

CST3504/3604 Lecture Notes

A Practical Approach to Database Development Using Oracle Server Express Edition (Part 4)

CST3604 Lecture 1B (Part 2) – Stored Procedures & Functions

(Part 2 of 2)

(Lecture Notes 1B)

Prof. Abel Angel Rodriguez

Part 4 – Database Development in Oracle XE 11g Creating Stored Procedures & Functions (Part 2)

4.2 Cursors & Stored Procedures for Select Statements that return multiple records

4.2.1 CURSORS & STORED PROCEDURES – Using Cursors to Execute SELECT STATEMENTS that Return Multiple Records Inside Stored Procedures

- As we discussed in last section, **inside Stored Procedures SELECT INTO STATEMENT can ONLY RETURN ONE RECORD.**
- If you need to create a Stored Procedure with a SELECT STATEMENT that returns **MORE THAN ONE RECORD**, than **YOU NEED TO USE A CURSOR INSIDE THE STORED PROCEDURE.**
- In this section, we will combine CURSORS with STORED PROCEDURE in order to be able to display multiple records from a SELECT STATEMENT.

Understanding Cursors & STORED PROCEDURES

- Below are examples of using CURSORS inside STORE PROCEDURES.
- The process is the same as CURSORS in a script, except we DON'T need to create a BLOCK OF CODE (DECLARE, BEGIN & END statement) since the STORED PROCEDURE already has the mechanism to implement a BLOCK OF CODE.
- Simply integrate the CURSOR syntax into a STORED PROCEDURE syntax.
- This will be shown in examples below.

Examples of Stored Procedures with NO PARAMETERS (Usually Not Best Practice)

Example 1 – Stored Procedure that executes SELECT QUERY that returns multiple records of all Female Customers Names only (No Parameters in Stored Procedure)

Target Code to Implement using Cursor

- This example we display the name of all female customers in the business using a CURSOR inside STORED PROCEDURE.
- **NOTE – this type of query is usually NOT BEST PRACTICE or NOT implemented in real applications because this will always return the records with gender being FEMALE ONLY! What about if you needed the MALE version of this, would it require another Query/Stored Procedure? Makes no sense. Why not have ONE QUERY that handles both FEMALE & MALE? We will see example of this in later examples.**
- Target SELECT QUERY & expected OUTPUT is shown below:

- Target SELECT QUERY:

```
--Select Query with WHERE clause
SELECT Name
FROM CUSTOMER
WHERE Gender = 'F' ;
```

- Expected output:

```
-- Script using CURSOR to display the following messages
Name of Female Customer Josephine Smith (Jo)
Name of Female Customer Mary Jones
Name of Female Customer Nancy Rivera
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Design:

```
--Define/declare Stored Procedure
CREATE OR REPLACE PROCEDURE usp_FemaleCustomerNames
IS
    --declare variable needed for SELECT INTO statement
    v_Name VARCHAR2(50);

    --Declare Cursor that selects all female customers
    CURSOR cur_Customer IS
        -- Query cursor will point to results
        SELECT Name
        FROM Customer
        WHERE Gender = 'F';

    BEGIN
        --Open Cursor
        OPEN cur_Customer;
        --Use PL/SQL language control or loop to display each record pointed by cursor
        LOOP
            --Fetch Cursor
            FETCH cur_Customer INTO v_Name;
            --Exit loop if pointing to END OF FILE (cursor pointing to null)
            EXIT WHEN cur_Customer%NOTFOUND;
            --Display each record
            DBMS_OUTPUT.PUT_LINE('Name of Female Customer '||v_Name);

        END LOOP;
        --Close Cursor
        CLOSE cur_Customer;
    END usp_FemaleCustomerNames;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar below has various icons for file operations. The main workspace has tabs for 'Start Page' and 'Stored Procedures & Functions Scripts.sql'. The 'Connections' section shows 'OracleXE_DBMS_DBDeveloper1_Connection'. The 'Worksheet' tab is active, displaying the following PL/SQL code:

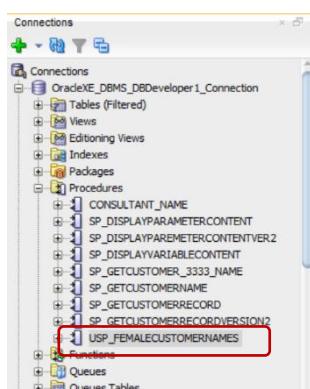
```
466
467     OPEN    cur_Customer;
468   LOOP
469
470     FETCH  cur_Customer INTO v_Name;
471     EXIT WHEN cur_Customer%NOTFOUND;
472     --Display each record
473     DBMS_OUTPUT.PUT_LINE('Name of Female Customer '||v_Name);
474
475   END LOOP;
476
477   CLOSE   cur_Customer;
478 END usp_FemaleCustomerNames;
```

A red box highlights the code area. Below it, the 'Script Output' pane shows the message "Procedure USP_FEMALECUSTOMERNAMES compiled" in a red-bordered box. The status bar at the bottom indicates "Line 478 Column 31".

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

□ In SQL Developer:

- Execute as follows:

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on

- 2) Execute/Compile the Script using the following syntax

EXECUTE ProcedureName;

- 3) The statement looks as follows:

EXECUTE sp_FemaleCustomerNames;

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDDeveloper1_Connection'. The 'Procedures' section is expanded, showing several stored procedures, with 'sp_FemaleCustomerNames' selected. The main workspace is a 'Worksheet' tab where the following PL/SQL code is written:

```
473   DBMS_OUTPUT.PUT_LINE('Name of Female Customer '||v_Name);
474
475   END LOOP;
476
477   CLOSE cur_Customer;
478   END usp_FemaleCustomerNames;
479
480 --Executing/testing Stored Procedure
481 --Turning server output on
482 SET SERVEROUTPUT ON;
483
484 -- Executing stored procedure
485 EXECUTE usp_FemaleCustomerNames;
486
487 -- Stored Procedure to display all female records using method 1
488 CREATE OR REPLACE PROCEDURE sp_FemaleCustomerDetails1
489 AS
```

A red box highlights the command 'EXECUTE usp_FemaleCustomerNames;'. The bottom right pane, titled 'Script Output', shows the results of the execution:

```
PL/SQL procedure successfully completed.
Name of Female Customer Josephine Smith (Jo)
Name of Female Customer Mary Jones
Name of Female Customer Nancy Rivera
```

A red box highlights the output text.

Example 2 – Stored Procedure that executes SELECT QUERY that returns multiple records will all Female Customers & all their attributes/columns

Target Code to Implement using Cursor

- ❑ This example we display the name of all female customers in the business using cursor.
- ❑ **NOTE – this type of query is usually NOT BEST PRACTICE or NOT implemented in real applications because this will always return the records with gender being FEMALE ONLY! What about if you needed the MALE version of this, would it require another Query/Stored Procedure? Makes no sense. Why not have ONE QUERY that handles both FEMALE & MALE? We will see example of this in later example.**
- ❑ Target SELECT QUERY & expected OUTPUT is shown below:

- Target SELECT QUERY:

```
--Select Query with WHERE clause
SELECT Customer_ID, Name, BDate, Address, Phone, Gender, Email
FROM CUSTOMER
WHERE Gender = 'F' ;
```

- Expected Output:

```
-- Script using CURSOR to display the following messages
Female Customer info: 1111 Josephine Smith (Jo) 01-JAN-81 111
Glendwood Road, Brooklyn NY 11210 718-282-1111 F
josephine.smith@comp.com
```

```
Female Customer info: 3333 Mary Jones 03-MAR-73 333 Flatlands
Avenue, Brooklyn NY 11203 718 333-3333 F mjones@xyz.com
```

```
Female Customer info: 5555 Nancy Rivera 05-MAY-75 555
Metropolitan Avenue, Brooklyn NY 11205 718 555-5555 F
nrivera@xyz.com
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
-- Stored Procedure to display all female records using method 1
CREATE OR REPLACE PROCEDURE usp_FemaleCustomerDetails1
IS
    --declare variable needed for SELECT INTO statement
    v_CustomerID  NUMBER(4);
    v_Name         VARCHAR2(50);
    v_BDate        DATE;
    v_Address      VARCHAR2(200);
    v_Phone        VARCHAR2(20);
    v_Gender       CHAR(1);
    v_Email        VARCHAR2(50);

    --Declare Cursor
    CURSOR cur_Customer IS

        -- Query cursor will point to results
        SELECT Customer_ID, Name, Bdate, Address, Phone, Gender, Email
        FROM Customer
        WHERE Gender = 'F';
    --Start Execution section
BEGIN
    --Open Cursor
    OPEN     cur_Customer; -- open cursor for use
    --loop to display each record returned by cursor
    --Use PL/SQL language control or loop to display each record pointed by cursor
    LOOP
        -- Fetch cursor data
        FETCH   cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone,
                           v_Gender, v_Email;
        EXIT WHEN cur_Customer%NOTFOUND;
        --Display each record
        DBMS_OUTPUT.PUT_LINE('Female Customer info: '||v_CustomerID||' '||v_Name||
                           ' '||v_BDate||' '||v_Address||' '||v_Phone||' '||v_Gender||' '|
                           ||v_Email);

    END LOOP;

    CLOSE   cur_Customer; --close cursor
END usp_FemaleCustomerDetails1;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:

The screenshot shows the Oracle SQL Developer interface. The SQL Worksheet pane contains PL/SQL code for a stored procedure named USP_FEMALECUSTOMERDETAILS1. The code retrieves female customer details from a table named Customer. The Script Output pane shows the message "Procedure USP_FEMALECUSTOMERDETAILS1 compiled".

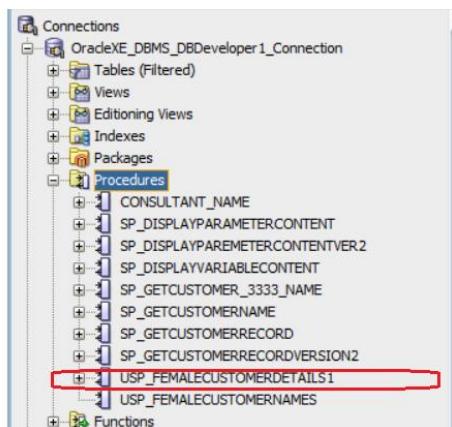
```
FROM Customer
WHERE Gender = 'F';
--Start Execution section
BEGIN
--Open Cursor
OPEN cur_Customer; -- open cursor for use
--loop to display each record returned by cursor
--Use PL/SQL language control or loop to display each record pointed by cursor
LOOP
-- Fetch cursor data
  FETCH cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone, v_Gender, v_Email;
  EXIT WHEN cur_CustomerNOTFOUND;
  --Display each record
  DBMS_OUTPUT.PUT_LINE('Female Customer info: '||v_CustomerID|| ' '||v_Name|| ' '||v_BDate|| ' '
    ||v_Address|| ' '||v_Phone|| ' '||v_Gender|| ' '||v_Email);
END LOOP;
CLOSE cur_Customer; --close cursor
END usp_FEMALECUSTOMERDETAILS1;
```

Procedure USP_FEMALECUSTOMERDETAILS1 compiled

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

□ In SQL Developer:

- Execute as follows:

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on

- 2) Execute/Compile the Script using the following syntax

EXECUTE ProcedureName;

- 3) The statement looks as follows:

EXECUTE sp_FemaleCustomerDetails1;

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDeveloper1_Connection'. The 'Procedures' section is expanded, showing several stored procedures including 'sp_FemaleCustomerDetails1'. The main workspace is a 'Worksheet' tab where the following PL/SQL code is written:

```

508 --loop to display each record returned by cursor
509 --Use PL/SQL language control or loop to display each record pointed by cursor
510 BEGIN
511   LOOP
512     -- Fetch cursor data
513     FETCH cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone, v_Gender, v_Email;
514     EXIT WHEN cur_Customer%NOTFOUND;
515     --Display each record
516     DBMS_OUTPUT.PUT_LINE('Female Customer info: '||v_CustomerID||' '||v_Name||' '||v_BDate||' '
517                               ||v_Address||' '||v_Phone||' '||v_Gender||' '||v_Email);
518   END LOOP;
519   CLOSE cur_Customer; --close cursor
520
521 END usp_FemaleCustomerDetails1;
522
523
524 --Turning server output on
525 SET SERVEROUTPUT ON;
526
527 -- Executing stored procedure
528 EXECUTE usp_FemaleCustomerDetails1;
529
530
531

```

A red box highlights the command 'SET SERVEROUTPUT ON;' and the execution line 'EXECUTE usp_FemaleCustomerDetails1;'. The 'Script Output' tab at the bottom shows the results of the execution:

```

Female Customer info: 1111 Josephine Smith (Jo) 01-JAN-81 111 Glendwood Road, Brooklyn NY 11210 718-282-1111 F josephine.smith@comp.com
Female Customer info: 3333 Mary Jones 03-MAR-73 333 Flatlands Avenue, Brooklyn NY 11203 718 333-3333 F mjones@xyz.com
Female Customer info: 5555 Nancy Rivera 05-MAY-75 555 Metropolitan Avenue, Brooklyn NY 11205 718 555-5555 F nrivera@xyz.com

```

Example 3 – Stored Procedure that executes SELECT QUERY that returns multiple records will all Female Customers & all their attributes/columns using the * character.

Target Code to Implement using Cursor

- ❑ This example we display the name of all female customers in the business using cursor just like the previous example, except that this time we use the * character in the query.
- ❑ **NOTE – this type of query is usually NOT BEST PRACTICE or NOT implemented in real applications because this will always return the records with gender being FEMALE ONLY! What about if you needed the MALE version of this, would it require another Query/Stored Procedure? Makes no sense. Why not have ONE QUERY that handles both FEMALE & MALE? We will see example of this in later examples.**
- ❑ Obviously, we expect the same results as the previous example.

- Target SELECT QUERY:

```
--Select Query with WHERE clause
SELECT *
FROM CUSTOMER
WHERE Gender = 'F' ;
```

- Expected OUTPUT is shown below

```
-- Script using CURSOR to display the following messages
Female Customer info: 1111 Josephine Smith (Jo) 01-JAN-81 111
Glendwood Road, Brooklyn NY 11210 718-282-1111 F
josephine.smith@comp.com

Female Customer info: 3333 Mary Jones 03-MAR-73 333 Flatlands
Avenue, Brooklyn NY 11203 718 333-3333 F mjones@xyz.com

Female Customer info: 5555 Nancy Rivera 05-MAY-75 555
Metropolitan Avenue, Brooklyn NY 11205 718 555-5555 F
nrivera@xyz.com
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
-- Stored Procedure to display all female records using method 1
CREATE OR REPLACE PROCEDURE usp_FemaleCustomerDetails1
IS
    --declare variable needed for SELECT INTO statement
    v_CustomerID  NUMBER(4);
    v_Name         VARCHAR2(50);
    v_BDate        DATE;
    v_Address      VARCHAR2(200);
    v_Phone        VARCHAR2(20);
    v_Gender       CHAR(1);
    v_Email        VARCHAR2(50);

    --Declare Cursor
    CURSOR cur_Customer IS

        -- Query cursor will point to results
        SELECT *
        FROM Customer
        WHERE Gender = 'F';
    --Start Execution section
BEGIN
    --Open Cursor
    OPEN   cur_Customer; -- open cursor for use
    --loop to display each record returned by cursor
    --Use PL/SQL language control or loop to display each record pointed by cursor
    LOOP
        -- Fetch cursor data
        FETCH   cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone,
                           v_Gender, v_Email;
        EXIT WHEN cur_Customer%NOTFOUND;
        --Display each record
        DBMS_OUTPUT.PUT_LINE('Female Customer info: '||v_CustomerID||' '||v_Name||
                           ' '||v_BDate||' '||v_Address||' '||v_Phone||' '||v_Gender||' '
                           ||v_Email);

    END LOOP;

    CLOSE   cur_Customer; --close cursor
END usp_FemaleCustomerDetails1;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : C:\Users\arod\Documents\NYCTC(Local)\Oracle Database Notes\Oracle Developer Scripts\Stored Procedures & Functions Scripts.sql". The left sidebar is the Connections tree, showing a single connection named "OracleXE_DBMS_DBDeveloper1_Connection" which contains several schema objects like Tables, Views, Procedures, etc. The main area has two tabs: "SQL Worksheet" and "Query Builder". The "SQL Worksheet" tab is active, displaying the following PL/SQL code:

```
--Declare Cursor
CURSOR cur_Customer IS
    -- Query cursor will point to results
    SELECT *
    FROM Customer
    WHERE Gender = 'F';

BEGIN
    OPEN cur_Customer;
    LOOP
        FETCH cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone, v_Gender, v_Email;
        EXIT WHEN cur_Customer%NOTFOUND;
        --Display each record
        DBMS_OUTPUT.PUT_LINE('Female Customer info: '||v_CustomerID||' '||v_Name||' '||v_BDate||'
                                ||v_Address||' '||v_Phone||' '||v_Gender||' '||v_Email);
    END LOOP;

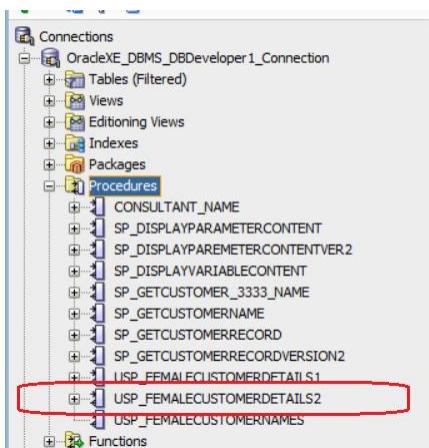
    CLOSE cur_Customer;
END usp_FemaleCustomerDetails2;
```

The "Script Output" pane at the bottom shows the message: "Procedure USP_FEMALECUSTOMERDETAILS2 compiled".

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

□ In SQL Developer:

- Execute as follows:

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax

EXECUTE ProcedureName;

- 3) The statement looks as follows:

EXECUTE sp_FemaleCustomerDetails2;

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows a single connection named 'OracleXE_DBMS_DBDDeveloper1_Connection'. The central workspace contains a SQL Worksheet with the following script:

```
551 BEGIN
552   OPEN cur_Customer;
553   LOOP
554     FETCH cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone, v_Gender, v_Email;
555     EXIT WHEN cur_Customer%NOTFOUND;
556     --Display each record
557     DBMS_OUTPUT.PUT_LINE('Female Customer info: ||v_CustomerID|| ' ||v_Name|| ' ||v_BDate|| '
558                           ||v_Address|| ' ||v_Phone|| ' ||v_Gender|| ' ||v_Email||');
559   END LOOP;
560
561   CLOSE cur_Customer;
562   END usp_FemaleCustomerDetails2;
563
564
565   --Turning server output on
566   SET SERVEROUTPUT ON;
567
568   -- Executing stored procedure
569   EXECUTE usp_FemaleCustomerDetails2;
570
571
572
573
574
```

The 'Script Output' tab at the bottom shows the results of the execution:

```
Female Customer info: 1111 Josephine Smith (Jo) 01-JAN-81 111 Glendale Road, Brooklyn NY 11210 718-282-1111 F josephine.smith@comp.com
Female Customer info: 3333 Mary Jones 03-MAR-73 333 Flatlands Avenue, Brooklyn NY 11203 718 333-3333 F mjones@xyz.com
Female Customer info: 5555 Nancy Rivera 05-MAY-75 555 Metropolitan Avenue, Brooklyn NY 11205 718 555-5555 F nrivera@xyz.com
```

Example 3 – Stored Procedure that executes SELECT QUERY that returns ALL records from Customer table & all attributes/columns

Target Code to Implement using Cursor

- ❑ This example we display the name of all female customers in the business using cursor just like the previous example, except that this time we use the * character in the query.
- ❑ **NOTE – this type of query is usually NOT BEST PRACTICE or NOT implemented in real applications because this type of query returns ALL RECORDS from a table. In practical applications, returning all records make no sense. Consider a table with 100,000 records, would it make sense to return that to a user screen? NO. This type of query is used for TEACHING & TESTING PURPOSE ONLY!**
- ❑ Obviously, we expect the same results as the previous example.

- Target SELECT QUERY:

```
--Select Query with WHERE clause
SELECT *
FROM CUSTOMER;
```

- Expected OUTPUT is shown below

```
-- Stored Procedure to display the following messages
CUSTOMER INFORMATION
Customer ID: 1111
Customer Name: Josephine Smith (Jo)
Customer Date of Birth: 01-JAN-81
Customer Address: 111 Glendwood Road, Brooklyn NY 11210
Customer Phone: 718-282-1111
Customer Gender: F
Customer Email: josephine.smith@comp.com
```

```
CUSTOMER INFORMATION
Customer ID: 3333
Customer Name: Mary Jones
Customer Date of Birth: 03-MAR-73
Customer Address: 333 Flatlands Avenue, Brooklyn NY 11203
Customer Phone: 718-333-3333
Customer Gender: F
Customer Email: mjones@xyz.com
```

```
CUSTOMER INFORMATION
Customer ID: 5555
Customer Name: Nancy Rivera
Customer Date of Birth: 05-MAY-75
Customer Address: 555 Metropolitan Avenue, Brooklyn NY 11205
Customer Phone: 718-555-5555
Customer Gender: F
Customer Email: nrivera@xyz.com
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
-- Stored Procedure to display all female records using method 1
CREATE OR REPLACE PROCEDURE usp_AllCustomers
IS
    --declare variable needed for SELECT INTO statement
    v_CustomerID  NUMBER(4);
    v_Name         VARCHAR2(50);
    v_BDate        DATE;
    v_Address      VARCHAR2(200);
    v_Phone        VARCHAR2(20);
    v_Gender       CHAR(1);
    v_Email        VARCHAR2(50);

    --Declare Cursor
    CURSOR cur_Customer IS
        -- Query cursor will point to results
        SELECT *
        FROM Customer;

    --Start Execution section
BEGIN
    --Open Cursor
    OPEN     cur_Customer; -- open cursor for use
    --loop to display each record returned by cursor
    --Use PL/SQL language control or loop to display each record pointed by cursor
    LOOP
        -- Fetch cursor data
        FETCH   cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone,
                           v_Gender, v_Email;
        EXIT WHEN cur_Customer%NOTFOUND;

        --Display each record
        DBMS_OUTPUT.PUT_LINE('CUSTOMER INFORMATION ');
        DBMS_OUTPUT.PUT_LINE('CustomerID: '||v_CustomerID);
        DBMS_OUTPUT.PUT_LINE('Customer Name: '||v_Name);
        DBMS_OUTPUT.PUT_LINE('Customer DOB: '||v_BDate);
        DBMS_OUTPUT.PUT_LINE('Customer Address: '||v_Address);
        DBMS_OUTPUT.PUT_LINE('Customer Phone: '||v_Phone);
        DBMS_OUTPUT.PUT_LINE('Customer Gender: '||v_Gender);
        DBMS_OUTPUT.PUT_LINE('Customer Email: '||v_Email);
        DBMS_OUTPUT.PUT_LINE(' ' ); -- empty line for formatting

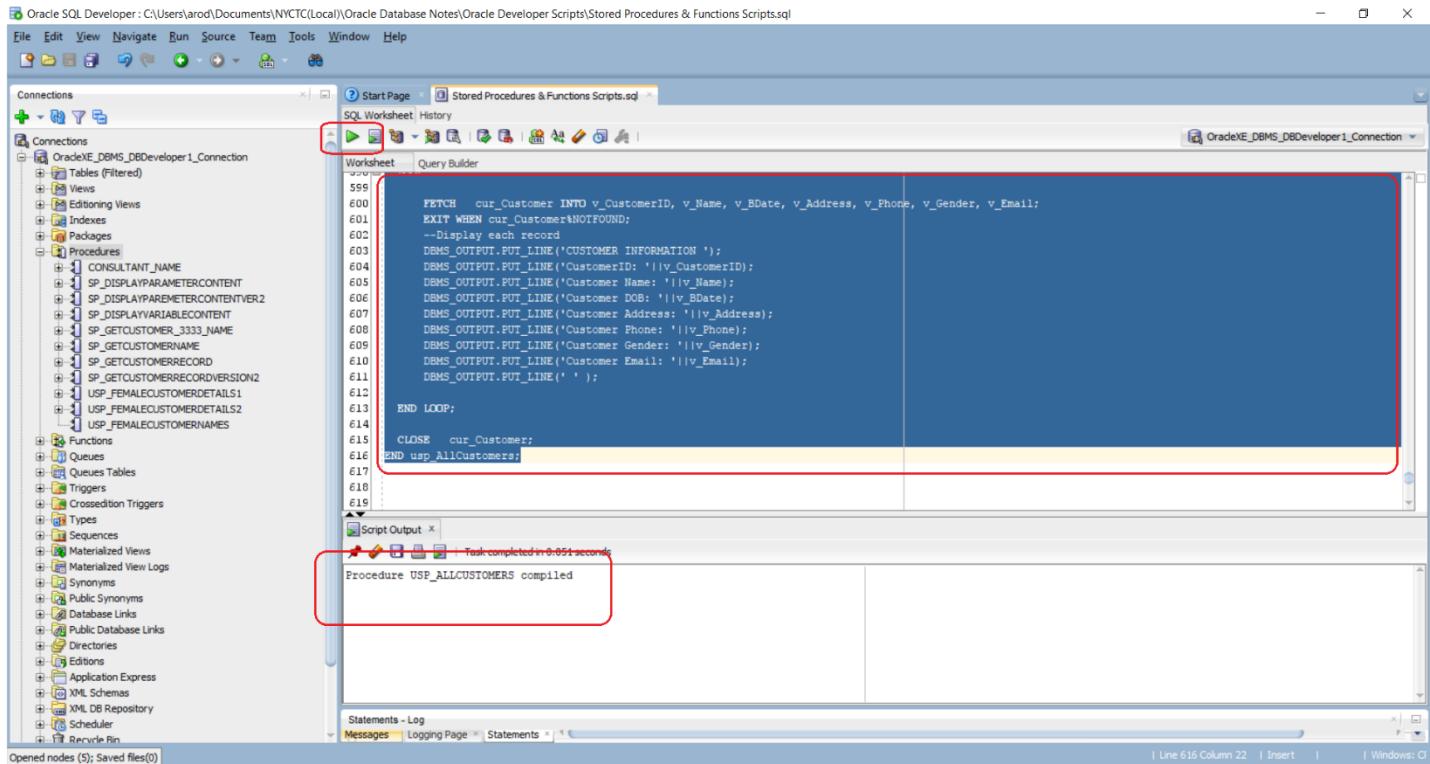
    END LOOP;

    CLOSE   cur_Customer; --close cursor
END usp_AllCustomers;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:



Oracle SQL Developer : C:\Users\arod\Documents\NYCTC\Local\Oracle Database Notes\Oracle Developer Scripts\Stored Procedures & Functions Scripts.sql

File Edit View Navigate Run Source Team Tools Window Help

Connections

Start Page Stored Procedures & Functions Scripts.sql

OracleXE_DBMS_DBDDeveloper1_Connection

Worksheet Query Builder

```
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
```

FETCH cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone, v_Gender, v_Email;
EXIT WHEN cur_Customer%NOTFOUND;
--Display each record
DBMS_OUTPUT.PUT_LINE('CUSTOMER INFORMATION ');
DBMS_OUTPUT.PUT_LINE('CustomerID: '||v_CustomerID);
DBMS_OUTPUT.PUT_LINE('Customer Name: '||v_Name);
DBMS_OUTPUT.PUT_LINE('Customer DOB: '||v_BDate);
DBMS_OUTPUT.PUT_LINE('Customer Address: '||v_Address);
DBMS_OUTPUT.PUT_LINE('Customer Phone: '||v_Phone);
DBMS_OUTPUT.PUT_LINE('Customer Gender: '||v_Gender);
DBMS_OUTPUT.PUT_LINE('Customer Email: '||v_Email);
DBMS_OUTPUT.PUT_LINE(' ');
END LOOP;
CLOSE cur_Customer;
END usp_AllCustomers;

Script Output

Task completed in 0.051 seconds

Procedure USP_ALLCUSTOMERS compiled

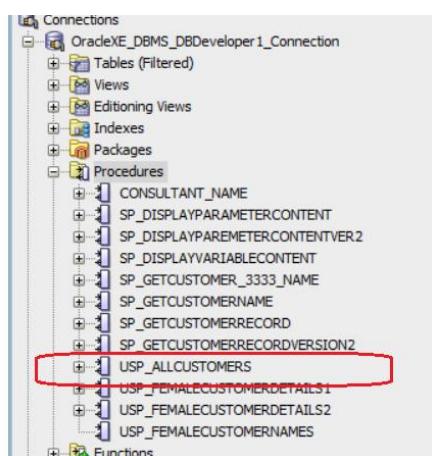
Statements - Log Messages Logging Page Statements

Opened nodes (5); Saved files(0)

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

□ In SQL Developer:

- Execute as follows:

1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on

2) Execute/Compile the Script using the following syntax

EXECUTE ProcedureName;

3) The statement looks as follows:

EXECUTE usp_AllCustomers;

4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left is the Connections tree, showing a single connection named 'OracleXE_DBMS_DBDeveloper1_Connection'. The central area has a 'Worksheet' tab open with the following SQL script:

```
CREATE OR REPLACE PROCEDURE usp_AllCustomers
AS
BEGIN
    --Turning server output on
    SET SERVEROUTPUT ON;
    -- Executing stored procedure
    EXECUTE usp_AllCustomers;
END;
/
```

The 'Script Output' tab at the bottom shows the results of the execution:

```
CUSTOMER INFORMATION
CustomerID: 3333
Customer Name: Mary Jones
Customer DOB: 03-MAR-73
Customer Address: 333 Flatlands Avenue, Brooklyn NY 11203
Customer Phone: 718 333-3333
Customer Gender: F
Customer Email: mjones@xyz.com

CUSTOMER INFORMATION
CustomerID: 5555
Customer Name: Nancy Rivera
Customer DOB: 05-MAY-75
Customer Address: 555 Metropolitan Avenue, Brooklyn NY 11205
Customer Phone: 718 555-5555
Customer Gender: F
Customer Email: nrivera@xyz.com

CUSTOMER INFORMATION
CustomerID: 6666
```

A red box highlights the 'Script Output' tab and the results of the execution.

Examples of Using CURSOR inside Stored Procedures with PARAMETERS (Best Practice) to execute SELECT Queries that return MORE THAN ONE RECORD

- Now we look at examples of STORED PROCEDURES PROPERLY DESIGNED to be GENERIC and execute queries that return one or more records and uses PARAMETERS to pass information/data needed to complete the Query.

Attention! Use CURSOR Inside Stored Procedures for Queries that CAN RETURN MORE THAN ONE RECORD!

- Below I integrated the syntax for Stored Procedure with parameters with CURSOR syntax:

```
CREATE [OR REPLACE] PROCEDURE procedure_name [(Param1, Param2, Param3.  
Param(n))]  
IS | AS  
--Declaration section (variables, cursor & other declarations)  
  
--Declaring the cursor & POINTING it to the results of a SELECT Statement  
--Note that the SELECT statement will use all or some of the parameters to  
--complete the SELECT with the data it needs for execution  
CURSOR Cursor_Name IS SELECT Statement;  
BEGIN  
--Oracle PL/SQL statements/executable section  
    -- open cursor for use  
    OPEN Cursor_Name;  
  
    -- Fetch & assign row cursor points to into variables  
    FETCH Cursor_Name INTO Var1, Var2, Var3, Var(n);  
  
        -- Consume/process variables as needed here!  
  
    -- Close cursor when done fetching and processing  
    CLOSE Cursor_Name;  
  
END [procedure_name];
```

Syntax for various components of Stored Procedure:

Description	Syntax	Location
Parameter declaration syntax:	Param(X) = ParameterName IN OUT IN OUT OracleDataType	Procedure Header
Variable declaration syntax:	VariableName OracleDataType;	IS AS block
Variable assignment syntax:	VariableName := value;	BEGIN block

Example 1 – Stored Procedure that executes SELECT QUERY that returns multiple records of MALE OR FEMALE Customers & all their attributes/columns using Parameters

Target Code to Implement using Cursor

- This example we display all attributes or columns of either Male or Female customers in the business using cursor.
- **NOTE – this type of query is usually IS BEST PRACTICE** or implemented in real applications because this will return either MALE OR FEMALE gender! By USING PARAMETERS, WE CAN MAKE THE QUERY GENERIC and can handle multiple criteria.
- Target SELECT QUERY & expected OUTPUT is shown below:

- Target SELECT QUERY:

```
--Select Query with WHERE clause
SELECT Customer_ID, Name, BDate, Address, Phone, Gender, Email
FROM CUSTOMER
WHERE Gender = ?;
```

- Expected Output for female:

```
-- Script using CURSOR to display the following messages
Customer info: 1111 Josephine Smith (Jo) 01-JAN-81 111
Glendwood Road, Brooklyn NY 11210 718-282-1111 F
josephine.smith@comp.com

Customer info: 3333 Mary Jones 03-MAR-73 333 Flatlands Avenue,
Brooklyn NY 11203 718 333-3333 F mjones@xyz.com

Customer info: 5555 Nancy Rivera 05-MAY-75 555 Metropolitan
Avenue, Brooklyn NY 11205 718 555-5555 F nrivera@xyz.com
```

- Expected Output for male:

```
-- Script using CURSOR to display the following messages
Customer info: 2222 Angel Rodriguez 01-JAN-72 222 Park Avenue,
New York NY 10030 212-222-2222 M arod@xyz.com

Customer info: 6666 Frank Hum 04-JUN-76 666 74th Street,
Flushing NY 11340 718 666-6666 M fhum@comp.com
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
-- Stored Procedure to display all female records using method 1
CREATE OR REPLACE PROCEDURE usp_CustomerByGender(p_Gender IN CHAR)
IS
    --declare variable needed for SELECT INTO statement
    v_CustomerID  NUMBER(4);
    v_Name         VARCHAR2(50);
    v_BDate        DATE;
    v_Address      VARCHAR2(200);
    v_Phone        VARCHAR2(20);
    v_Gender       CHAR(1);
    v_Email        VARCHAR2(50);

    --Declare Cursor
    CURSOR cur_Customer IS

        -- Query cursor will point to results
        SELECT Customer_ID, Name, Bdate, Address, Phone, Gender, Email
        FROM Customer
        WHERE Gender = p_Gender;          -- using parameter in query

    --Start Execution section
BEGIN
    --Open Cursor
    OPEN   cur_Customer; -- open cursor for use
    --loop to display each record returned by cursor
    --Use PL/SQL language control or loop to display each record pointed by cursor
    LOOP
        -- Fetch cursor data
        FETCH   cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone,
                    v_Gender, v_Email;
        EXIT WHEN cur_Customer%NOTFOUND;
        --Display each record
        DBMS_OUTPUT.PUT_LINE('Customer info: '||v_CustomerID||' '||v_Name||
        ' '||v_BDate||' '||v_Address||' '||v_Phone||' '||v_Gender||' '|
        ||v_Email);

    END LOOP;

    CLOSE   cur_Customer; --close cursor
END usp_CustomerByGender;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - Enter code in script
 - Execute/Compile the Script
 - Verify Procedure was successfully compiled
- Steps shown below:

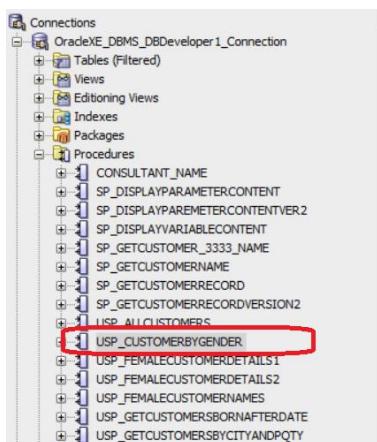
The screenshot shows the Oracle SQL Developer interface. On the left is the Navigator window displaying database objects like Tables, Views, Procedures, etc. The main area is a Worksheet tab showing PL/SQL code for a procedure named USP_CUSTOMERBYGENDER. The code uses a cursor to select customer information and then loops to output each record. A red box highlights the 'Run' button in the toolbar above the worksheet. Another red box highlights the 'Script Output' window at the bottom, which shows the message 'Procedure USP_CUSTOMERBYGENDER compiled'. The status bar at the bottom right indicates 'Task completed in 0.239 seconds'.

```
1  CURSOR cur_Customer IS
2    -- Query cursor will point to results
3    SELECT Customer_ID, Name, Bdate, Address, Phone, Gender, Email
4    FROM Customer
5    WHERE Gender = p_Gender;
6
7  BEGIN
8    --Start Execution section
9
10   --Open Cursor
11   OPEN cur_Customer; -- open cursor for use
12   --loop to display each record returned by cursor
13   --Use PL/SQL language control or loop to display each record pointed by cursor
14   LOOP
15     -- Fetch cursor data
16     FETCH cur_Customer INTO v_CustomerID, v_Name, v_BDate, v_Address, v_Phone, v_Gender, v_Email;
17     EXIT WHEN cur_Customer%NOTFOUND;
18     --Display each record
19     DBMS_OUTPUT.PUT_LINE('Female Customer info: '||v_CustomerID|| ' ' ||v_Name|| ' ' ||v_BDate|| '
20                               ||v_Address|| ' ' ||v_Phone|| ' ' ||v_Gender|| ' ' ||v_Email);
21   END LOOP;
22
23   CLOSE cur_Customer; --close cursor
24
25 END usp_CustomerByGender;
```

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- In the Navigator Window select Procedure object.
- Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

□ In SQL Developer we execute the procedure using several methods to show example of various approach:

- **Option #1 – Passing VALUE as argument to Stored Procedure for FEMALE:**

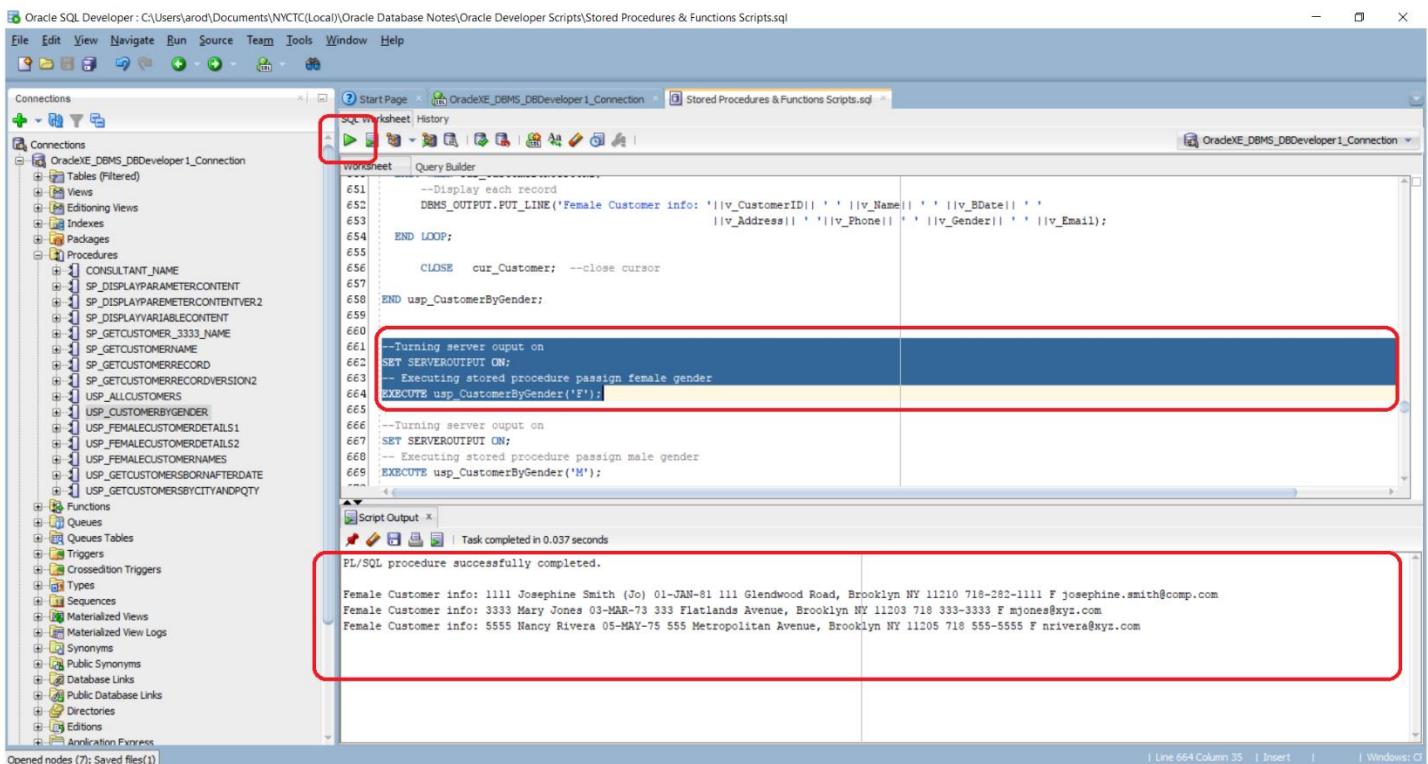
- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax using parameters

EXECUTE ProcedureName (value);

- 3) We execute the statement with the following argument for female:

EXECUTE usp_CustomerByGender ('F');

- 4) Message Window should indicate if execution was successful & show output of results



- **Option #2 – Passing VALUE as argument to Stored Procedure for MALE:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax using parameters

EXECUTE ProcedureName (Value);

- 3) We execute the statement with the following argument for female:

EXECUTE usp_CustomerByGender ('M');

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left is the Connections tree, showing a single connection named "OracleXE_DBMS_DBDDeveloper1_Connection". The main area has two tabs: "Worksheet" and "Query Builder". The "Worksheet" tab contains the following SQL code:

```

660 :--Turning server output on
661 SET SERVEROUTPUT ON;
662 -- Executing stored procedure passign female gender
663 EXECUTE usp_CustomerByGender('F');
664
665 :--Turning server output on
666 SET SERVEROUTPUT ON;
667 -- Executing stored procedure passign male gender
668 EXECUTE usp_CustomerByGender('M');
669
670 :--Turning server output on
671 SET SERVEROUTPUT ON;
672 -- Executing stored procedure passign variable with Female Gender
673 v_Gender := 'F';
674 EXECUTE usp_CustomerByGender(v_Gender);
675
676 :--Turning server output on
677 SET SERVEROUTPUT ON;
678

```

The "Script Output" tab at the bottom shows the results of the execution:

```

PL/SQL procedure successfully completed.

Female Customer info: 7777 Michael Triolo 07-JUL-77 777 Church Avenue, Brooklyn NY 11201 718 777-7777 M mtriolo@xyz.com
Female Customer info: 2222 Angel Rodriguez 02-FEB-72 222 Park Avenue, New York, NY 10030 212 222-2222 M arod@xyz.com
Female Customer info: 6666 Frank Hum 04-JUN-76 666 74th Street, Flushing NY 1134 718 666-6666 M fhum@comp.com

```

- **Option #3 – Passing VARIABLE as argument to Stored Procedure for FEMALE:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) **RECAP!** – Recall that to declare variables, assign variables & use variables in your SCRIPTS, you need to use a BLOCK OF CODE:

```

DECLARE
    -- Declaration section

BEGIN
    -- Execution section (Assign variable, execute code)

END ;

```

- 3) **NEW!** – when executing or calling a Stored Procedure from within a BLOCK OF CODE in your Script, you **DO NOT NEED** the **KEYWORD EXECUTE**! Therefore, from WITHIN A BLOCK OF CODE the following syntax using parameters & passing a VARIABLE as argument is as follows:

ProcedureName (variable);

- 4) We create a variable to store the value of the gender:

```
v_TheGender      CHAR(1) ;
```

- 5) We assign the variable the value of the gender:

```
v_Gender := 'F' ;
```

- 6) We execute the statement with the following argument for female:

```
usp_CustomerByGender(v_Gender) ;
```

- 7) Put it all in a BLOCK OF CODE:

```

DECLARE
    -- Declaration section
v_TheGender      CHAR(1) ;

BEGIN
    -- Execution section (Assign variable, execute code)
v_Gender := 'F' ;

usp_CustomerByGender(v_Gender) ;

END ;

```

8) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left is the Connections tree, showing a single connection named 'OracleXE_DBMS_DBDDeveloper1_Connection'. The main workspace has two tabs: 'Start Page' and 'Stored Procedures & Functions Scripts.sql'. The 'Stored Procedures & Functions Scripts.sql' tab is active and contains the following PL/SQL code:

```
675 -- Executing stored procedure passing variable with Female Gender
676 -- Remember, to create variables, assign variables you need to use a
677 -- BLOCK OF CODE in a SCRIPT, thus we need DECLARE, BEGIN, END statement
678
679 --Turning server output on
680 SET SERVEROUTPUT ON;
681 DECLARE
682   v_TheGender      CHAR(1);           --Declare variable
683
684 BEGIN
685   v_TheGender := 'F';                --Assigned variable
686
687   -- New Syntax = EXECUTE keyword NOT NEEDED when calling from
688   -- within BLOCK OF CODE
689   usp_CustomerByGender(v_TheGender);  --Pass variable as argument
690
691 END;
692
693
694
```

Below the code, the 'Script Output' window displays the results of the execution:

```
PL/SQL procedure successfully completed.

Female Customer info: 1111 Josephine Smith (Jo) 01-JAN-81 111 Glendale Road, Brooklyn NY 11210 718-282-1111 F josephine.smith@comp.com
Female Customer info: 3333 Mary Jones 03-MAR-73 333 Flatlands Avenue, Brooklyn NY 11203 718 333-3333 F mjones@xyz.com
Female Customer info: 5555 Nancy Rivera 05-MAY-75 555 Metropolitan Avenue, Brooklyn NY 11205 718 555-5555 F nrivera@xyz.com
```

- **Option #4 – Passing VARIABLE as argument to Stored Procedure for MALE:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) **RECAP!** – Recall that to declare variables, assign variables & use variables in your SCRIPTS, you need to use a BLOCK OF CODE:

```

DECLARE
    -- Declaration section

BEGIN
    -- Execution section (Assign variable, execute code)

END ;

```

- 3) **NEW!** – when executing or calling a Stored Procedure from within a BLOCK OF CODE in your Script, you **DO NOT NEED** the **KEYWORD EXECUTE**! Therefore, from WITHIN A BLOCK OF CODE the following syntax using parameters & passing a VARIABLE as argument is as follows:

ProcedureName (variable);

- 4) We create a variable to store the value of the gender:

```
v_TheGender      CHAR(1) ;
```

- 5) We assign the variable the value of the gender:

```
v_Gender := 'M' ;
```

- 6) We execute the statement with the following argument for female:

```
usp_CustomerByGender(v_Gender) ;
```

- 7) Put it all in a BLOCK OF CODE:

```

DECLARE
    -- Declaration section
    v_TheGender      CHAR(1) ;

BEGIN
    -- Execution section (Assign variable, execute code)
    v_Gender := 'M' ;

    usp_CustomerByGender(v_Gender) ;

END ;

```

8) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree displays a single connection named 'OracleXE_DBMS_DBDDeveloper1_Connection' containing various database objects like Tables, Views, Procedures, Functions, etc. The main workspace is a 'Worksheet' tab where a PL/SQL script is being run. The script is as follows:

```
682
683 -- Executing stored procedure passign variable with MAEL Gender
684 -- Remember, to create variables, assign variables you need to use a
685 -- BLOCK OF CODE in a SCRIPT, thus we need DECLARE, BEGIN, END statement
686
687
688 --Turning server ouput on
689 SET SERVEROUTPUT ON;
690
691 DECLARE
692     v_TheGender    CHAR(1);           --Declare variable
693
694 BEGIN
695     v_TheGender := 'M';             --Assigned variable
696
697     -- New Syntax = EXECUTE keyword NOT NEEDED when calling from
698     -- within BLOCK OF CODE
699     --usp_CustomerByGender(v_TheGender);          --Pass variable as argument
700
701 END;
```

The 'Script Output' window at the bottom shows the results of the execution:

```
PL/SQL procedure successfully completed.

Female Customer info: 7777 Michael Triolo 07-JUL-77 777 Church Avenue, Brooklyn NY 11201 718 777-7777 M mtriolo@xyz.com
Female Customer info: 2222 Angel Rodriguez 02-FEB-72 222 Park Avenue, New York, NY 10030 212 222-2222 M arod@xyz.com
Female Customer info: 6666 Frank Hum 04-JUN-76 666 74th Street, Flushing NY 1134 718 666-6666 M fhum@comp.com
```

Example 2 – Stored Procedure Stored Procedure that displays the Name & Gender of all records for customers born a certain Birth Date grouped by gender (Using Parameters)

Target Code to Implement using Cursor

- ❑ This example we display the name & gender of customers born after a date.
- ❑ **NOTE – this type of query is usually BEST PRACTICE or implemented in real applications because this will return the records FOR ANY DATE!**

- ❑ The target query & results:

- Target SELECT QUERY:

```
--Select Query with WHERE clause
SELECT Name, Gender
FROM CUSTOMER
WHERE BDate > = ?
GROUP BY Gender, Name;
```

- Expected OUTPUT is shown below for date greater than FEBRUARY 02, 1972

```
-- Script using CURSOR to display the following messages
Customer info: Josephine Smith (Jo) | F

Customer info: Mary Jones | F

Customer info: Nancy Rivera | F

Customer info: Michael Triolo | M

Customer info: Frank Hum | M
```

- Another possible expected Output for date greater than DECEMBER 12, 1980

```
-- Script using CURSOR to display the following messages
Customer info: Josephine Smith (Jo) | F
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
-- Stored Procedure that displays all records for customers born after
-- a date passed by parameter grouped by gender

CREATE OR REPLACE PROCEDURE usp_GetCustomersBornAfterDate(pDate IN DATE)
IS
    --declare variable needed for SELECT INTO statement

    v_Name      VARCHAR2(50);
    v_Gender     CHAR(1);

    --Declare Cursor
    CURSOR cur_Customer IS

        -- Query cursor will point to results
        SELECT Name, Gender
        FROM Customer
        WHERE BDate > pDate           -- using parameter in query
        GROUP BY Gender, Name;

    --Start Execution section
BEGIN
    --Open Cursor
    OPEN   cur_Customer; -- open cursor for use
    --loop to display each record returned by cursor
    --Use PL/SQL language control or loop to display each record pointed by cursor
    LOOP
        -- Fetch cursor data
        FETCH   cur_Customer INTO v_CustomerID, v_Name, v_Gender;
        EXIT WHEN cur_Customer%NOTFOUND;
        --Display each record
        DBMS_OUTPUT.PUT_LINE('Customer info: '||v_Name|| ' | ' ||v_Gender);

    END LOOP;

    CLOSE   cur_Customer; --close cursor
END usp_GetCustomersBornAfterDate;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDeveloper1_Connection'. The 'Procedures' node is expanded, showing several procedures like 'CONSULTANT_NAME', 'SP_DISPLAYPARAMETERCONTENT', etc., and one procedure highlighted with a red box: 'USP_GETCUSTOMERSBORNAFTERDATE'. The main workspace is a 'Worksheet' tab showing the PL/SQL code for this procedure. The code defines a cursor 'cur_Customer' to select data from a table, loops through it, and prints the results. The bottom status bar indicates the procedure was compiled successfully.

```
GROUP BY Gender, Name;
--Start Execution section
BEGIN
--Open Cursor
OPEN cur_Customer; -- open cursor for use
--loop to display each record returned by cursor
LOOP
-- Fetch cursor data
  FETCH cur_Customer INTO v_Name, v_Gender;
  EXIT WHEN cur_Customer%NOTFOUND;
--Display each record
  DBMS_OUTPUT.PUT_LINE('Customer Info: ' || v_Name|| ' | ' ||v_Gender);
END LOOP;
CLOSE cur_Customer; --close cursor
END usp_GetCustomersBornAfterDate;
```

Procedure USP_GETCUSTOMERSBORNAFTERDATE compiled

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

- In SQL Developer:

- Option #1 – Passing VALUE as argument to Stored Procedure for FEMALE:**

- Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- Execute/Compile the Script using the following syntax using parameters

EXECUTE *ProcedureName* (*value*) ;

- We execute the statement with the following argument for female:

EXECUTE usp_GetCustomersBornAfterDate ('02-FEB-72') ;

- Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows a single connection named "OracleXE_DBMS_DBDeveloper1_Connection". The central workspace has a "Worksheet" tab open, containing the following PL/SQL script:

```
742   LOOP
743     -- Fetch cursor data
744     FETCH cur_Customer INTO v_Name, v_Gender;
745     EXIT WHEN cur_Customer%NOTFOUND;
746     --Display each record
747     DBMS_OUTPUT.PUT_LINE('Customer Info: ' ||v_Name|| ' | ' ||v_Gender);
748   END LOOP;
749
750   CLOSE cur_Customer; --close cursor
751
752 END usp_GetCustomersBornAfterDate;
753
754
755 --Turning server ouput on
756 SET SERVEROUTPUT ON;
757
758 -- Executing stored procedure
759 EXECUTE usp_GetCustomersBornAfterDate('02-FEB-72');
760
761
762
```

A red box highlights the command `EXECUTE usp_GetCustomersBornAfterDate('02-FEB-72');`. Below the worksheet, the "Script Output" window displays the results of the execution:

```
PL/SQL procedure successfully completed.

Customer Info: Josephine Smith (Jo) | F
Customer Info: Mary Jones | F
Customer Info: Nancy Rivera | F
Customer Info: Michael Trioio | M
Customer Info: Frank Hum | M
```

The "Script Output" window also includes a note: "Task completed in 0.031 seconds".

▪ **Option #2 – Passing VALUE as argument to Stored Procedure for MALE:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax using parameters

EXECUTE ProcedureName (Value);

- 3) We execute the statement with the following argument for female:

EXECUTE usp_GetCustomersBornAfterDate('12-DEC-80');

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left is the Object Navigator pane, which lists various database objects like Tables, Views, Procedures, Functions, and Sequences under the connection 'OracleXE_DBMS_DBDeveloper_1_Connection'. In the center is the 'Worksheet' tab where the following PL/SQL script is written:

```

754 :--Turning server output on
755 SET SERVEROUTPUT ON;
756
757 :-- Executing stored procedure
758 EXECUTE usp_GetCustomersBornAfterDate('01-FEB-72');
759
760
761
762
763 :--Turning server output on
764 SET SERVEROUTPUT ON;
765
766 :-- Executing stored procedure
767 EXECUTE usp_GetCustomersBornAfterDate('12-DEC-80');
768
769
770
771
772
773
774

```

The lines from 763 to 767 are highlighted with a red rectangle. Below the worksheet is the 'Script Output' tab, which displays the results of the execution:

```

PL/SQL procedure successfully completed.

Customer Info: Josephine Smith (Jo) | F

```

The 'Script Output' tab is also highlighted with a red rectangle. The status bar at the bottom right indicates 'Task completed in 0.019 seconds'.

- **Option #3 – Passing VARIABLE as argument to Stored Procedure for date = FEBRUARY 02, 1972:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) **RECAP!** – Recall that to declare variables, assign variables & use variables in your SCRIPTS, you need to use a BLOCK OF CODE:

```

DECLARE
    -- Declaration section

BEGIN
    -- Execution section (Assign variable, execute code)

END ;

```

- 3) **NEW!** – when executing or calling a Stored Procedure from within a BLOCK OF CODE in your Script, you **DO NOT NEED** the **KEYWORD EXECUTE**! Therefore, from WITHIN A BLOCK OF CODE the following syntax using parameters & passing a VARIABLE as argument is as follows:

ProcedureName (variable);

- 4) We create a variable to store the value of the gender:

```
v_TheDate      DATE;
```

- 5) We assign the variable the value of the gender:

```
v_ TheDate:= '02-FEB-72';
```

- 6) We execute the statement with the following argument for female:

```
usp_GetCustomersBornAfterDate(v_TheDate);
```

- 7) Put it all in a BLOCK OF CODE:

```

DECLARE
    -- Declaration section
    v_TheDate      DATE;

BEGIN
    -- Execution section (Assign variable, execute code)
    v_ TheDate:= '02-FEB-72';

    usp_GetCustomersBornAfterDate(v_TheDate);

END ;

```

8) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDeveloper1_Connection'. The 'Procedures' node is expanded, showing numerous stored procedures like SP_CONSULTANT_NAME, SP_DISPLAYPARAMETERCONTENT, etc. The central workspace has a 'Worksheet' tab open, containing the following PL/SQL code:

```
769 -- Executing stored procedure passign variable with DATE
770 -- Remember, to create variables, assign variables you need to use a
771 -- BLOCK OF CODE in a SCRIPT, thus we need DECLARE, BEGIN, END statement
772
773
774 --Turning server ouput on
775 SET SERVEROUTPUT ON;
776
777 DECLARE
778   v_TheDate      DATE;          --Declare variable
779
780 BEGIN
781   v_TheDate := '02-FEB-72';      --Assigned variable
782
783   -- New Syntax = EXECUTE keyword NOT NEEDED when calling from
784   -- within BLOCK OF CODE
785   usp_GetCustomersBornAfterDate(v_TheDate);    --Pass variable as argument
786
787 END;
788
789
```

The code is highlighted with a red rectangle. Below the code, the 'Script Output' window shows the results of the execution:

```
PL/SQL procedure successfully completed.

Customer Info: Josephine Smith (Jo) | F
Customer Info: Mary Jones | F
Customer Info: Nancy Rivera | F
Customer Info: Michael Triolo | M
Customer Info: Frank Hum | M
```

The 'Script Output' window is also highlighted with a red rectangle.

- **Option #3 – Passing VARIABLE as argument to Stored Procedure for date = DECEMBER 12, 1980:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) **RECAP!** – Recall that to declare variables, assign variables & use variables in your SCRIPTS, you need to use a BLOCK OF CODE:

```

DECLARE
    -- Declaration section

BEGIN
    -- Execution section (Assign variable, execute code)

END ;

```

- 3) **NEW!** – when executing or calling a Stored Procedure from within a BLOCK OF CODE in your Script, you **DO NOT NEED** the **KEYWORD EXECUTE**! Therefore, from WITHIN A BLOCK OF CODE the following syntax using parameters & passing a VARIABLE as argument is as follows:

ProcedureName (variable);

- 4) We create a variable to store the value of the gender:

```
v_TheDate      DATE;
```

- 5) We assign the variable the value of the gender:

```
v_ TheDate:= '12-DEC-80';
```

- 6) We execute the statement with the following argument for female:

```
usp_GetCustomersBornAfterDate(v_TheDate);
```

- 7) Put it all in a BLOCK OF CODE:

```

DECLARE
    -- Declaration section
v_TheDate      DATE;

BEGIN
    -- Execution section (Assign variable, execute code)
v_ TheDate:= '12-DEC-80';

usp_GetCustomersBornAfterDate(v_TheDate);

END ;

```

8) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left is a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDDeveloper_1_Connection'. The 'Procedures' node is expanded, showing various stored procedures like SP_GETCUSTOMERNAME, SP_GETCUSTOMERRECORD, etc. In the center is a 'Worksheet' tab where a PL/SQL script is being run. The script is as follows:

```
790 :-- Executing stored procedure passign variable with MAEL Gender
791 :-- Remember, to create variables, assign variables you need to use a
792 :-- BLOCK OF CODE in a SCRIPt, thus we need DECLARE, BEGIN, END statement
793
794
795
796 :--Turning server ouput on
797 SET SERVEROUTPUT ON;
798 DECLARE
799   v_TheDate      DATE;          --Declare variable
800
801 BEGIN
802   v_TheDate := '12-DEC-80';      --Assigned variable
803
804   -- New Syntax = EXECUTE keyword NOT NEEDED when calling from
805   -- within BLOCK OF CODE
806   usp_GetCustomersBornAfterDate(v_TheDate);      --Pass variable as argument
807
808 END;
809
810
```

Below the worksheet, the 'Script Output' window shows the results of the execution:

```
PL/SQL procedure successfully completed.

Customer Info: Josephine Smith (Jo) | F
```

Example 3 – Stored Procedure Stored Procedure that displays the Name, address, email, product description & quantity of all customers who live in a city and have purchased more than X products. Business Objectives is to give them a special discount (Using Parameters)

Target Code to Implement using Cursor

- This example we display the name & gender of customers born after a date.
- **NOTE – this type of query is usually BEST PRACTICE or implemented in real applications because this stored procedure will return the records FOR ANY CITY & PURCHASE AMOUNT!**
- The target query & results:
 - Target SELECT QUERY using JOIN & LIKE statement to search for a string in a column:

```
--Select Query with WHERE clause
SELECT Customer.Name, Customer.Address, Customer.Email,
Product.Description, Purchase.Quantity
FROM CUSTOMER, PURCHASE, PRODUCT
WHERE CUSTOMER.CUSTOMER_ID = PURCHASE.CUSTOMER_ID AND
PURCHASE.PRODUCT_ID = PRODUCT.PRODUCT_ID
AND CUSTOMER.Address LIKE '%'||p_City||'%'
AND PURCHASE.Quantity >= p_PQuantity;
```

- Expected OUTPUT is shown below for customer who live in BROOKLYN & purchase quantity >= to 2 products

```
-- Script using CURSOR to display the following messages
Customer info: Josephine Smith (Jo) | 111 Glendwood Road,
Brooklyn NY 11210 | josephine.smith@comp.com | High Definition
TV | 2

Customer info: Nancy Rivera | 555 Metropolitan Avenue,
Brooklyn NY 11205 | nrivera@xyz.com | Laptop Computer | 4
```

- Expected OUTPUT is shown below for customer who live in BROOKLYN & purchase quantity >= to 3 products

```
-- Script using CURSOR to display the following messages
Customer info: Nancy Rivera | 555 Metropolitan Avenue,
Brooklyn NY 11205 | nrivera@xyz.com | Laptop Computer | 4
```

- Expected OUTPUT is shown below for customer who live in BROOKLYN & purchase quantity >= to 3 products

```
-- Script using CURSOR to display the following messages
Customer info: Angel Rodriguez | 222 Park Avenue, New York, NY
10030 | arod@xyz.com | High Power Washing Machine | 1

Customer info: Angel Rodriguez | 222 Park Avenue, New York, NY
10030 | arod@xyz.com | High Power Washing Machine | 1
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
-- Stored Procedure that displays all records for customers born after
-- a date passed by parameter grouped by gender

CREATE OR REPLACE PROCEDURE usp_GetCustomersByCityAndPQty(p_City IN VARCHAR2, p_PQuantity
IN NUMBER)
IS
    --declare variable needed for SELECT INTO statement

    v_Name          VARCHAR2(50);
    v_Address       VARCHAR2(200);
    v_Email         VARCHAR2(50);
    v_Prod_Description VARCHAR2(400);
    v_Purchase_Qty  NUMBER(5,0);

    --Declare Cursor
    CURSOR cur_Customer IS

        -- Query cursor will point to results
        SELECT Customer.Name, Customer.Address, Customer.Email,
               Product.Description, Purchase.Quantity
        FROM CUSTOMER, PURCHASE, PRODUCT
        WHERE CUSTOMER.CUSTOMER_ID = PURCHASE.CUSTOMER_ID AND
              PURCHASE.PRODUCT_ID = PRODUCT.PRODUCT_ID
        AND CUSTOMER.Address LIKE '%'||p_City|| '%' AND PURCHASE.Quantity >= p_PQuantity;

    --Start Execution section
BEGIN
    --Open Cursor
    OPEN   cur_Customer; -- open cursor for use
    --loop to display each record returned by cursor
    --Use PL/SQL language control or loop to display each record pointed by cursor
    LOOP
        -- Fetch cursor data
        FETCH   cur_Customer INTO v_Name, v_Address, v_Email,
                           v_Prod_Description, v_Purchase_Qty;
        EXIT WHEN cur_Customer%NOTFOUND;
        --Display each record
        DBMS_OUTPUT.PUT_LINE('Customer info: ' || v_Name || ' | ' || v_Address || ' | '
        || v_Email || ' | ' || v_Prod_Description || ' | ' || v_Purchase_Qty);

    END LOOP;

    CLOSE   cur_Customer; --close cursor
END usp_GetCustomersByCityAndPQty;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The left sidebar displays a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDeveloper1_Connection'. The main area has two tabs: 'Worksheet' and 'Script Output'. The 'Worksheet' tab contains the following PL/SQL code:

```
551   FOR cur_Customer IN v_Cust
552   EXIT WHEN cur_Customer%NOTFOUND;
553   --Display each record
554   DBMS_OUTPUT.PUT_LINE('Customer Info: ' || v_Name || ' | ' || v_Address || ' |
555   || v_Email || ' | ' || v_Prod_Description || ' | ' || v_Purchase_Qty);
556
557   CLOSE cur_Customer; --close cursor
558
559
560 END usp_GetCustomersByCityAndPQty;
561
562
563 --Turning server output on
564 SET SERVEROUTPUT ON;
565
566 -- Executing stored procedure
567 EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn', 3);
568
569
570 EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn');
571
572
573 --EXECUTE user.GetCustomersByCityAndPQty(7);
```

The 'Script Output' tab shows the result of the compilation:

```
Procedure USP_GETCUSTOMERSBYCITYANDPQTY compiled
```

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

- In SQL Developer:

- Option #1 – Passing VALUE as argument to Stored Procedure for CITY = Brooklyn & Quantity = 2:**

- Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- Execute/Compile the Script using the following syntax using parameters

EXECUTE ProcedureName (value1, value2);

- We execute the statement with the following argument for female:

EXECUTE usp_GetCustomersByCityAndPQty ('Brooklyn', 2);

- Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under 'Connections' (OracleXE_DBMS_DBDeveloper1_Connection). The central workspace has a 'Script' tab open with the following PL/SQL code:

```
B64    END LOOP;
B65
B66    CLOSE cur_Customer; --close cursor
B67
B68  END usp_GetCustomersByCityAndPQty;
B69
B70  --Turning server output on
B71  SET SERVEROUTPUT ON;
B72
B73  -- Executing stored procedure passign city and quantiy as values
B74  EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn', 2);
B75
B76
B77  --Turning server output on
B78  SET SERVEROUTPUT OFF;
B79
B80  -- Executing stored procedure passign city and quantiy as values
B81  EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn', 3);
B82
B83
B84
```

Below the code, the 'Script Output' tab shows the results of the execution:

```
PL/SQL procedure successfully completed.

Customer Info: Josephine Smith (Jo) | 111 Glendale Road, Brooklyn NY 11210 | josephine.smith@comp.com | High Definition TV | 2
Customer Info: Nancy Rivera | 555 Metropolitan Avenue, Brooklyn NY 11205 | nrivera@xyz.com | Laptop Computer | 4
```

▪ **Option #2 – Passing VALUE as argument to Stored Procedure for CITY = Brooklyn & Quantity = 3:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax using parameters

EXECUTE ProcedureName (value1, value2);

- 3) We execute the statement with the following argument for female:

EXECUTE usp_GetCustomersByCityAndPQty ('Brooklyn', 3);

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows the 'OracleXE_DBMS_DBDeveloper1_Connection' with various schema objects like Tables, Views, Procedures, Functions, etc. The central workspace has a 'Worksheet' tab open with the following PL/SQL code:

```

--Turning server output on
SET SERVEROUTPUT ON;
-- Executing stored procedure passign city and quantiy as values
EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn', 3);
--Turning server cuput on
SET SERVEROUTPUT ON;
-- Executing stored procedure passign city and quantiy as values
EXECUTE usp_GetCustomersByCityAndPQty('New York', 1);

```

Below the worksheet is a 'Script Output' tab which displays the results of the execution:

```

PL/SQL procedure successfully completed.

Customer Info: Nancy Rivera | 555 Metropolitan Avenue, Brooklyn NY 11205 | nrivera@xyz.com | Laptop Computer | 4

```

A red box highlights the 'Worksheet' tab and the executed code. Another red box highlights the 'Script Output' tab and its contents.

▪ **Option #2 – Passing VALUE as argument to Stored Procedure for CITY = New York & Quantity = 1:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax using parameters

```
EXECUTE ProcedureName (value1, value2);
```

- 3) We execute the statement with the following argument for female:

```
EXECUTE usp_GetCustomersByCityAndPQty('New York', 1);
```

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows the 'OracleXE_DBMS_DBDeveloper1_Connection' node expanded, revealing various database objects like Tables, Views, Procedures, Functions, and Sequences. The 'Procedures' node is selected. In the center, the 'SQL Worksheet' tab is active, containing the following PL/SQL code:

```

865
866
867 --Turning server output on
868 SET SERVEROUTPUT ON;
869
870 -- Executing stored procedure passign city and quantity as values
871 EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn', 3);
872
873 --Turning server output on
874 SET SERVEROUTPUT ON;
875
876 -- Executing stored procedure passign city and quantity as values
877 EXECUTE usp_GetCustomersByCityAndPQty('New York', 1);
878
879 -- Executing stored procedure passign variables with city & quantity
880
881 -- Remember, to create variables, assign variables you need to use a
882 -- BLOCK OF CODE in a SCRIPT - thus we need DECLARF BEGIN END statement.
883
884
885
886

```

The 'Script Output' tab at the bottom shows the execution results:

```

PL/SQL procedure successfully completed.

Customer Info: Angel Rodriguez | 222 Park Avenue, New York, NY 10030 | arod@xyz.com | High Power Washing Machine | 1
Customer Info: Angel Rodriguez | 222 Park Avenue, New York, NY 10030 | arod@xyz.com | High Power Dryer Machine | 1

```

- **Option #3 – Passing VARIABLE as argument to Stored Procedure for CITY = Brooklyn & Quantity = 2:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) **RECAP!** – Recall that to declare variables, assign variables & use variables in your SCRIPTS, you need to use a BLOCK OF CODE:

```

DECLARE
    -- Declaration section

BEGIN
    -- Execution section (Assign variable, execute code)

END ;

```

- 3) **NEW!** – when executing or calling a Stored Procedure from within a BLOCK OF CODE in your Script, you **DO NOT NEED** the **KEYWORD EXECUTE**! Therefore, from WITHIN A BLOCK OF CODE the following syntax using parameters & passing a VARIABLE as argument is as follows:

ProcedureName (variable1, variable2);

- 4) We create a variable to store the value of the gender:

```
v_City      VARCHAR2(20) ;
```

- 5) We assign the variable the value of the gender:

```
v_Qty      NUMBER(5,0) ;
```

- 6) We execute the statement with the following argument for female:

```
usp_GetCustomersByCityAndPQty(v_City,v_Qty);
```

- 7) Put it all in a BLOCK OF CODE:

```

DECLARE
    -- Declaration section
    v_City      VARCHAR2(20) ;
    v_Qty      NUMBER(5,0) ;

BEGIN
    -- Execution section (Assign variable, execute code)
    v_City := 'Brooklyn';
    v_Qty := 2;

    usp_GetCustomersByCityAndPQty(v_City,v_Qty);

END ;

```

8) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays database connections and objects. The main area has a red box around the 'Worksheet' tab where a PL/SQL script is being run. Another red box highlights the 'Script Output' tab at the bottom, which shows the execution results.

```
896
897
898 --- Executing stored procedure passign variables with city & quantity
899 --- Remember, to create variables, assign variables you need to use a
900 --- BLOCK OF CODE in a SCRIPT, thus we need DECLARE, BEGIN, END statement
901
902
903 --Turning server ouput on
904 SET SERVEROUTPUT ON;
905
906 DECLARE
907   v_City      VARCHAR2(20);          --Declare variable
908   v_Qty       NUMBER(5,0);          --Declare variable
909
910 BEGIN
911   v_City := 'Brooklyn';           --Assigned variable
912   v_Qty := 2;                   --Assigned variable
913
914   -- New Syntax = EXECUTE keyword NOT NEEDED when calling from
915   -- within BLOCK OF CODE
916   usp_GetCustomersByCityAndPQty(v_City,v_Qty);    --Pass variables as arguments
917
918 END;
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
900
```

PL/SQL procedure successfully completed.

Customer Info: Josephine Smith (Jo) | 111 Glendwood Road, Brooklyn NY 11210 | josephine.smith@comp.com | High Definition TV | 2
Customer Info: Nancy Rivera | 555 Metropolitan Avenue, Brooklyn NY 11205 | nrivera@xyz.com | Laptop Computer | 4

- **Option #3 – Passing VARIABLE as argument to Stored Procedure for CITY = New York & Quantity = 1:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) **RECAP!** – Recall that to declare variables, assign variables & use variables in your SCRIPTS, you need to use a BLOCK OF CODE:

```

DECLARE
    -- Declaration section

BEGIN
    -- Execution section (Assign variable, execute code)

END ;

```

- 3) **NEW!** – when executing or calling a Stored Procedure from within a BLOCK OF CODE in your Script, you **DO NOT NEED** the **KEYWORD EXECUTE**! Therefore, from WITHIN A BLOCK OF CODE the following syntax using parameters & passing a VARIABLE as argument is as follows:

ProcedureName (variable1, Variable2);

- 4) We create a variable to store the value of the gender:

```
v_City      VARCHAR2(20);
```

- 5) We assign the variable the value of the gender:

```
v_Qty      NUMBER(5,0);
```

- 6) We execute the statement with the following argument for female:

```
usp_GetCustomersByCityAndPQty(v_City,v_Qty);
```

- 7) Put it all in a BLOCK OF CODE:

```

DECLARE
    -- Declaration section
    v_City      VARCHAR2(20);
    v_Qty      NUMBER(5,0);

BEGIN
    -- Execution section (Assign variable, execute code)
    v_City := New York;
    v_Qty := 1;

    usp_GetCustomersByCityAndPQty(v_City,v_Qty);

END ;

```

8) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left is the Connections tree, showing a single connection named "OracleXE_DBMS_DBDDeveloper1_Connection". The main workspace contains a "Worksheet" tab with the following PL/SQL code:

```
910 . . .
911 . . .
912 -- Executing stored procedure passign variables with city & quantity
913 -- Remember, to create variables, assign variables you need to use a
914 -- BLOCK OF CODE in a SCRIPT, thus we need DECLARE, BEGIN, END statement
915
916 --Turning server output on
917 SET SERVEROUTPUT ON;
918
919 DECLARE
920   v_City      VARCHAR2(20);          --Declare variable
921   v_Qty       NUMBER(5,0);          --Declare variable
922
923 BEGIN
924   v_City := 'New York';           --Assigned variable
925   v_Qty := 1;                   --Assigned variable
926
927   -- New Syntax = EXECUTE keyword NOT NEEDED when calling from
928   -- within BLOCK OF CODE
929   --usp_GetCustomersByCityAndPQty(v_City,v_Qty);           --Pass variables as arguments
930
931   usp_GetCustomersByCityAndPQty(v_City,v_Qty);
932
933 END;
```

The "Script Output" tab at the bottom shows the execution results:

```
PL/SQL procedure successfully completed.

Customer Info: Angel Rodriguez | 222 Park Avenue, New York, NY 10030 | arod@xyz.com | High Power Washing Machine | 1
Customer Info: Angel Rodriguez | 222 Park Avenue, New York, NY 10030 | arod@xyz.com | High Power Dryer Machine | 1
```

Example 4 – Stored Procedure that DISPLAYS the Name, address, phone & email of all customers who HAVE NOT purchased any products. Business objectives is to give these customers a special promotional discount as incentive to buy our products and become first time customers

Target Code to Implement using Cursor

- This example we display the name & gender of customers born after a date.
- NOTE – this type of query is usually BEST PRACTICE or implemented in real applications because this stored procedure will return the records FOR ANY CITY & PURCHASE AMOUNT!**
- The target query & results:
 - Target SELECT QUERY using JOIN & LIKE statement to search for a string in a column:

```
--Select Query with WHERE clause
SELECT CUSTOMER.Name, CUSTOMER.Address, CUSTOMER.Phone, CUSTOMER.Email
FROM CUSTOMER
WHERE NOT EXISTS(SELECT PURCHASE.Customer_ID
                  FROM PURCHASE
                  WHERE PURCHASE.Customer_ID = CUSTOMER.Customer_ID);
```

- Expected OUTPUT is shown below for customer who live in **BROOKLYN** & purchase quantity \geq to **2** products

```
-- Script using CURSOR to display the following messages
The following Customers have NOT purchased our products. Send them a
promotional offer:
Customer Info: Frank Hum | 666 74th Street, Flushing NY 1134 | 718 666-
6666 | fhum@comp.com
Customer Info: Mary Jones | 333 Flatlands Avenue, Brooklyn NY 11203 |
718 333-3333 | mjones@xyz.com
Customer Info: Michael Triolo | 777 Church Avenue, Brooklyn NY 11201 |
718 777-7777 | mtriolo@xyz.com
```

Implementation Steps

Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
--Stored Procedure that DISPLAYS the Name, address, phone & email of all customers who  
--Have NOT purchased any products  
--Business objectives is to give these customers a special promotional discount as  
--incentive to buy our products and become first time customers
```

```
CREATE OR REPLACE PROCEDURE usp_GetCustomersNoPurchase  
IS
```

```
--declare variable needed for SELECT INTO statement  
  
v_Name          VARCHAR2(50);  
v_Address       VARCHAR2(200);  
v_Email         VARCHAR2(50);  
v_Phone         VARCHAR2(20);  
  
--Declare Cursor  
CURSOR cur_Customer IS  
  
    -- Query cursor will point to results  
SELECT CUSTOMER.Name, CUSTOMER.Address, CUSTOMER.Phone, CUSTOMER.Email  
FROM CUSTOMER  
WHERE NOT EXISTS(SELECT PURCHASE.Customer_ID  
                  FROM PURCHASE  
                  WHERE PURCHASE.Customer_ID = CUSTOMER.Customer_ID);  
  
--Start Execution section  
  
BEGIN  
    --Display a message  
    DBMS_OUTPUT.PUT_LINE('The following Customers have NOT purchased our products.  
                         Send them a promotional offer: ');  
    --Open Cursor  
    OPEN   cur_Customer; -- open cursor for use  
    --loop to display each record returned by cursor  
    --Use PL/SQL language control or loop to display each record pointed by cursor  
    LOOP  
        -- Fetch cursor data  
        FETCH   cur_Customer INTO v_Name, v_Address, v_Phone, v_Email;  
        EXIT WHEN cur_Customer%NOTFOUND;  
        --Display each record  
        DBMS_OUTPUT.PUT_LINE('Customer info: ' || v_Name || ' | ' || v_Address || ' | '  
                           || v_Phone || ' | ' || v_Email);  
  
    END LOOP;  
  
    CLOSE   cur_Customer; --close cursor  
  
END usp_GetCustomersNoPurchase;
```

Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- In SQL Developer:

- In the script, do the following:
 - 1) Enter code in script
 - 2) Execute/Compile the Script
 - 3) Verify Procedure was successfully compiled
- Steps shown below:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays a tree view of database objects under 'Connections' for 'OracleXE_DBMS_DBDeveloper1_Connection'. The main workspace has two tabs: 'Stored Procedures & Functions Scripts.sql' and 'DML Test Queries1.sql'. The 'Stored Procedures & Functions Scripts.sql' tab contains the following PL/SQL code:

```

1001 --> PROCEDURE USP_GETCUSTOMERSNOPURCHASE IS
1002   --> V_VIPFORLINE ARE FOLLOWING CUSTOMERS HAVE NOT PURCHASED OUR PRODUCTS. SEND THEM A PROMOTIONAL OFFER.
1003   CURSOR cur_Customer FOR
1004     SELECT * FROM CUSTOMER WHERE CustomerID NOT IN (SELECT CustomerID FROM PURCHASE);
1005   v_Name VARCHAR2(50), v_Address VARCHAR2(100), v_Phone NUMBER(10,0), v_Email VARCHAR2(50);
1006   BEGIN
1007     OPEN cur_Customer; -- open cursor for use
1008     LOOP
1009       FETCH cur_Customer INTO v_Name, v_Address, v_Phone, v_Email;
1010       EXIT WHEN cur_Customer%NOTFOUND;
1011       DBMS_OUTPUT.PUT_LINE('Customer Info: ' || v_Name || ' ' || v_Address || ' ' ||
1012                             v_Phone || ' ' || v_Email);
1013     END LOOP;
1014
1015     CLOSE cur_Customer; --close cursor
1016   END;
1017
1018   END usp_GetCustomersNoPurchase;

```

The 'Script Output' window at the bottom shows the message: 'Procedure USP_GETCUSTOMERSNOPURCHASE compiled'.

Step 3: In the Oracle SQL Developer Navigator Window, Verify Schema Object was created for Stored Procedure

- In SQL Developer:

- 1) In the Navigator Window select Procedure object.
- 2) Verify Schema Object exists



Step 4: In the Oracle SQL Developer Script Windows TEST the Stored Procedure by EXECUTING it

□ In SQL Developer:

- **Calling this stored procedure that takes no parameters:**

- 1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
- 2) Execute/Compile the Script using the following syntax using parameters

EXECUTE ProcedureName;

- 3) We execute the statement with the following argument for female:

EXECUTE usp_GetCustomersNoPurchase;

- 4) Message Window should indicate if execution was successful & show output of results

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows two connections: 'OracleXE_DBMS_DBDeveloper1_Connection' and 'OracleXE_DBMS_System_Connection'. The main area has a 'Worksheet' tab open with the following PL/SQL script:

```
1013 OPEN cur_Customer; -- open cursor for use
1014   --loop to display each record returned by cursor
1015   --Use PL/SQL language control or loop to display each record pointed by cursor
1016 LOOP
1017   -- Fetch cursor data
1018   FETCH cur_Customer INTO v_Name, v_Address, v_Phone, v_Email;
1019   EXIT WHEN cur_Customer%NOTFOUND;
1020   --Display each record
1021   DBMS_OUTPUT.PUT_LINE('Customer Info: ' ||v_Name|| ' | ' ||v_Address|| ' |
1022   ||v_Phone|| ' | ' ||v_Email);
1023 END LOOP;
1024
1025 CLOSE cur_Customer; --close cursor
1026
1027 END usp_GetCustomersNoPurchase;
1028
1029
1030
1031 --Turning server output on
1032 SET SERVEROUTPUT ON;
1033
1034 -- Executing stored procedure passign city and quantiy as values
1035 EXECUTE usp_GetCustomersNoPurchase;
1036
1037
```

The 'Script Output' tab at the bottom displays the results of the execution:

```
The following Customers have NOT purchased our products. Send them a promotional offer:
Customer Info: Frank Hum | 666 74th Street, Flushing NY 1134 | 718 666-6666 | fhum@comp.com
Customer Info: Mary Jones | 333 Flatlands Avenue, Brooklyn NY 11203 | 718 333-3333 | mjones@xyz.com
Customer Info: Michael Triolo | 777 Church Avenue, Brooklyn NY 11201 | 718 777-7777 | mtriolo@xyz.com
```