MEMO

FROM: Momodou lamin Keita Database Development
TO: Bill Cunningham
DATE: 13/12/15

SUBJECT: DBAS 2200 ASSG 4

EXISTING SYSTEM

The existing system is the scenario information included in the Problem-Solving Cases 6-10 from Lesson B in the supplied text.

STATEMENT OF REQUIREMENT

I will supply necessary analysis in the form of pseudocode, narrative, and/or diagrammatic tools, my actual code solution, and a screenshot of the SQL Developer output for each of the four cases above.

ANALYSIS:

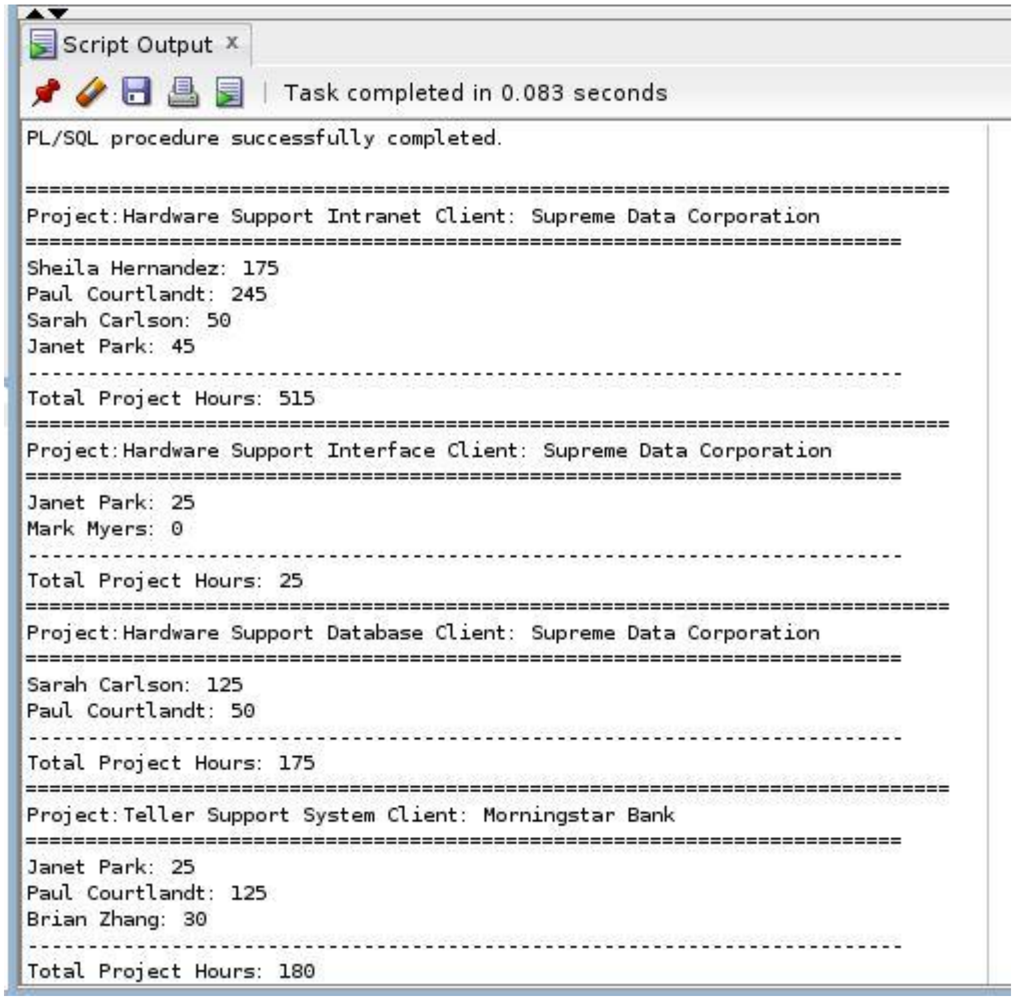
Case6:

```
SET SERVEROUTPUT ON


DECLARE


CURSOR project_cursor is

select project.P_ID, project.project_name, client.client_name

from project

inner join CLIENT on CLIENT.CLIENT_ID = project.CLIENT_ID;

project_cursor_row project_cursor%ROWTYPE;

ProjectID NUMBER;

CURSOR employees_cursor is

select PRO.P_ID, PRO.PROJECT_NAME,C.C_FIRST ,C.C_LAST, P.TOTAL_HOURS

FROM PROJECT_CONSULTANT P

inner join CONSULTANT C ON C.C_ID = P.C_ID

inner join PROJECT PRO ON PRO.P_ID = P.P_ID

where p.P_ID = ProjectID;

employees_cursor_row  employees_cursor%ROWTYPE;

CURSOR total_cursor is
```

```
select sum(P.TOTAL_HOURS) as total_hours
        FROM PROJECT_CONSULTANT P
        inner join CONSULTANT C ON C.C_ID = P.C_ID
        inner join PROJECT PRO ON PRO.P_ID = P.P_ID
        where PRO.P_ID = PROJECTID;
        total_cursor_row  total_cursor%ROWTYPE;
BEGIN
OPEN project_cursor;
FETCH project_cursor INTO project_cursor_row;


LOOP
EXIT WHEN project_cursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('=================================================================
===========');
   DBMS_OUTPUT.PUT_LINE('Project:' ||project_cursor_row.project_name||
                   ' ' || 'Client: '||
                   project_cursor_row.client_name);

DBMS_OUTPUT.PUT_LINE('=================================================================
=======');
   ProjectID :=project_cursor_row.P_ID;
   OPEN employees_cursor;
   FETCH employees_cursor INTO employees_cursor_row;
   OPEN total_cursor;
   FETCH total_cursor INTO total_cursor_row;
   LOOP
   EXIT WHEN employees_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(employees_cursor_row.C_FIRST ||
                      ' ' ||
                      employees_cursor_row.C_LAST||
                      ': ' ||
                      employees_cursor_row.TOTAL_HOURS);
```

```
          FETCH employees_cursor INTO employees_cursor_row;
           END LOOP;
           DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------------------');
           DBMS_OUTPUT.PUT_LINE('Total Project Hours: '|| total_cursor_row.total_hours);
           CLOSE employees_cursor;
            CLOSE total_cursor;
           FETCH project_cursor INTO project_cursor_row;
  END LOOP;
  CLOSE project_cursor;
  END;
```

OUTPUT



Script Output

Task completed in 0.083 seconds

```
PL/SQL procedure successfully completed.

=============================================================================
Project:Hardware Support Intranet Client: Supreme Data Corporation
=========================================================================
Sheila Hernandez: 175
Paul Courtlandt: 245
Sarah Carlson: 50
Janet Park: 45
-------------------------------------------------------------------------
Total Project Hours: 515
=============================================================================
Project:Hardware Support Interface Client: Supreme Data Corporation
=========================================================================
Janet Park: 25
Mark Myers: 0
-------------------------------------------------------------------------
Total Project Hours: 25
=============================================================================
Project:Hardware Support Database Client: Supreme Data Corporation
=========================================================================
Sarah Carlson: 125
Paul Courtlandt: 50
-------------------------------------------------------------------------
Total Project Hours: 175
=============================================================================
Project:Teller Support System Client: Morningstar Bank
=========================================================================
Janet Park: 25
Paul Courtlandt: 125
Brian Zhang: 30
-------------------------------------------------------------------------
Total Project Hours: 180
```

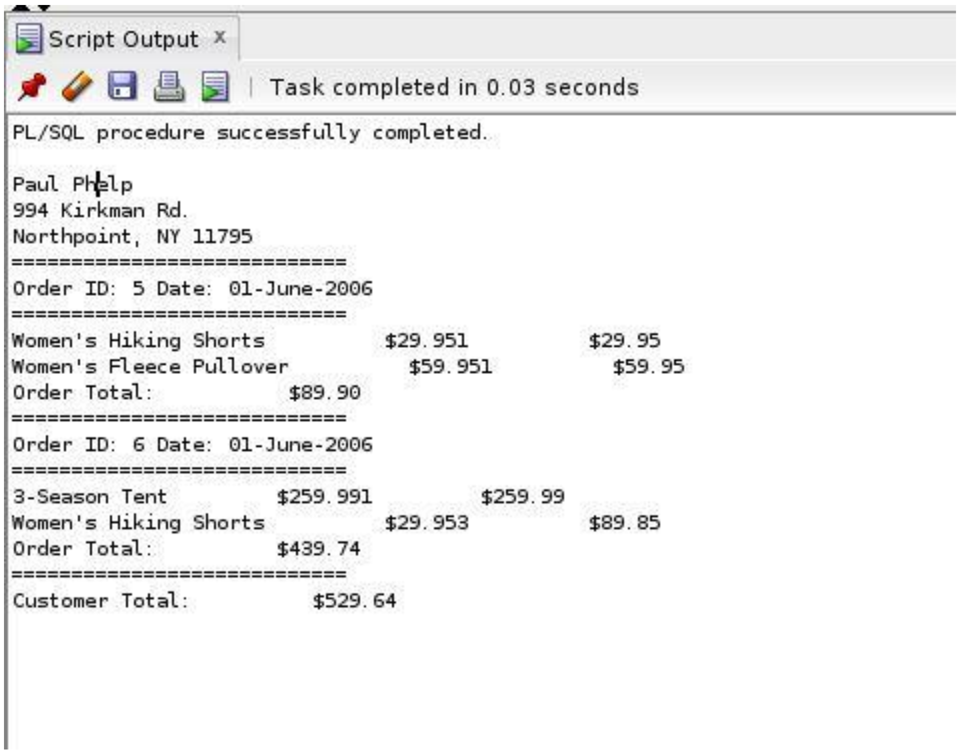
Case 7:

```
SET SERVEROUTPUT ON

DECLARE
  --variables for row data
  cust_id customer.c_id%TYPE;
  first_name customer.c_first%TYPE;
  last_name customer.c_last%TYPE;
  address customer.c_address%TYPE;
  city customer.c_city%TYPE;
  state customer.c_state%TYPE;
  zip customer.c_zip%TYPE;
  order_id order_line.o_id%TYPE;
  total_value NUMBER(10,2);
  order_total NUMBER(10,2) := 0;
  customer_total NUMBER(10,2) := 0;

  --cursor #1 - iterate through orders
  CURSOR order_cursor IS
    SELECT o.o_id, o.o_date
    FROM orders o
    WHERE o.c_id = cust_id;
  order_cursor_row order_cursor%ROWTYPE;

  --cursor # 2 - iterate through items in order
  CURSOR item_cursor IS
    SELECT i.item_desc, inv.inv_price, ol.ol_quantity
    FROM order_line ol
    INNER JOIN inventory inv ON ol.inv_id = inv.inv_id
    INNER JOIN item i ON i.item_id = inv.item_id
    WHERE ol.o_id = order_id;
  item_cursor_row item_cursor%ROWTYPE;

BEGIN

  --query to select customer info
  SELECT cust.c_id, cust.c_first, cust.c_last, cust.c_address, cust.c_city, cust.c_state, cust.c_zip
  INTO cust_id, first_name, last_name, address, city, state, zip
  FROM customer cust
  WHERE cust.c_id = 4;

  --output customer info
  DBMS_OUTPUT.PUT_LINE(first_name || ' ' || last_name);
  DBMS_OUTPUT.PUT_LINE(address);
  DBMS_OUTPUT.PUT_LINE(city || ', ' || state || ' ' || zip);

  --cursor #1
  OPEN order_cursor;
  FETCH order_cursor INTO order_cursor_row;
  LOOP
    EXIT WHEN order_cursor%NOTFOUND;
    --output order id and date
    DBMS_OUTPUT.PUT_LINE('===========================');
    DBMS_OUTPUT.PUT_LINE('Order ID: ' || order_cursor_row.o_id ||
                         ' ' ||
                         'Date: ' || TO_CHAR(order_cursor_row.o_date, 'dd-FMMonth-yyyy'));
    DBMS_OUTPUT.PUT_LINE('===========================');

    order_id := order_cursor_row.o_id;

    --cursor #2
```

```
   OPEN item_cursor;
   FETCH item_cursor INTO item_cursor_row;
   LOOP
     EXIT WHEN item_cursor%NOTFOUND;

      --calculate total value of each item in order
      total_value := item_cursor_row.inv_price * item_cursor_row.ol_quantity;

      --output items in order
      DBMS_OUTPUT.PUT_LINE(item_cursor_row.item_desc ||
                      '' ||
                      TO_CHAR(item_cursor_row.inv_price, '$999,999,999.99') ||
                      '' ||
                      item_cursor_row.ol_quantity ||
                      '' ||
                      TO_CHAR(item_cursor_row.inv_price * item_cursor_row.ol_quantity,
'$999,999,999.99'));

      --add item total to order total
      order_total := order_total + total_value;

     FETCH item_cursor INTO item_cursor_row;
   END LOOP;
   CLOSE item_cursor;

   --output order total
   DBMS_OUTPUT.PUT_LINE('Order Total: ' || TO_CHAR(order_total, '$999,999,999.99'));

   --add order total to customer total
   customer_total := customer_total + order_total;

   FETCH order_cursor INTO order_cursor_row;
```

```
  END LOOP;
  CLOSE order_cursor;

  --output customer total
  DBMS_OUTPUT.PUT_LINE('============================');
  DBMS_OUTPUT.PUT_LINE('Customer Total: ' || TO_CHAR(customer_total, '$999,999,999.99'));


END;
```

Output:



```
PL/SQL procedure successfully completed.

Paul Phelp
994 Kirkman Rd.
Northpoint, NY 11795
============================
Order ID: 5 Date: 01-June-2006
============================
Women's Hiking Shorts         $29.951         $29.95
Women's Fleece Pullover         $59.951         $59.95
Order Total:           $89.90
============================
Order ID: 6 Date: 01-June-2006
============================
3-Season Tent         $259.991         $259.99
Women's Hiking Shorts         $29.953         $89.85
Order Total:          $439.74
============================
Customer Total:          $529.64
```

Case 8:

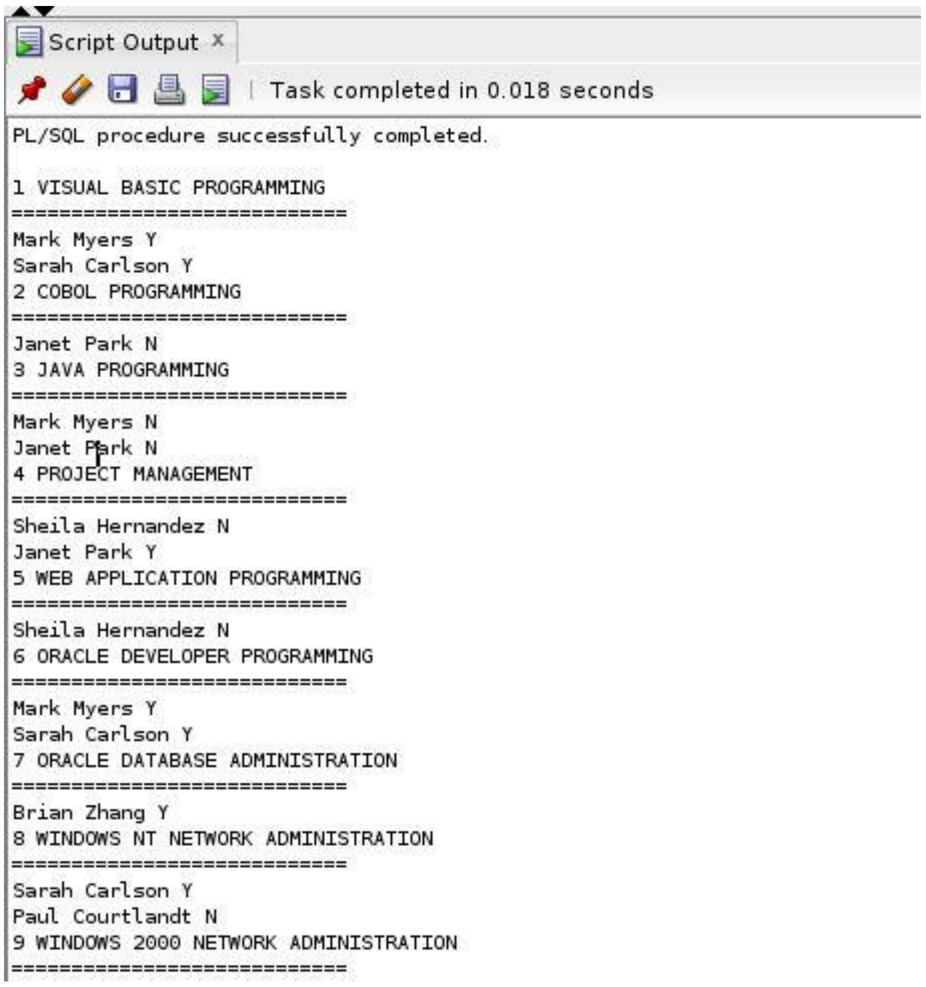
SET SERVEROUTPUT ON

DECLARE
  con_skill NUMBER(3,0);

```
  --cursor #1 for skills
  CURSOR skill_cursor IS
    SELECT s.skill_id, s.skill_description
    FROM skill s;

  --cursor #2 for consultants
  CURSOR consultant_skill_cursor IS
    SELECT con.c_first, con.c_last, con_s.certification
    FROM consultant con
    INNER JOIN consultant_skill con_s ON con_s.c_id = con.c_id
    WHERE con_s.skill_id = con_skill;
  consultant_skill_cursor_row consultant_skill_cursor%ROWTYPE;

  --create table of skill id and skill description
  TYPE s_id IS TABLE OF skill.skill_id%TYPE;
  TYPE s_description IS TABLE OF skill.skill_description%TYPE;

  skill_ids s_id;
  skill_descriptions s_description;
  inx1 PLS_INTEGER;

BEGIN
  OPEN skill_cursor;
  --use cursor to fill out skill table
  FETCH skill_cursor BULK COLLECT INTO skill_ids, skill_descriptions;
  CLOSE skill_cursor;

  --loop through skill table
  FOR inx1 IN 1..skill_ids.count LOOP
  skill_descriptions(inx1) := UPPER(skill_descriptions(inx1));
```

```
 --output skill id and description
 DBMS_OUTPUT.PUT_LINE (skill_ids(inx1) ||' ' || skill_descriptions(inx1));

 con_skill := skill_ids(inx1);

  DBMS_OUTPUT.PUT_LINE('===========================');

 --use consultant cursor to display all consultants with specific skill
 OPEN consultant_skill_cursor;
   FETCH consultant_skill_cursor INTO consultant_skill_cursor_row;
   LOOP
     EXIT WHEN consultant_skill_cursor%NOTFOUND;

        --output name and certification status
       DBMS_OUTPUT.PUT_LINE (consultant_skill_cursor_row.c_first
                        || ' '
                        || consultant_skill_cursor_row.c_last
                        || ' '
                        || consultant_skill_cursor_row.certification);

     FETCH consultant_skill_cursor INTO consultant_skill_cursor_row;
   END LOOP;
   CLOSE consultant_skill_cursor;


 END LOOP;
END;
```

Output:



```
PL/SQL procedure successfully completed.

1 VISUAL BASIC PROGRAMMING
============================
Mark Myers Y
Sarah Carlson Y
2 COBOL PROGRAMMING
============================
Janet Park N
3 JAVA PROGRAMMING
============================
Mark Myers N
Janet Park N
4 PROJECT MANAGEMENT
============================
Sheila Hernandez N
Janet Park Y
5 WEB APPLICATION PROGRAMMING
============================
Sheila Hernandez N
6 ORACLE DEVELOPER PROGRAMMING
============================
Mark Myers Y
Sarah Carlson Y
7 ORACLE DATABASE ADMINISTRATION
============================
Brian Zhang Y
8 WINDOWS NT NETWORK ADMINISTRATION
============================
Sarah Carlson Y
Paul Courtlandt N
9 WINDOWS 2000 NETWORK ADMINISTRATION
============================
```

Case 9:

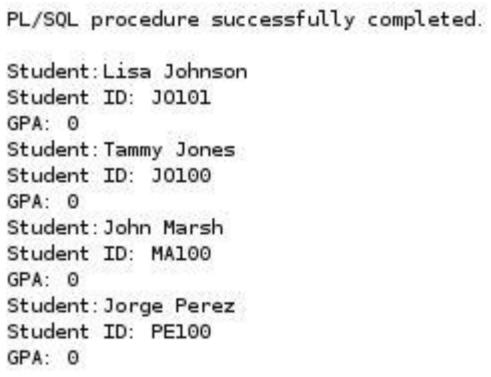
```
SET SERVEROUTPUT ON


DECLARE
total Number:=0;
grades number:=0;
gpa number:=0;
cursor student_cursor is
select  distinct s.s_id,s.s_first,s.s_last
```

```
from student s

left outer join enrollment e on e.s_id=s.s_id

where e.grade is NOT NULL;


student_cursor_row  student_cursor%ROWTYPE;


StudentID VARCHAR2(5);


cursor student_grade_cursor is

select s.s_id,e.grade

from student s

left outer join enrollment e on e.s_id=s.s_id

where s.s_id=StudentID

AND e.grade is not null;


student_grade_cursor_row student_grade_cursor%ROWTYPE;


BEGIN


OPEN student_cursor;

FETCH student_cursor into student_cursor_row;


LOOP

EXIT WHEN student_cursor%NOTFOUND;

DBMS_OUTPUT.PUT_LINE('Student:' ||student_cursor_row.s_first||

                    ' ' ||  student_cursor_row.s_last);


StudentID:=student_grade_cursor_row.s_id;

   Open student_grade_cursor;

   Fetch student_grade_cursor into student_grade_cursor_row;

   Loop
```

```
Exit when student_grade_cursor%NOTFOUND;
    if(student_grade_cursor_row.grade= 'A') then
    total:= total+4;
    elsif(student_grade_cursor_row.grade ='B') then
    total:=total+3;
    elsif(student_grade_cursor_row.grade ='C') then
    total:=total+2;
    elsif(student_grade_cursor_row.grade ='D') then
    total:=total+1;
    elsif(student_grade_cursor_row.grade ='F') then
    total:=total+0;
    else
    total :=total;
    END IF;
    grades :=grades+1;
    Fetch student_grade_cursor into student_grade_cursor_row;
    END LOOP;
    close student_grade_cursor;
    if(grades =0) then
    gpa :=0;
    else
    gpa :=total/grades;
    END IF;
    DBMS_OUTPUT.PUT_LINE('Student ID: '|| student_cursor_row.s_id);
    DBMS_OUTPUT.PUT_LINE('GPA: ' || round(gpa,2));

    gpa := 0;
    total:=0;
    grades:=0;
    Fetch student_cursor into student_cursor_row;
    END LOOP;
```

END;

Output:



```
PL/SQL procedure successfully completed.

Student:Lisa Johnson
Student ID: JO101
GPA: 0
Student:Tammy Jones
Student ID: JO100
GPA: 0
Student:John Marsh
Student ID: MA100
GPA: 0
Student:Jorge Perez
Student ID: PE100
GPA: 0
```

Case9 is Incomplete

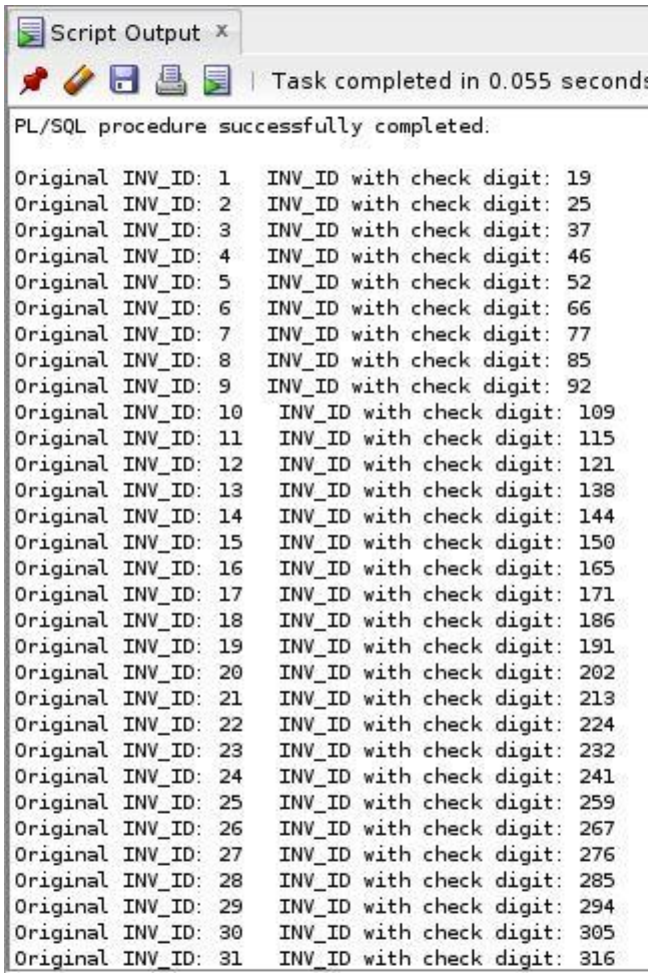
Case 10:

```
SET SERVEROUTPUT ON

DECLARE
  --variable for the new id including check digit
  new_inv_id NUMBER(11,0);
  --variable for the check digit
  check_digit NUMBER(2,0);
  --variable for the sum of the multiplication array
  check_sum NUMBER(11,0) := 0;

  --array of inv_ids
  type inv_char_arr is table of char(1) index by pls_integer;
  l_inv_arr inv_char_arr;

  --array of item_ids
  type item_char_arr is table of char(1) index by pls_integer;
  l_item_arr item_char_arr;
  --array for inv_ids * item_ids
  type sum_arr is table of NUMBER(10,0) index by pls_integer;
  l_sum_arr sum_arr;

  --cursor to iterate through items in inventory
  CURSOR inv_cursor IS
    SELECT inv.inv_id, inv.item_id
    FROM inventory inv;
  inv_cursor_row inv_cursor%ROWTYPE;

BEGIN
  OPEN inv_cursor;
  FETCH inv_cursor INTO inv_cursor_row;
  LOOP
    EXIT WHEN inv_cursor%NOTFOUND;

    --split inv_id into char array
    for i in 1 .. length(TO_CHAR(inv_cursor_row.inv_id))
      loop
      l_inv_arr(i) := substr( TO_CHAR(inv_cursor_row.inv_id), i, 1 );
    end loop;

    --split item_id into char array
    for i in 1 .. length(TO_CHAR(inv_cursor_row.item_id))
      loop
      l_item_arr(i) := substr( TO_CHAR(inv_cursor_row.item_id), i, 1 );
    end loop;

    --multiply both arrays together to get sum array
    for i in 1 .. l_inv_arr.count
```

```
     loop

     --if item_id is shorter than inv_id, use zeros for extra indexes
     IF (l_item_arr.EXISTS(i)) THEN
       l_sum_arr(i) := l_item_arr(i) * l_inv_arr(i);
       check_sum := check_sum + l_sum_arr(i);
     ELSE
       l_sum_arr(i) := 0 * l_inv_arr(i);
       check_sum := check_sum + l_sum_arr(i);
     END IF;
   end loop;

   --remainder of check_sum / 11
   check_sum := MOD(check_sum, 11);

   --check digit is 11 minus check_sum
   check_digit := 11-check_sum;

   --change result to single digit if = 10 or 11
   IF(check_digit = 10) THEN
     check_digit := 0;
   ELSIF(check_digit = 11) THEN
     check_digit := 1;
   END IF;

   --concatenate check digit to end of inv_id
   new_inv_id := TO_NUMBER(TO_CHAR(inv_cursor_row.inv_id) || TO_CHAR(check_digit));

   DBMS_OUTPUT.PUT_LINE('Original INV_ID: ' || inv_cursor_row.inv_id || '   INV_ID with check digit: '
|| new_inv_id);

   FETCH inv_cursor INTO inv_cursor_row;
```

```
  END LOOP;

  CLOSE inv_cursor;

END;
```

Output:



```
PL/SQL procedure successfully completed.

Original INV_ID: 1   INV_ID with check digit: 19
Original INV_ID: 2   INV_ID with check digit: 25
Original INV_ID: 3   INV_ID with check digit: 37
Original INV_ID: 4   INV_ID with check digit: 46
Original INV_ID: 5   INV_ID with check digit: 52
Original INV_ID: 6   INV_ID with check digit: 66
Original INV_ID: 7   INV_ID with check digit: 77
Original INV_ID: 8   INV_ID with check digit: 85
Original INV_ID: 9   INV_ID with check digit: 92
Original INV_ID: 10   INV_ID with check digit: 109
Original INV_ID: 11   INV_ID with check digit: 115
Original INV_ID: 12   INV_ID with check digit: 121
Original INV_ID: 13   INV_ID with check digit: 138
Original INV_ID: 14   INV_ID with check digit: 144
Original INV_ID: 15   INV_ID with check digit: 150
Original INV_ID: 16   INV_ID with check digit: 165
Original INV_ID: 17   INV_ID with check digit: 171
Original INV_ID: 18   INV_ID with check digit: 186
Original INV_ID: 19   INV_ID with check digit: 191
Original INV_ID: 20   INV_ID with check digit: 202
Original INV_ID: 21   INV_ID with check digit: 213
Original INV_ID: 22   INV_ID with check digit: 224
Original INV_ID: 23   INV_ID with check digit: 232
Original INV_ID: 24   INV_ID with check digit: 241
Original INV_ID: 25   INV_ID with check digit: 259
Original INV_ID: 26   INV_ID with check digit: 267
Original INV_ID: 27   INV_ID with check digit: 276
Original INV_ID: 28   INV_ID with check digit: 285
Original INV_ID: 29   INV_ID with check digit: 294
Original INV_ID: 30   INV_ID with check digit: 305
Original INV_ID: 31   INV_ID with check digit: 316
```

RECOMMENDATION

I recommend the answers and justifications developed in the Analysis section above for satisfaction of the Statement of Requirement.