Since the procedure contains user-defined functions, these require function prototypes at the head of the procedure.

```
/* ************************  Function Prototypes ********************** */

FUNCTION AssignCountry RETURNS INTEGER
        ( INPUT pcCountry AS CHARACTER ) FORWARD.

FUNCTION GetEmployeeCount RETURNS INTEGER
        (  ) FORWARD.
```

The Main Block contains code that gets executed when the procedure starts up, which fills the temp-table with all the employees in the AutoEdge database.

```
/* ***************************  Main Block  *************************** */

BUFFER ttEmployee:ATTACH-DATA-SOURCE (DATA-SOURCE srcEmployee:HANDLE ).
DATASET dsEmployee:FILL ().   /* All 18 Employees */
OPEN QUERY qEmployee PRESELECT EACH ttEmployee.
```

The Main Block also contains an **ON CLOSE** block of code which will do cleanup if the calling procedure applies the **CLOSE** event to the persistent procedure handle.

```
ON CLOSE OF THIS-PROCEDURE
DO:
   CLOSE QUERY qEmployee.
   EMPTY TEMP-TABLE ttEmployee.
   DELETE PROCEDURE THIS-PROCEDURE.
END.
```

There is one internal procedure, which sets the **EmployeeNotes** value for all employees matching the **EmployeePosition** value passed in.

```
/* **********************  Internal Procedures  *********************** */

PROCEDURE InitializeNotes:

/*------------------------------------------------------------------------------
Purpose:       Assign a value to all empty EmployeeNotes.
Notes:          Done for a specific EmployeePosition.

------------------------------------------------------------------------------*/

DEFINE INPUT PARAMETER pcPosition AS CHARACTER.

    FOR EACH ttEmployee WHERE ttEmployee.EmployeePosition = pcPosition:
        ttEmployee.EmployeeNotes = "Initial note for this " + pcPosition.
    END.
END PROCEDURE.
```

The first user-defined function assigns a **BirthCountry** value to all employees that don't have one, and keeps a count of the number of rows modified.