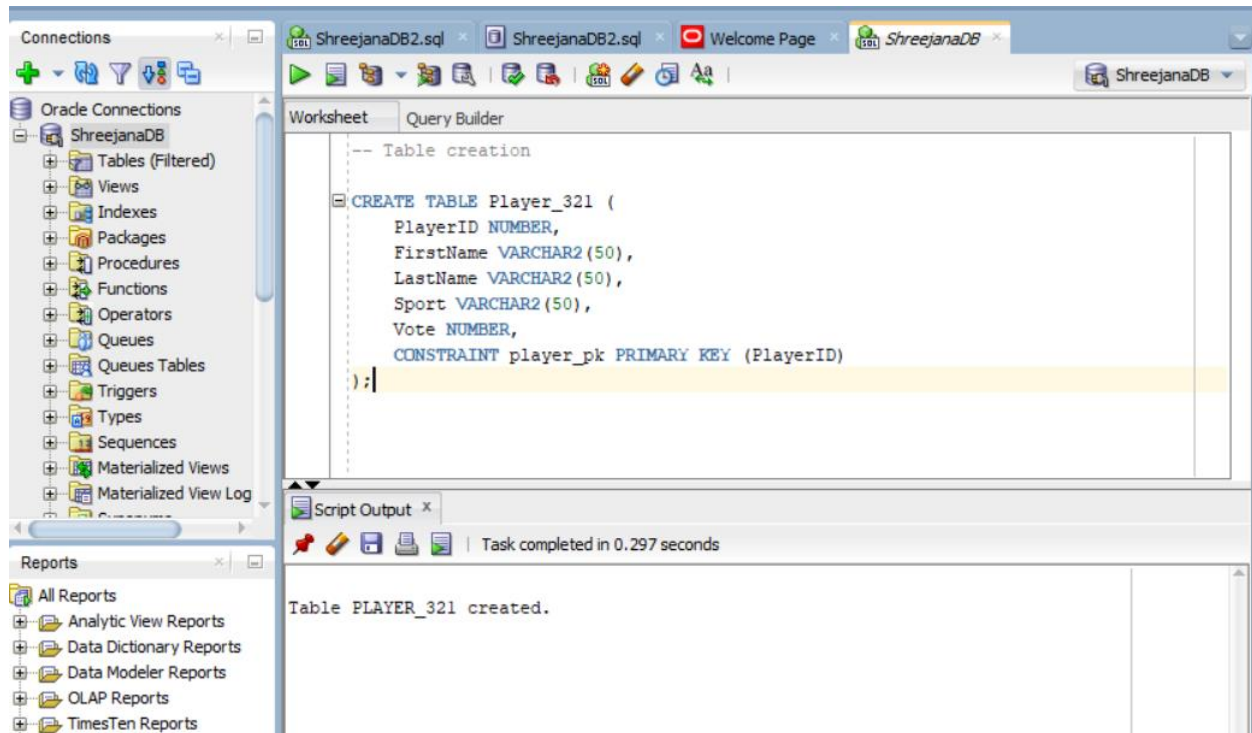


Table creation



Inserting the data to the table

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a tree view of the 'ShreejanaDB' database, including Tables, Views, Indexes, Packages, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, Materialized Views, and Materialized View Log. The 'Reports' pane is also visible, showing a list of report types. The main workspace is divided into two panes: 'Worksheet' and 'Query Builder'. The 'Worksheet' pane contains an SQL query to insert data into the 'Player_321' table. The 'Script Output' pane at the bottom shows the execution results, indicating that four rows were successfully inserted.

```
-- Inserting data to the table above

INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (1, 'Michael', 'Jordan', 'Basketball', 98);
INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (2, 'Serena', 'Williams', 'Tennis', 95);
INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (3, 'Lionel', 'Messi', 'Soccer', 99);
INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (4, 'Usain', 'Bolt', 'Athletics', 97);
```

Script Output x

Task completed in 0.124 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Connections

- Oracle Connections
 - ShreejanaDB
 - Tables (Filtered)
 - Views
 - Indexes
 - Packages
 - Procedures
 - Functions
 - Operators
 - Queues
 - Queues Tables
 - Triggers
 - Types
 - Sequences
 - Materialized Views
 - Materialized View Log

Reports

- All Reports
- Analytic View Reports
- Data Dictionary Reports
- Data Modeler Reports
- OLAP Reports
- TimesTen Reports

Worksheet

```

INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (1, 'Michael', 'Jordan', 'Basketball', 98);
INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (2, 'Serena', 'Williams', 'Tennis', 95);
INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (3, 'Lionel', 'Messi', 'Soccer', 99);
INSERT INTO Player_321 (PlayerID, FirstName, LastName, Sport, Vote)
VALUES (4, 'Usain', 'Bolt', 'Athletics', 97);

select * from player_321;
  
```

Script Output x Query Result x

SQL | All Rows Fetched: 4 in 0.066 seconds

PLAYERID	FIRSTNAME	LASTNAME	SPORT	VOTE
1	Michael	Jordan	Basketball	98
2	Serena	Williams	Tennis	95
3	Lionel	Messi	Soccer	99
4	Usain	Bolt	Athletics	97

Question 1: Implicit cursor

Connections

- Oracle Connection
 - ShreejanaDB
 - Tables (Fi
 - Views
 - Indexes
 - Packages
 - Procedure
 - Functions
 - Operators
 - Queues
 - Queues T2
 - Triggers
 - Types
 - Sequences
 - Materialize
 - Materialize

Worksheet

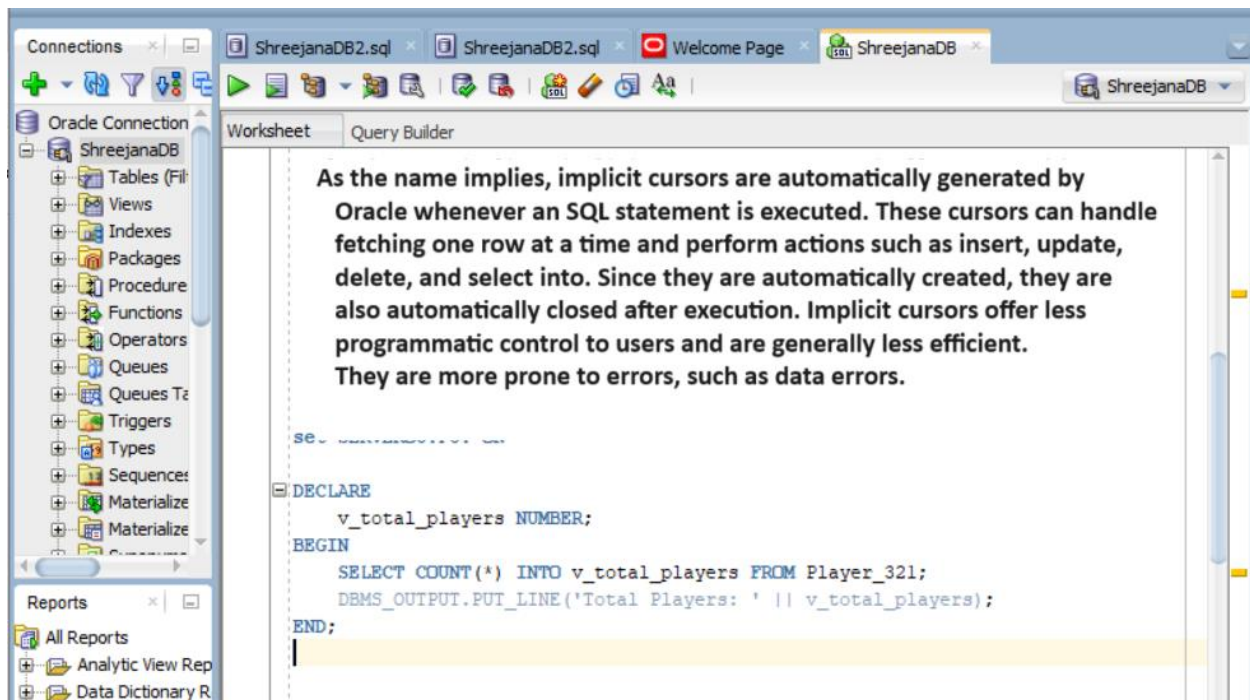
```

-- Implicit cursor

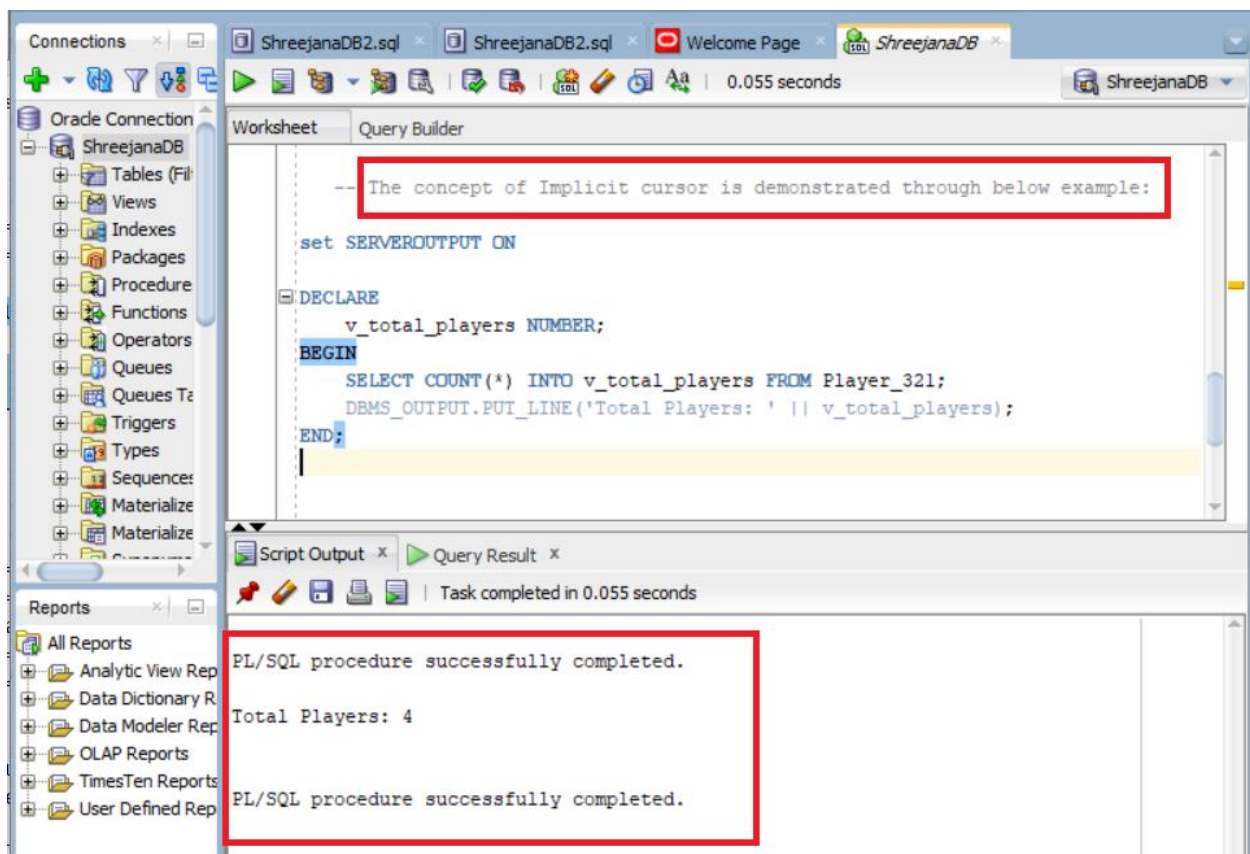
--
--
/* Cursor is a pointer which is pointing to the memory area. Cursors can be
broadly divided into implicit and explicit cursors.

As the name implies, implicit cursors are automatically generated by
Oracle whenever an SQL statement is executed. These cursors can handle
fetching one row at a time and perform actions such as insert, update,
delete, and select into. Since they are automatically created, they are
also automatically closed after execution. Implicit cursors offer less
programmatic control to users and are generally less efficient.
They are more prone to errors, such as data errors. */

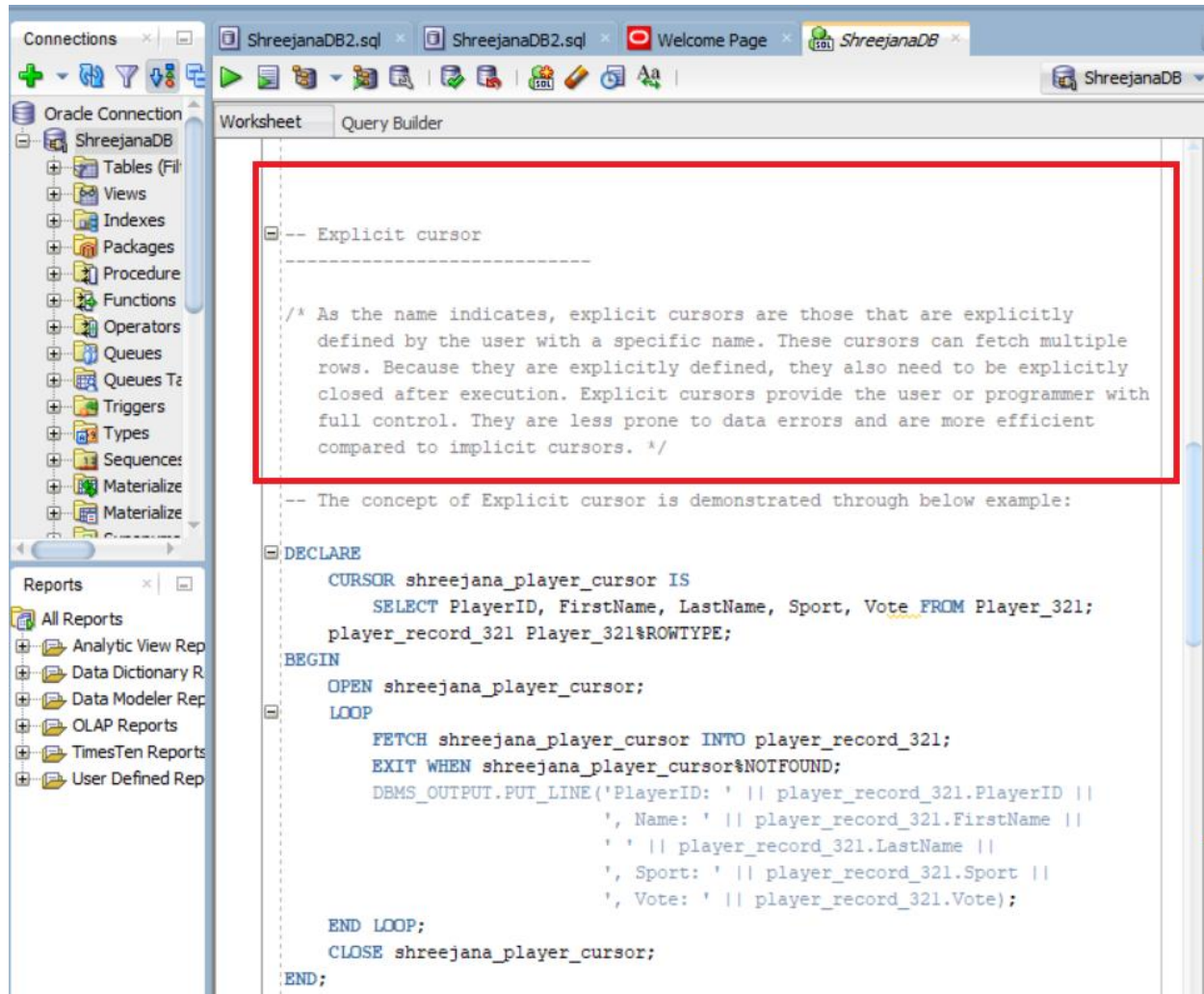
-- The concept of Implicit cursor is demonstrated through below example:
  
```



code and output



Question 2: Explicit Cursor



The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows a connection to 'ShreejanaDB'. The main workspace is divided into 'Worksheet' and 'Query Builder' tabs. A red rectangle highlights a comment block in the SQL editor. Below this, a PL/SQL block demonstrates the use of an explicit cursor.

```
-- Explicit cursor
-----

/* As the name indicates, explicit cursors are those that are explicitly
defined by the user with a specific name. These cursors can fetch multiple
rows. Because they are explicitly defined, they also need to be explicitly
closed after execution. Explicit cursors provide the user or programmer with
full control. They are less prone to data errors and are more efficient
compared to implicit cursors. */

-- The concept of Explicit cursor is demonstrated through below example:

DECLARE
    CURSOR shreejana_player_cursor IS
        SELECT PlayerID, FirstName, LastName, Sport, Vote FROM Player_321;
    player_record_321 Player_321%ROWTYPE;
BEGIN
    OPEN shreejana_player_cursor;
    LOOP
        FETCH shreejana_player_cursor INTO player_record_321;
        EXIT WHEN shreejana_player_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('PlayerID: ' || player_record_321.PlayerID ||
                               ', Name: ' || player_record_321.FirstName ||
                               ' ' || player_record_321.LastName ||
                               ', Sport: ' || player_record_321.Sport ||
                               ', Vote: ' || player_record_321.Vote);
    END LOOP;
    CLOSE shreejana_player_cursor;
END;
```

The screenshot shows the Oracle SQL Developer interface. On the left is the 'Connections' pane with a tree view of the 'ShreejanaDB' schema, including Tables, Views, Indexes, Packages, Procedure, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, Materialized Views, and Materialized Views. Below this is the 'Reports' pane with a tree view of 'All Reports', including Analytic View Rep, Data Dictionary R, Data Modeler Rep, OLAP Reports, TimesTen Reports, and User Defined Rep. The main window is titled 'Worksheet' and 'Query Builder'. It contains a red-bordered box with the following text:

As the name indicates, explicit cursors are those that are explicitly defined by the user with a specific name. These cursors can fetch multiple rows. Because they are explicitly defined, they also need to be explicitly closed after execution. Explicit cursors provide the user or programmer with full control. They are less prone to data errors and are more efficient compared to implicit cursors.

Below the red box, the text reads: -- The concept of Explicit cursor is demonstrated through below example:

```
DECLARE
  CURSOR shreejana_player_cursor IS
    SELECT PlayerID, FirstName, LastName, Sport, Vote FROM Player_321;
  player_record_321 Player_321%ROWTYPE;
BEGIN
  OPEN shreejana_player_cursor;
  LOOP
    FETCH shreejana_player_cursor INTO player_record_321;
    EXIT WHEN shreejana_player_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('PlayerID: ' || player_record_321.PlayerID ||
      ', Name: ' || player_record_321.FirstName ||
      ' ' || player_record_321.LastName ||
      ', Sport: ' || player_record_321.Sport ||
      ', Vote: ' || player_record_321.Vote);
  END LOOP;
  CLOSE shreejana_player_cursor;
END;
```

code and output

The screenshot displays the Oracle SQL Developer interface. The left sidebar shows the 'Connections' pane with 'ShreejanaDB' selected, and the 'Reports' pane. The main window is split into a 'Worksheet' and a 'Query Builder' tab. The 'Worksheet' tab contains a PL/SQL procedure that demonstrates an explicit cursor. The procedure is as follows:

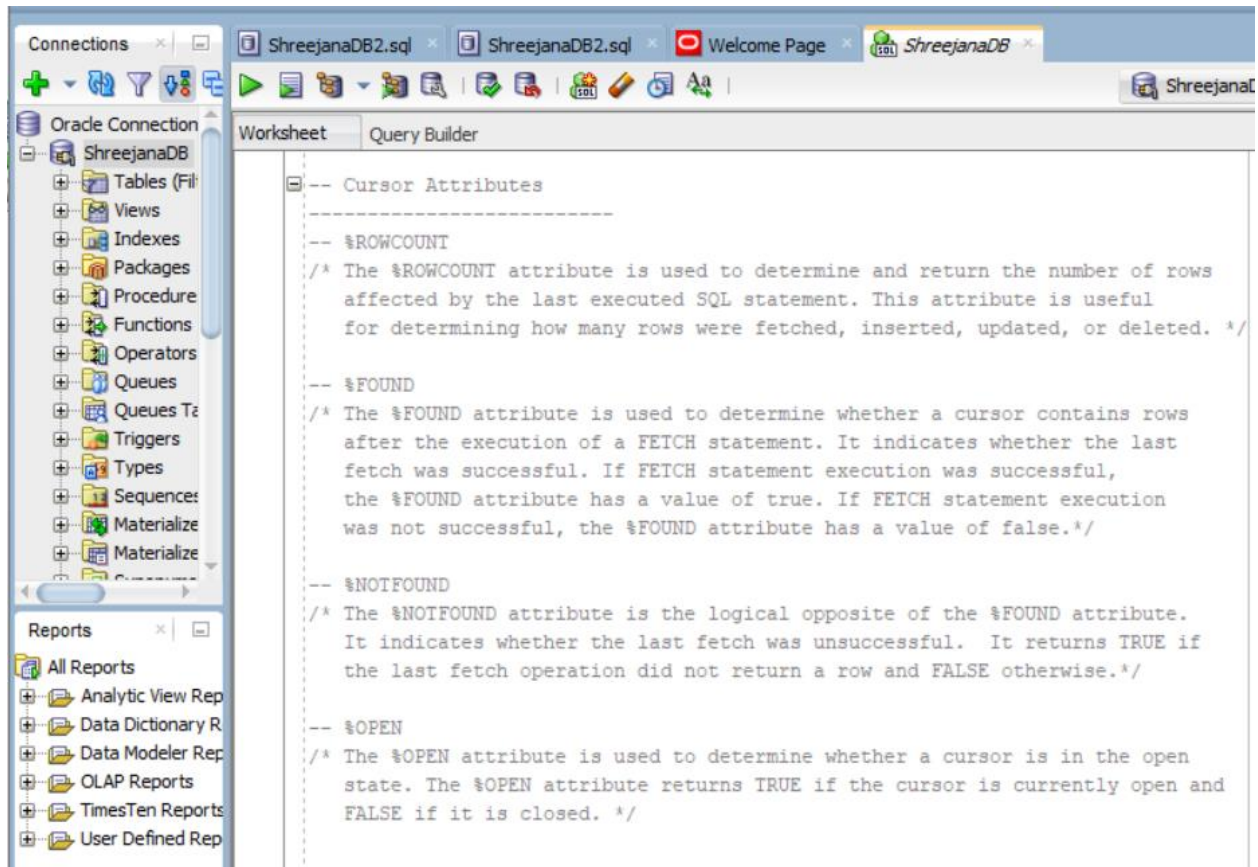
```
DECLARE
  CURSOR shreejana_player_cursor IS
    SELECT PlayerID, FirstName, LastName, Sport, Vote FROM Player_321;
  player_record_321 Player_321%ROWTYPE;
BEGIN
  OPEN shreejana_player_cursor;
  LOOP
    FETCH shreejana_player_cursor INTO player_record_321;
    EXIT WHEN shreejana_player_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('PlayerID: ' || player_record_321.PlayerID ||
      ', Name: ' || player_record_321.FirstName ||
      ' ' || player_record_321.LastName ||
      ', Sport: ' || player_record_321.Sport ||
      ', Vote: ' || player_record_321.Vote);
  END LOOP;
  CLOSE shreejana_player_cursor;
END;
```

Below the code, the 'Script Output' pane shows the results of the procedure execution, which took 0.097 seconds. The output lists four players with their IDs, names, sports, and votes.

```
PlayerID: 1, Name: Michael Jordan, Sport: Basketball, Vote: 98
PlayerID: 2, Name: Serena Williams, Sport: Tennis, Vote: 95
PlayerID: 3, Name: Lionel Messi, Sport: Soccer, Vote: 99
PlayerID: 4, Name: Usain Bolt, Sport: Athletics, Vote: 97

PL/SQL procedure successfully completed.
```

Question 3: Cursor Attributes



The screenshot displays the Oracle SQL Developer application. The top toolbar includes icons for connections, execution, and editing. The left sidebar shows the 'Oracle Connection' tree with 'ShreejanaDB' selected, and a 'Reports' section below it. The main workspace is divided into 'Worksheet' and 'Query Builder' tabs. The 'Worksheet' tab is active, showing a document titled 'Cursor Attributes' with the following content:

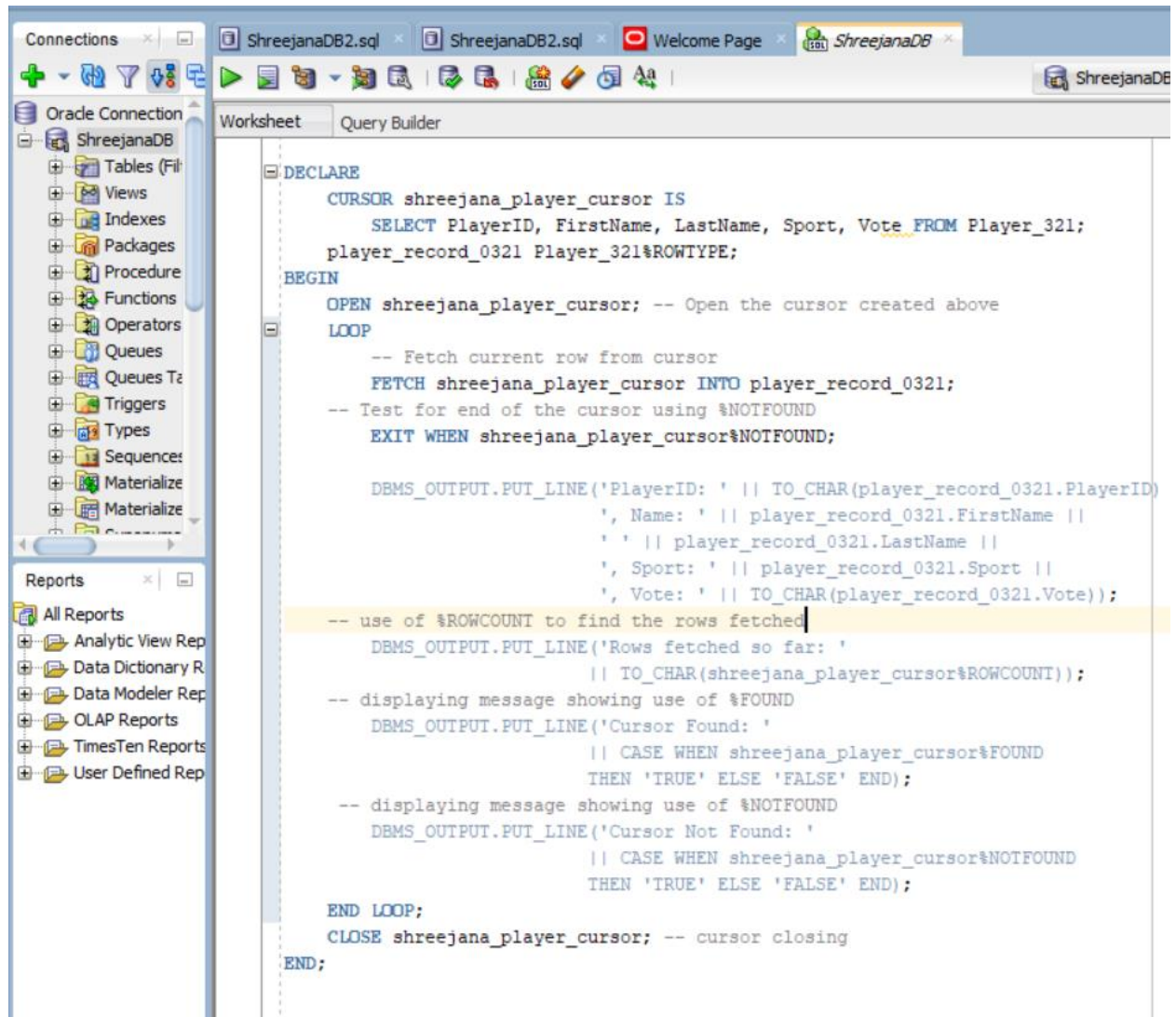
```
-- Cursor Attributes
-----
-- %ROWCOUNT
/* The %ROWCOUNT attribute is used to determine and return the number of rows
affected by the last executed SQL statement. This attribute is useful
for determining how many rows were fetched, inserted, updated, or deleted. */

-- %FOUND
/* The %FOUND attribute is used to determine whether a cursor contains rows
after the execution of a FETCH statement. It indicates whether the last
fetch was successful. If FETCH statement execution was successful,
the %FOUND attribute has a value of true. If FETCH statement execution
was not successful, the %FOUND attribute has a value of false.*/

-- %NOTFOUND
/* The %NOTFOUND attribute is the logical opposite of the %FOUND attribute.
It indicates whether the last fetch was unsuccessful. It returns TRUE if
the last fetch operation did not return a row and FALSE otherwise.*/

-- %OPEN
/* The %OPEN attribute is used to determine whether a cursor is in the open
state. The %OPEN attribute returns TRUE if the cursor is currently open and
FALSE if it is closed. */
```


code



The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' pane shows 'ShreejanaDB' selected. Below it, the 'Reports' pane lists various report types. The main 'Worksheet' tab contains a PL/SQL script. The script declares a cursor named 'shreejana_player_cursor' and uses a loop to fetch data from the 'Player_321' table. It includes comments for testing and outputting row counts and found/not found status.

```
DECLARE
    CURSOR shreejana_player_cursor IS
        SELECT PlayerID, FirstName, LastName, Sport, Vote FROM Player_321;
    player_record_0321 Player_321%ROWTYPE;
BEGIN
    OPEN shreejana_player_cursor; -- Open the cursor created above
    LOOP
        -- Fetch current row from cursor
        FETCH shreejana_player_cursor INTO player_record_0321;
        -- Test for end of the cursor using %NOTFOUND
        EXIT WHEN shreejana_player_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('PlayerID: ' || TO_CHAR(player_record_0321.PlayerID)
                               || ', Name: ' || player_record_0321.FirstName ||
                               ' ' || player_record_0321.LastName ||
                               ', Sport: ' || player_record_0321.Sport ||
                               ', Vote: ' || TO_CHAR(player_record_0321.Vote));

        -- use of %ROWCOUNT to find the rows fetched
        DBMS_OUTPUT.PUT_LINE('Rows fetched so far: '
                               || TO_CHAR(shreejana_player_cursor%ROWCOUNT));

        -- displaying message showing use of %FOUND
        DBMS_OUTPUT.PUT_LINE('Cursor Found: '
                               || CASE WHEN shreejana_player_cursor%FOUND
                                    THEN 'TRUE' ELSE 'FALSE' END);

        -- displaying message showing use of %NOTFOUND
        DBMS_OUTPUT.PUT_LINE('Cursor Not Found: '
                               || CASE WHEN shreejana_player_cursor%NOTFOUND
                                    THEN 'TRUE' ELSE 'FALSE' END);

    END LOOP;
    CLOSE shreejana_player_cursor; -- cursor closing
END;
```

Output

The screenshot displays the Oracle SQL Developer environment. The 'Connections' pane on the left shows the 'ShreejanaDB' connection. The 'Query Builder' window at the top shows the following SQL code:

```
EXIT WHEN shreejana_player_cursor%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('PlayerID: ' || TO_CHAR(player_record_0321.PlayerID) ||
                    ', Name: ' || player_record_0321.FirstName ||
                    ' ' || player_record_0321.LastName ||
                    ', Sport: ' || player_record_0321.Sport ||
                    ', Vote: ' || TO_CHAR(player_record_0321.Vote));

-- use of %ROWCOUNT to find the rows fetched
DBMS_OUTPUT.PUT_LINE('Rows fetched so far: ' || player_record_0321.ROWCOUNT);
```

The 'Script Output' window at the bottom shows the execution results, which are highlighted by a red box:

```
Task completed in 0.056 seconds.

PlayerID: 1, Name: Michael Jordan, Sport: Basketball, Vote: 98
Rows fetched so far: 1
Cursor Found: TRUE
Cursor Not Found: FALSE
PlayerID: 2, Name: Serena Williams, Sport: Tennis, Vote: 95
Rows fetched so far: 2
Cursor Found: TRUE
Cursor Not Found: FALSE
PlayerID: 3, Name: Lionel Messi, Sport: Soccer, Vote: 99
Rows fetched so far: 3
Cursor Found: TRUE
Cursor Not Found: FALSE
PlayerID: 4, Name: Usain Bolt, Sport: Athletics, Vote: 97
Rows fetched so far: 4
Cursor Found: TRUE
Cursor Not Found: FALSE

PL/SQL procedure successfully completed.
```