Drexel University / CCI / INFO365

Homework Assignment 3

(due through Blackboard before class on 3/07/2018)

Things you should please do, lest you should lose my tender affection:

- Somehow incorporate your last name into the name of the solution file you upload.
- Make sure your name is inside the document itself, at the beginning or in the header.
- All code must be text inside of the document that I can copy and paste – do not include screenshots of code.
- Make sure that your work is original – do not copy your code from or share your code with anybody else in class.

Certification of Academic Honesty (see syllabus for details)

===

I certify that:

- **This homework assignment is entirely my own work.**
- **I have not quoted the words of any other person from a printed source or a website without indicating what has been quoted and providing an appropriate citation.**
- **I have not submitted this paper / project to satisfy the requirements of any other course.**

Signature: **Joseph Mulray**

Date : **3/6/2018**

==

1. Random Junk

I have a table called RandomJunk. Marvel at its beauty:

```
CREATE TABLE RandomJunk
(
 TextJunk VARCHAR2(10),
 NumJunk INTEGER
)
```

Now, for a limited time, you can have one, too! Run the attached code:

hw1_randomjunksetup.sql

Your table should now have 200 rows of random junk in it. Your assignment is to write an anonymous PL/SQL block that perform all of the following tasks.

Loop through every row in your table. For each row, check to see if the value of NumJunk is between the values returned by two separate calls to:

CEIL(DBMS_RANDOM.VALUE(0,1000))

Don't store the values you get by calling this expression – you should be making fresh calls to these for each row. (Basically, you're trying to compare each row against a range of values that was determined specifically for that row. Don't just create one range and use it for all rows.)

If you want to use the BETWEEN clause (e.g., BETWEEN v_val1 AND v_val2) to check whether the value of NumJunk is between the two values you generate for each row, note that BETWEEN expects its first argument to be lower in value than its second argument. (You can read more here: http://www.frein.com/what-between-means-in-sql-not-what-youd-think/.) So, you will have to compare the random values you generated for that row and use the lower one first.

If the value of NumJunkfor a given row is within the range between the two calls to the expression above, then you should set the value of TextJunk in the RandomJunk table to 'UPDATED' for that single row only.

However, if the value of NumJunk for a given row is not within the range between the two calls to the expression above, then you should add the TextJunk value for that row to some kind of collection variable, such as a varray. (Use the same collection variable for all rows.)

Using DBMS_OUTPUT, print out the NumJunk value for each row in RandomJunk that now has a TextJunk value of 'UPDATED'. Then, print out the contents of your collection.

Show your code and output.

```
DECLARE
        CURSOR num_cur IS
        SELECT * FROM RandomJunk FOR UPDATE;

        TYPE prod_varray IS VARRAY(200) OF RandomJunk%ROWTYPE;
        varray_prod PROD_VARRAY := prod_varray();

        rand1 NUMBER;
        rand2 NUMBER;

BEGIN

        FOR v_junk IN num_cur
        LOOP
                -- Update the random values each loop iteration
                rand1:=  CEIL(DBMS_RANDOM.VALUE(0,1000));
                rand2 :=  CEIL(DBMS_RANDOM.VALUE(0,1000));

                IF rand1 <= rand2 THEN
                        IF v_junk.NUMJUNK BETWEEN rand1 AND rand2 THEN
```

```
                    UPDATE RandomJunk
                    SET TEXTJUNK='UPDATED'
                    WHERE CURRENT OF num_cur;

                ELSE
                    varray_prod.EXTEND;
                    varray_prod(varray_prod.count) := v_junk;

                END IF;
            ELSE
                IF v_junk.NUMJUNK BETWEEN rand2 AND rand1 THEN
                    UPDATE RandomJunk
                    SET TEXTJUNK='UPDATED'
                    WHERE CURRENT OF num_cur;

                ELSE
                    varray_prod.EXTEND;
                    varray_prod(varray_prod.count) := v_junk;
                END IF;

            END IF;

        END LOOP;
        COMMIT;


        FOR i in varray_prod.FIRST .. varray_prod.LAST LOOP
            DBMS_OUTPUT.PUT_LINE(i || ' : ' || varray_prod(i).TEXTJUNK);
        END LOOP;

END;
/
```

OUTPUT:

Pasted some of the output below was 140 rows just pasted the first 50.

```
SQL> DECLARE
        CURSOR num_cur IS
        SELECT * FROM RandomJunk FOR UPDATE;

        TYPE prod_varray IS VARRAY(200) OF RandomJunk%ROWTYPE;
        varray_prod PROD_VARRAY := prod_varray();

        rand1 NUMBER;
        rand2 NUMBER;

BEGIN

        FOR v_junk IN num_cur
        LOOP
            -- Update the random values each loop iteration
            rand1:=  CEIL(DBMS_RANDOM.VALUE(0,1000));
```

```
                rand2 :=  CEIL(DBMS_RANDOM.VALUE(0,1000));

                IF rand1 <= rand2 THEN
                        IF v_junk.NUMJUNK BETWEEN rand1 AND rand2 THEN
                                UPDATE RandomJunk
                                SET TEXTJUNK='UPDATED'
                                WHERE CURRENT OF num_cur;

                        ELSE
                                varray_prod.EXTEND;
                                varray_prod(varray_prod.count) := v_junk;

                        END IF;
                ELSE
                        IF v_junk.NUMJUNK BETWEEN rand2 AND rand1 THEN
                                UPDATE RandomJunk
                                SET TEXTJUNK='UPDATED'
                                WHERE CURRENT OF num_cur;

                        ELSE
                                varray_prod.EXTEND;
                                varray_prod(varray_prod.count) := v_junk;
                        END IF;

                END IF;

        END LOOP;
        COMMIT;


        FOR i in varray_prod.FIRST .. varray_prod.LAST LOOP
                DBMS_OUTPUT.PUT_LINE(i || ' : ' || varray_prod(i).TEXTJUNK);
        END LOOP;

END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
 27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49   50
 51   52
1 : EALGNIRYAO
2 : VTSXBJCBJN
3 : RTFDQJBSTQ
4 : RNGVGEXPKK
5 : HANDVYMOLL
6 : HTDMFOIXQN
7 : COYBZYMLFU
8 : VAMJOCOCGM
9 : GCNCHKSJOP
10 : HVGKSRPVCI
```

```
11 : RZIMWHBTRF
12 : TGBDBIIKTH
13 : PIDRULGAKK
14 : UZRBFLWZZV
15 : YDIJOMHVAP
16 : TMCZHGHVIG
17 : ZSGCNXZCNT
18 : NLOZRTBKUP
19 : NHOVFCQMXA
20 : FGVHOOUPSC
21 : HMTTJQBKDU
22 : OOMFDJWCSZ
23 : ITCOMJUMQW
24 : HAYMHNUGXP
25 : JAGCEDRHLB
26 : MATTXQUSLR
27 : GKMFZYNXEG
28 : LFVHFJXGDR
29 : PPSNDXSUEC
30 : CRSPXEQODD
31 : DSZEXDCIUU
32 : JFNVEUTLNF
33 : CWQSZNKIUS
34 : WICEWZJEIB
35 : QODZRPQODW
36 : FVHIPCOFGX
37 : ZBWCQICJYO
38 : MWSUICBGQD
39 : WUBKINRHPK
40 : BNGHVDUJKZ
41 : OHAIZKQQKV
42 : NSSRIWSYOZ
43 : DBXODWBGRY
44 : RIBWLEXJTT
45 : SFLRNFHJVL
46 : WKEGQUPWYB
47 : YNRKIZDAJK
48 : OVMGTUYHDX
49 : OSAJIXJRHN
50 : VRHSVDHKUZ
```

2. Timestamp Testing

The Oracle keyword *systimestamp* returns a timestamp based on the current value of the system clock. Here is a query you could use to get just the microseconds portion of the current timestamp:

```
SELECT
```

```
        EXTRACT(SECOND FROM SYSTIMESTAMP)
                - TRUNC(EXTRACT(SECOND FROM SYSTIMESTAMP))
FROM dual;
```

("dual" is a fake table in Oracle – it's a way to dummy out a SQL statement when all you want to do is to select some system-generated value such as sysdate or systimestamp)

Create an anonymous block that checks the value of systimestamp, prints out this value, and then handles the value based on the rules below.

If the microseconds are less than or equal to 0.2, then print the phrase "Dirty deeds done dirt cheap."

If the microseconds are greater than 0.2 but less than or equal to 0.4, then print the phrase "John Wayne Gacy was a clown – that's freaky." (Print it exactly this way – no changes!)

If the microseconds are greater than 0.4 but less than or equal to 0.6, then print the phrase "Most XKCD cartoons are over my head."

If the microseconds are greater than 0.6 but less than or equal to 0.8, then print the phrase "Most Garfield cartoons are over my head."

If the microseconds are greater than 0.8, then print the phrase "How did Nicholas Cage ever become an action star?"

If the microseconds do not fall into any of the above categories, then print the phrase "I think I messed up one of my conditions."

Show your code, and the results of running the code 10 times. You can run it manually 10 times if you like – you don't have to control this part in the code itself.

```
DECLARE
        time number;

BEGIN
        SELECT
        EXTRACT(SECOND FROM SYSTIMESTAMP)
        - TRUNC(EXTRACT(SECOND FROM SYSTIMESTAMP)) INTO time
        FROM dual;

        DBMS_OUTPUT.PUT_LINE('Time: ' || time);

        IF time <= .2 THEN
                DBMS_OUTPUT.PUT_LINE('Dirty deeds done dirt cheap.');
        ELSIF time >.2 AND time <=.4 THEN
                DBMS_OUTPUT.PUT_LINE('John Wayne Gacy was a clown - that''s freaky.');
        ELSIF time >.4 AND time <=.6 THEN
                DBMS_OUTPUT.PUT_LINE('Most XKCD cartoons are over my head.');
        ELSIF time >.6 AND time <=.8 THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Most Garfield cartoons are over my head.');
    ELSIF time >.8 THEN
            DBMS_OUTPUT.PUT_LINE('How did Nicholas Cage ever become an action star?');
    ELSE
            DBMS_OUTPUT.PUT_LINE('I think I messed up one of my conditions.');
    END IF;

END;
/
```

OUTPUT:

Pasted from server output for the one entry and just pasted the output for the rest of the 9 outputs.

```
SQL> DECLARE
    time number;

BEGIN
    SELECT
    EXTRACT(SECOND FROM SYSTIMESTAMP)
    - TRUNC(EXTRACT(SECOND FROM SYSTIMESTAMP)) INTO time
    FROM dual;

    DBMS_OUTPUT.PUT_LINE('Time: ' || time);

    IF time <= .2 THEN
            DBMS_OUTPUT.PUT_LINE('Dirty deeds done dirt cheap.');
    ELSIF time >.2 AND time <=.4 THEN
            DBMS_OUTPUT.PUT_LINE('John Wayne Gacy was a clown – that''s freaky.');
    ELSIF time >.4 AND time <=.6 THEN
            DBMS_OUTPUT.PUT_LINE('Most XKCD cartoons are over my head.');
    ELSIF time >.6 AND time <=.8 THEN
            DBMS_OUTPUT.PUT_LINE('Most Garfield cartoons are over my head.');
    ELSIF time >.8 THEN
            DBMS_OUTPUT.PUT_LINE('How did Nicholas Cage ever become an action star?');
    ELSE
            DBMS_OUTPUT.PUT_LINE('I think I messed up one of my conditions.');
    END IF;

END;
/
  2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
27
Time: .210758
John Wayne Gacy was a clown - that's freaky.

PL/SQL procedure successfully completed.

SQL>
```

```
Time: .187666
Dirty deeds done dirt cheap.
PL/SQL procedure successfully completed.


Time: .900243
How did Nicholas Cage ever become an action star?
PL/SQL procedure successfully completed.


Time: .309161
John Wayne Gacy was a clown - that's freaky.
PL/SQL procedure successfully completed.

Time: .173185
Dirty deeds done dirt cheap.
PL/SQL procedure successfully completed.

Time: .188798
Dirty deeds done dirt cheap.
PL/SQL procedure successfully completed.


Time: .821061
How did Nicholas Cage ever become an action star?
PL/SQL procedure successfully completed.

Time: .510481
Most XKCD cartoons are over my head.
PL/SQL procedure successfully completed.

Time: .326685
John Wayne Gacy was a clown - that's freaky.
PL/SQL procedure successfully completed.

Time: .151164
Dirty deeds done dirt cheap.
PL/SQL procedure successfully completed.
```

3. Product Logic

Execute the following code to create a Products table.

hw1_productssetup.sql

Write an anonymous block that inspects all of the records from the Products table in descending order of the values in the ProdName field.

For each product record, if

a) the ProdName contains the string “Mega”
   AND
b) and the ProdName of the record inspected immediately prior to the current record contains the string “Glider”

then print out the ProductId and ProdName of the current product record.

If a product record does not meet the criteria (a and b) outlined above, then print out the value “NO MATCH FOR PRODUCT.”

Obviously, the first record inspected has no prior records to consider, and hence does not satisfy condition b.

Show your code and the output from your block.

```
DECLARE
        CURSOR prod_cur IS
        SELECT * FROM Products
        ORDER BY PRODUCTID DESC;

        previous Products%ROWTYPE;

BEGIN
        FOR v_prod IN prod_cur
        LOOP

        IF v_prod.PRODNAME LIKE '%Mega%' AND previous.PRODNAME LIKE '%Glider%' THEN
                DBMS_OUTPUT.PUT_LINE('PRODUCTID:' || v_prod.PRODUCTID || '
PRODNAME:' || v_prod.PRODNAME);

        ELSE
                DBMS_OUTPUT.PUT_LINE('NO MATCH FOR PRODUCT');

        -- Set record to previous value for comparison on next product
        previous := v_prod;

        END IF;

        END LOOP;
END;
```

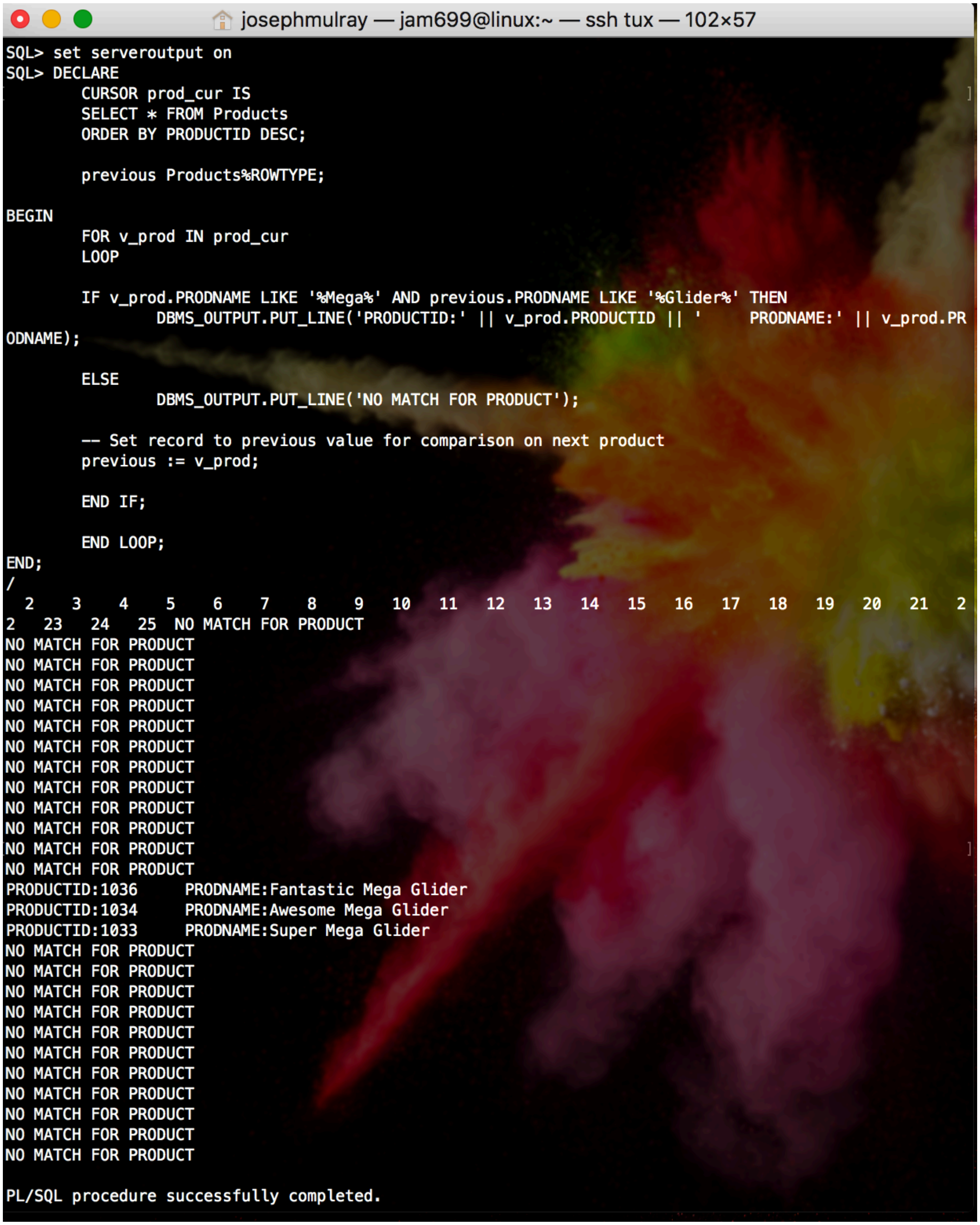
/

OUTPUT:

```
SQL>
DECLARE
        CURSOR prod_cur IS
        SELECT * FROM Products
        ORDER BY PRODUCTID DESC;

        previous Products%ROWTYPE;

BEGIN
        FOR v_prod IN prod_cur
        LOOP

        IF v_prod.PRODNAME LIKE '%Mega%' AND previous.PRODNAME LIKE '%Glider%' THEN
                DBMS_OUTPUT.PUT_LINE('PRODUCTID:'      ||      v_prod.PRODUCTID      ||      '
PRODNAME:' || v_prod.PRODNAME);
                -- DBMS_OUTPUT.PUT_LINE();

        ELSE
                DBMS_OUTPUT.PUT_LINE('NO MATCH FOR PRODUCT');

        -- Set record to previous value for comparison on next product
        previous := v_prod;

        END IF;

        END LOOP;
END;
/SQL>  2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
26
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
PRODUCTID:1036      PRODNAME:Fantastic Mega Glider
PRODUCTID:1034      PRODNAME:Awesome Mega Glider
```

```
PRODUCTID:1033       PRODNAME:Super Mega Glider
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT

PL/SQL procedure successfully completed.

SQL>
```

Just outputted a screenshot I could fit below of the block executing.



```
SQL> set serveroutput on
SQL> DECLARE
        CURSOR prod_cur IS
        SELECT * FROM Products
        ORDER BY PRODUCTID DESC;

        previous Products%ROWTYPE;

BEGIN
        FOR v_prod IN prod_cur
        LOOP

        IF v_prod.PRODNAME LIKE '%Mega%' AND previous.PRODNAME LIKE '%Glider%' THEN
                DBMS_OUTPUT.PUT_LINE('PRODUCTID:' || v_prod.PRODUCTID || '      PRODNAME:' || v_prod.PR
ODNAME);

        ELSE
                DBMS_OUTPUT.PUT_LINE('NO MATCH FOR PRODUCT');

        -- Set record to previous value for comparison on next product
        previous := v_prod;

        END IF;

        END LOOP;
END;
/
  2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   2
2   23   24   25  NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
PRODUCTID:1036     PRODNAME:Fantastic Mega Glider
PRODUCTID:1034     PRODNAME:Awesome Mega Glider
PRODUCTID:1033     PRODNAME:Super Mega Glider
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT
NO MATCH FOR PRODUCT

PL/SQL procedure successfully completed.
```

<<The End>>