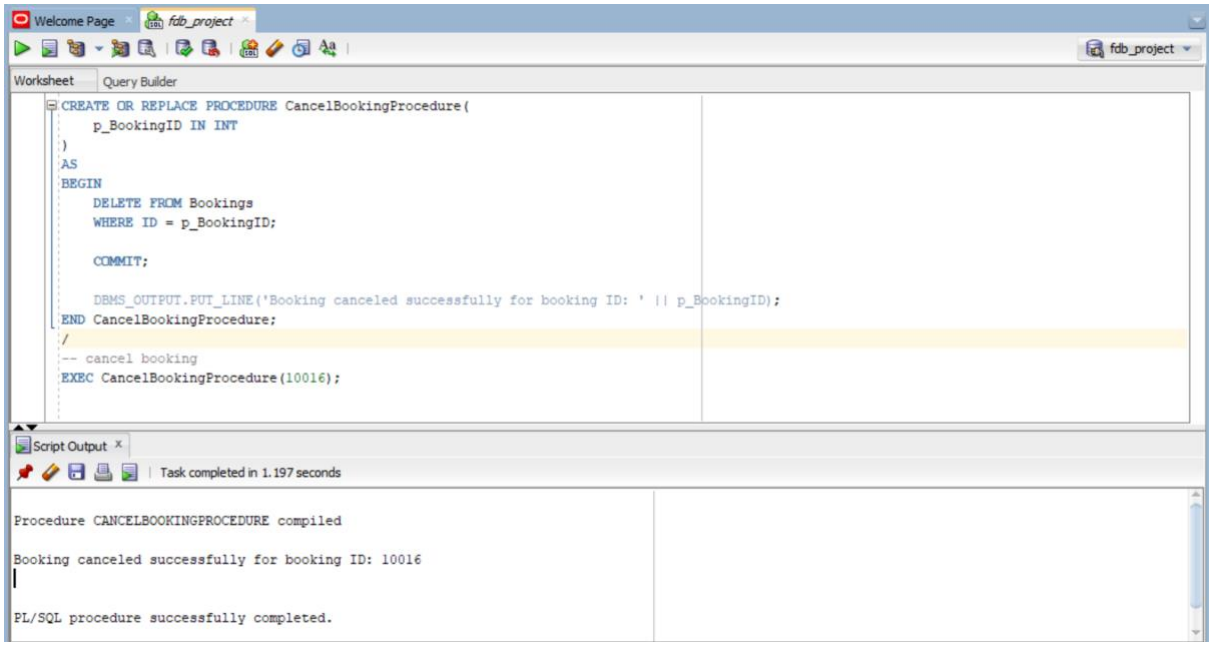
FROM Guest
WHERE ID = 2001;

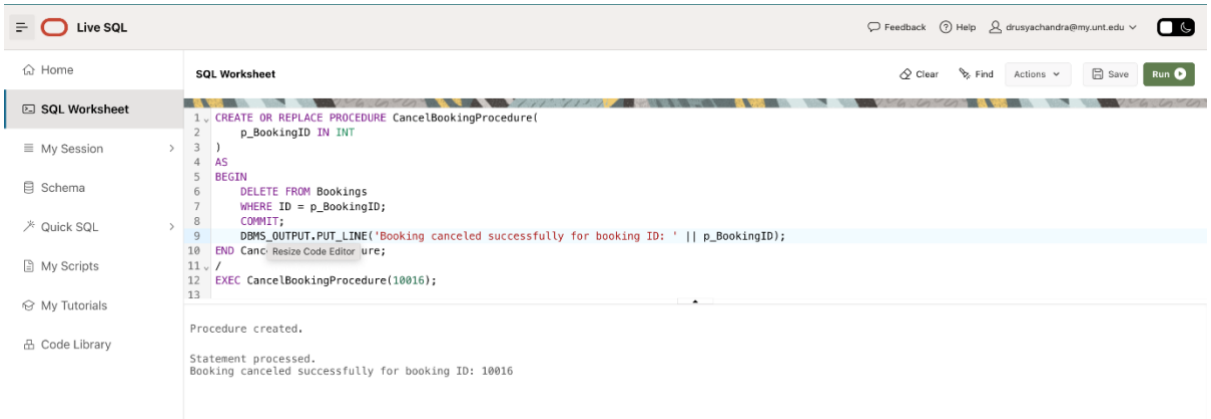




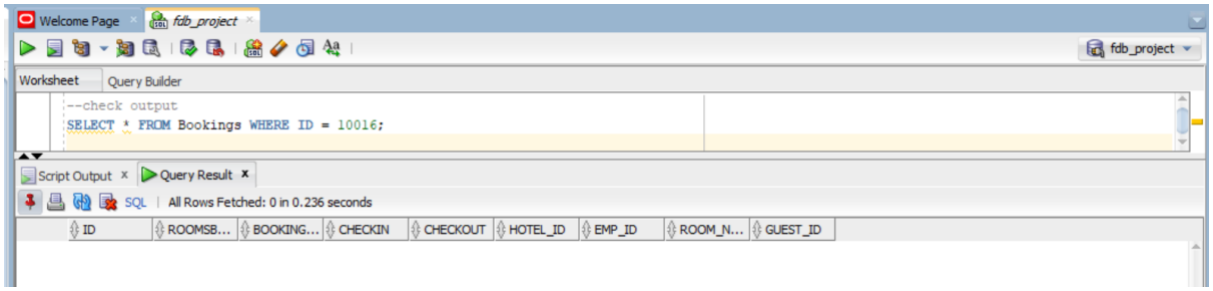
2. **CancelBookingProcedure: This stored procedure cancels a booking for a guest based on the provided booking ID.**

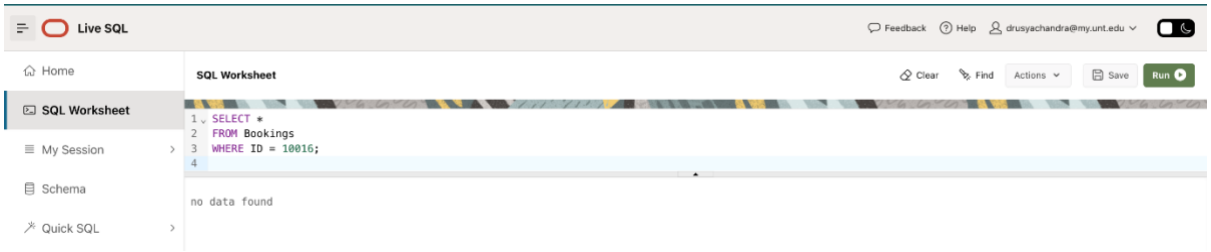
```
CREATE OR REPLACE PROCEDURE CancelBookingProcedure(
   p_BookingID IN INT
)
AS
BEGIN
   DELETE FROM Bookings
   WHERE ID = p_BookingID;

   COMMIT;

   DBMS_OUTPUT.PUT_LINE('Booking canceled successfully for booking ID: ' ||
p_BookingID);
END CancelBookingProcedure;
/
-- cancel booking
EXEC CancelBookingProcedure(10016);
```





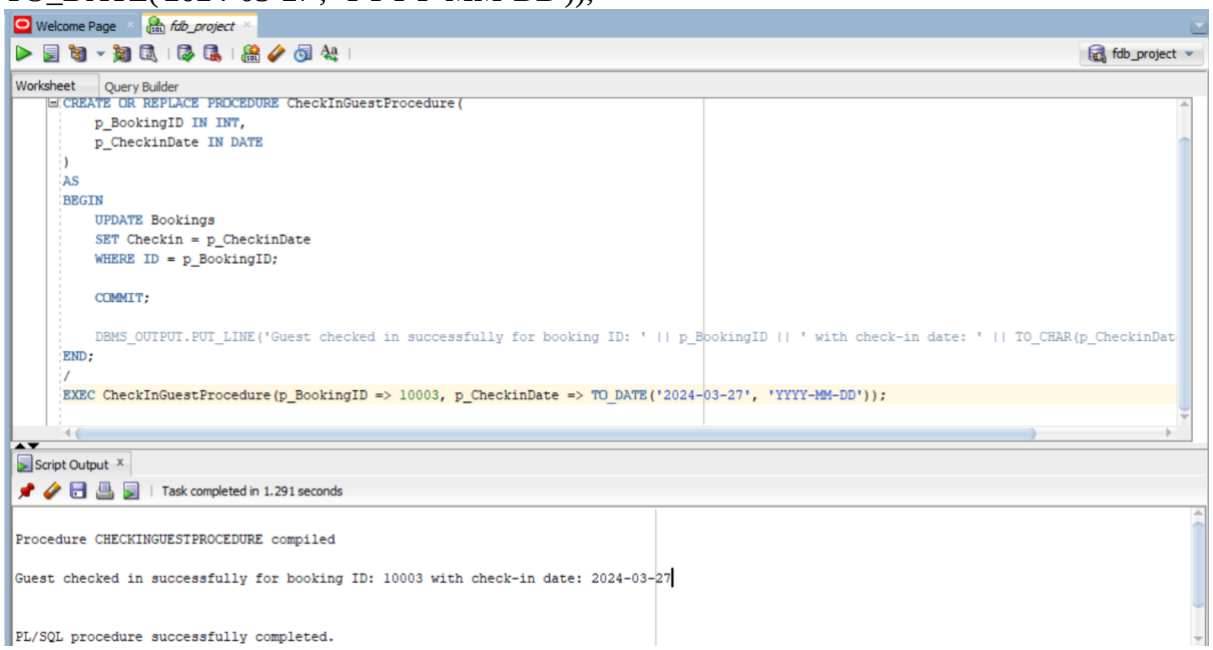
--CHECK output
SELECT *
FROM Bookings
WHERE ID = 10016;

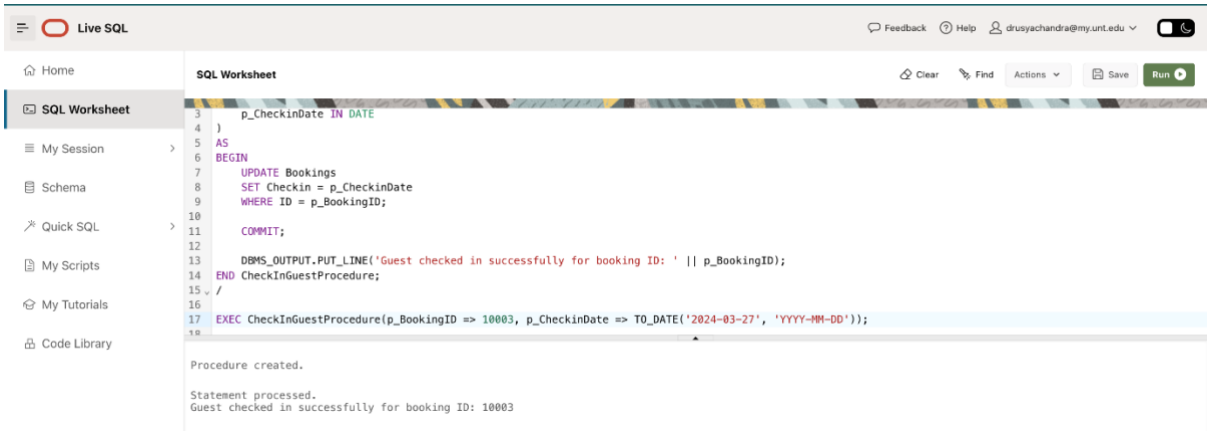




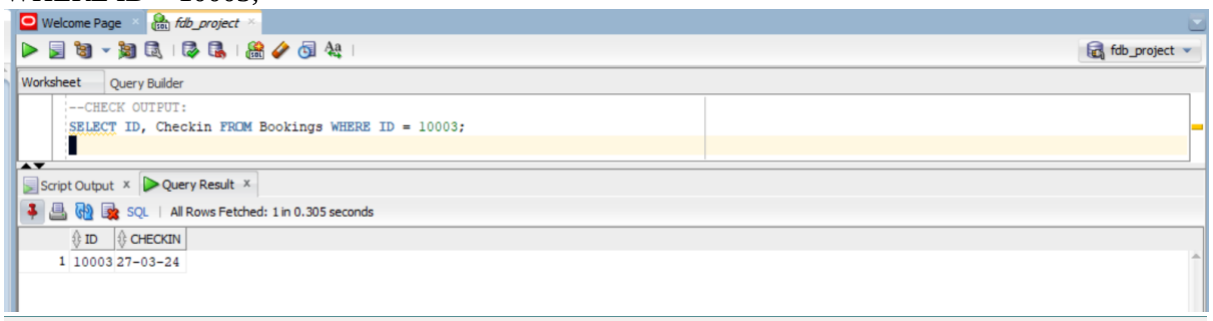
3. **CheckInGuestProcedure: This stored procedure updates the check-in date for a guest's booking based on the provided booking ID.**

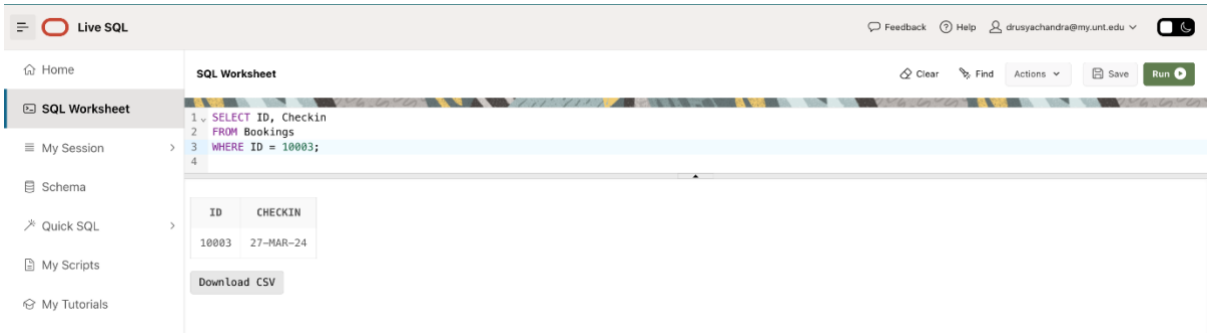
```
CREATE OR REPLACE PROCEDURE CheckInGuestProcedure(
   p_BookingID IN INT
)
AS
BEGIN
   UPDATE Bookings
   SET Checkin = SYSDATE
   WHERE ID = p_BookingID;

   COMMIT;

   DBMS_OUTPUT.PUT_LINE('Guest checked in successfully for booking ID: ' ||
p_BookingID);
END CheckInGuestProcedure;
/
EXEC CheckInGuestProcedure(p_BookingID => 10003, p_CheckinDate =>
TO_DATE('2024-03-27', 'YYYY-MM-DD'));
```





CHECK OUTPUT:
SELECT ID, Checkin
FROM Bookings
WHERE ID = 10003;

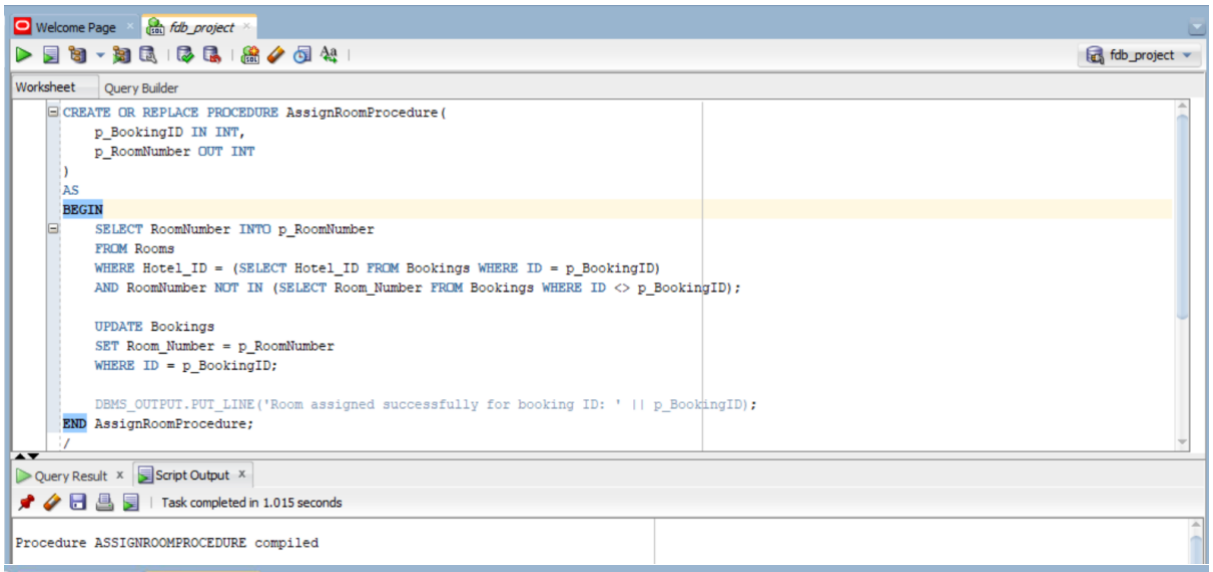


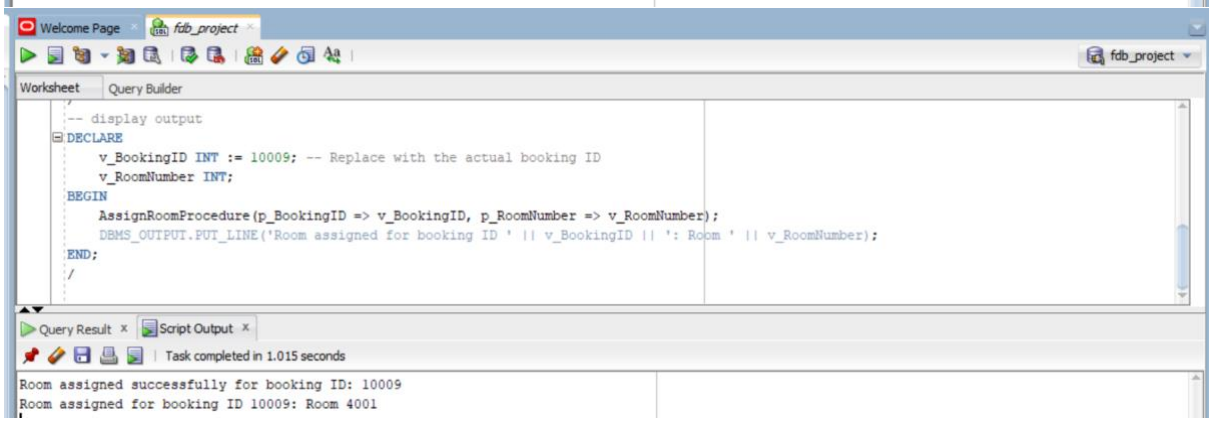


4. **AssignRoomProcedure: This procedure assigns a room to a guest based on their booking and availability.**

```
CREATE OR REPLACE PROCEDURE AssignRoomProcedure(
   p_BookingID IN INT,
   p_RoomNumber OUT INT
)
AS
BEGIN
   SELECT RoomNumber INTO p_RoomNumber
   FROM Rooms
   WHERE Hotel_ID = (SELECT Hotel_ID FROM Bookings WHERE ID = p_BookingID)
```

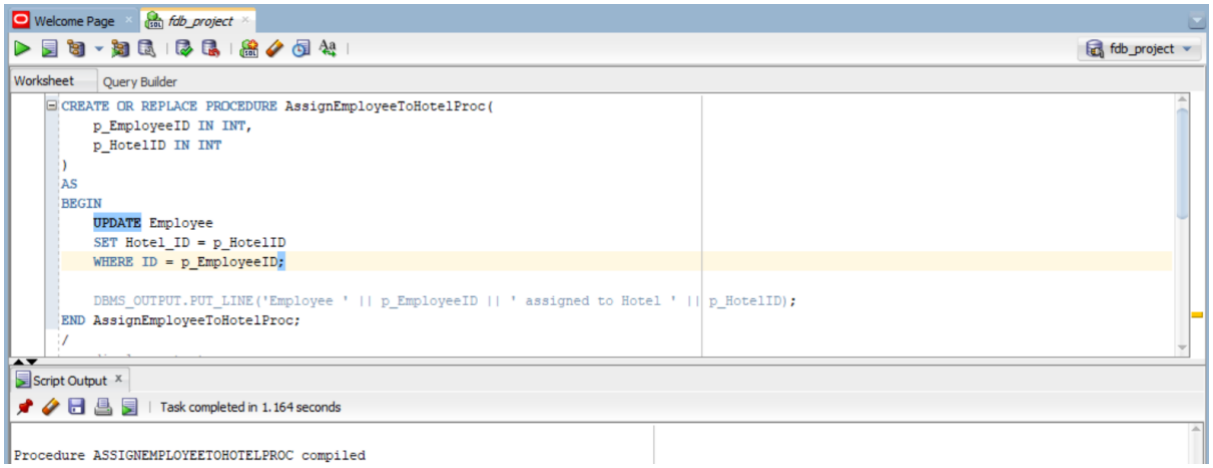
```
    AND RoomNumber NOT IN (SELECT Room_Number FROM Bookings WHERE ID <>
p_BookingID);

    UPDATE Bookings
    SET Room_Number = p_RoomNumber
    WHERE ID = p_BookingID;

    DBMS_OUTPUT.PUT_LINE('Room assigned successfully for booking ID: ' ||
p_BookingID);
END AssignRoomProcedure;
/
-- display output
DECLARE
    v_BookingID INT := 10009; -- Replace with the actual booking ID
    v_RoomNumber INT;
BEGIN
    AssignRoomProcedure(p_BookingID => v_BookingID, p_RoomNumber =>
v_RoomNumber);
    DBMS_OUTPUT.PUT_LINE('Room assigned for booking ID ' || v_BookingID || ': Room '
|| v_RoomNumber);
END;
/
```

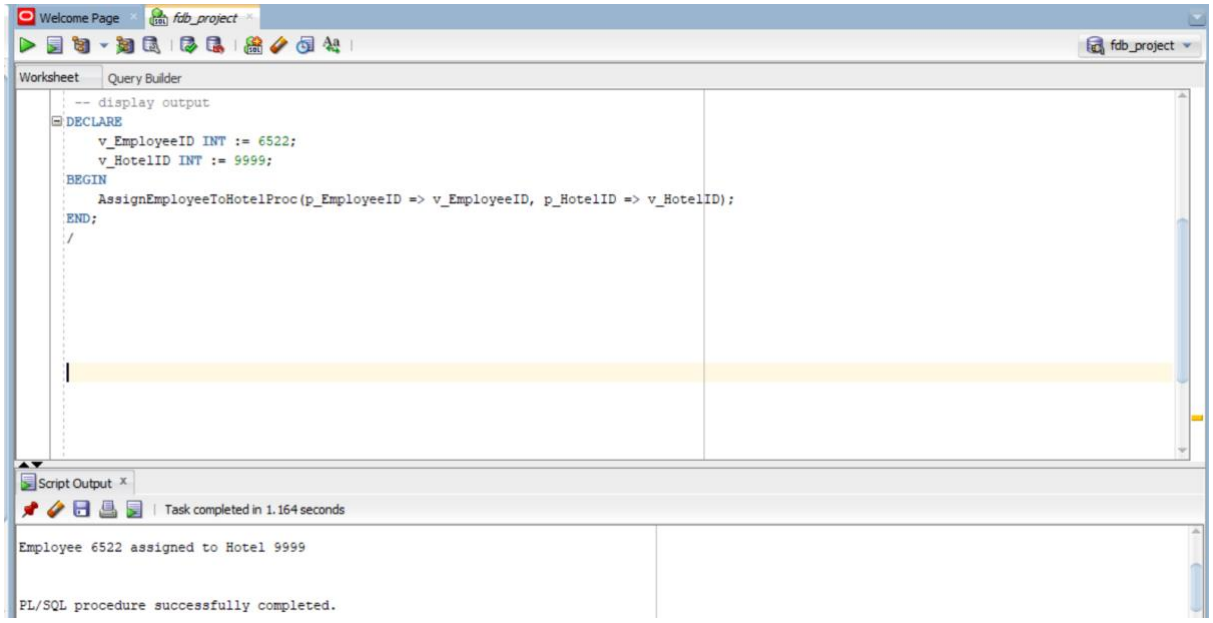




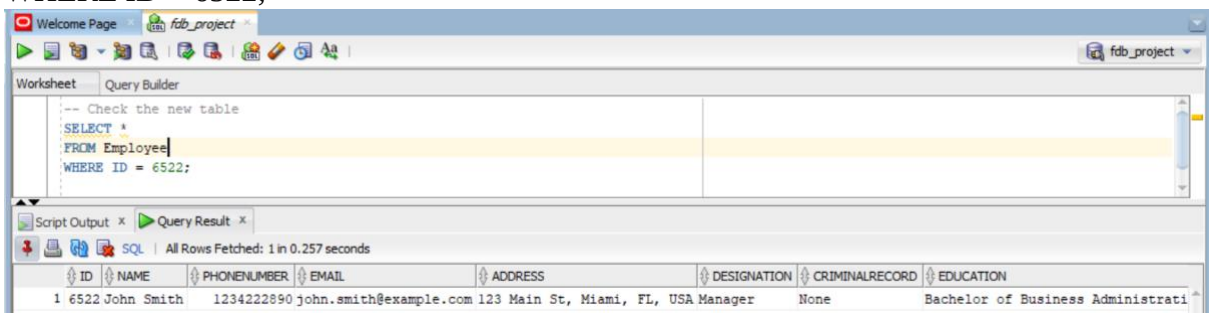
5. **AssignEmployeeToHotelProc: This procedure assigns an employee to a specific hotel within the chain.**

```
CREATE OR REPLACE PROCEDURE AssignEmployeeToHotelProc(
   p_EmployeeID IN INT,
   p_HotelID IN INT
)
AS
BEGIN
   UPDATE Employee
   SET Hotel_ID = p_HotelID
   WHERE ID = p_EmployeeID;

   DBMS_OUTPUT.PUT_LINE('Employee ' || p_EmployeeID || ' assigned to Hotel ' ||
p_HotelID);
END AssignEmployeeToHotelProc;
/
 -- display output
DECLARE
   v_EmployeeID INT := 6522;
   v_HotelID INT := 9999;
BEGIN
   AssignEmployeeToHotelProc(p_EmployeeID => v_EmployeeID, p_HotelID =>
v_HotelID);
END;
/
```





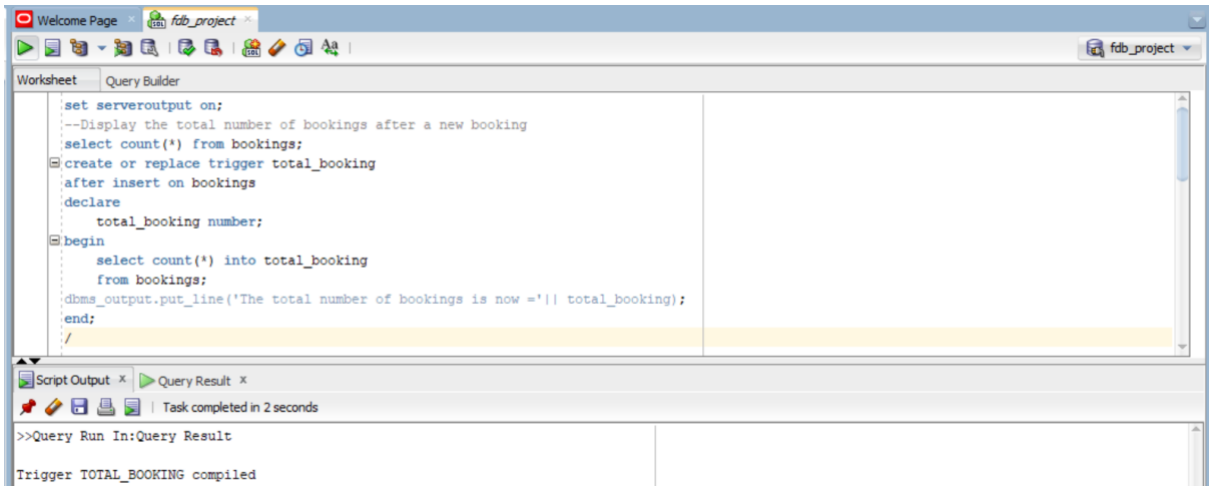
-- Check the new table
SELECT *
FROM Employee
WHERE ID = 6522;



Trigges:

1. Display the total number of bookings after a new booking

```
select count(*) from bookings;
create or replace trigger total_booking
after insert on bookings
declare
        total_booking number;
begin
        select count(*) into total_booking
   from bookings;
dbms_output.put_line('The total number of bookings is now ='|| total_booking);
end;
/
```



2. Display the total number of amenities after a new amenity is being added

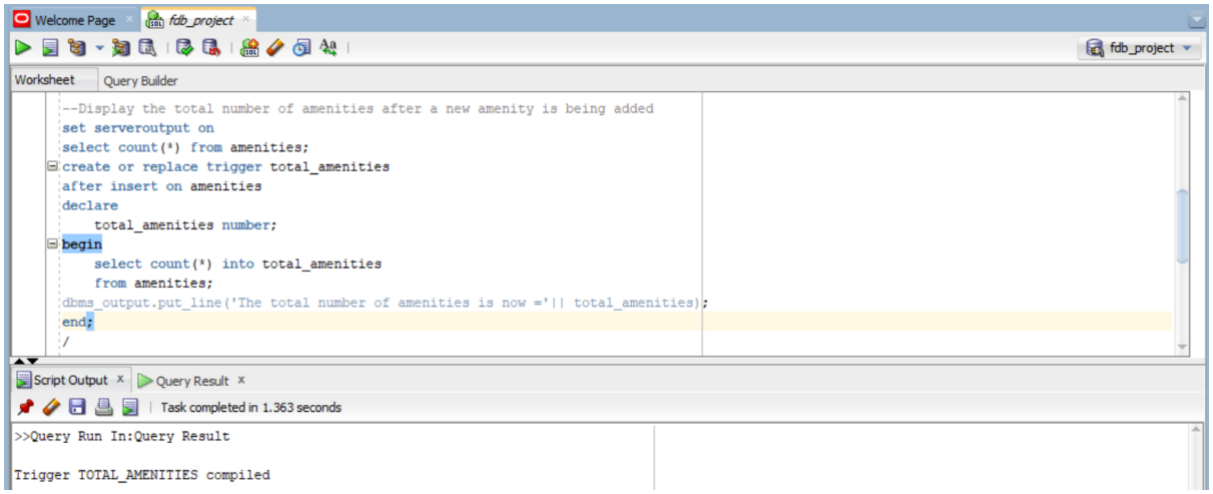
```
select count(*) from amenities;
create or replace trigger total_amenities
after insert on amenities
declare
        total_amenities number;
begin
        select count(*) into total_amenities
```

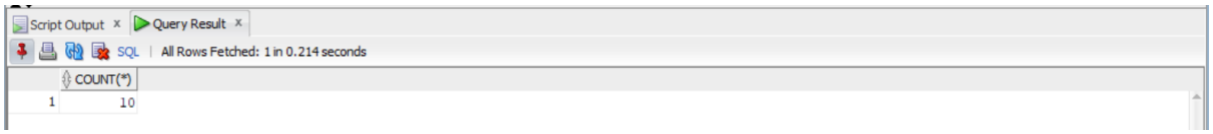
from amenities;

dbms_output.put_line('The total number of amenities is now ='|| total_amenities);

end;

/





3. Display the total number of packages after a new package is being added

select count(*) from packages;

create or replace trigger total_packages

after insert on packages

declare

total_packages number;

begin

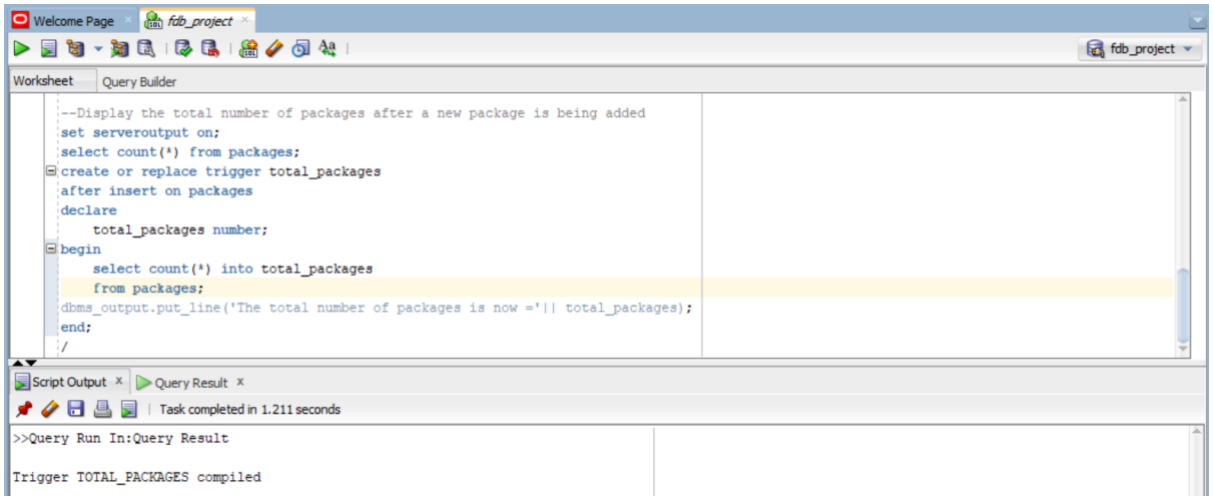
select count(*) into total_packages

from packages;

dbms_output.put_line('The total number of packages is now ='|| total_packages);

end;

/

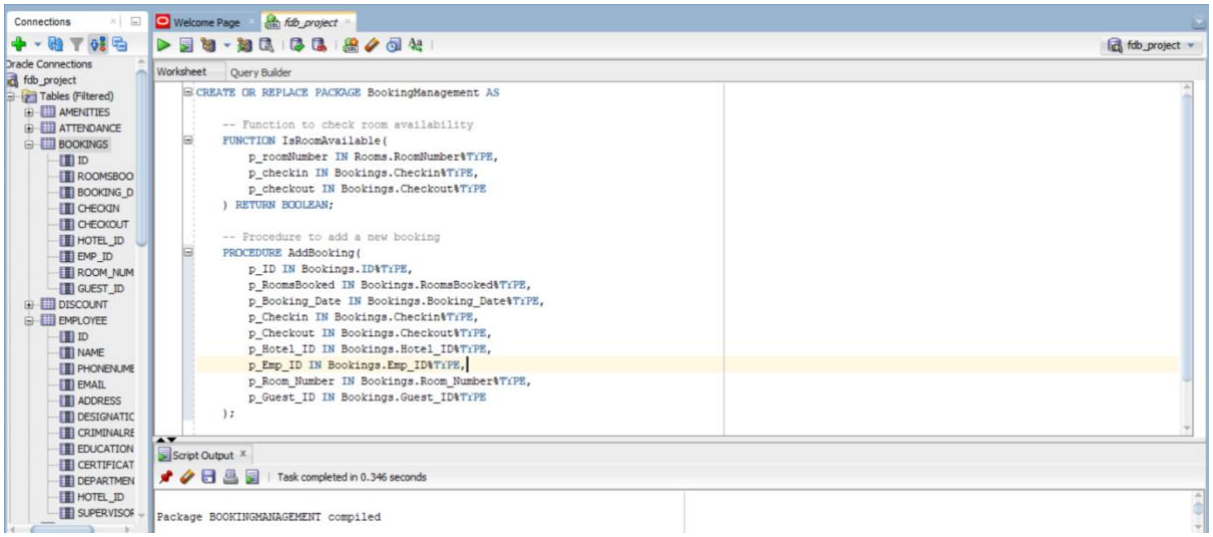

Welcome Page
fdb_project
Worksheet
Query Builder
--Display the total number of packages after a new package is being added
set serveroutput on;
select count(*) from packages;
create or replace trigger total_packages
after insert on packages
declare
total_packages number;
begin
select count(*) into total_packages
from packages;
dbms_output.put_line('The total number of packages is now ='|| total_packages);
end;
/
Script Output
Query Result
Task completed in 1.211 seconds
>>Query Run In:Query Result
Trigger TOTAL_PACKAGES compiled


Script Output
Query Result
SQL
All Rows Fetched: 1 in 0.108 seconds
COUNT(*)
1
10

Package:

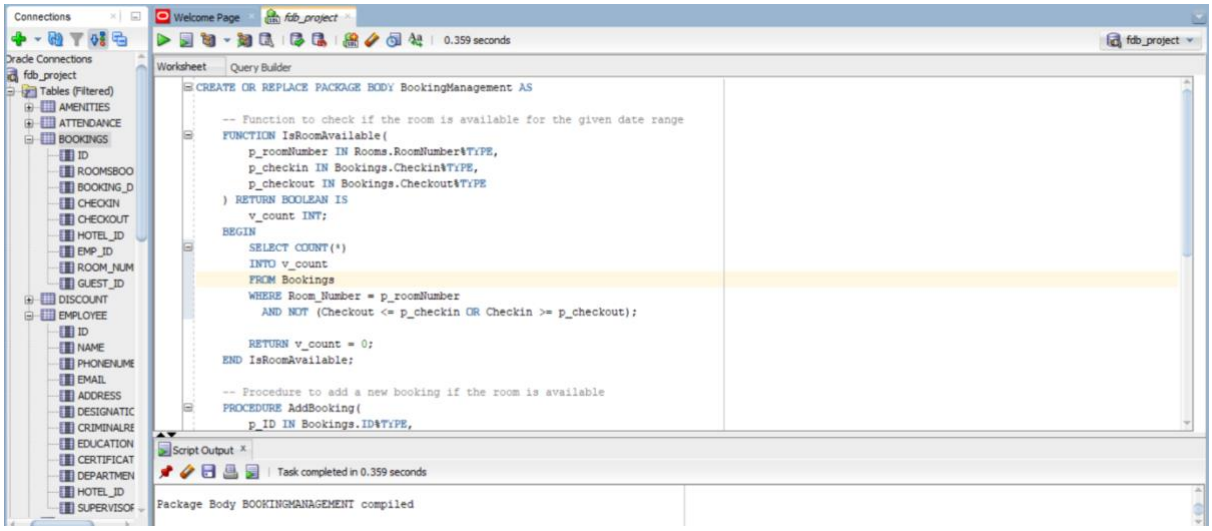
BookingManagement Package with multiple functions and procedures

```
CREATE OR REPLACE PACKAGE BookingManagement AS

    -- Function to check room availability
    FUNCTION IsRoomAvailable(
p_roomNumber IN Rooms.RoomNumber%TYPE,
p_checkin IN Bookings.Checkin%TYPE,
p_checkout IN Bookings.Checkout%TYPE
    ) RETURN BOOLEAN;

    -- Procedure to add a new booking
    PROCEDURE AddBooking(
p_ID IN Bookings.ID%TYPE,
p_RoomsBooked IN Bookings.RoomsBooked%TYPE,
p_Booking_Date IN Bookings.Booking_Date%TYPE,
p_Checkin IN Bookings.Checkin%TYPE,
p_Checkout IN Bookings.Checkout%TYPE,
p_Hotel_ID IN Bookings.Hotel_ID%TYPE,
p_Emp_ID IN Bookings.Emp_ID%TYPE,
p_Room_Number IN Bookings.Room_Number%TYPE,
p_Guest_ID IN Bookings.Guest_ID%TYPE
    );

END BookingManagement;
/
```



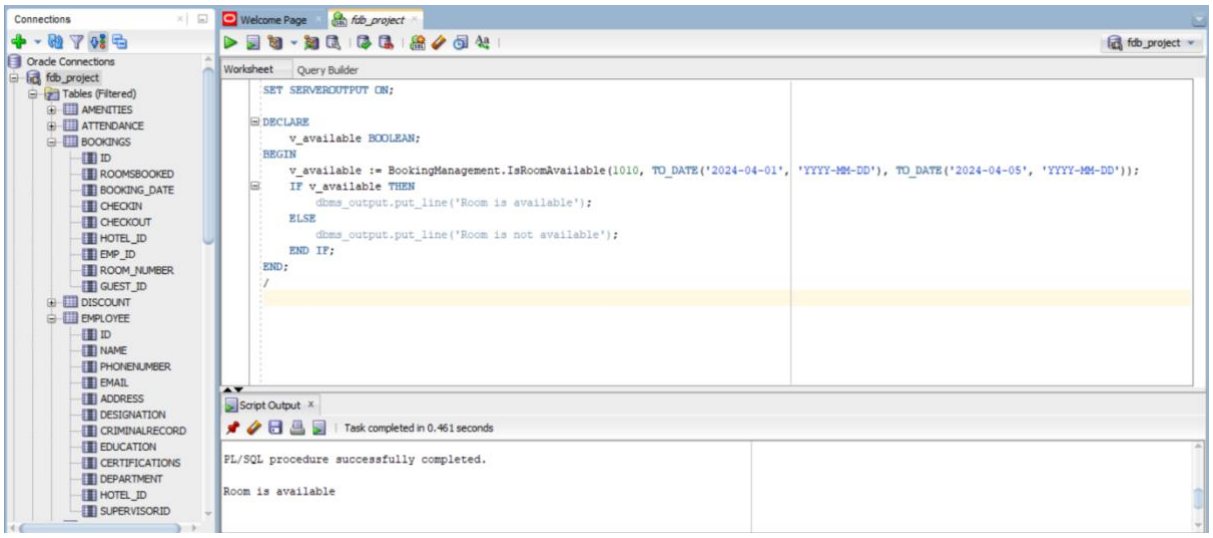

```
CREATE OR REPLACE PACKAGE BODY BookingManagement AS

    -- Function to check if the room is available for the given date range
    FUNCTION IsRoomAvailable(
p_roomNumber IN Rooms.RoomNumber%TYPE,
p_checkin IN Bookings.Checkin%TYPE,
p_checkout IN Bookings.Checkout%TYPE
    ) RETURN BOOLEAN IS
v_count INT;
    BEGIN
        SELECT COUNT(*)
        INTO v_count
        FROM Bookings
        WHERE Room_Number = p_roomNumber
          AND NOT (Checkout <= p_checkin OR Checkin>= p_checkout);

        RETURN v_count = 0;
    END IsRoomAvailable;

    -- Procedure to add a new booking if the room is available
    PROCEDURE AddBooking(
p_ID IN Bookings.ID%TYPE,
p_RoomsBooked IN Bookings.RoomsBooked%TYPE,
```

```
p_Booking_Date IN Bookings.Booking_Date%TYPE,

p_Checkin IN Bookings.Checkin%TYPE,

p_Checkout IN Bookings.Checkout%TYPE,

p_Hotel_ID IN Bookings.Hotel_ID%TYPE,

p_Emp_ID IN Bookings.Emp_ID%TYPE,

p_Room_Number IN Bookings.Room_Number%TYPE,

p_Guest_ID IN Bookings.Guest_ID%TYPE

    ) IS

    BEGIN

        IF IsRoomAvailable(p_Room_Number, p_Checkin, p_Checkout) THEN

            INSERT INTO Bookings(ID, RoomsBooked, Booking_Date, Checkin, Checkout, Hotel_ID,
Emp_ID, Room_Number, Guest_ID)

            VALUES (p_ID, p_RoomsBooked, p_Booking_Date, p_Checkin, p_Checkout, p_Hotel_ID,
p_Emp_ID, p_Room_Number, p_Guest_ID);

        ELSE

            RAISE_APPLICATION_ERROR(-20001, 'Room not available for the selected dates.');

        END IF;

    END AddBooking;


END BookingManagement;

/
```



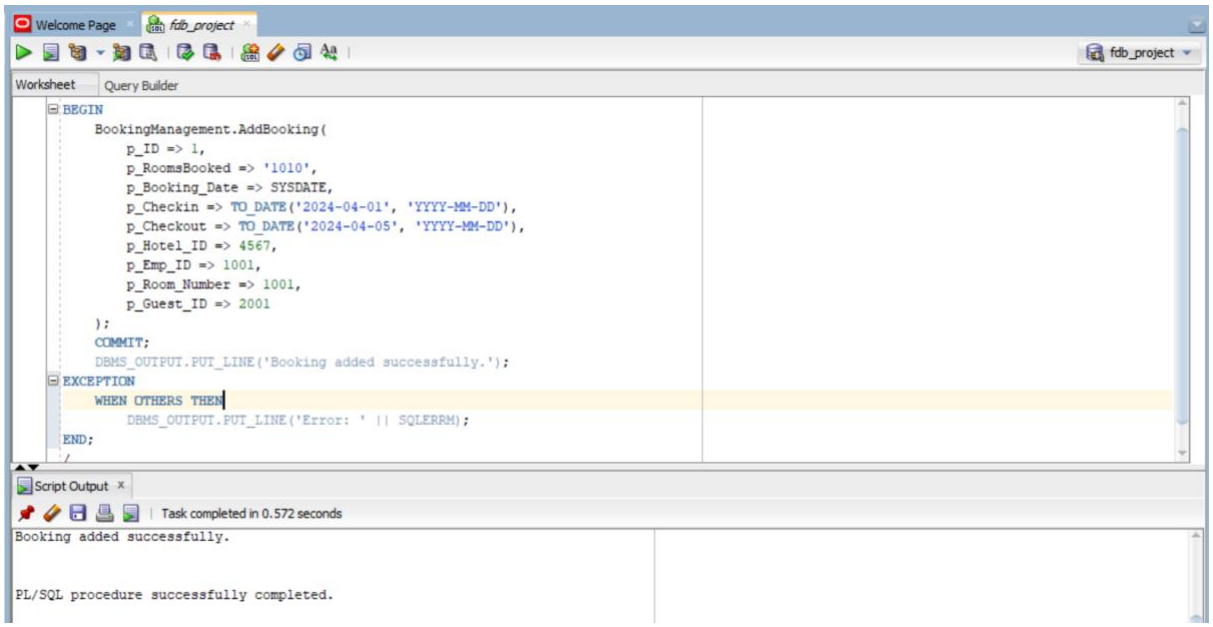

```
DECLARE
```

```
v_available BOOLEAN;
BEGIN
v_available := BookingManagement.IsRoomAvailable(1010, TO_DATE('2024-04-01', 'YYYY-MM-DD'), TO_DATE('2024-04-05', 'YYYY-MM-DD'));
   IF v_available THEN
      DBMS_OUTPUT.PUT_LINE('Room is available');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Room is not available');
   END IF;
END;
/
```




```
SET SERVEROUTPUT ON;
BEGIN
BookingManagement.AddBooking(
p_ID => 1,
p_RoomsBooked => '1010',
p_Booking_Date => SYSDATE,
p_Checkin => TO_DATE('2024-04-01', 'YYYY-MM-DD'),
p_Checkout => TO_DATE('2024-04-05', 'YYYY-MM-DD'),
p_Hotel_ID => 4567,
p_Emp_ID => 1001,
p_Room_Number => 1001,
p_Guest_ID => 2001
```

```
    );
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Booking added successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



INDIVIDUAL CONTRIBUTIONS:

For this week's project submission, I worked on the stored functions and stored procedures. I assisted in writing queries and their corresponding PL/SQL code and made sure we got the correct outputs for the QueHotel database.

I also assisted with the documentation this week, formatting the document wherever necessary. I also organized timely and productive meetings to discuss the project and update the project wherever necessary.