A Simple Method for Generating Excel Reports from Oracle using the ExcelDocumentType

A couple of years ago, I created a user defined type called the ExcelDocumentType that allows a developer to create a custom Excel (XML) document using PL/SQL. The user defined type has been well received, but a few folks have commented on how tedious the coding can become when creating multiple reports (with a similar tabular layout). I came to the same conclusion recently, and have created a method that greatly simplifies the process. The method makes use of a couple of new user defined types and a PL/SQL package (all provided for your use). It makes creating a multi worksheet document (with a different query for each worksheet) a breeze and with very little ExcelDocumentType coding involved.

The Basics

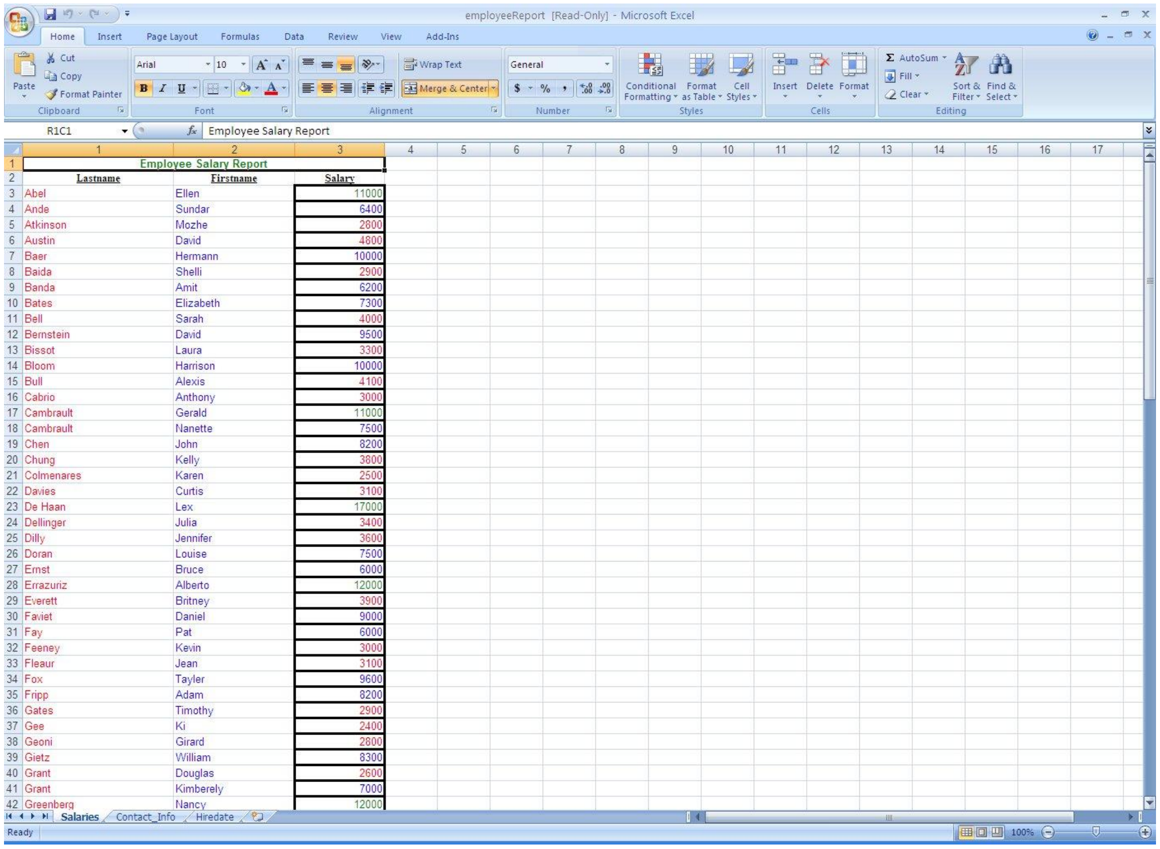
As mentioned above, this new method for generating ExcelDocumentType reports uses a few Oracle user defined types and a small utility package called ExcelDocUtils. Before I get into the details of the PL/SQL package and the supporting user defined types, let's look at a code sample that will generate an Excel document containing three worksheets. For this example I used the EMPLOYEES table from the Oracle HR demo schema.

```
/**
*  This example covers a few features:
*  - Multiple worksheets with multiple queries
*  - Creating Styles and applying them to columns
*  - Worksheet Title (spanning multiple cells)
*  - Conditional Formatting for a range of cells in a worksheet
*  - Sending finished report to a web browser (call it thru a PL/SQL DAD ...)
*  - Hyperlinked cells
*/

CREATE OR REPLACE PROCEDURE employeeReport AS


   v_sql_salary        VARCHAR2(200) := 'SELECT
''ExcelHRef:::#Hiredate!A1:::''||last_name,first_name,salary FROM hr.employees ORDER BY
last_name,first_name';
   v_sql_contact       VARCHAR2(200) := 'SELECT last_name,first_name,phone_number,email  FROM
hr.employees ORDER BY last_name,first_name';
   v_sql_hiredate      VARCHAR2(200) := 'SELECT
last_name,first_name,to_char(hire_date,''MM/DD/YYYY'') hire_date FROM hr.employees ORDER BY
last_name,first_name';

   excelReport         ExcelDocumentType := ExcelDocumentType();
   documentArray       ExcelDocumentLine := ExcelDocumentLine();

   v_worksheet_rec     ExcelDocTypeUtils.T_WORKSHEET_DATA := NULL;
   v_worksheet_array   ExcelDocTypeUtils.WORKSHEET_TABLE  :=
ExcelDocTypeUtils.WORKSHEET_TABLE();

   v_sheet_title       ExcelDocTypeUtils.T_SHEET_TITLE := NULL;

   -- Objects for Defining Document Styles (Optional)

   v_style_def         ExcelDocTypeUtils.T_STYLE_DEF := NULL;
   v_style_array       ExcelDocTypeUtils.STYLE_LIST  := ExcelDocTypeUtils.STYLE_LIST();

   -- Object for Defining Conditional Formating (Optional)

   v_condition_rec         ExcelDocTypeUtils.T_CONDITION      := NULL;
   v_condition_array       ExcelDocTypeUtils.CONDITIONS_TABLE :=
ExcelDocTypeUtils.CONDITIONS_TABLE();

   -- Conditions are applied to a range of cells ... there can be more than grouping of format conditions per
worksheet.
   v_conditional_format_rec   ExcelDocTypeUtils.T_CONDITIONAL_FORMATS;
   v_conditional_format_array ExcelDocTypeUtils.CONDITIONAL_FORMATS_TABLE :=
ExcelDocTypeUtils.CONDITIONAL_FORMATS_TABLE();

BEGIN

  -- Define Styles (Optional)
    v_style_def.p_style_id     := 'LastnameStyle';
    v_style_def.p_text_color   := 'Red';

    ExcelDocTypeUtils.addStyleType(v_style_array,v_style_def);

    v_style_def := NULL;
    v_style_def.p_style_id          := 'SheetTitleStyle';
    v_style_def.p_align_horizontal  := 'Center';
    v_style_def.p_bold              := 'Y';
    v_style_def.p_text_color        := 'Green';

    ExcelDocTypeUtils.addStyleType(v_style_array,v_style_def);

    v_style_def := NULL;
    v_style_def.p_style_id     := 'FirstnameStyle';
    v_style_def.p_text_color   := 'Blue';

    ExcelDocTypeUtils.addStyleType(v_style_array,v_style_def);

    -- Style that includes custom borders around numbers
    v_style_def := NULL;
    v_style_def.p_style_id         := 'NumberStyle';
    v_style_def.p_number_format    := '$###,###,###.00';
    v_style_def.p_align_horizontal := 'Right';
    v_style_def.p_custom_xml         := '<Borders>'||
                                '<Border ss:Position="Left" ss:LineStyle="Continuous" ss:Weight="3"/>'||
                                '<Border ss:Position="Right" ss:LineStyle="Continuous" ss:Weight="3"/>'||
                                '<Border ss:Position="Top" ss:LineStyle="Continuous" ss:Weight="3"/>'||
                                '<Border ss:Position="Bottom" ss:LineStyle="Continuous" ss:Weight="3"/>'||
                              '</Borders>';
```

```
 ExcelDocTypeUtils.addStyleType(v_style_array,v_style_def);

-- Define Sheet Title
v_sheet_title.title      := 'Employee Salary Report';

-- Must Less than or Equal to the max number of columns on the worksheet.
v_sheet_title.cell_span  := '3';
v_sheet_title.style      := 'SheetTitleStyle';

v_worksheet_rec.title    := v_sheet_title;


-- Add conditional formating for Salary Ranges ... color code salary amounts
-- across three different ranges.

v_condition_rec.qualifier    := 'Between';
v_condition_rec.value        := '0,5000';
v_condition_rec.format_style := 'color:red';

ExcelDocTypeUtils.addConditionType(v_condition_array,v_condition_rec);

v_condition_rec.qualifier    := 'Between';
v_condition_rec.value        := '5001,10000';
v_condition_rec.format_style := 'color:blue';

ExcelDocTypeUtils.addConditionType(v_condition_array,v_condition_rec);


v_condition_rec.qualifier    := 'Between';
v_condition_rec.value        := '10001,1000000';
v_condition_rec.format_style := 'color:green';

ExcelDocTypeUtils.addConditionType(v_condition_array,v_condition_rec);

-- Format range for Column 3 starting at row 2 and going to row 65000 ...
v_conditional_format_rec.range      := 'R2C3:R65000C3';
v_conditional_format_rec.conditions := v_condition_array;

ExcelDocTypeUtils.addConditionalFormatType(v_conditional_format_array,v_conditional_format_rec);

v_worksheet_rec.worksheet_cond_formats := v_conditional_format_array;

-- Salary
v_worksheet_rec.query             := v_sql_salary;
v_worksheet_rec.worksheet_name    := 'Salaries';
v_worksheet_rec.col_count         := 3;
v_worksheet_rec.col_width_list    := '25,20,15';
v_worksheet_rec.col_header_list   := 'Lastname,Firstname,Salary';
v_worksheet_rec.col_datatype_list := 'String,String,Number';
v_worksheet_rec.col_style_list    := 'LastnameStyle,FirstnameStyle,NumberStyle';

ExcelDocTypeUtils.addWorksheetType(v_worksheet_array,v_worksheet_rec);

v_worksheet_rec := NULL;

-- Contact
```

```
   v_worksheet_rec.query           := v_sql_contact;
   v_worksheet_rec.worksheet_name  := 'Contact_Info';
   v_worksheet_rec.col_count       := 4;
   v_worksheet_rec.col_width_list  := '25,25,25,25';
   v_worksheet_rec.col_header_list := 'Lastname,Firstname,Phone,Email';
   v_worksheet_rec.col_style_list    := 'LastnameStyle,FirstnameStyle,,';

   ExcelDocTypeUtils.addWorksheetType(v_worksheet_array,v_worksheet_rec);
   v_worksheet_rec := NULL;

   -- Hiredate
   v_worksheet_rec.query           := v_sql_hiredate;
   v_worksheet_rec.worksheet_name  := 'Hiredate';
   v_worksheet_rec.col_count       := 3;
   v_worksheet_rec.col_width_list  := '25,20,20';
   v_worksheet_rec.col_header_list := 'Lastname,Firstname,Hiredate';
   v_worksheet_rec.col_style_list    := 'LastnameStyle,FirstnameStyle,,';

   ExcelDocTypeUtils.addWorksheetType(v_worksheet_array,v_worksheet_rec);

   excelReport := ExcelDocTypeUtils.createExcelDocument(v_worksheet_array,v_style_array);

   excelReport.displayDocument;

END;
/
```

The code above generates the following Excel document:



employeeReport [Read-Only] - Microsoft Excel

R1C1 Employee Salary Report

	1	2	3
1	Employee Salary Report		
2	Lastname	Firstname	Salary
3	Abel	Ellen	11000
4	Ande	Sundar	6400
5	Atkinson	Mozhe	2800
6	Austin	David	4800
7	Baer	Hermann	10000
8	Baida	Shelli	2900
9	Banda	Amit	6200
10	Bates	Elizabeth	7300
11	Bell	Sarah	4000
12	Bernstein	David	9500
13	Bissot	Laura	3300
14	Bloom	Harrison	10000
15	Bull	Alexis	4100
16	Cabrio	Anthony	3000
17	Cambrault	Gerald	11000
18	Cambrault	Nanette	7500
19	Chen	John	8200
20	Chung	Kelly	3800
21	Colmenares	Karen	2500
22	Davies	Curtis	3100
23	De Haan	Lex	17000
24	Dellinger	Julia	3400
25	Dilly	Jennifer	3600
26	Doran	Louise	7500
27	Ernst	Bruce	6000
28	Errazuriz	Alberto	12000
29	Everett	Britney	3900
30	Faviet	Daniel	9000
31	Fay	Pat	6000
32	Feeney	Kevin	3000
33	Fleaur	Jean	3100
34	Fox	Tayler	9600
35	Fripp	Adam	8200
36	Gates	Timothy	2900
37	Gee	Ki	2400
38	Geoni	Girard	2800
39	Gietz	William	8300
40	Grant	Douglas	2600
41	Grant	Kimberely	7000
42	Greenberg	Nancy	12000

So, what are we looking at here? First, a drastic reduction in the amount of code it takes to generate a basic Excel report. Second, we see some interesting data structures that allow us to accomplish that feat. The code above centers around three structures and a new PL/SQL package called *ExcelDocTypeUtils*:

(1) A **record type** called *T_WORKSHEET_DATA* that holds: the query string that generates the worksheet data; the worksheet name; the number of columns displayed in the sheet; a list containing the column width for each displayed column; a list of column headers. This record is defined in the ExcelDocTypeUtils package.
(2) A **collection object** of *T_WORKSHEET_DATA* records called *WORKSHEET_TABLE*. Each item in the collection represents one worksheet in the Excel document.
(3) The *ExcelDocumentType* is that third structure. The *WORKSHEET_TABLE* collection is passed to a function in the *ExcelDocTypeUtils* package that returns a fully populated *ExcelDocumentType* object.
(4) The *ExcelDocTypeUtils* package contains a single public utility function called *createExcelDocument*. This function creates and returns an *ExcelDocumentType* object based upon the *WORKSHEET_TABLE* input parameter.

The Code

I have provided a link in this blog entry that will allow you to download the code and all of the required objects. Download it and give it a try. I have been using at it at my place of employment to generate scheduled reports for various departments and individuals. So far so good!

For those who just want to take a look at the code, here is the code for the ExcelDocTypeUtils package:

```
CREATE OR REPLACE PACKAGE ExcelDocTypeUtils AS


  /* These constants are associated with the creation hyperlink cells.

    hyperlinked data should look like: ExcelHRef:::#Sheet1!A1:::Sheet1 (<hrefIndicator>:::<target>:::<label>)
        or ExcelHref:::http://www.google.com:::Google

  */

  HREF_INDICATOR CONSTANT VARCHAR2(9) := 'ExcelHRef';
  HREF_SEP_CHAR  CONSTANT VARCHAR2(3)  := ':::';

  TYPE t_refcursor IS REF CURSOR;

  pv_result_table  RESULT_TABLE := RESULT_TABLE();
  /* This type allows the user to create a title row at the top of a worksheet */

  TYPE T_SHEET_TITLE IS RECORD(
     title      VARCHAR2(1000),
     cell_span  NUMBER(12),
     style      VARCHAR2(200)
  );

  /* This type allows the user to add conditional formatting to worksheet cells. */

  TYPE T_CONDITION IS RECORD(
     qualifier    VARCHAR2(200),
     value        VARCHAR2(200),
     format_style VARCHAR2(500)
  );

  /* An Array of COND that allows the user to add multiple conditions to a worksheet. */
  TYPE CONDITIONS_TABLE IS TABLE OF T_CONDITION;


  TYPE T_CONDITIONAL_FORMATS IS RECORD(
    range            VARCHAR2(200),
    conditions       CONDITIONS_TABLE
  );

  TYPE CONDITIONAL_FORMATS_TABLE IS TABLE OF T_CONDITIONAL_FORMATS;


  /* This record contains all of the components required to create an Excel Report worksheet. */
  TYPE T_WORKSHEET_DATA IS RECORD(
     query                  VARCHAR2(4000),
     title                  T_SHEET_TITLE,
     worksheet_name         VARCHAR2(50),
     worksheet_cond_formats CONDITIONAL_FORMATS_TABLE,
     col_count              NUMBER(3),
     col_width_list         VARCHAR2(500),
     col_caption            VARCHAR2(2000),
     col_header_list        VARCHAR2(2000),
     col_datatype_list      VARCHAR2(4000),
     col_style_list         VARCHAR2(5000),
     col_formula_list       VARCHAR2(4000)
  );


  /* An Array of T_WORKSHEET_DATA allows us to create an excel document with multiple worksheets based
on
    different queries. */
  TYPE WORKSHEET_TABLE IS TABLE OF T_WORKSHEET_DATA;


  /* This record structure matches the createStyle method of the ExcelDocumentType. */
  TYPE T_STYLE_DEF IS RECORD(
                         p_style_id        VARCHAR2(50),
                         p_font            VARCHAR2(50),
                         p_ffamily         VARCHAR2(50),
                         p_fsize           VARCHAR2(50),
                         p_bold            VARCHAR2(1),
                         p_italic          VARCHAR2(1),
                         p_underline       VARCHAR2(1),
                         p_text_color      VARCHAR2(50),
                         p_cell_color      VARCHAR2(50),
                     p_cell_pattern     VARCHAR2(50),
                     p_align_vertical   VARCHAR2(50),
                     p_align_horizontal VARCHAR2(50),
                     p_wrap_text        VARCHAR2(1),
                     p_number_format    VARCHAR2(100),
                     p_custom_xml       VARCHAR2(4000)
                   );


  /* Collection of styles that can applied to cells */
  TYPE STYLE_LIST IS TABLE OF T_STYLE_DEF;


  /* These four procedures are convenience procedures that make it easy to a record to it associated collection. */

  PROCEDURE addStyleType(p_style_array IN OUT NOCOPY STYLE_LIST, p_style_rec T_STYLE_DEF);

  PROCEDURE addWorksheetType(p_worksheet_data IN OUT NOCOPY WORKSHEET_TABLE,
p_worksheet_rec T_WORKSHEET_DATA);

  PROCEDURE addConditionType(p_condition_data IN OUT NOCOPY CONDITIONS_TABLE,
p_condition_rec T_CONDITION);

  PROCEDURE addConditionalFormatType(p_cond_format_data IN OUT NOCOPY
CONDITIONAL_FORMATS_TABLE, p_cond_format_rec T_CONDITIONAL_FORMATS);

  /* This function creates the Excel Document based on the data passed in the p_worksheet_data collection. */

  FUNCTION createExcelDocument(p_worksheet_data WORKSHEET_TABLE,
                     p_style_data     STYLE_LIST := STYLE_LIST()) RETURN ExcelDocumentType;


END;
/
sho err;

/****************************************************************************************
****************************************************************************************/

CREATE OR REPLACE PACKAGE BODY ExcelDocTypeUtils AS


/*

  Function that returns the element at the requested position in a delimited string.

*/
FUNCTION getStringElement(p_string    VARCHAR2,
                  p_element   NUMBER,
                  p_delimiter VARCHAR2 := ',',
                  p_level     NUMBER   := 0)   RETURN VARCHAR2
IS

  v_string   VARCHAR2(2000) := NULL;
  v_element  VARCHAR2(2000) := NULL;
  v_next     VARCHAR2(2000) := NULL;

  v_level    NUMBER(4)      := 0;


BEGIN
```

```
   v_level := p_level + 1;

   v_element  := substr(p_string||p_delimiter,1,instr(p_string||p_delimiter,p_delimiter)-1);

   -- need to look ahead to make sure we handle the null elements.
   v_next     := substr(p_string||p_delimiter,instr(p_string||p_delimiter,p_delimiter),length(p_delimiter));

   IF ((v_level >= p_element) OR (v_element IS NULL AND v_next != p_delimiter)) THEN

      RETURN v_element;

   ELSE

      v_string :=
substr(p_string||p_delimiter,instr(p_string||p_delimiter,p_delimiter)+length(p_delimiter),length(p_string));

      RETURN getStringElement(v_string,p_element,p_delimiter,v_level);

   END IF;


END;

/*============================================================================*/
FUNCTION isHRefData(p_data VARCHAR2 := NULL) RETURN BOOLEAN
IS

   v_return BOOLEAN := FALSE;

BEGIN

   IF INSTR(p_data,HREF_INDICATOR,1) > 0 THEN

     v_return := TRUE;

   END IF;


   RETURN v_return;

END;

/*============================================================================*/

FUNCTION getLinkTarget(p_data VARCHAR2 := NULL) RETURN VARCHAR2
IS

   v_target VARCHAR2(2000) := NULL;

BEGIN

   v_target := getStringElement(p_data,2,HREF_SEP_CHAR);

   RETURN v_target;

END;

/*============================================================================*/

FUNCTION getLinkLabel(p_data VARCHAR2 := NULL) RETURN VARCHAR2
IS
```

```
   v_label VARCHAR2(2000) := NULL;

BEGIN

   v_label := getStringElement(p_data,3,HREF_SEP_CHAR);

   RETURN v_label;

END;

/*==========================================================================*/
/*
    This function executes the given query and returns the data in a RESULT_TABLE Collection object.

*/

FUNCTION buildDataSet(p_query_string VARCHAR2 := NULL,
                      p_col_count    NUMBER   := 0) RETURN RESULT_TABLE
IS


  v_row_symbol       VARCHAR2(20)   := 'v_row';
  v_row_fetch        VARCHAR2(1000) := NULL;
  v_row_extend       NUMBER(3)      := p_col_count;

  v_query            VARCHAR2(16000) := p_query_string;

  v_result_proc      VARCHAR2(32000) := 'DECLARE'||chr(10)||
                                        ' TYPE t_refcursor IS REF CURSOR; '||chr(10)||
                                        ' v_row T_ROW := T_ROW(); '||chr(10)||
                                        ' v_query VARCHAR2(4000) := ''<q>''; '||chr(10)||
                                        ' v_refcur t_refcursor; '||chr(10)||
                                       'BEGIN '||chr(10)||
                                       '   OPEN v_refcur FOR v_query; '||chr(10)||
                                       '   LOOP '||chr(10)||
                                       '    v_row.extend(<e>); '||chr(10)||
                                       '    FETCH v_refcur INTO <f> ;'||chr(10)||
                                       '    EXIT WHEN v_refcur%NOTFOUND; '||chr(10)||
                                       '    ExcelDocTypeUtils.pv_result_table.EXTEND; '||chr(10)||
                                       '    ExcelDocTypeUtils.pv_result_table(ExcelDocTypeUtils.pv_result_table.COUNT) :=
v_row; '||chr(10)||
                                       '    v_row.DELETE; '||chr(10)||
                                       '   END LOOP; '||chr(10)||
                                       'END; ';

BEGIN

   FOR x IN 1 .. v_row_extend LOOP

      v_row_fetch := v_row_fetch||v_row_symbol||'('||x||'),';

   END LOOP;

   v_row_fetch := RTRIM(v_row_fetch,',');

   v_result_proc := REPLACE(v_result_proc,'<q>',REPLACE(v_query,'''',''''''));

   v_result_proc := REPLACE(v_result_proc,'<e>',to_char(v_row_extend));

   v_result_proc := REPLACE(v_result_proc,'<f>',v_row_fetch);
   pv_result_table := RESULT_TABLE();

   EXECUTE IMMEDIATE v_result_proc;

   RETURN pv_result_table;

END;

/*==================================================================================*/

PROCEDURE addStyleType(p_style_array IN OUT NOCOPY STYLE_LIST, p_style_rec T_STYLE_DEF)
IS
BEGIN

   p_style_array.EXTEND;
   p_style_array(p_style_array.COUNT) := p_style_rec;

END;

/*==================================================================================*/

PROCEDURE addWorksheetType(p_worksheet_data IN OUT NOCOPY WORKSHEET_TABLE,
p_worksheet_rec T_WORKSHEET_DATA)
IS
BEGIN

   p_worksheet_data.EXTEND;
   p_worksheet_data(p_worksheet_data.COUNT) := p_worksheet_rec;

END;

/*==================================================================================*/

PROCEDURE addConditionType(p_condition_data IN OUT NOCOPY CONDITIONS_TABLE,
p_condition_rec T_CONDITION)
IS
BEGIN

   p_condition_data.EXTEND;
   p_condition_data(p_condition_data.COUNT) := p_condition_rec;

END;

/*==================================================================================*
/

PROCEDURE addConditionalFormatType(p_cond_format_data IN OUT NOCOPY
CONDITIONAL_FORMATS_TABLE, p_cond_format_rec T_CONDITIONAL_FORMATS)
IS
BEGIN

   p_cond_format_data.EXTEND;
   p_cond_format_data(p_cond_format_data.COUNT) := p_cond_format_rec;

END;

/*==================================================================================*/
/*

   This functiom constructs and returns an ExcelDocumentType based upon the parameters passed
   in by the WORKSHEET_TABLE type parameter.
```

```
*/
FUNCTION createExcelDocument(p_worksheet_data WORKSHEET_TABLE,
                             p_style_data     STYLE_LIST := STYLE_LIST()) RETURN ExcelDocumentType
IS

  resultDocument        ExcelDocumentType;
  v_row                 T_ROW := T_ROW();
  v_results             RESULT_TABLE := RESULT_TABLE();

  v_title               T_SHEET_TITLE := NULL;

  v_style               T_STYLE_DEF := NULL;

  v_default_col_width   NUMBER(3)    := 30;
  v_col_width           NUMBER(3)    := 0;

  v_default_data_type   VARCHAR2(6)  := 'String';
  v_data_type           VARCHAR2(20) := NULL;
  v_data_style          VARCHAR2(50) := NULL;

  v_style_list          VARCHAR2(4000) := ';';

  v_count_rows          NUMBER(10);
  v_formula             VARCHAR2(100);

BEGIN

  BEGIN

     COMMIT;

  EXCEPTION
     WHEN OTHERS THEN NULL;
  END;



  resultDocument := ExcelDocumentType();

  -- Open Document
  resultDocument.documentOpen;

  -- Define Customs Styles
  resultDocument.stylesOpen;

  resultDocument.defaultStyle;

  /* Style for Column Header Row */
  resultDocument.createStyle(p_style_id =>'ColumnHeader',
                             p_font     =>'Times New Roman',
                             p_ffamily  =>'Roman',
                             p_fsize    =>'10',
                             p_bold     =>'Y',
                             p_underline =>'Single',
                             p_align_horizontal=>'Center',
                             p_align_vertical=>'Bottom');


  FOR x IN 1 .. p_style_data.COUNT LOOP

    v_style := p_style_data(x);
    v_style_list := v_style_list||';'||UPPER(v_style.p_style_id);

    resultDocument.createStyle(p_style_id         => UPPER(v_style.p_style_id),
                               p_font             => v_style.p_font,
                               p_ffamily          => v_style.p_ffamily,
                               p_fsize            => v_style.p_fsize,
                               p_bold             => v_style.p_bold,
                               p_italic           => v_style.p_italic,
                               p_underline        => v_style.p_underline,
                               p_text_color       => v_style.p_text_color,
                               p_cell_color       => v_style.p_cell_color,
                               p_cell_pattern     => v_style.p_cell_pattern,
                               p_align_vertical   => v_style.p_align_vertical,
                               p_align_horizontal => v_style.p_align_horizontal,
                               p_wrap_text        => v_style.p_wrap_text,
                               p_number_format    => v_style.p_number_format,
                               p_custom_xml       => v_style.p_custom_xml);


  END LOOP;

  resultDocument.stylesClose;


  FOR ws_index IN 1 .. p_worksheet_data.COUNT LOOP

     -- Open Worksheets

     resultDocument.worksheetOpen(p_worksheet_data(ws_index).worksheet_name);


     -- Define Conditional Formatting

     BEGIN

        FOR cf_index IN 1 .. p_worksheet_data(ws_index).worksheet_cond_formats.COUNT LOOP


resultDocument.worksheetCondFormatOpen(p_worksheet_data(ws_index).worksheet_cond_formats(cf_index).ra
nge);

           BEGIN
              FOR cond_index IN 1 ..
p_worksheet_data(ws_index).worksheet_cond_formats(cf_index).conditions.COUNT LOOP


resultDocument.createCondFormat(p_qualifier=>p_worksheet_data(ws_index).worksheet_cond_formats(cf_index
).conditions(cond_index).qualifier,

p_value=>p_worksheet_data(ws_index).worksheet_cond_formats(cf_index).conditions(cond_index).value,

p_format_style=>p_worksheet_data(ws_index).worksheet_cond_formats(cf_index).conditions(cond_index).forma
t_style);


              END LOOP;
           EXCEPTION WHEN COLLECTION_IS_NULL THEN NULL;
           END;

           resultDocument.worksheetCondFormatClose;
        END LOOP;

     EXCEPTION WHEN COLLECTION_IS_NULL THEN NULL;

     END;

     -- Define Columns

     FOR colnum IN 1 .. p_worksheet_data(ws_index).col_count LOOP

        v_col_width :=
NVL(TO_NUMBER(getStringElement(p_worksheet_data(ws_index).col_width_list,colnum)),v_default_col_widt
h);

        resultDocument.defineColumn(p_width=>v_col_width);
     END LOOP;

     -- Sheet Title Row
     v_title := p_worksheet_data(ws_index).title;
     IF v_title.title IS NOT NULL THEN

        IF v_title.cell_span IS NULL OR v_title.cell_span >= p_worksheet_data(ws_index).col_count THEN

            v_title.cell_span := p_worksheet_data(ws_index).col_count-1;

        END IF;

        resultDocument.rowOpen;

resultDocument.addCell(p_style=>UPPER(v_title.style),p_data=>v_title.title,p_custom_attr=>'ss:MergeAcross="'
||v_title.cell_span||'"');
        resultDocument.rowClose;

     END IF;

     -- Caption Row

     IF p_worksheet_data(ws_index).col_caption IS NOT NULL THEN

        resultDocument.rowOpen;
        FOR colnum IN 1 .. p_worksheet_data(ws_index).col_count LOOP


resultDocument.addCell(p_style=>'ColumnHeader',p_data=>getStringElement(p_worksheet_data(ws_index).col_
caption,colnum));

        END LOOP;

        resultDocument.rowClose;

     END IF;


     -- Heading Row
     resultDocument.rowOpen;

     FOR colnum IN 1 .. p_worksheet_data(ws_index).col_count LOOP


resultDocument.addCell(p_style=>'ColumnHeader',p_data=>getStringElement(p_worksheet_data(ws_index).col_
header_list,colnum));
```

```
    END LOOP;

    resultDocument.rowClose;

    v_results := buildDataSet(p_worksheet_data(ws_index).query,
                              p_worksheet_data(ws_index).col_count);

    v_count_rows := v_results.COUNT;

    FOR r_index IN 1 .. v_results.COUNT LOOP

      resultDocument.rowOpen;

      v_row  := v_results(r_index);

      FOR c_index IN 1 .. v_row.COUNT LOOP

        v_data_type  :=
NVL(getStringElement(p_worksheet_data(ws_index).col_datatype_list,c_index),v_default_data_type);

        v_data_style :=
NVL(UPPER(getStringElement(p_worksheet_data(ws_index).col_style_list,c_index)),NULL);

        IF INSTR(v_style_list,v_data_style) = 0 THEN

          v_data_style := NULL;

        END IF;

                    IF isHRefData(v_row(c_index)) THEN

          resultDocument.addCell(p_data      => getLinkLabel(v_row(c_index)),
                                 p_data_type => v_data_type,
                                 p_style     => v_data_style,
                                                                                 p_HRef      =>
getLinkTarget(v_row(c_index)));

                    ELSE

          resultDocument.addCell(p_data      => v_row(c_index),
                                 p_data_type => v_data_type,
                                 p_style     => v_data_style);

                    END IF;

      END LOOP;

      v_row.DELETE;

      resultDocument.rowClose;

    END LOOP;

    v_results.DELETE;

    -- Formula Row
    IF p_worksheet_data(ws_index).col_formula_list IS NOT NULL THEN

      resultDocument.rowOpen;

      FOR colnum IN 1 .. p_worksheet_data(ws_index).col_count LOOP
```

```
          v_data_style :=
NVL(UPPER(getStringElement(p_worksheet_data(ws_index).col_style_list,colnum)),NULL);
          v_formula :=
replace(getStringElement(p_worksheet_data(ws_index).col_formula_list,colnum),'<ZMIN>',trim(to_char(v_count
_rows)));
          resultDocument.addCell(p_formula   => v_formula,
                              p_data_type => v_data_type,
                              p_style     => v_data_style);

        END LOOP;

        resultDocument.rowClose;

     END IF;

     resultDocument.worksheetClose;

  END LOOP;

  resultDocument.documentClose;

  RETURN resultDocument;

END;


/*=============*/
/* END PACKAGE */
/*=============*/
END;
/
```

Wrapping It Up

As with all of the code I post in my blog entries, please feel to use it and modify it as you see fit. Please feel free to contact me with any questions! I hope this makes generating Excel reports a little more easy and lot less tedious.