CSD 4203 – Database Programming

Term : 2023S

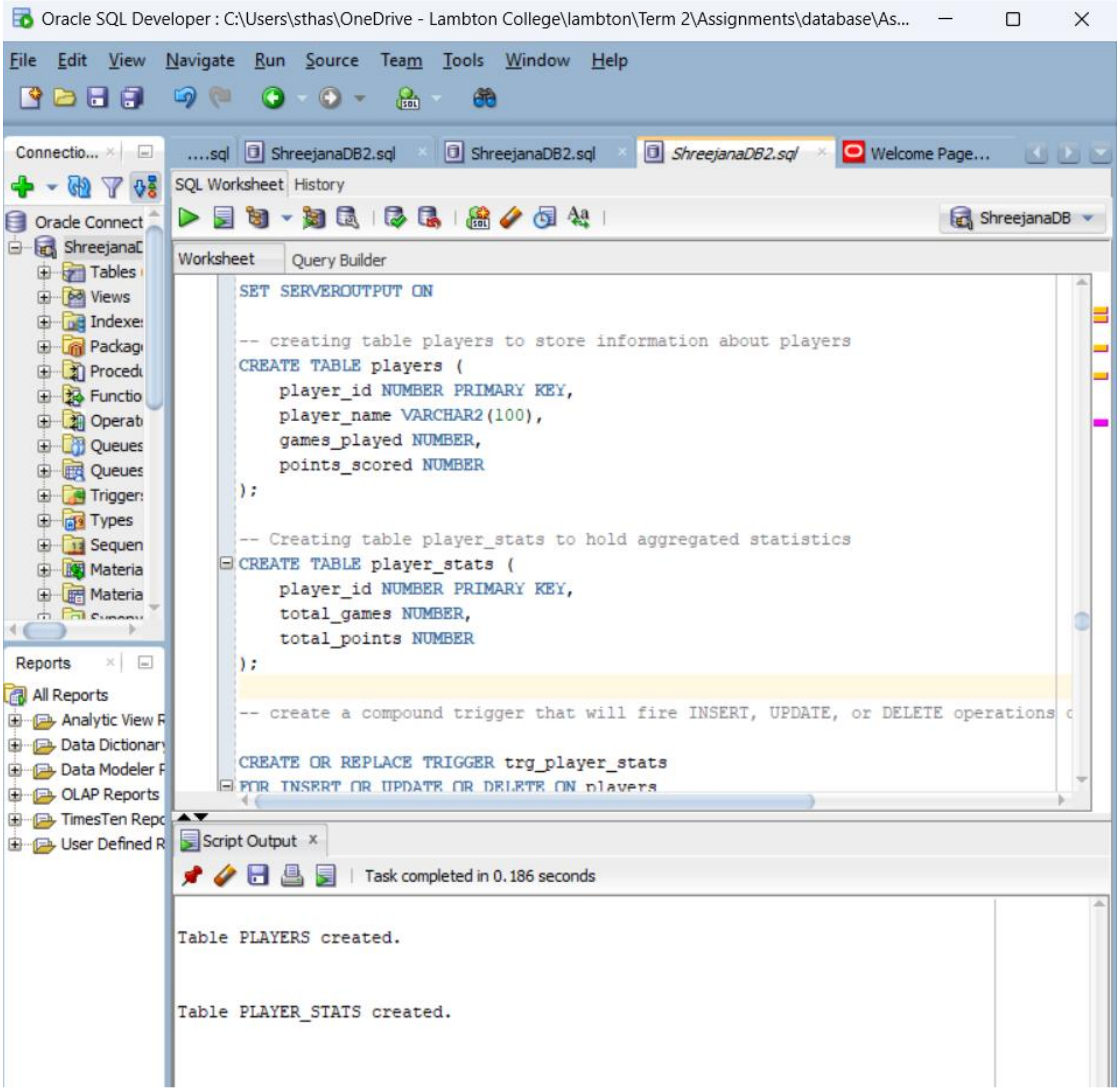
Student ID : C0930321

Student Name : Shreejana Shrestha

Assignment # 8

--

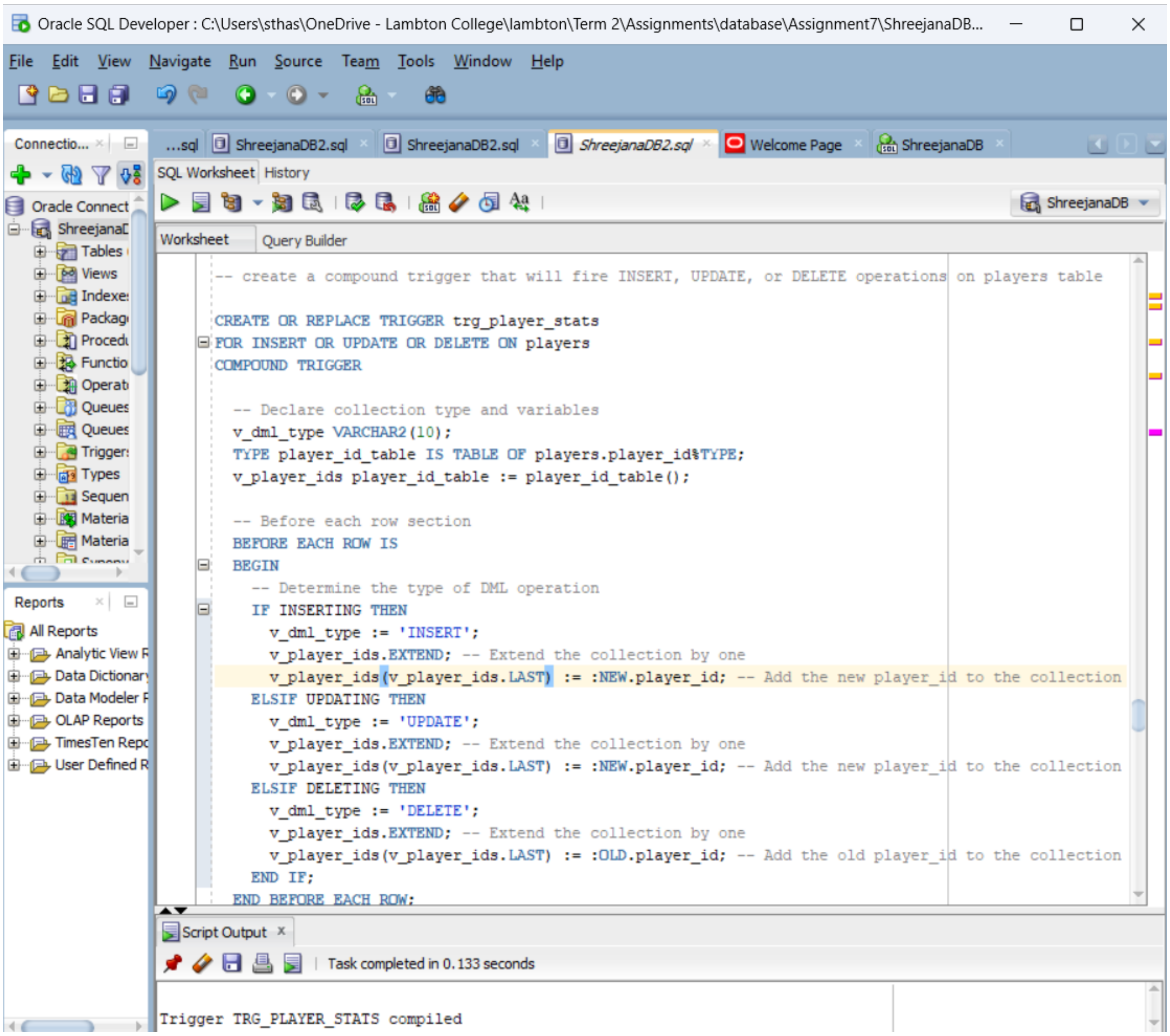
# Table creation



```
SET SERVEROUTPUT ON

-- creating table players to store information about players
CREATE TABLE players (
    player_id NUMBER PRIMARY KEY,
    player_name VARCHAR2(100),
    games_played NUMBER,
    points_scored NUMBER
);

-- Creating table player_stats to hold aggregated statistics
CREATE TABLE player_stats (
    player_id NUMBER PRIMARY KEY,
    total_games NUMBER,
    total_points NUMBER
);

-- create a compound trigger that will fire INSERT, UPDATE, or DELETE operations c

CREATE OR REPLACE TRIGGER trg_player_stats
FOR INSERT OR UPDATE OR DELETE ON players
```

```
Table PLAYERS created.


Table PLAYER_STATS created.
```

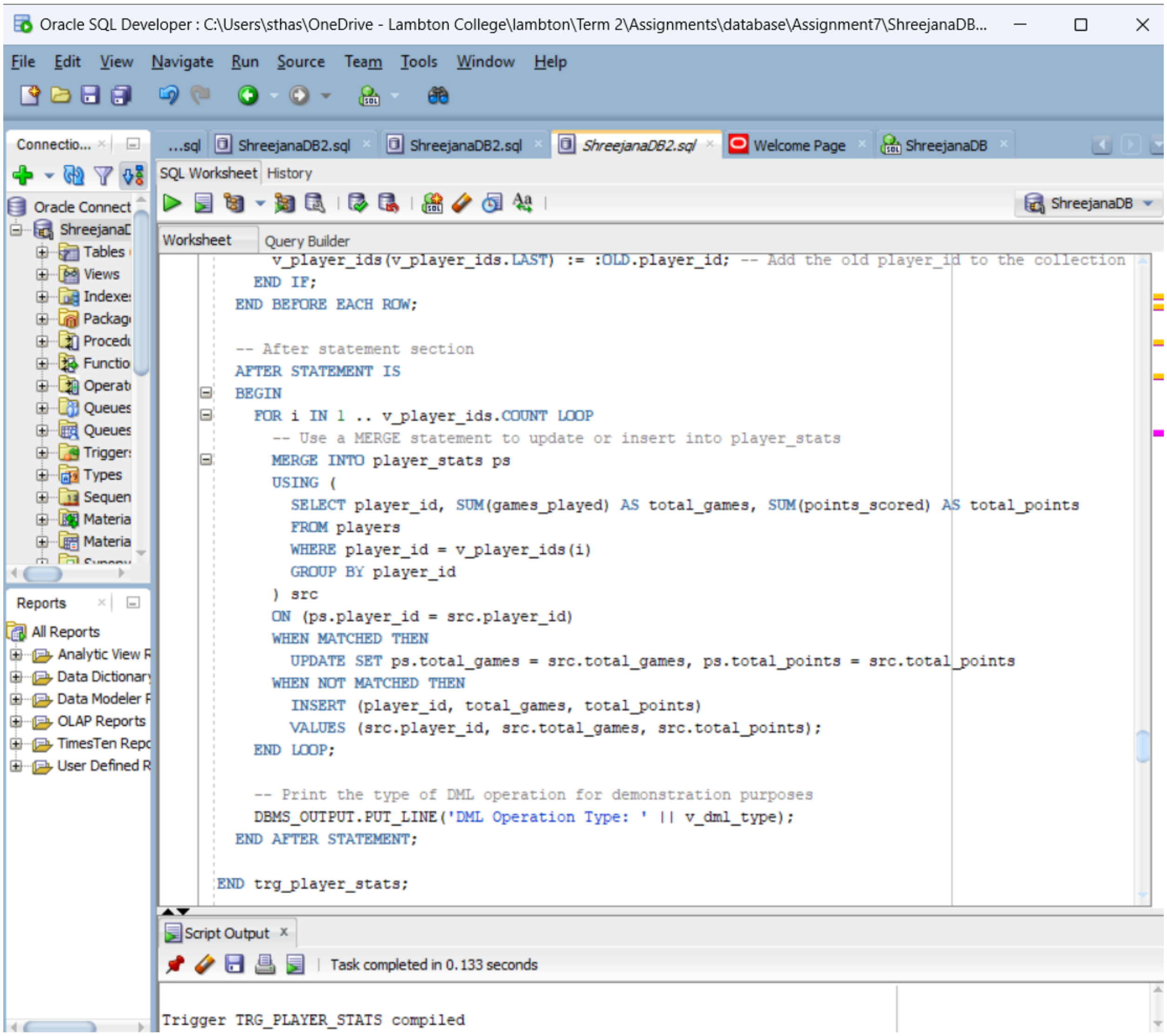
Compound Trigger creation



```
-- create a compound trigger that will fire INSERT, UPDATE, or DELETE operations on players table

CREATE OR REPLACE TRIGGER trg_player_stats
FOR INSERT OR UPDATE OR DELETE ON players
COMPOUND TRIGGER

  -- Declare collection type and variables
  v_dml_type VARCHAR2(10);
  TYPE player_id_table IS TABLE OF players.player_id%TYPE;
  v_player_ids player_id_table := player_id_table();

  -- Before each row section
  BEFORE EACH ROW IS
  BEGIN
    -- Determine the type of DML operation
    IF INSERTING THEN
      v_dml_type := 'INSERT';
      v_player_ids.EXTEND; -- Extend the collection by one
      v_player_ids(v_player_ids.LAST) := :NEW.player_id; -- Add the new player_id to the collection
    ELSIF UPDATING THEN
      v_dml_type := 'UPDATE';
      v_player_ids.EXTEND; -- Extend the collection by one
      v_player_ids(v_player_ids.LAST) := :NEW.player_id; -- Add the new player_id to the collection
    ELSIF DELETING THEN
      v_dml_type := 'DELETE';
      v_player_ids.EXTEND; -- Extend the collection by one
      v_player_ids(v_player_ids.LAST) := :OLD.player_id; -- Add the old player_id to the collection
    END IF;
  END BEFORE EACH ROW;
```

Task completed in 0.133 seconds

```
Trigger TRG_PLAYER_STATS compiled
```

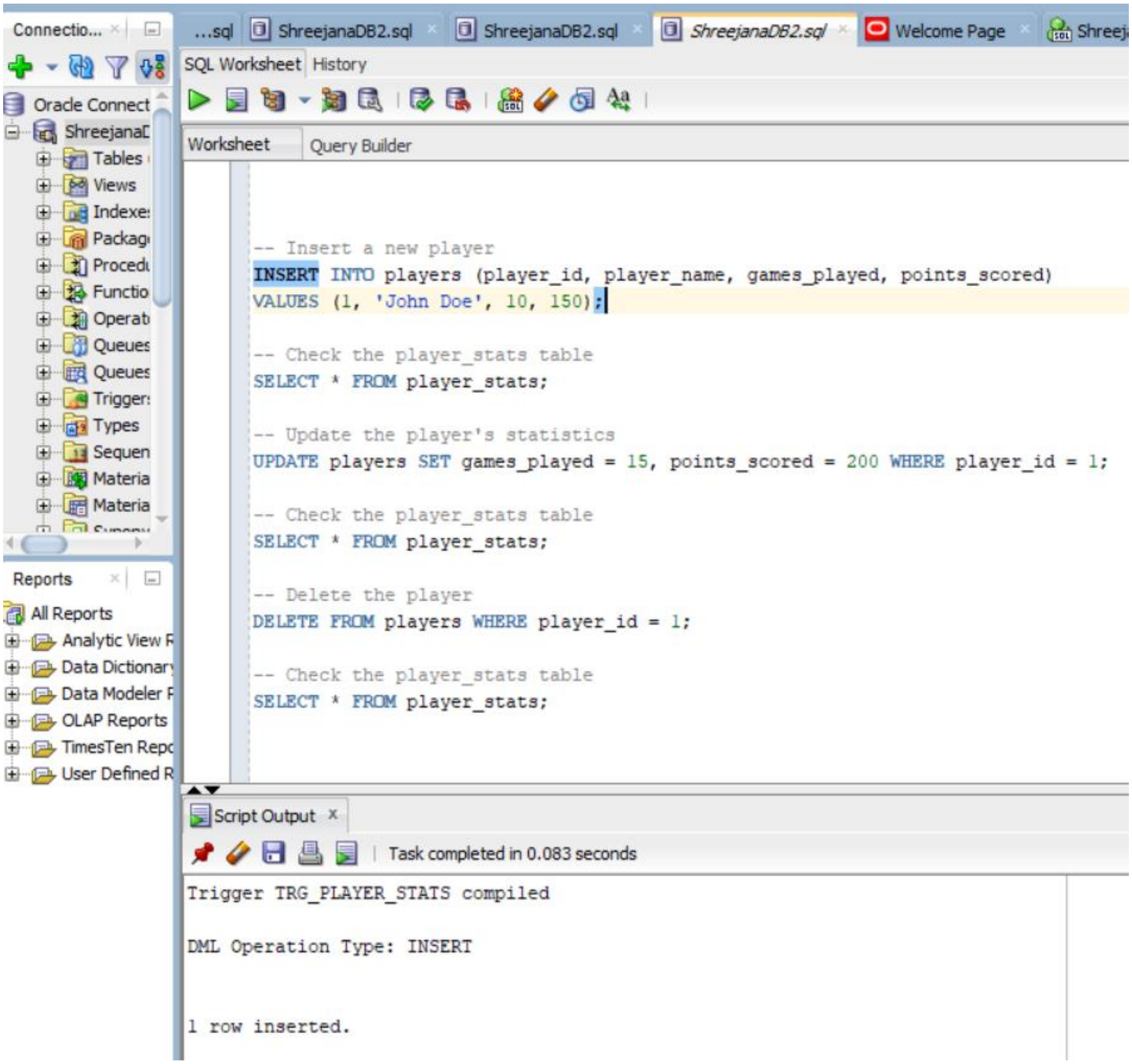


```
      v_player_ids(v_player_ids.LAST) := :OLD.player_id; -- Add the old player_id to the collection
    END IF;
  END BEFORE EACH ROW;

  -- After statement section
  AFTER STATEMENT IS
  BEGIN
    FOR i IN 1 .. v_player_ids.COUNT LOOP
      -- Use a MERGE statement to update or insert into player_stats
      MERGE INTO player_stats ps
      USING (
        SELECT player_id, SUM(games_played) AS total_games, SUM(points_scored) AS total_points
        FROM players
        WHERE player_id = v_player_ids(i)
        GROUP BY player_id
      ) src
      ON (ps.player_id = src.player_id)
      WHEN MATCHED THEN
        UPDATE SET ps.total_games = src.total_games, ps.total_points = src.total_points
      WHEN NOT MATCHED THEN
        INSERT (player_id, total_games, total_points)
        VALUES (src.player_id, src.total_games, src.total_points);
    END LOOP;

    -- Print the type of DML operation for demonstration purposes
    DBMS_OUTPUT.PUT_LINE('DML Operation Type: ' || v_dml_type);
  END AFTER STATEMENT;

END trg_player_stats;
```

```
Trigger TRG_PLAYER_STATS compiled
```

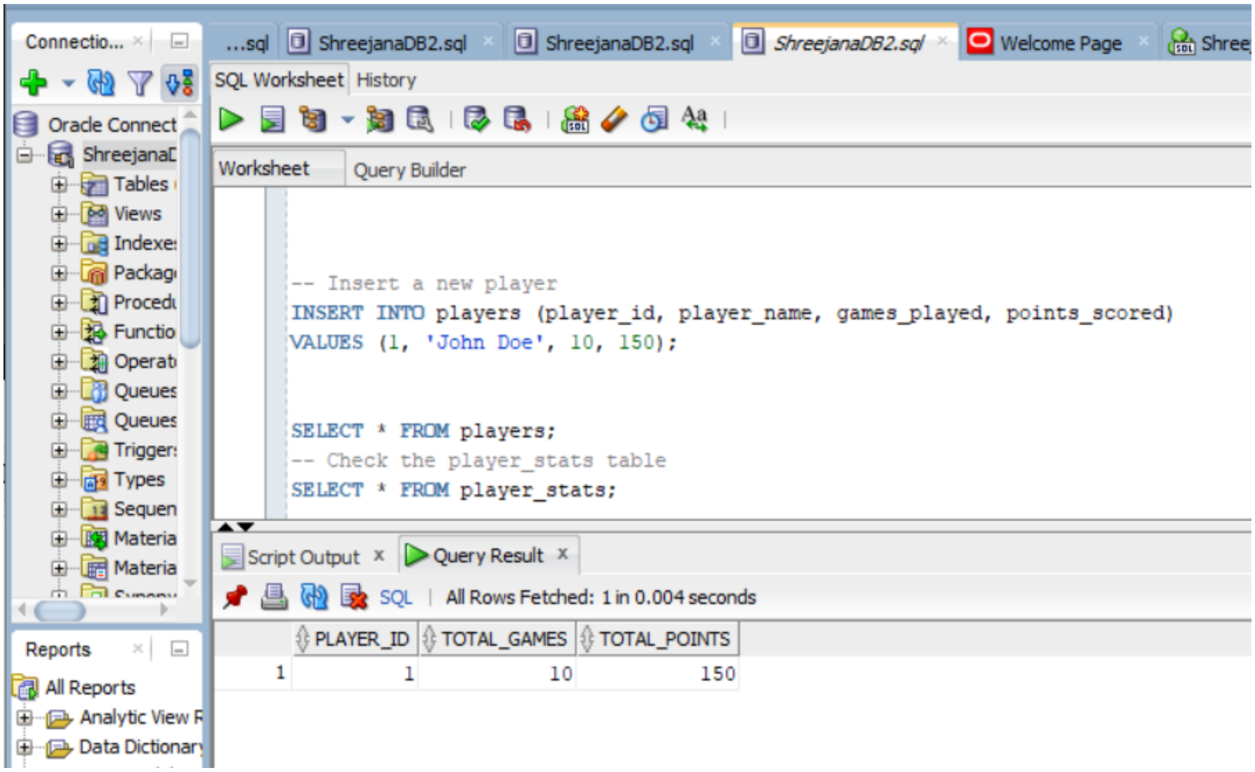
# Implementing the trigger through insert, update or delete

# INSERT



```
-- Insert a new player
INSERT INTO players (player_id, player_name, games_played, points_scored)
VALUES (1, 'John Doe', 10, 150);

-- Check the player_stats table
SELECT * FROM player_stats;

-- Update the player's statistics
UPDATE players SET games_played = 15, points_scored = 200 WHERE player_id = 1;

-- Check the player_stats table
SELECT * FROM player_stats;

-- Delete the player
DELETE FROM players WHERE player_id = 1;

-- Check the player_stats table
SELECT * FROM player_stats;
```

```
Trigger TRG_PLAYER_STATS compiled

DML Operation Type: INSERT


1 row inserted.
```

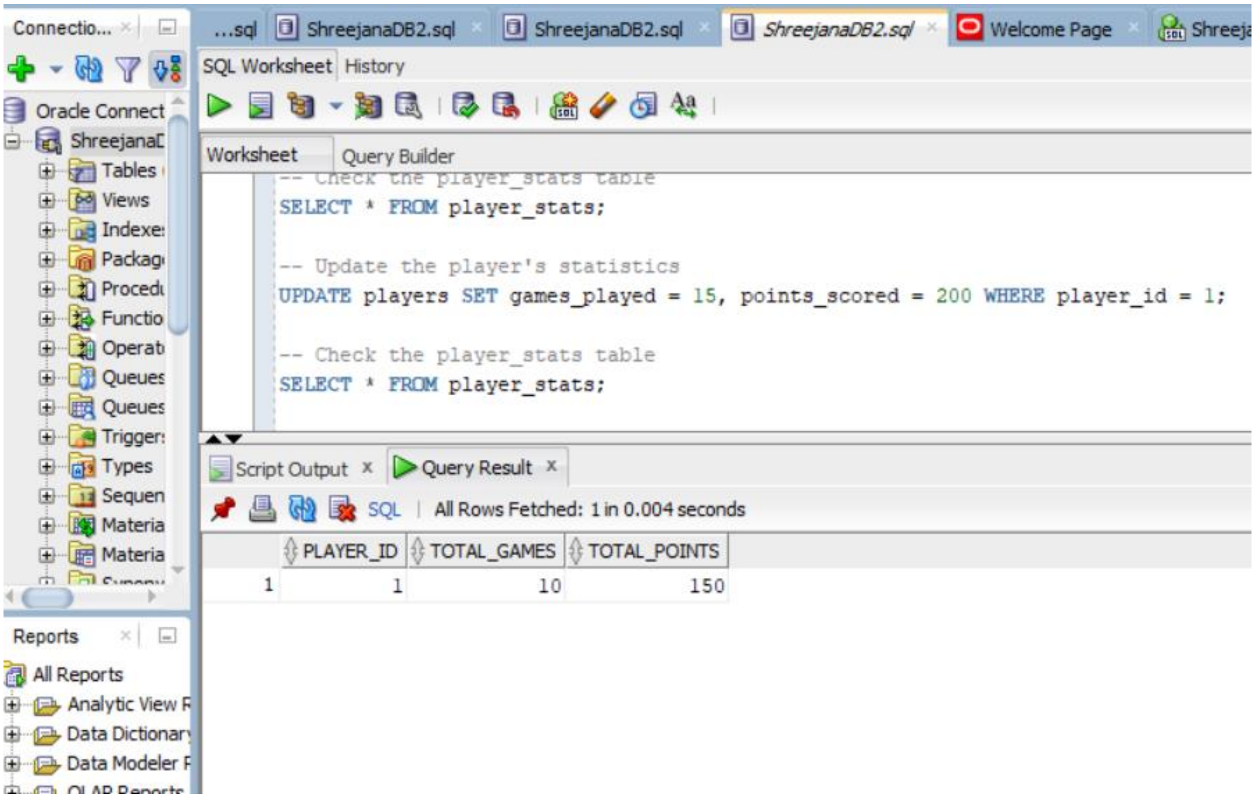


```
-- Insert a new player
INSERT INTO players (player_id, player_name, games_played, points_scored)
VALUES (1, 'John Doe', 10, 150);


SELECT * FROM players;
-- Check the player_stats table
SELECT * FROM player_stats;
```

All Rows Fetched: 1 in 0.004 seconds

	PLAYER_ID	TOTAL_GAMES	TOTAL_POINTS
1	1	10	150

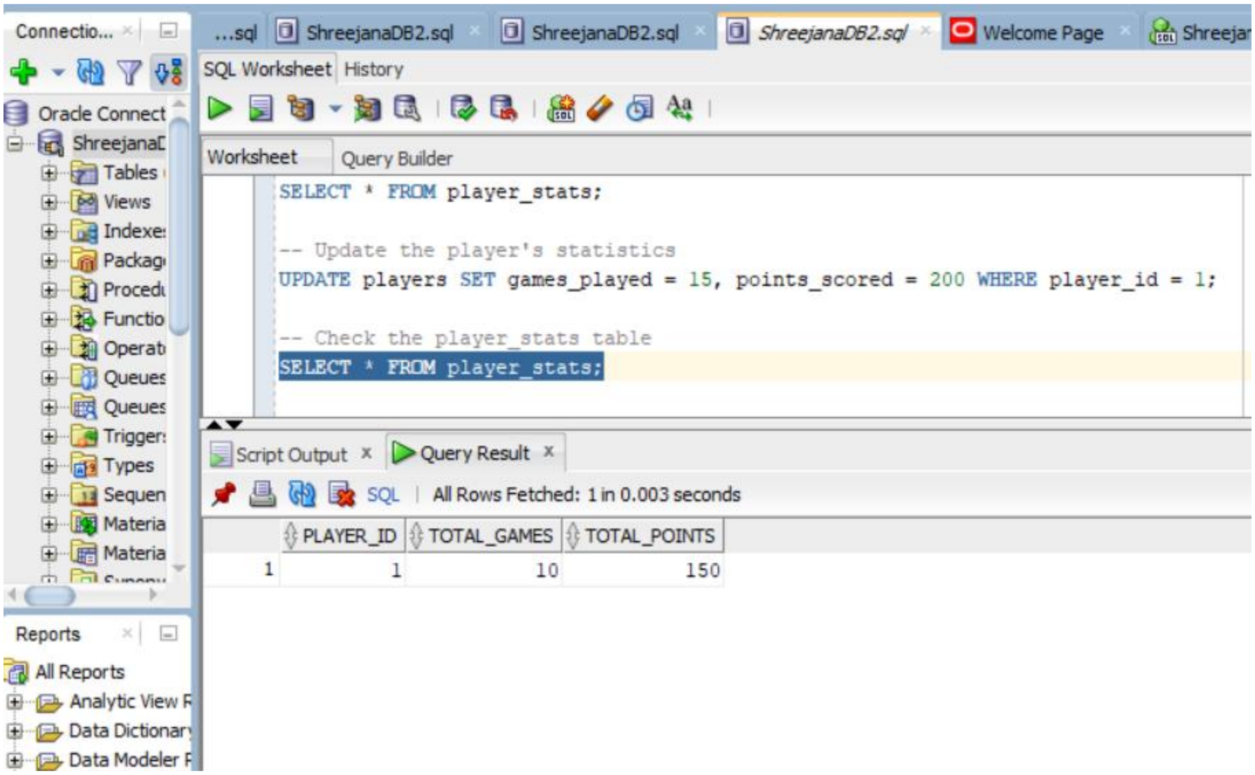
# UPDATE



```
-- Check the player_stats table
SELECT * FROM player_stats;

-- Update the player's statistics
UPDATE players SET games_played = 15, points_scored = 200 WHERE player_id = 1;

-- Check the player_stats table
SELECT * FROM player_stats;
```

All Rows Fetched: 1 in 0.004 seconds

	PLAYER_ID	TOTAL_GAMES	TOTAL_POINTS
1	1	10	150

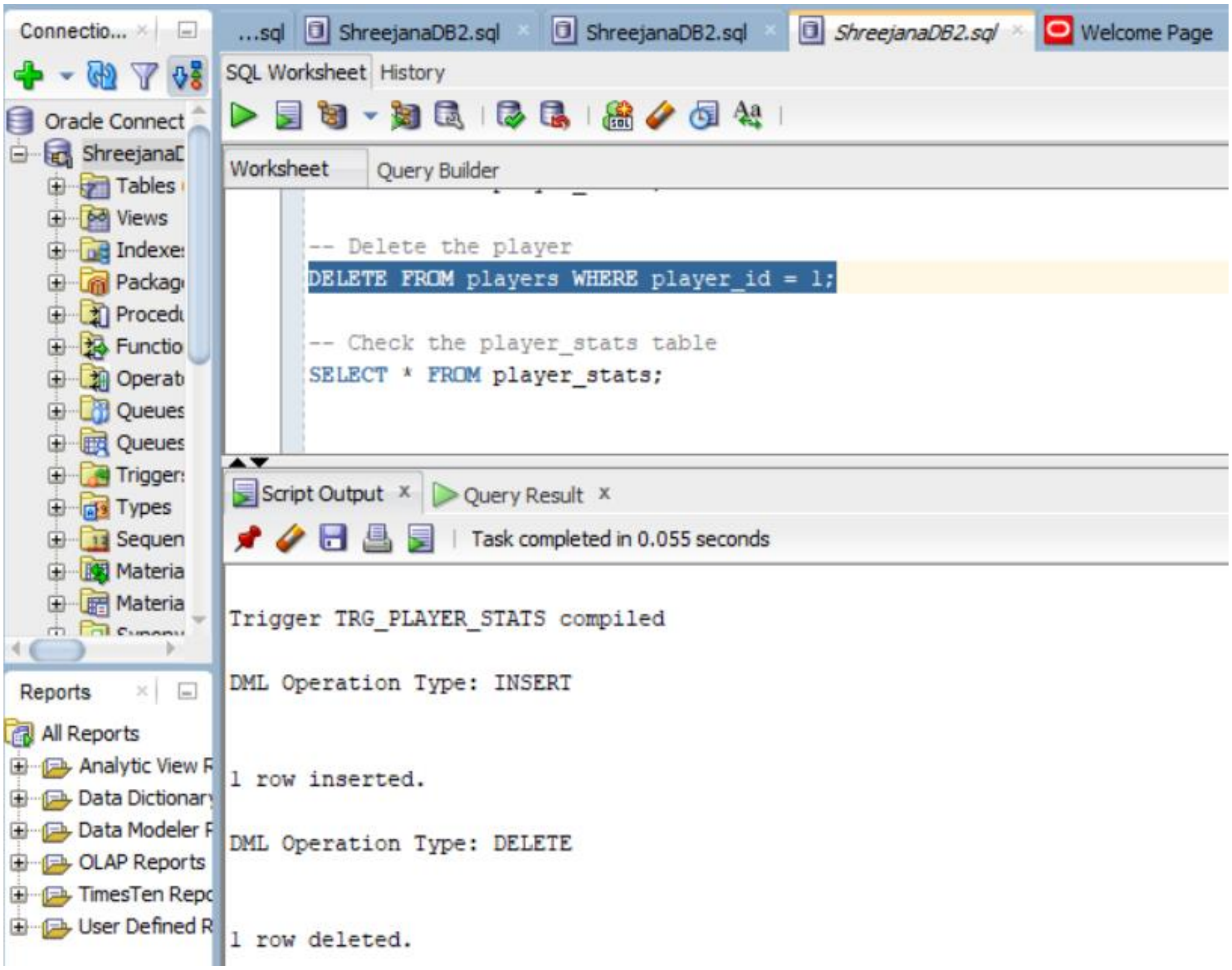


```
SELECT * FROM player_stats;

-- Update the player's statistics
UPDATE players SET games_played = 15, points_scored = 200 WHERE player_id = 1;

-- Check the player_stats table
SELECT * FROM player_stats;
```

All Rows Fetched: 1 in 0.003 seconds

	PLAYER_ID	TOTAL_GAMES	TOTAL_POINTS
1	1	10	150

# DELETE



```
-- Delete the player
DELETE FROM players WHERE player_id = 1;

-- Check the player_stats table
SELECT * FROM player_stats;
```

Task completed in 0.055 seconds

```
Trigger TRG_PLAYER_STATS compiled

DML Operation Type: INSERT


1 row inserted.

DML Operation Type: DELETE


1 row deleted.
```

# A compound trigger is a type of trigger in Oracle PL/SQL that groups multiple timing points (e.g., BEFORE EACH ROW, AFTER EACH ROW, BEFORE STATEMENT, AFTER STATEMENT) into a single trigger. This allows you to handle complex DML operations on a table more efficiently.

The **trg_player_stats** compound trigger updates the **player_stats** table based on changes to the players table. It collects affected player IDs during INSERT, UPDATE, or DELETE operations and processes them all at once after the DML operations are complete. This ensures that the **player_stats** table is accurately updated with the new or changed data, and avoids common issues like mutating table errors.