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

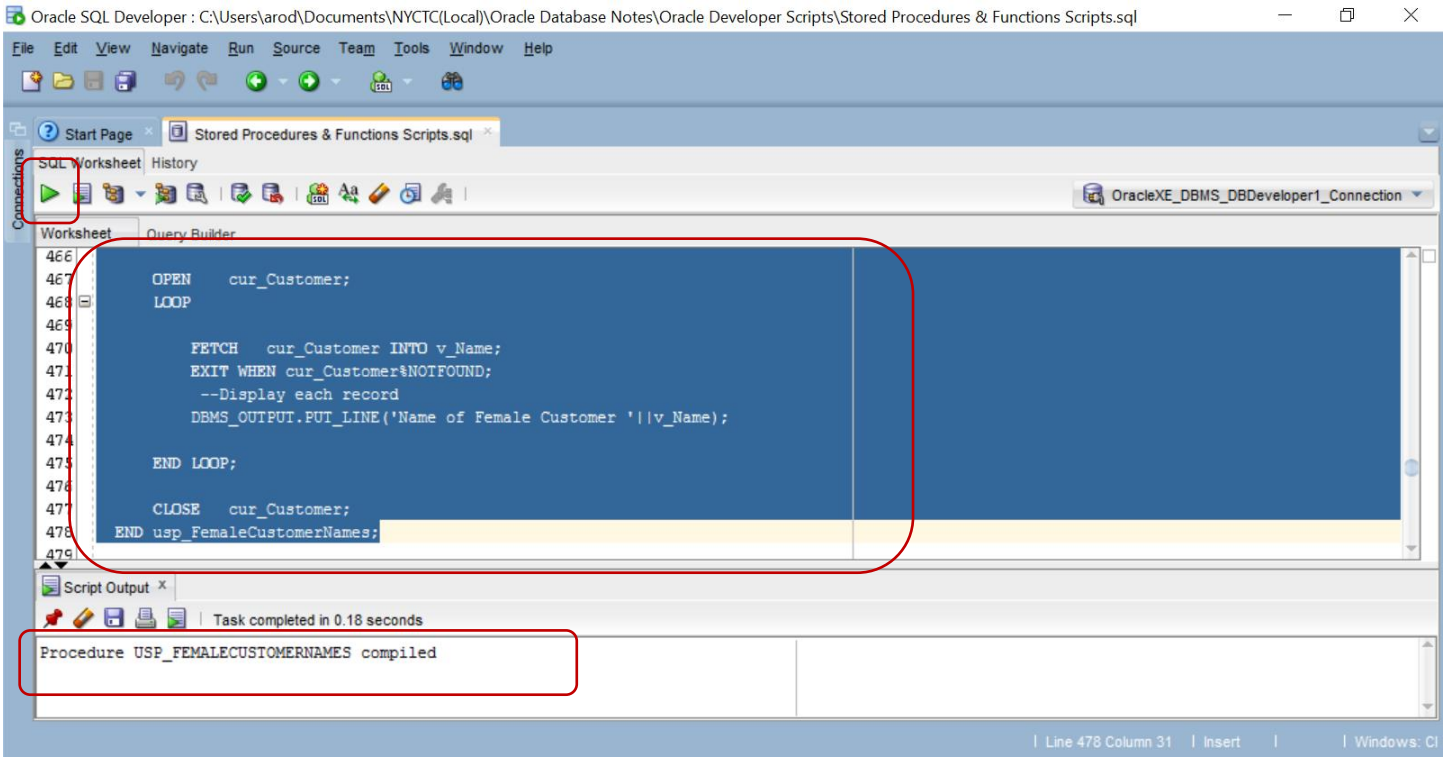
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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;
```

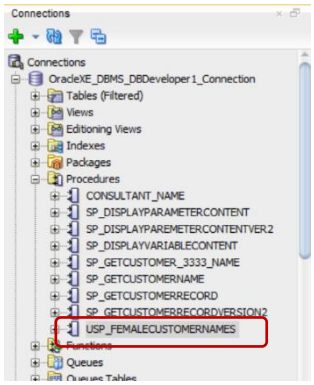
Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- 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:



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

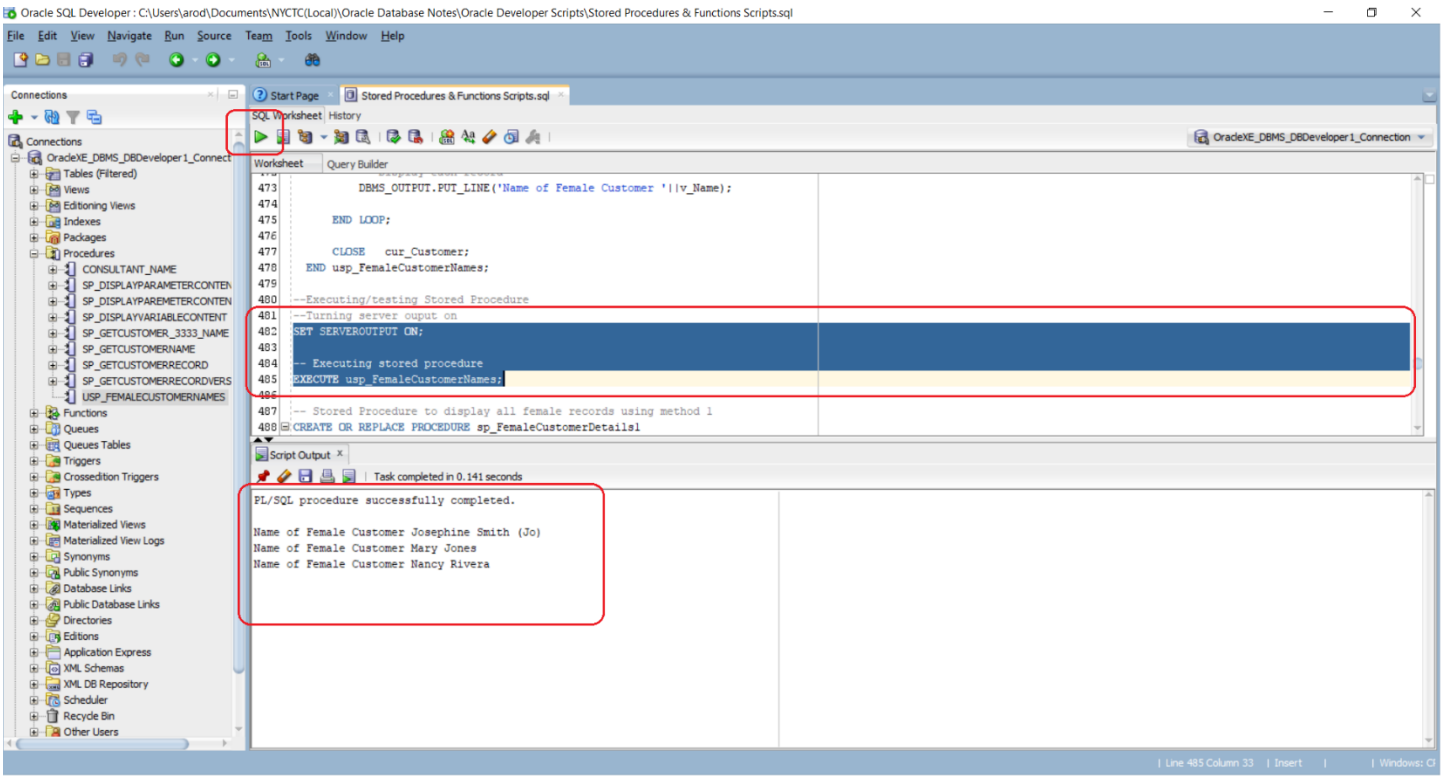
2) Execute/Compile the Script using the following syntax

```
EXECUTE ProcedureName;
```

3) The statement looks as follows:

```
EXECUTE sp_FemaleCustomerNames;
```

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



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

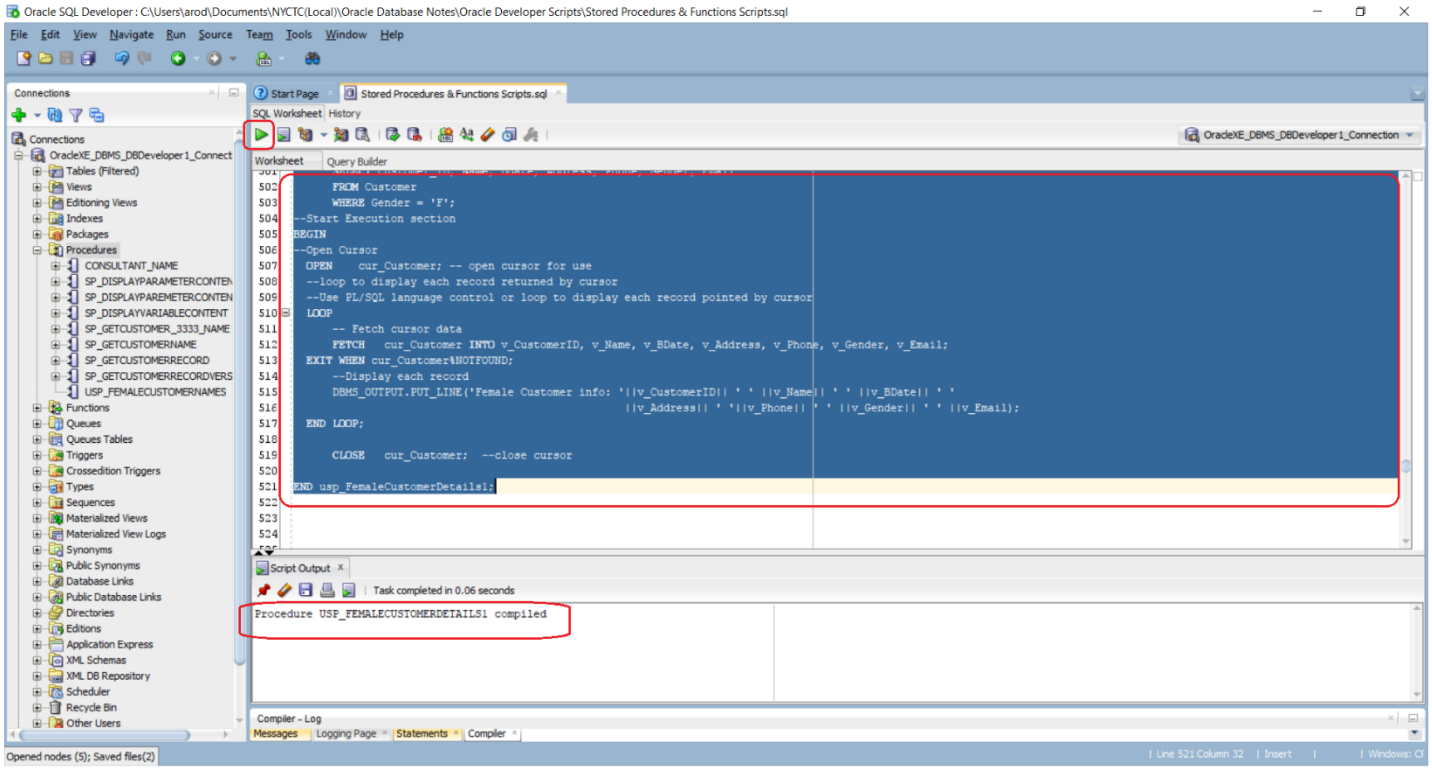
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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:



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

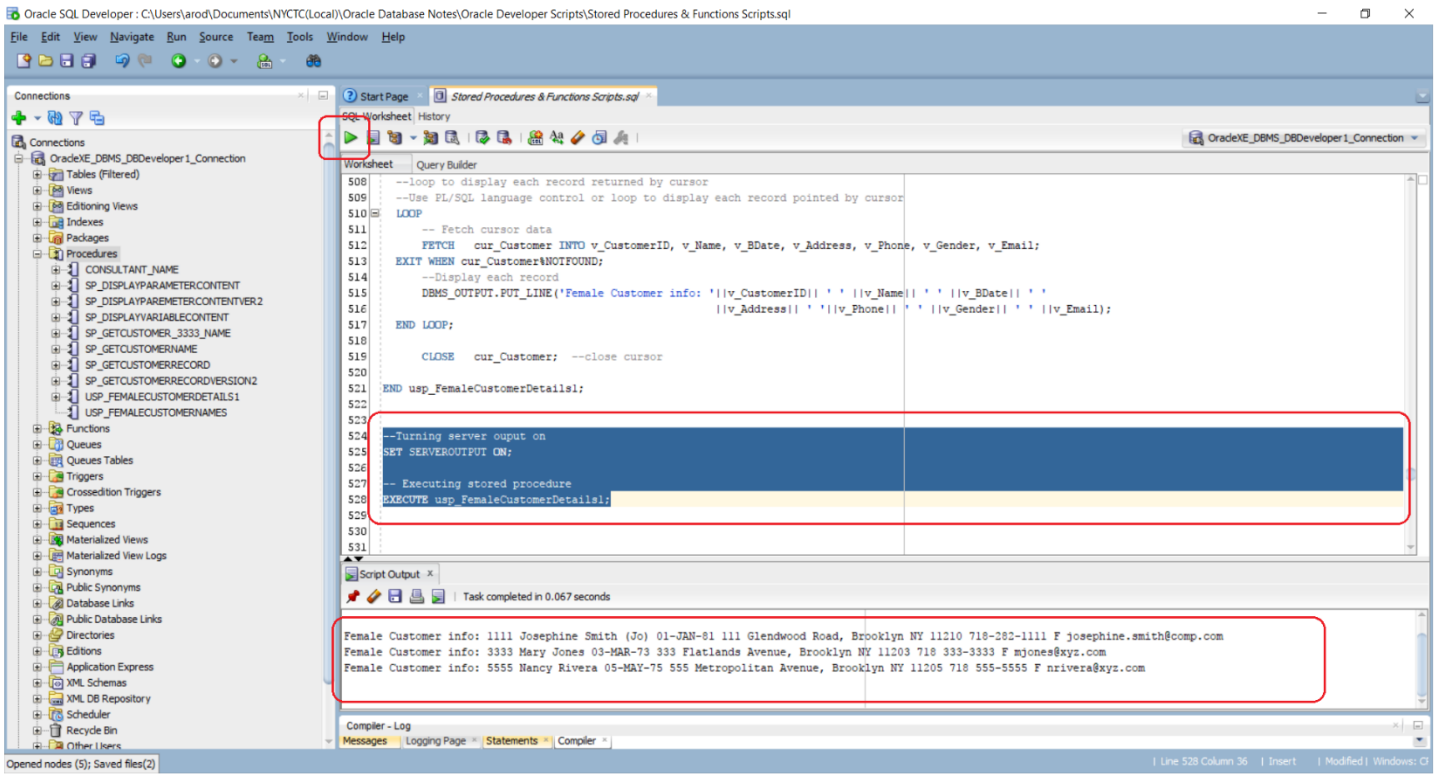
2) Execute/Compile the Script using the following syntax

```
EXECUTE ProcedureName;
```

3) The statement looks as follows:

```
EXECUTE sp_FemaleCustomerDetails1;
```

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



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

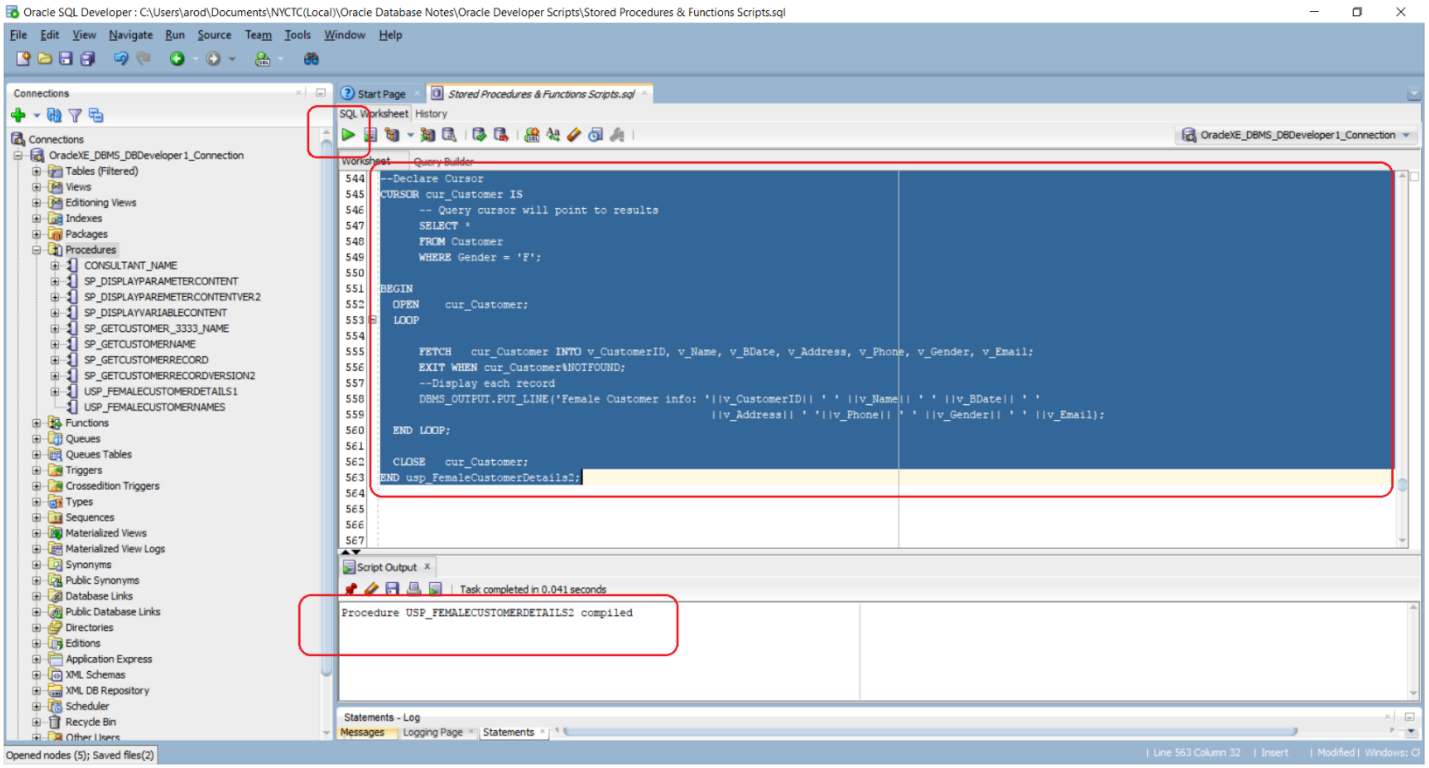
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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;
```

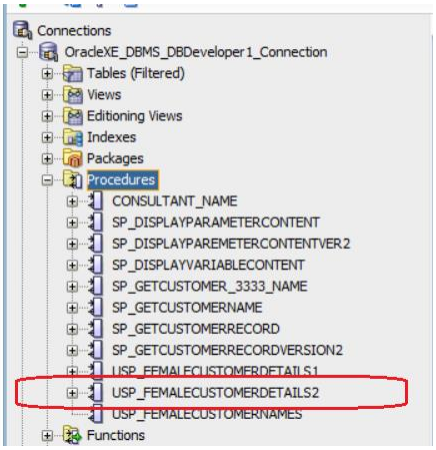
Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- 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:



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

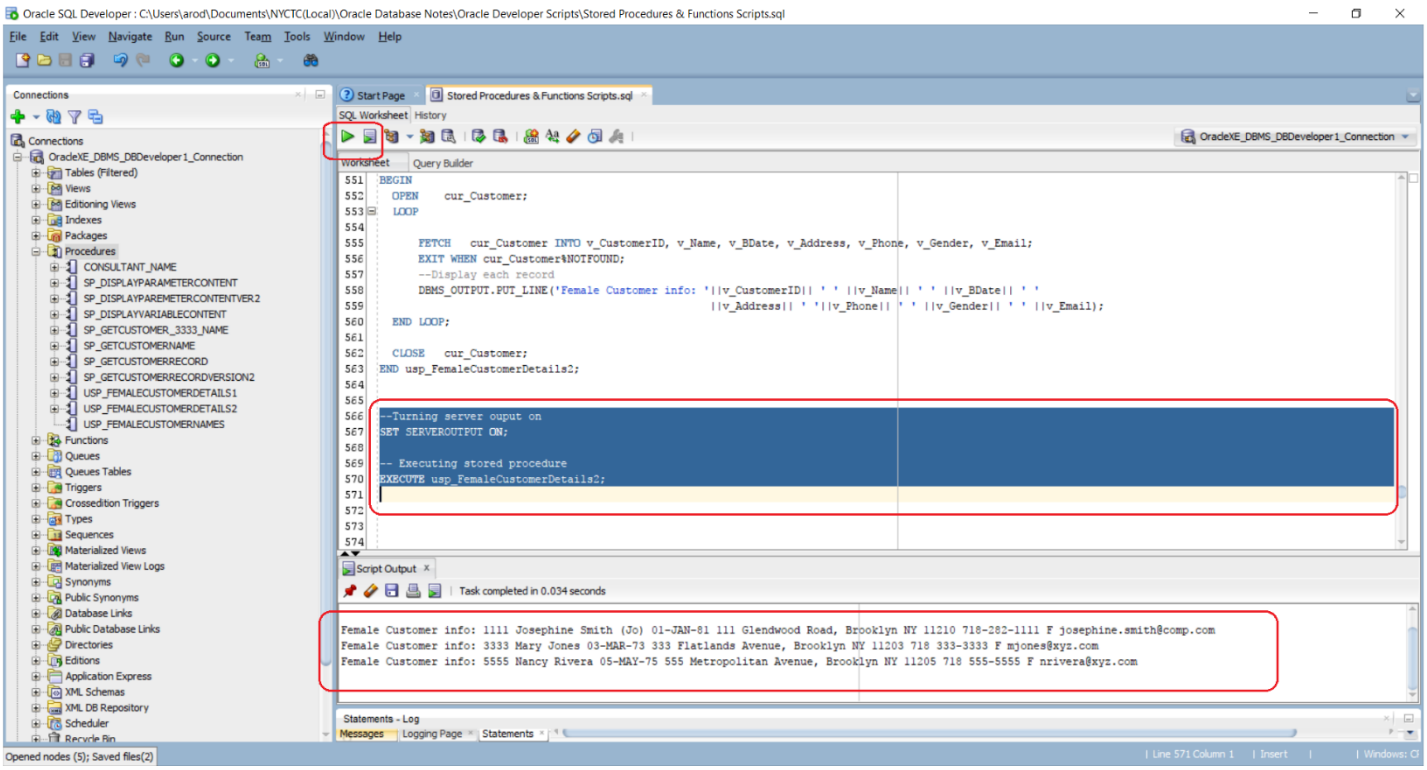
2) Execute/Compile the Script using the following syntax

```
EXECUTE ProcedureName;
```

3) The statement looks as follows:

```
EXECUTE sp_FemaleCustomerDetails2;
```

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



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

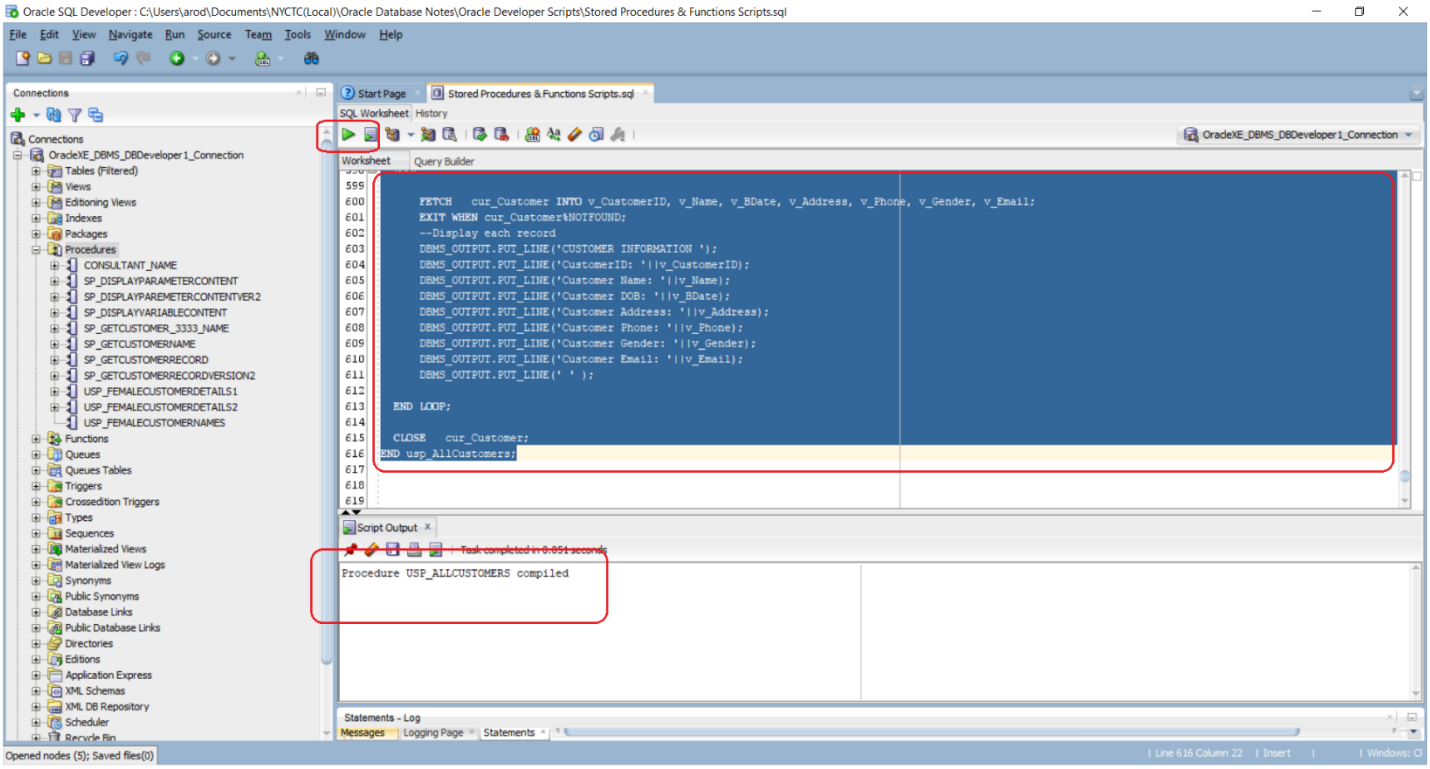
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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;
```

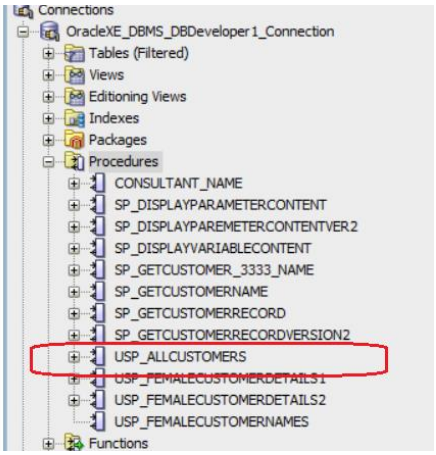
Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- 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:



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

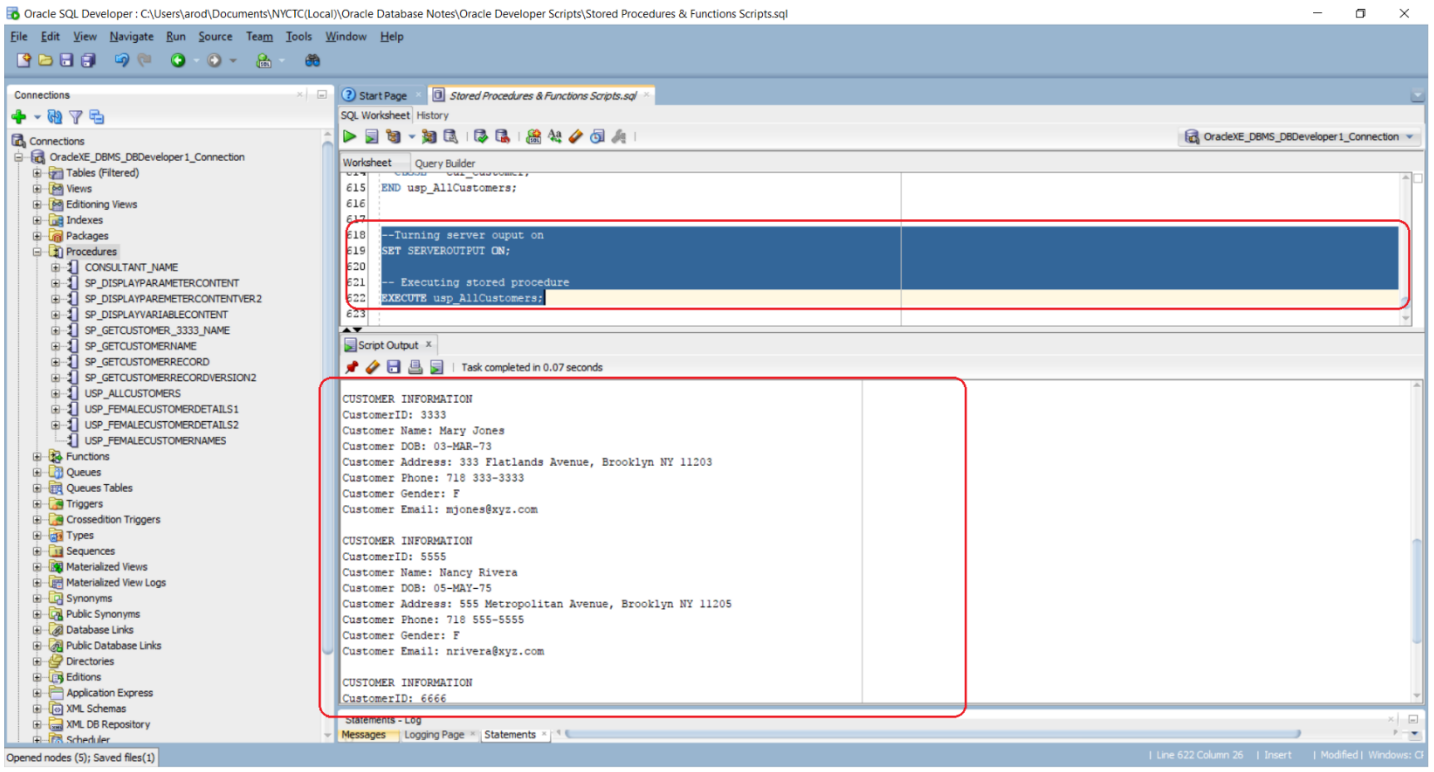
2) Execute/Compile the Script using the following syntax

```
EXECUTE ProcedureName;
```

3) The statement looks as follows:

```
EXECUTE usp_AllCustomers;
```

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



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) = ParemeterName 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

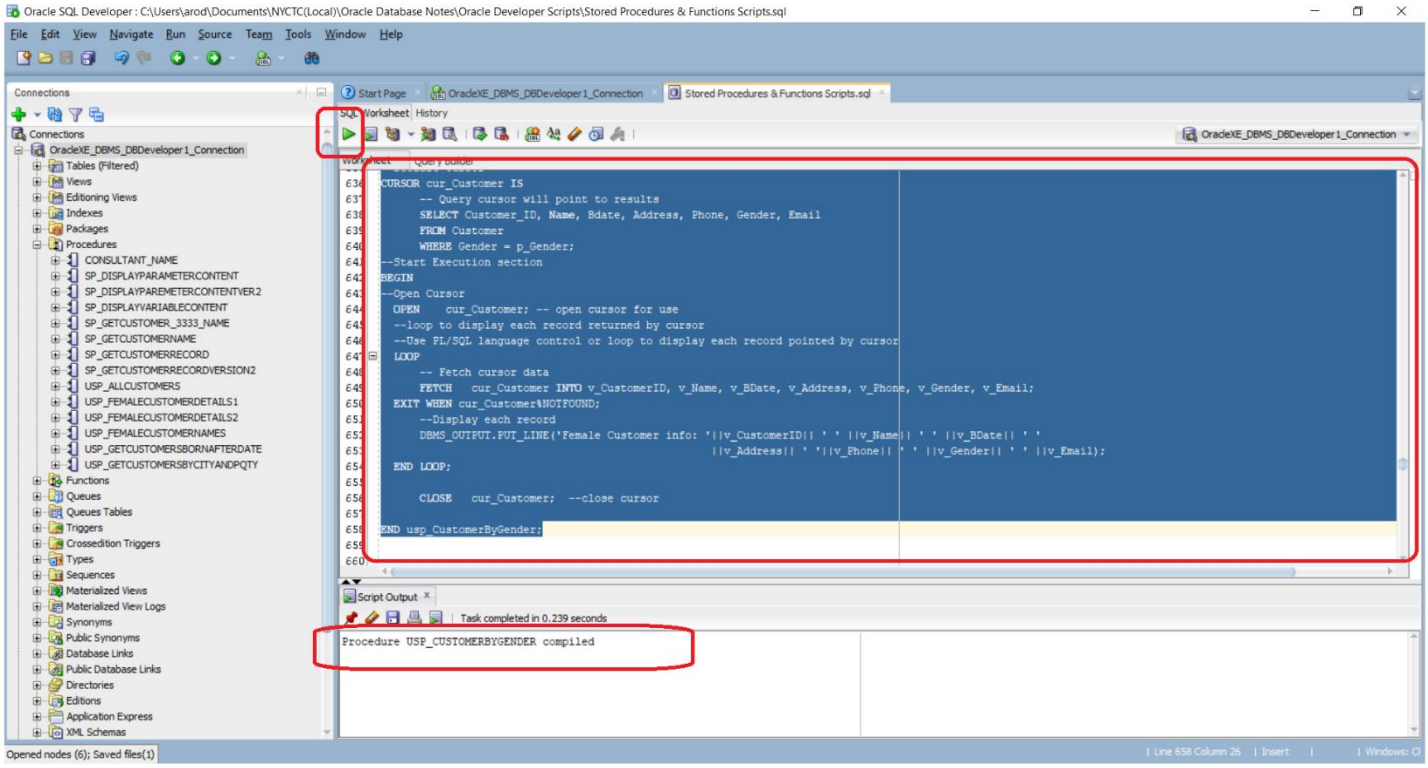
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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;
```

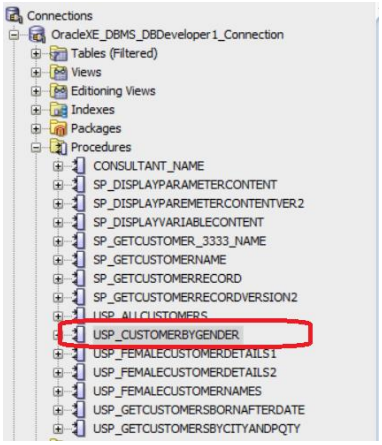
Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

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



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 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

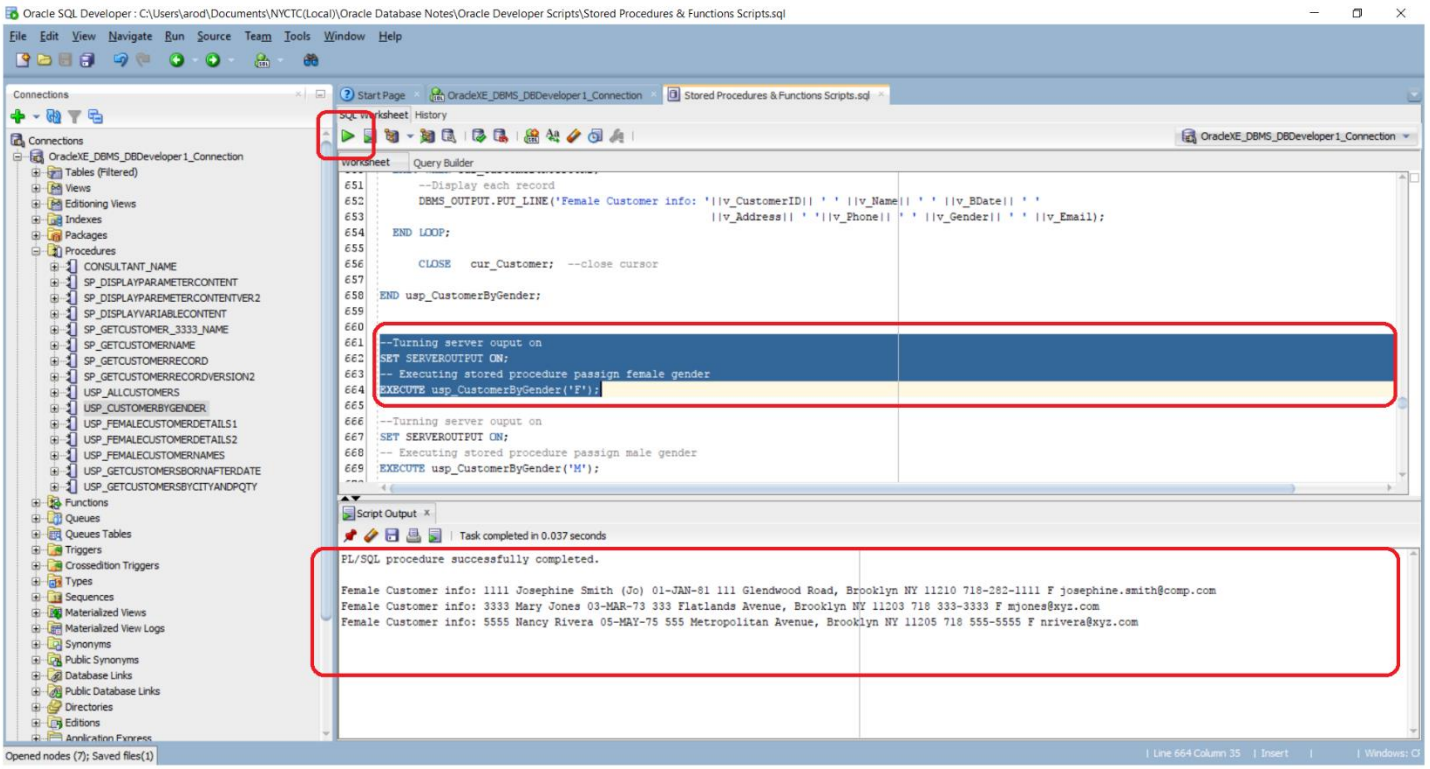
    2) Execute/Compile the Script using the following syntax using parameters

```
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

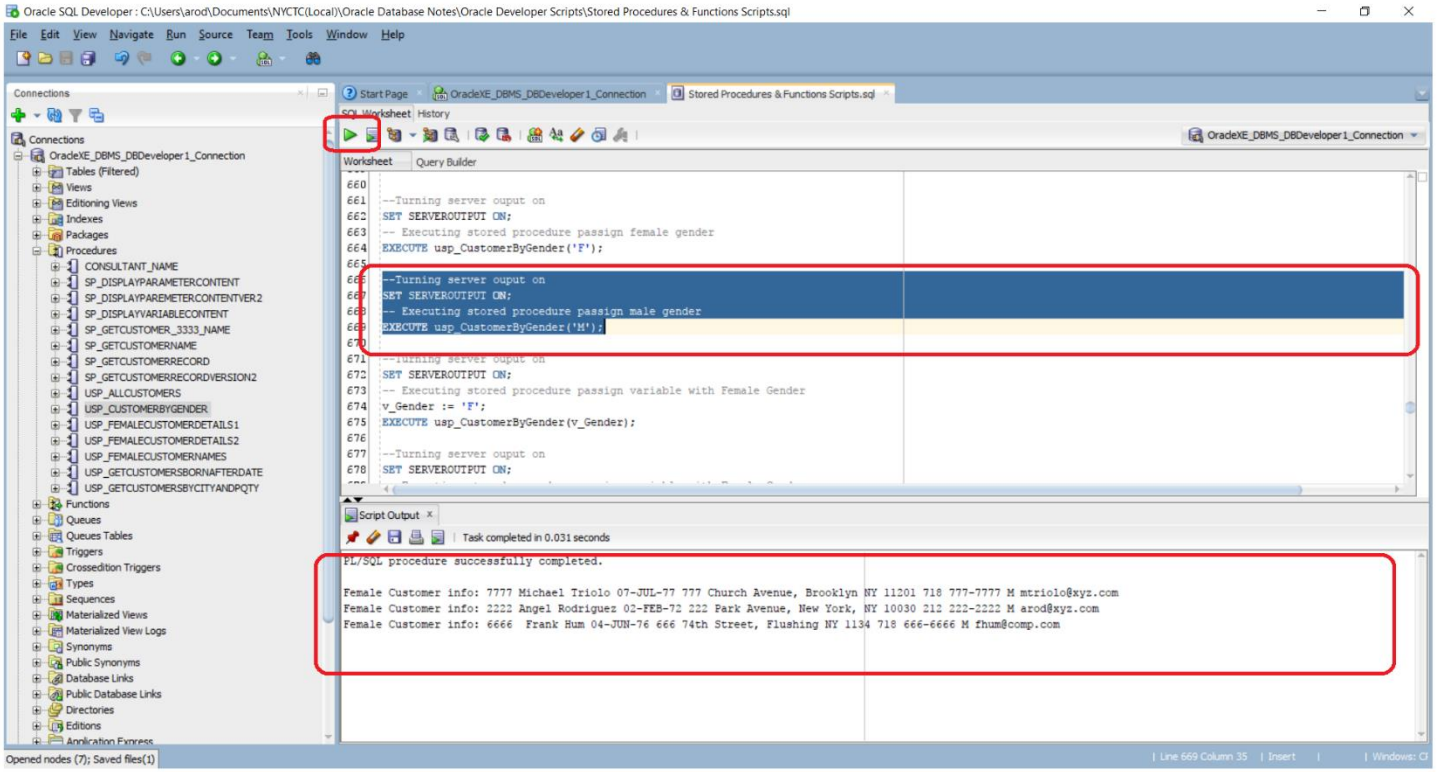
2) Execute/Compile the Script using the following syntax using parameters

```
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



- **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';
```

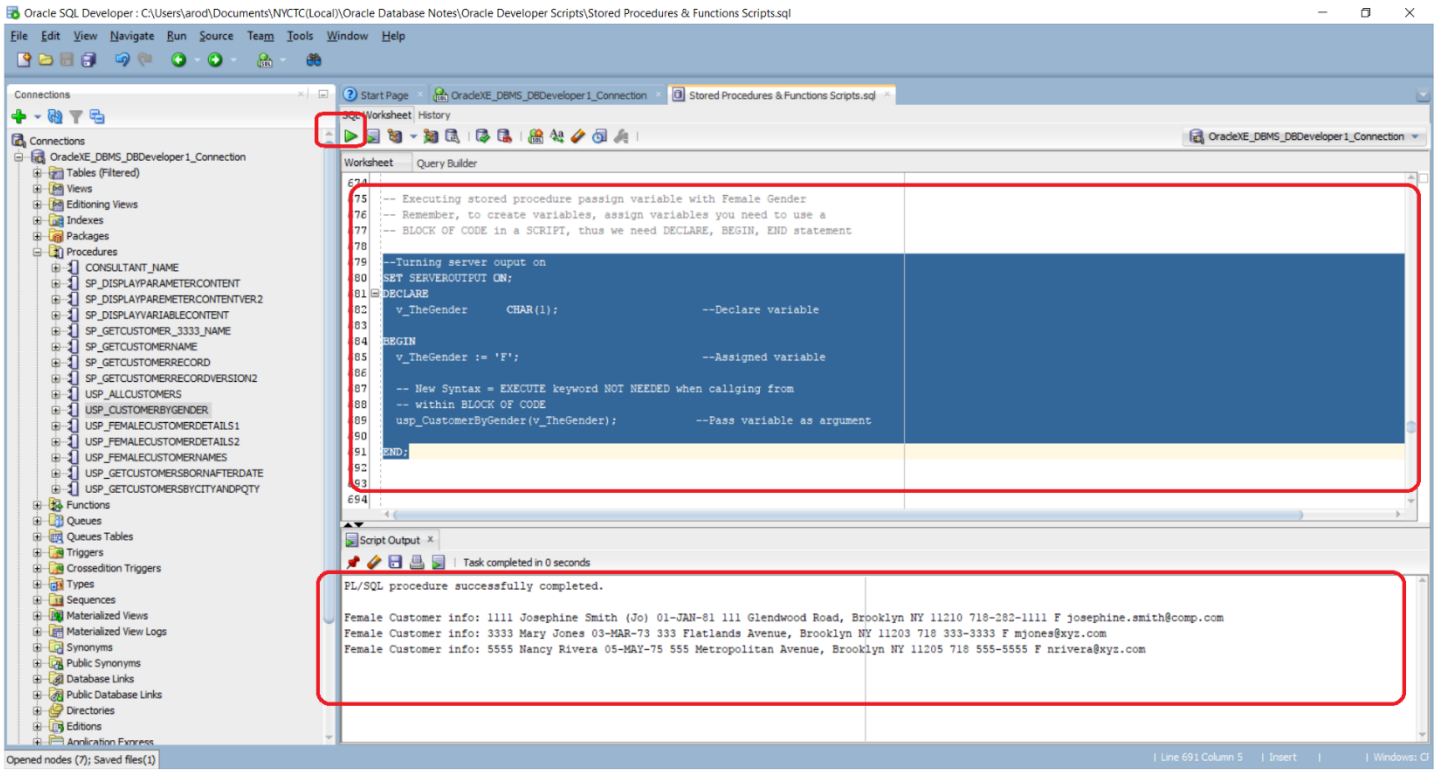
6) We execute the statement with the following argument for female:

```
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



- **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';
```

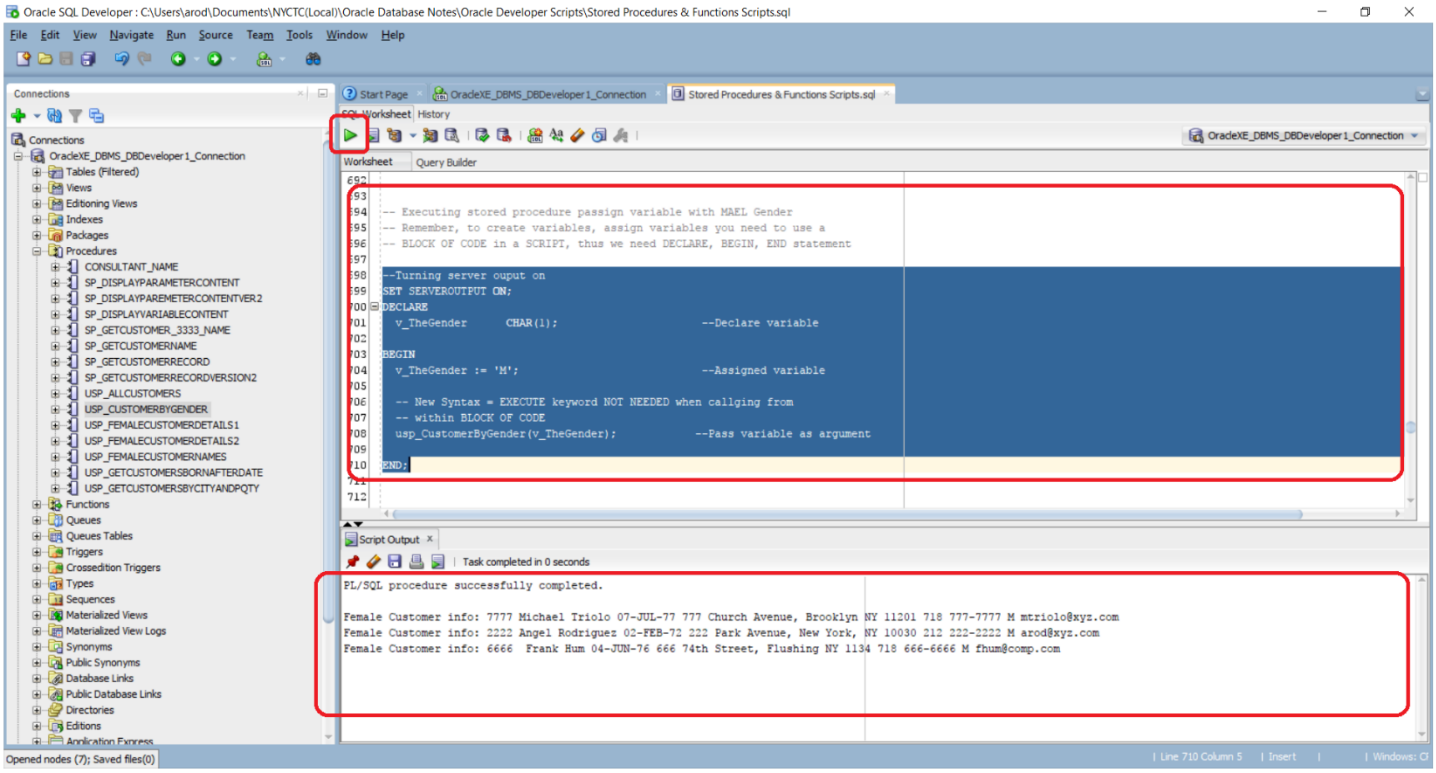
6) We execute the statement with the following argument for female:

```
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



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

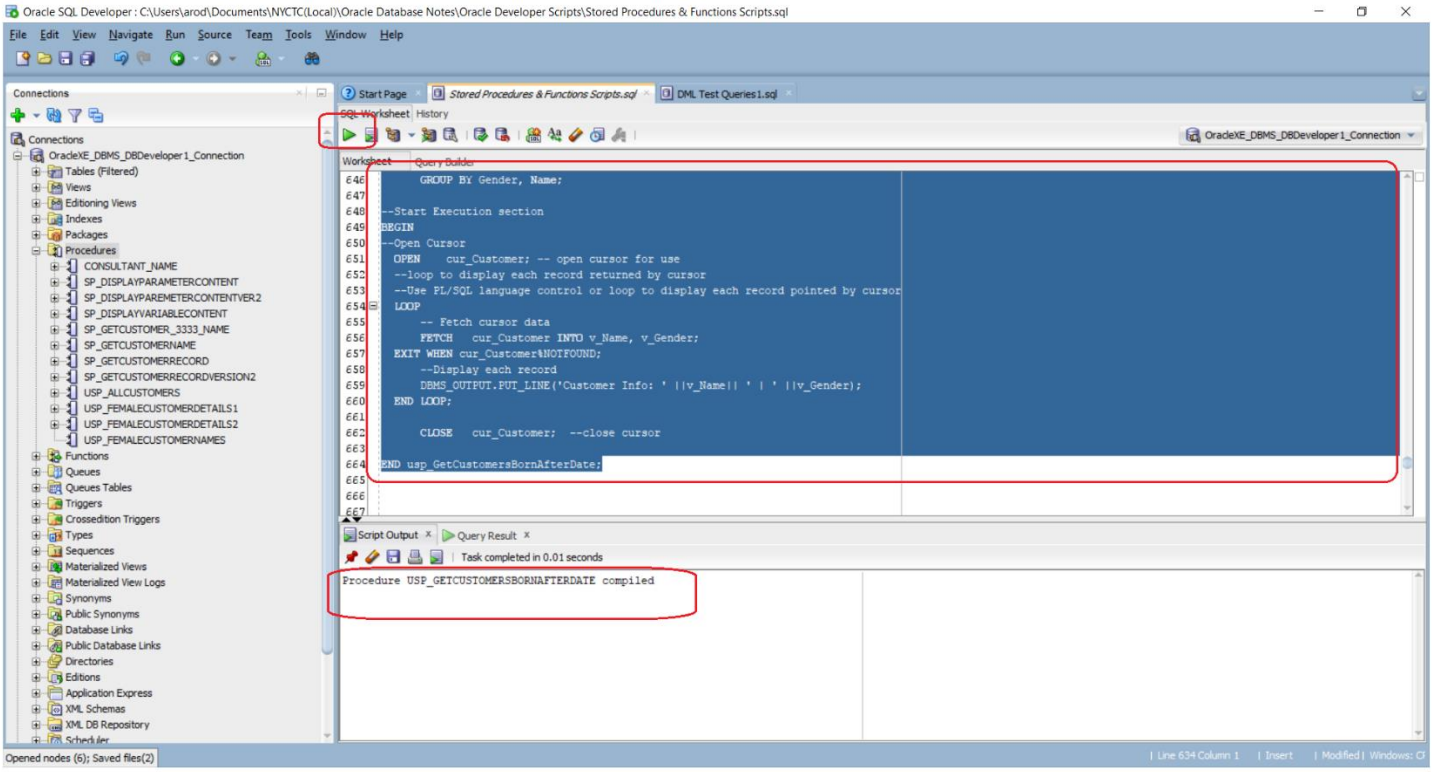
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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:



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:**

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

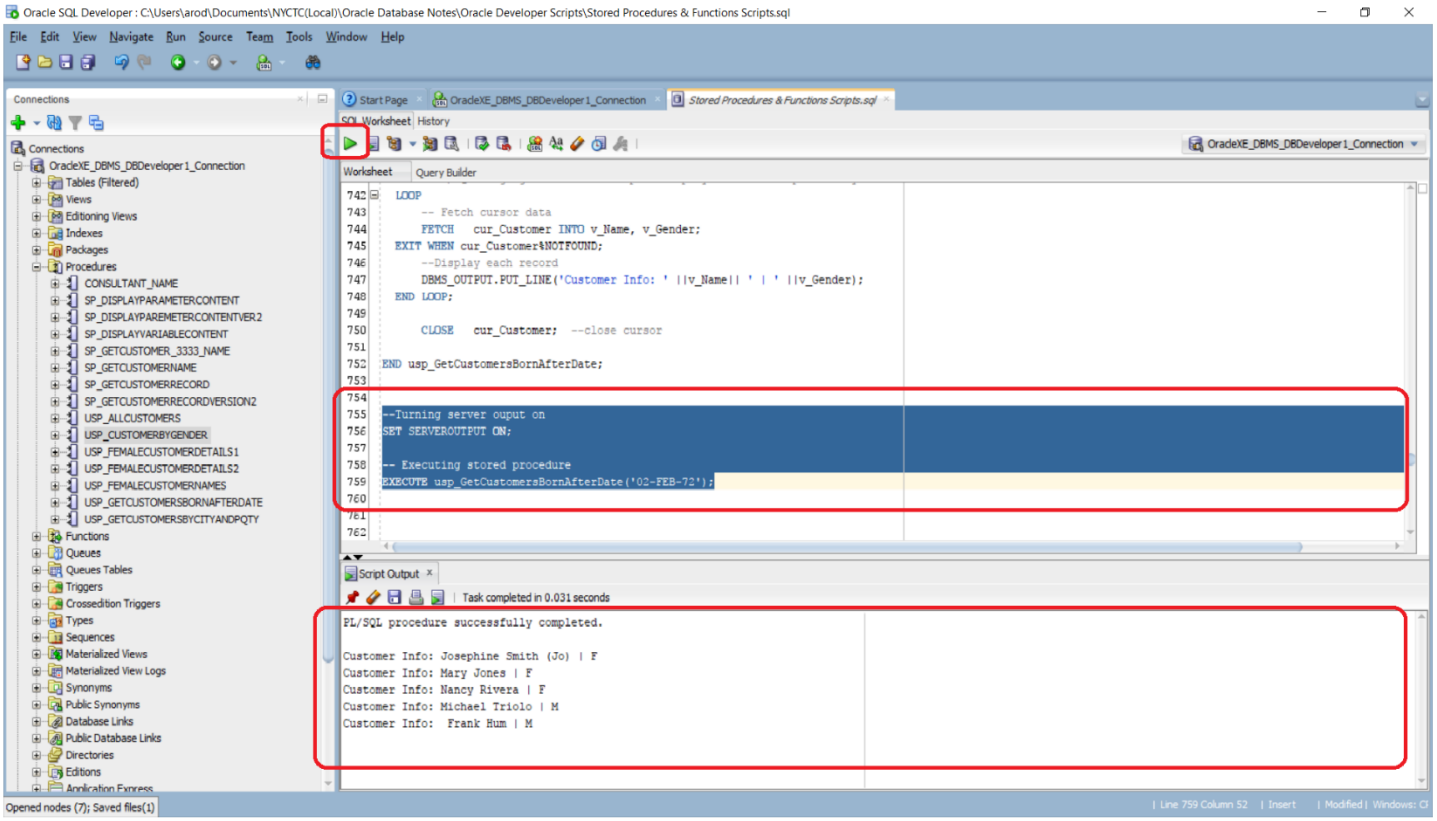
2) Execute/Compile the Script using the following syntax using parameters

```
EXECUTE ProcedureName(value);
```

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

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

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



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

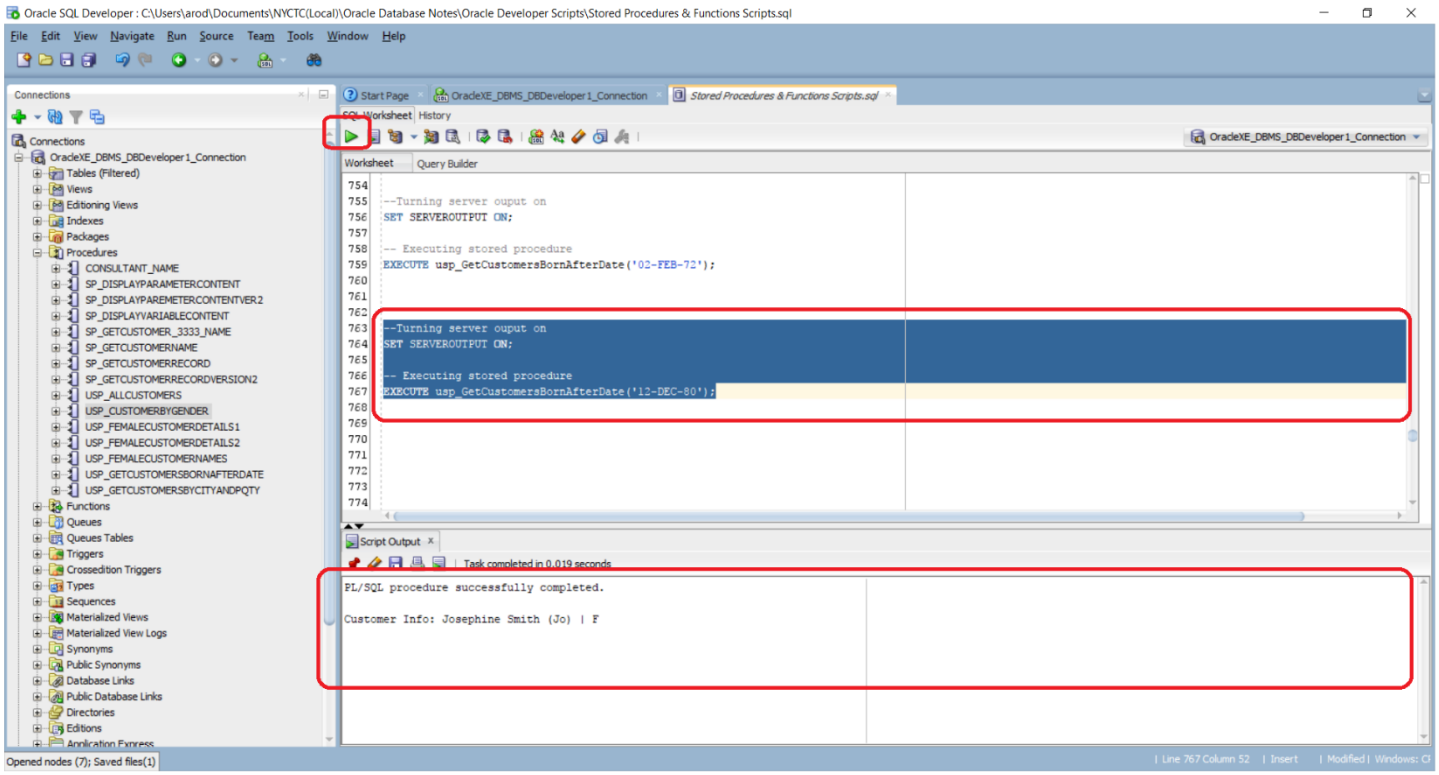
1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
2) Execute/Compile the Script using the following syntax using parameters

```
EXECUTE ProcedureName(Value);
```

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

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

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



- **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';
```

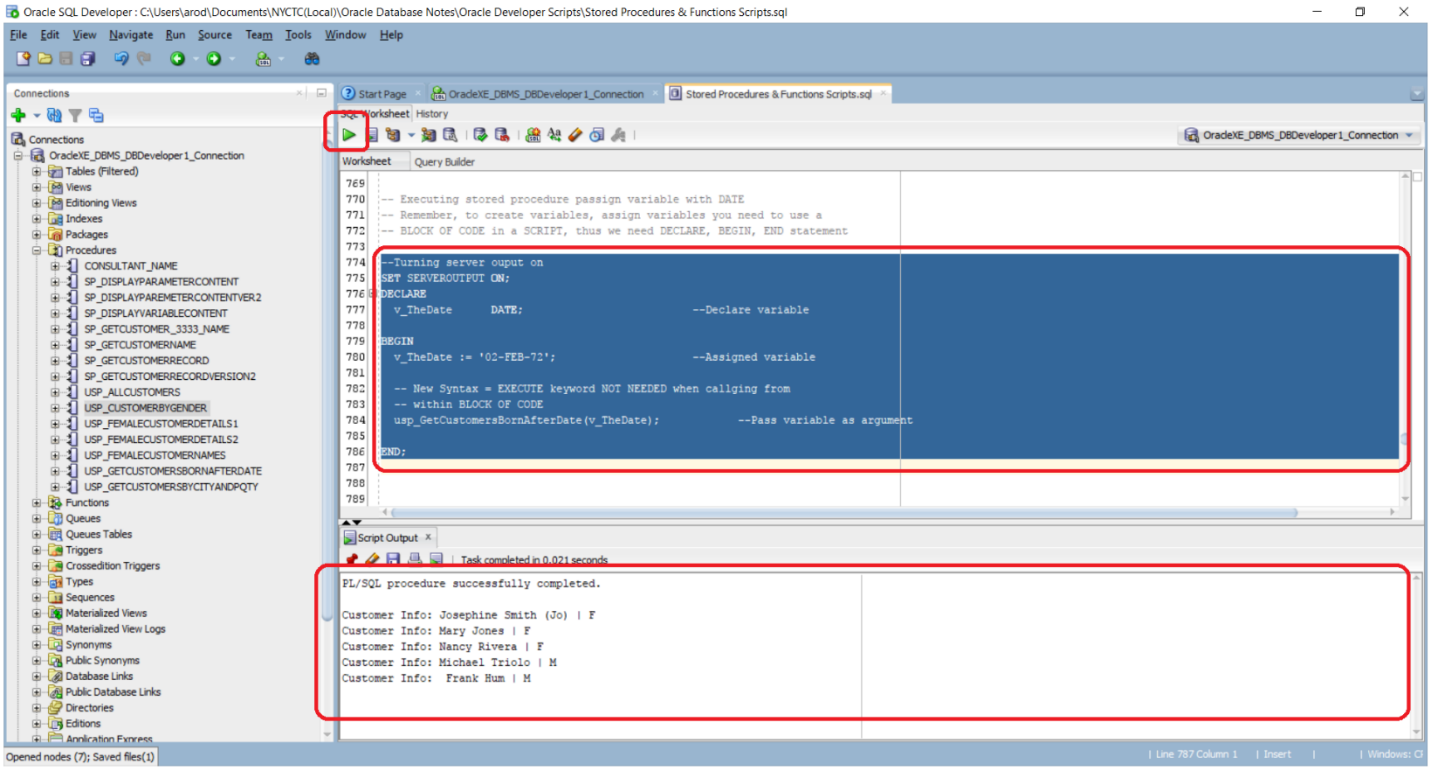
6) We execute the statement with the following argument for female:

```
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



- **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';
```

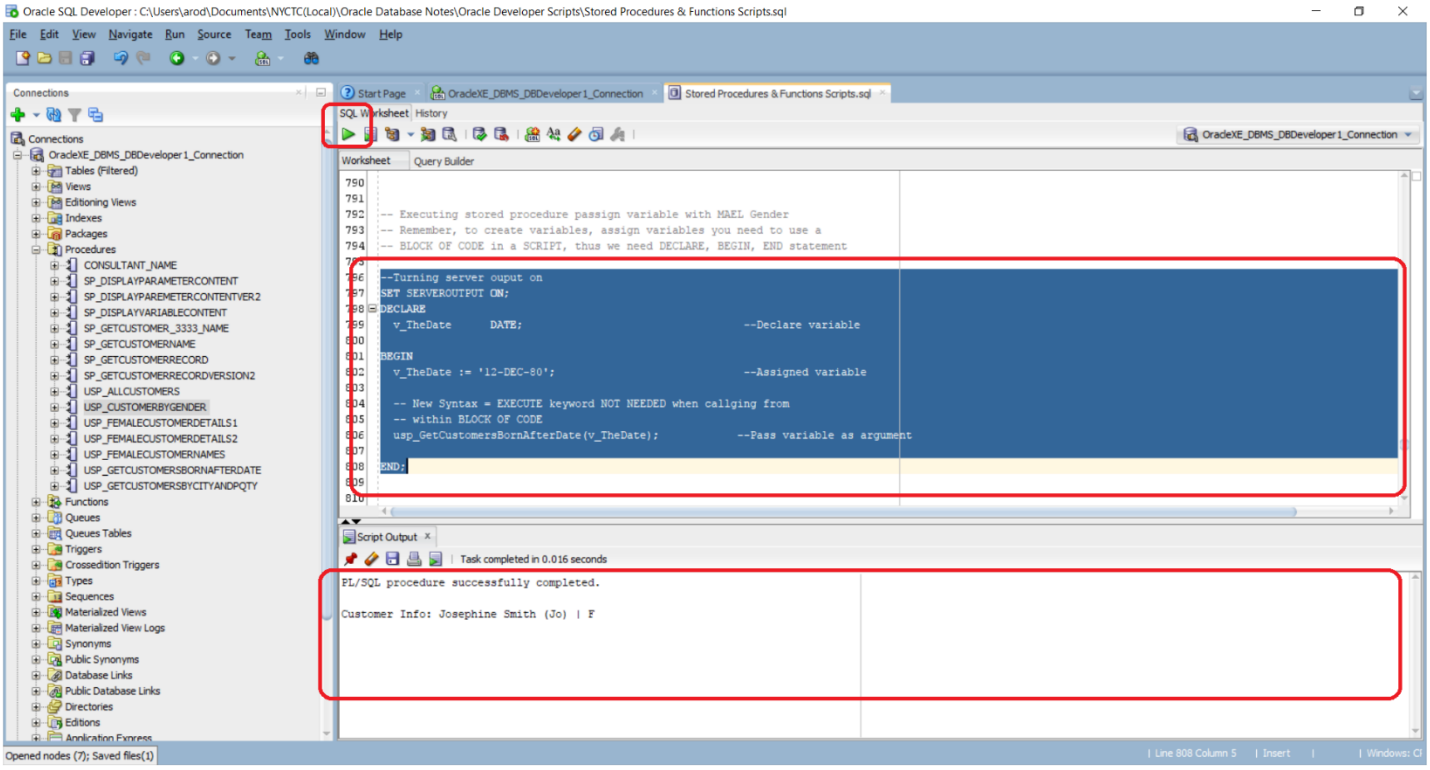
6) We execute the statement with the following argument for female:

```
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



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

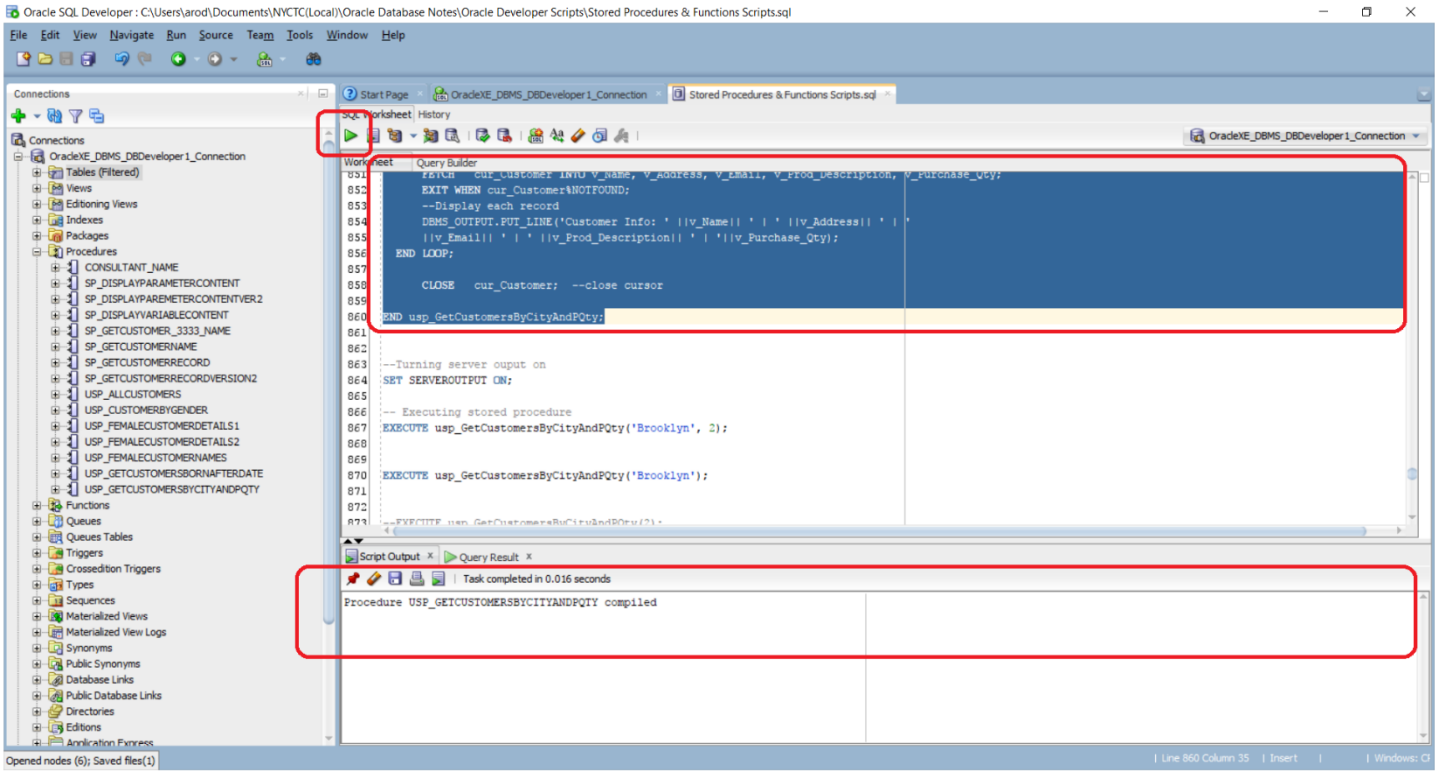
Step 1: Design/define the STORED PROCEDURE using CURSOR

- 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;
```

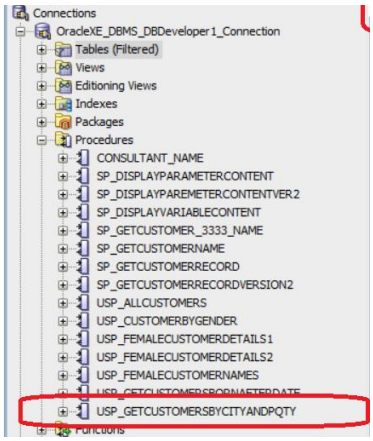
Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- 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:



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:**

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

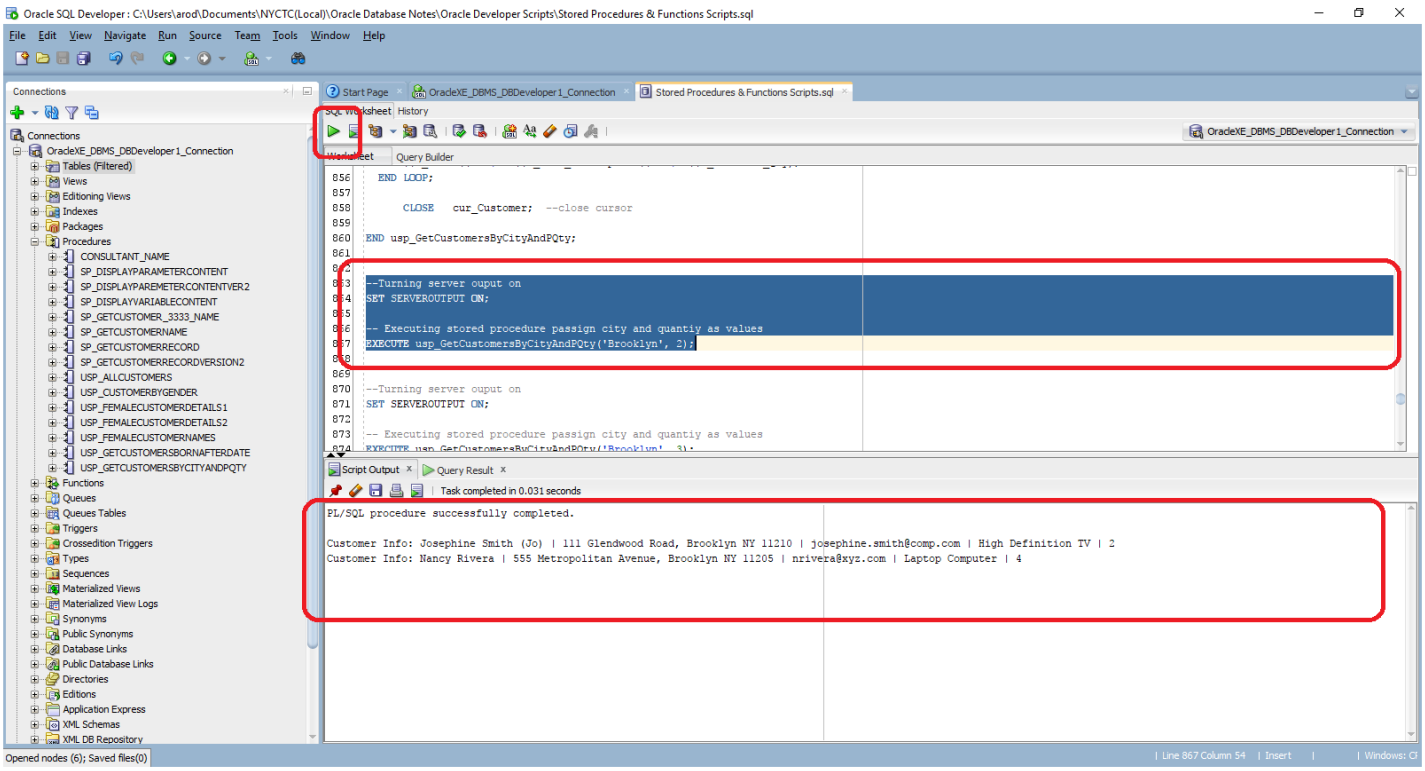
2) Execute/Compile the Script using the following syntax using parameters

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

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

```
EXECUTE usp_GetCustomersByCityAndPQty('Brooklyn', 2);
```

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



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

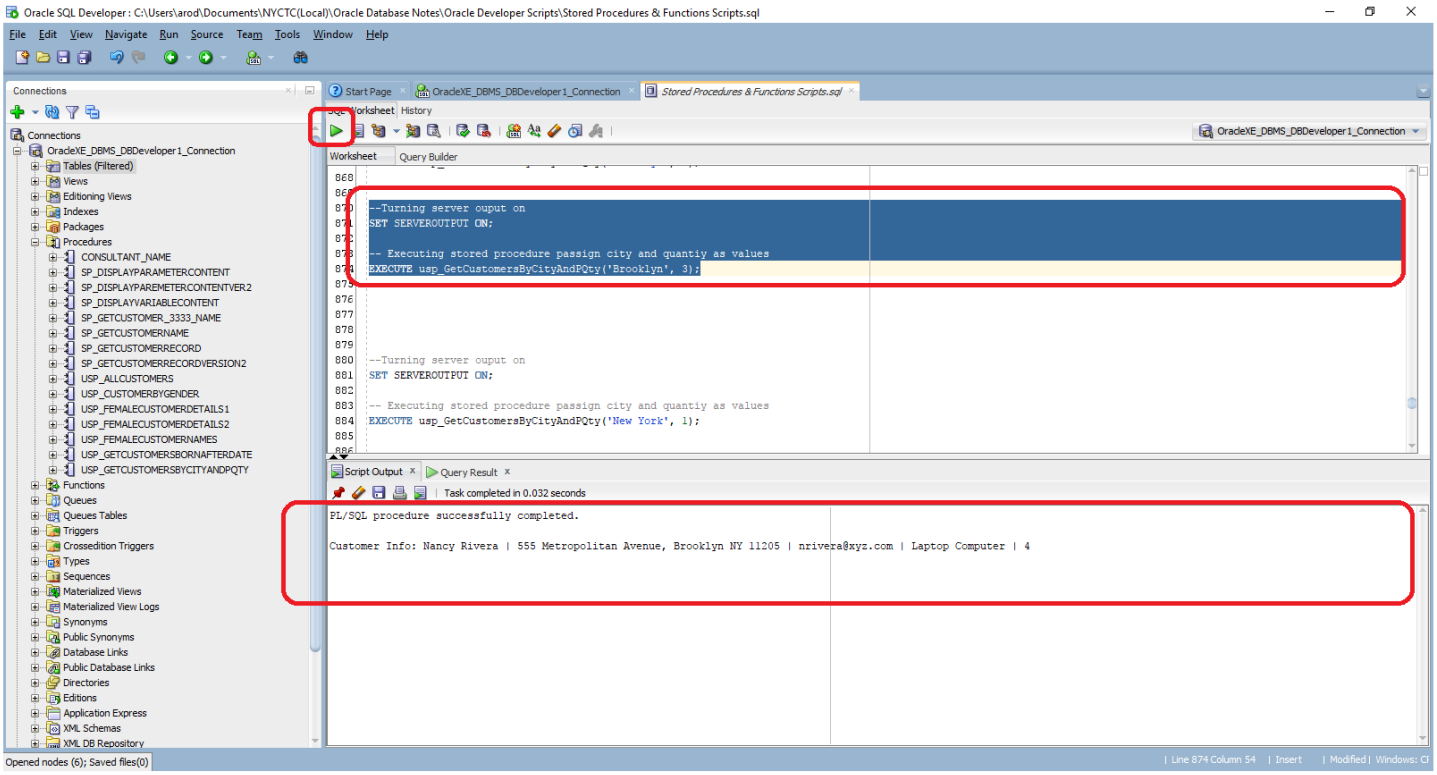
1) Since we are displaying data to the message window we need to **TURN SERVER OUTPUT ON**. Enter code to turn the server output on
2) Execute/Compile the Script using the following syntax using parameters

```
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



- **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

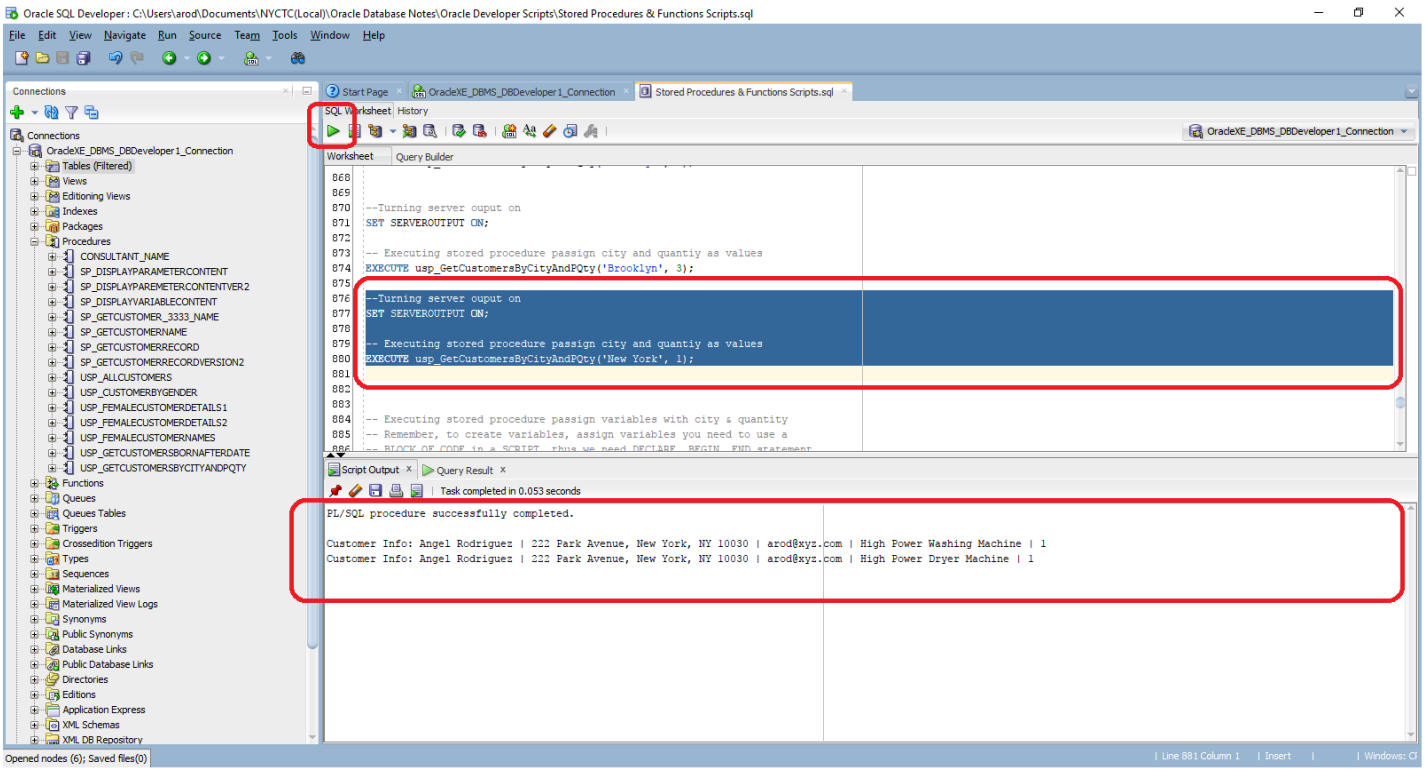
2) Execute/Compile the Script using the following syntax using parameters

```
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



- **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);
```

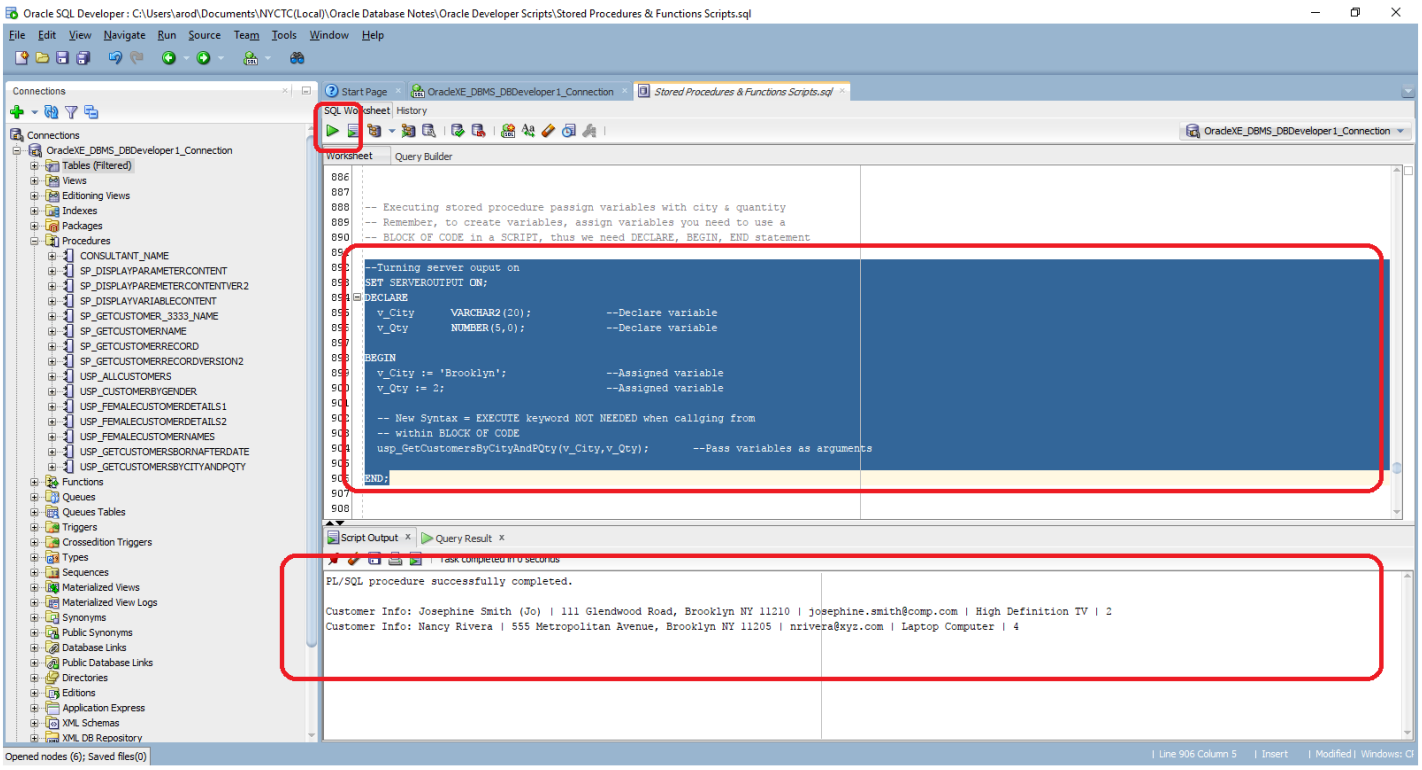
6) We execute the statement with the following argument for female:

```
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



- **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);
```

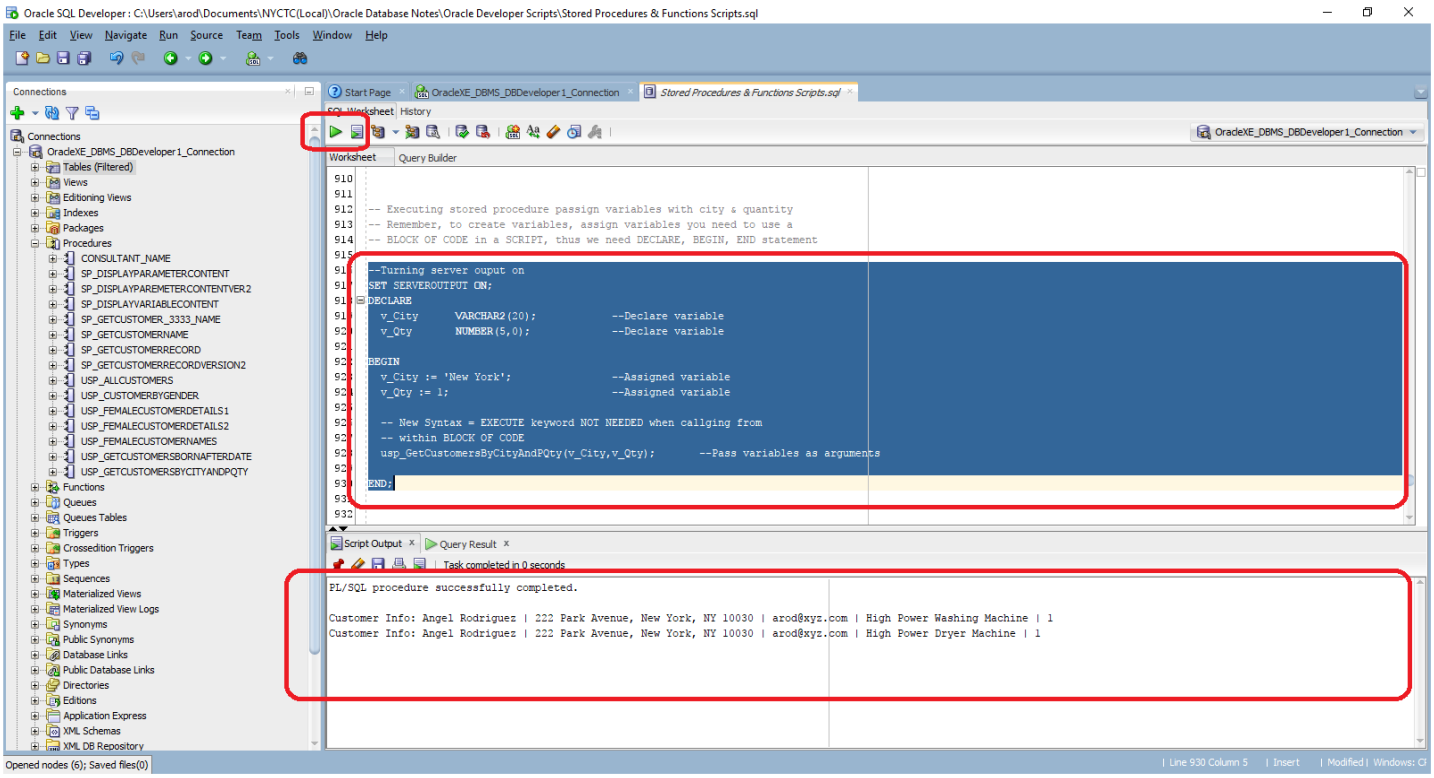
6) We execute the statement with the following argument for female:

```
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



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 >= 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

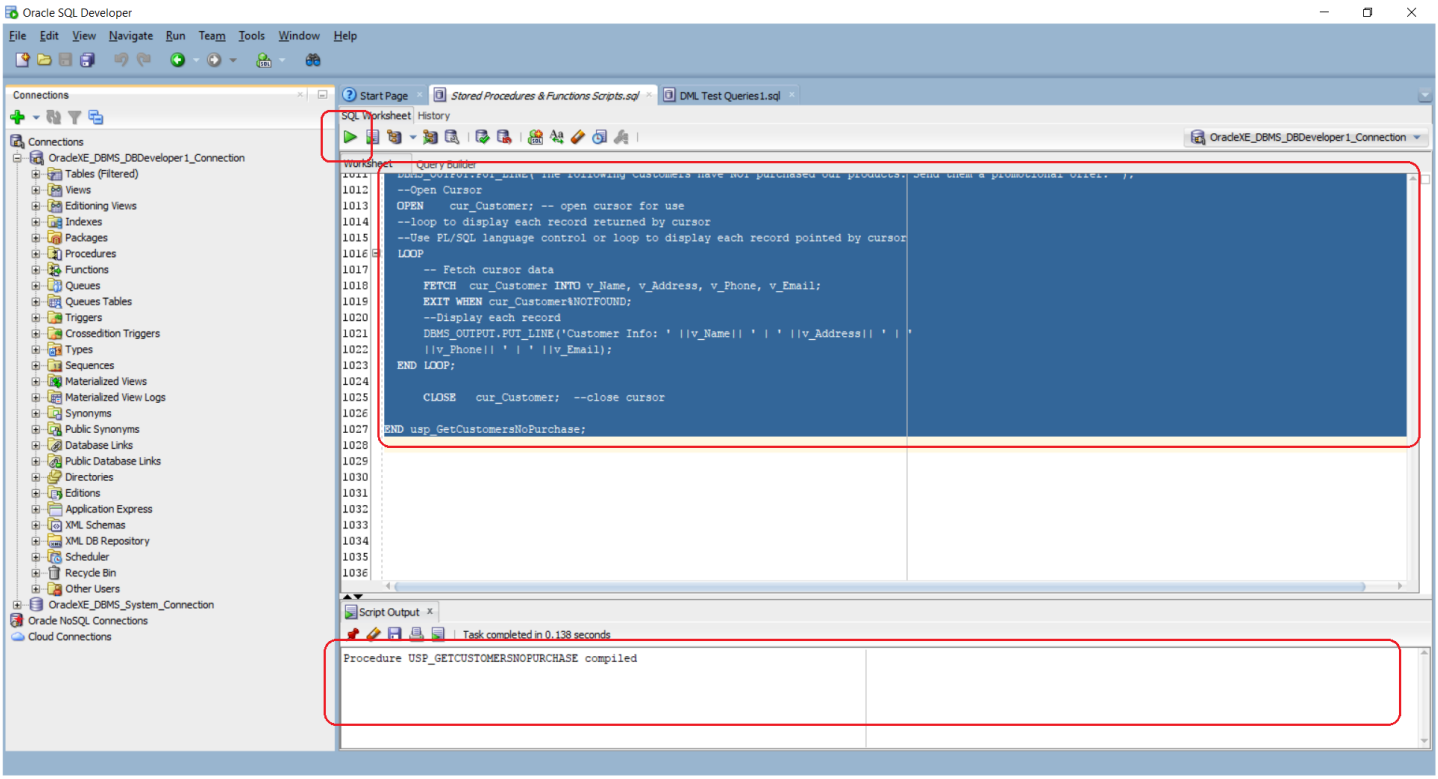
Step 1: Design/define the STORED PROCEDURE using CURSOR

- Stored Procedure Definition design:

```
--Stored Proedure 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 promotial discount as
--insentive 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 messsage
        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;
```

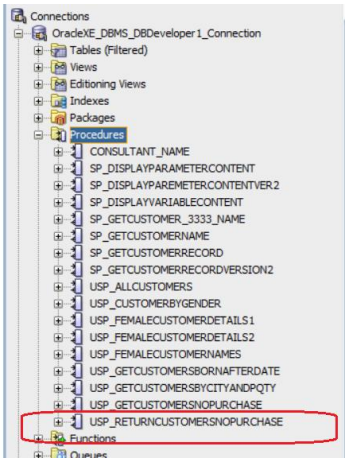
Step 2: In the Oracle SQL Developer Script Windows Enter the Code & Compile Stored Procedure

- 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:



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