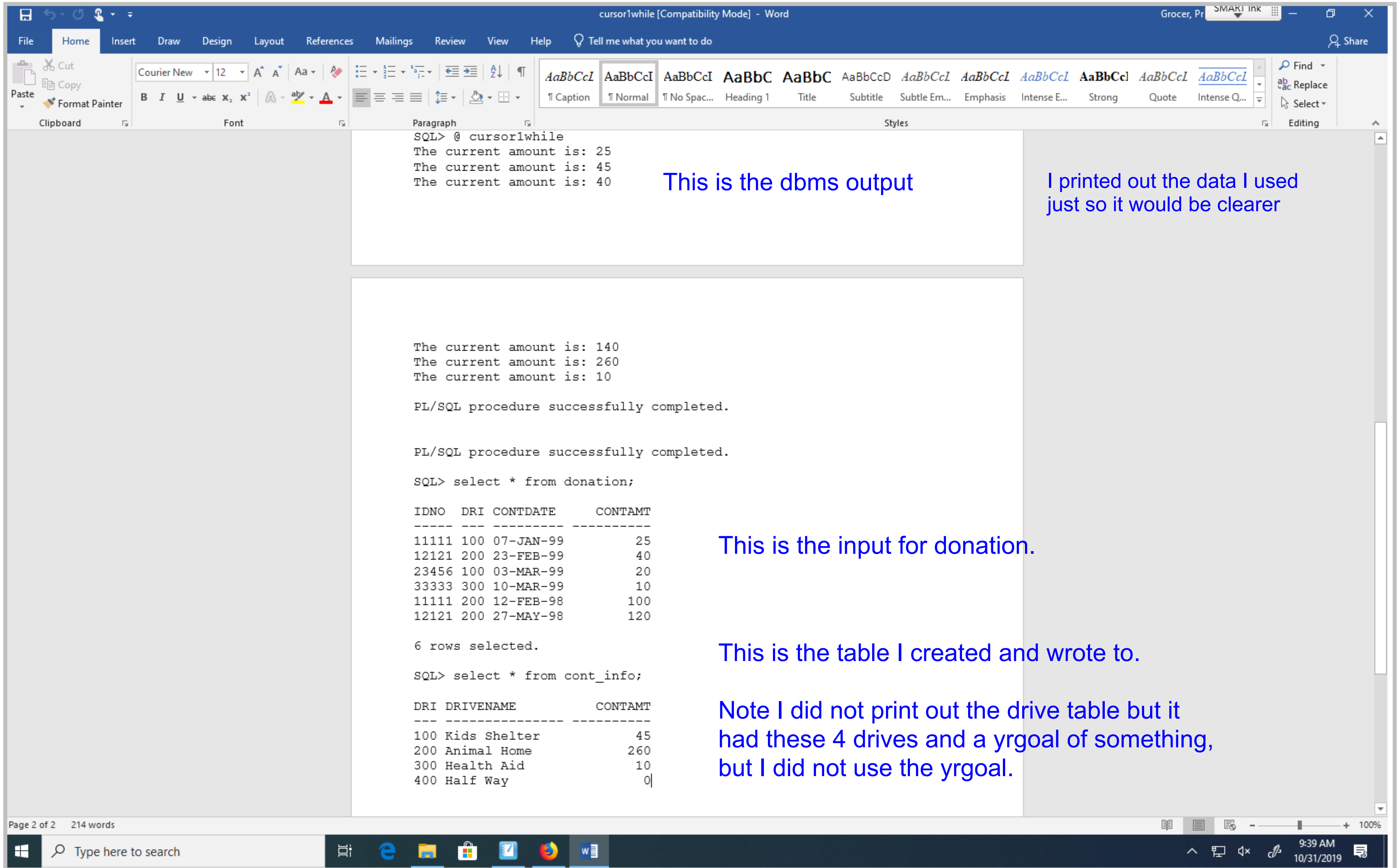

```
SQL> @ cursor1while
The current amount is: 25
The current amount is: 45
The current amount is: 40
```

This is the dbms output

I printed out the data I used just so it would be clearer

```
The current amount is: 140
The current amount is: 260
The current amount is: 10

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL> select * from donation;

IDNO  DRI CONTDATE     CONTAMT
----- --- --------- ----------
11111 100 07-JAN-99         25
12121 200 23-FEB-99         40
23456 100 03-MAR-99         20
33333 300 10-MAR-99         10
11111 200 12-FEB-98        100
12121 200 27-MAY-98        120

6 rows selected.

SQL> select * from cont_info;

DRI DRIVENAME          CONTAMT
--- --------------- ----------
100 Kids Shelter            45
200 Animal Home            260
300 Health Aid              10
400 Half Way                 0
```

This is the input for donation.

This is the table I created and wrote to.

Note I did not print out the drive table but it had these 4 drives and a yrgoal of something, but I did not use the yrgoal.

Title: Oct 31-9:28 AM (1 of 16)

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

drive_cursor

100 KS
200 AH
300 HA
400 HW

donation_cursor

The cursor does not get filled until it is opened.

This is the program I stepped through and it is in the notes - I also printed it on the data sheet but used this version to step through the code.

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME       CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

Type here to search

9:48 AM
10/31/2019

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

100
KS
25

drive_cursor

100	KS
200	AH
300	HA
400	HW

donation_cursor

25
20

V_tot_contamt

0

When I fetch from the drive_cursor 100 and Kids Shelter (KS) get stored in my variables. Then when I open donation_cursor it follows the select in the cursor and gets the contamt where the drivenno on the table record is equal to the v_drive_no that I just got when I fetched from the drive_cursor. There are two records with a driveno of 100 and there contamts are 25 and 20.

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME         CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

Type here to search

9:57 AM
10/31/2019

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

100
KS

~~25~~
20

drive_cursor

100 KS
200 AH
300 HA
400 HW

donation_cursor

25
20

V_tot_contamt

~~0~~
25

The current amount is: 25

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME        CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

Type here to search

9:59 AM
10/31/2019

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

100
KS

~~25~~
20

drive_cursor

100	KS
200	AH
300	HA
400	HW

donation_cursor

25
20

V_tot_contamt

~~0~~
~~25~~
45

table cont_info

100 KS 45

The current amount is: ~~25~~ 45

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME       CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

Type here to search

10:01 AM
10/31/2019

Continuation of first cursor examp

www.pgrocer.net/Cis50/curfstcont.html

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

~~100 KS~~ 200 AH

~~25~~ 20

drive_cursor

100	KS
200	AH
300	HA
400	HW

donation_cursor

40
100
120

V_tot_contamt

~~0~~
~~25~~
45

table cont_info

100 KS 45

The current amount is: 25
: 45

I closed the donation_cursor and opened it to get the matches to driveno 200.

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME       CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no       drive.driveno%TYPE;
  v_drive_name     drive.drivename%TYPE;
  v_contamt        donation.contamt%TYPE;
  v_tot_contamt    cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME       CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

100 KS

200 AH

~~25~~ 20

~~40~~ ~~100~~ 120

drive_cursor

100	KS
200	AH
300	HA
400	HW

donation_cursor

40

100

120

V_tot_contamt

∅

~~40~~

~~140~~

260

table cont_info

100 KS 45

200 AH 260

The current amount is: 25

45

40

140

260

Type here to search

10:10 AM
10/31/2019

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME          CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

I have opened donation_cursor for the third time and it now has 10 since there is only one record with driveno 300.

drive_cursor

100 KS
200 AH
300 HA
400 HW

donation_cursor

10

V_tot_contamt

0
40
140
260

table cont_info

100 KS 45
200 AH 260

The current amount is: 25
45
40
140
260

100 KS
200 AH
3 HA

25
20

40
100
120

10:12 AM 10/31/2019

Type here to search

Title: Oct 31-10:12 AM (8 of 16)

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

SQL CODE:

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
         FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME          CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

100 KS 200 AH 300 HA 100 ~~25~~ 20 ~~40~~ 10 ~~100~~ 120

drive_cursor

100 KS
200 AH
300 HA
400 HW

donation_cursor

10

v_tot_contamt

[∅ 16]

table cont_info

45

100 KS
200 AH 260
10
300 HA

The current amount is: 25
: 45
40
140
260
10

Type here to search

10:14 AM 10/31/2019

File Edit View History Bookmarks Tools Help

Continuation of first cursor exampl

www.pgrocer.net/Cis50/curfstcont.html

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     FETCH drive_cursor INTO v_drive_no, v_drive_name;
     WHILE drive_cursor%FOUND LOOP
       IF donation_cursor%ISOPEN THEN
          CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       FETCH donation_cursor INTO v_contamt;
       WHILE donation_cursor%FOUND LOOP
          v_tot_contamt := v_tot_contamt + v_contamt;
          dbms_output.put_line('The current amount is: '||v_tot_contamt);
          FETCH donation_cursor INTO v_contamt;
       END LOOP;
       INSERT into cont_info
         VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

SQL CODE:

```
SQL> @ cursor7b
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME       CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

drive_cursor

100 KS
200 AH
300 HA
400 HW

donation_cursor

nothing

v_tot_contamt

0

table cont_info

100 KS 45
200 AH 260
300 HA 10
400 HW 0

The current amount is: 25
: 45
40
140
260
10

First cursor example

www.pgrocer.net/Cis50/cursorexampl1.html

```
SET SERVEROUTPUT ON
DECLARE
  v_drive_no      drive.driveno%TYPE;
  v_drive_name    drive.drivename%TYPE;
  v_contamt       donation.contamt%TYPE;
  v_tot_contamt   cont_info.contamt%TYPE;
  CURSOR drive_cursor IS
     SELECT driveno, drivename FROM drive
     ORDER BY driveno;
  CURSOR donation_cursor IS
     SELECT contamt FROM donation
     WHERE v_drive_no = driveno
     ORDER BY driveno;
  BEGIN
     OPEN drive_cursor;
     LOOP
       FETCH drive_cursor INTO v_drive_no, v_drive_name;
       EXIT WHEN drive_cursor%NOTFOUND;
       IF donation_cursor%ISOPEN THEN
         CLOSE donation_cursor;
       END IF;
       OPEN donation_cursor;
       v_tot_contamt := 0;
       LOOP
         FETCH donation_cursor INTO v_contamt;
         EXIT WHEN donation_cursor%NOTFOUND;
         v_tot_contamt := v_tot_contamt + v_contamt;
         dbms_output.put_line('The current amount is: '||v_tot_contamt);
       END LOOP;
       INSERT into cont_info
          VALUES(v_drive_no, v_drive_name, v_tot_contamt);
       CLOSE donation_cursor;
     END LOOP;
     CLOSE drive_cursor;
END;
/
SET SERVEROUTPUT OFF
```

Not considered a good idea to have an EXIT embedded in a loop to prevent the whole loop from being executed.

Fetch
Loop
{
Fetch
Exit when

If found
Process
else
exit

SQL CODE:

```
SQL> @ cursor6a
The current amount is: 25
The current amount is: 45
The current amount is: 55
The current amount is: 105
The current amount is: 40
The current amount is: 75
The current amount is: 10
The current amount is: 20

PL/SQL procedure successfully completed.

Input truncated to 20 characters
SQL> SELECT * FROM cont_info;

DRI DRIVENAME         CONTAMT
--- --------------- ---------
100 Kids Shelter          105
```

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

Oracle PL/SQL Loops

One of the key things that PL/SQL offers is the ability to set up loops and to process records individually within the loop. There are several looping styles, we will start with what is defined as a BASIC loop. Within the basic loop, the exit statement is used to terminate the loop. The syntax for the BASIC LOOP is:

```
LOOP
     Processing code within the loop
     EXIT which can include a WHEN condition
        (note there can be multiple EXIT statements)
END LOOP;
```

First I am creating a table that I will use in at least some of the example that test the loop. It contains a record number, a counter and a column for whatever.

SQL CODE:

```
SQL> CREATE TABLE testloop
  2  (rec_no NUMBER(3), ctr NUMBER(3), data_in VARCHAR2(10));

Table created.

SQL> DESC testloop;
 Name                            Null?    Type
 ------------------------------- -------- ----
 REC_NO                                   NUMBER(3)
 CTR                                      NUMBER(3)
 DATA_IN                                  VARCHAR2(10)

SQL> edit basicloop1
```

This is an example of a basic loop. I have set up two variables, v_recno which I initialized at 1 and v_ctr which I initialized at 100. In the processing part of the block I inserted a LOOP which will insert the v_recno into the rec_no column on the table and the v_ctr into the ctr column of the table. After doing the insert, I incremented the v_recno by 1 and the v_ctr by 10. Then I used the IF to test if the v_recno was greater than 5. If it was I coded EXIT which will leave the loop. If it wasn't the if ended. When the END LOOP was encountered the loop was restarted which meant that another record was inserted, the variables were incremented and the test was made. This loop will continue until the IF statement tests true for v_recno > 5. At that point the EXIT statement takes processing outside the loop. Since there are no other instructions the block is complete and the message procedure is successfully completed will be displayed. I then did a select on the testloop table and displayed the records that were produced as a result of this loop. Notice that v_recno runs from 1 to 5. After the row with 5 is inserted, the code added one making v_recno 6 and that is when the IF statement caused the EXIT to happen. Notice also that v_ctr started at 100 and was incremented by 10 each pass through the loop. No data was put in the column data_in.

EXIT can be inserted multiple times in the code. From a programming point of view, I prefer EXIT once at the bottom of the loop to avoid confusion. Using EXIT to leave the loop before it is complete is a programming practice that I try to avoid.

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
```

Type here to search

10:28 AM 10/31/2019

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

```
SQL> SELECT * FROM testloop;

no rows selected
```

WHILE LOOP

The WHILE loop allows the programmer to put a condition on the WHILE statement that starts the loop. The syntax is:

WHILE condition LOOP
    Processing code
END LOOP;

This format calls for the condition to be evaluated at the start of each iteration of the loop. If the condition is TRUE then the loop is processed, if the condition is FALSE then the loop is not processed. With loops it is important to have something to exit the loop and something that impacts or changes the condition on which you base that exit.

In the example below, the WHILE condition is v_recno < 6 which means the processing will continue until v_recno is no longer less than 6. Adding 1 to v_recno is the statement that impacts the condtion. When the add results in 6, the loop is stopped and so the insert for rec_no 6 never happens.

SQL CODE:

```
SQL> edit whileloop1
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   WHILE v_recno < 6 LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ whileloop1
Input truncated to 1 characters

PL/SQL procedure successfully completed.
SQL> SELECT * FROM testloop;

   REC_NO       CTR DATA_IN
--------- --------- ----------
        1       100
        2       110
        3       120
```

Type here to search

10:29 AM
10/31/2019

File Edit View History Bookmarks Tools Help

PL/SQL Loops

www.pgrocer.net/Cis50/orloops.html

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=6;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   WHILE v_recno < 6 LOOP
     INSERT INTO testloop(rec_no, ctr)
        VALUES(v_recno, v_ctr);
     v_recno :=v_recno + 1;
     v_ctr := v_ctr + 10;
   END LOOP;
END;
/
```

SQL CODE:

```
SQL> @ whileloop2
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM testloop;

no rows selected
```

FOR LOOP

The FOR loop controls the iterations in the FOR statement. The syntax is:

FOR index in [REVERSE] lower_bound..upper_bound LOOP
        Processing code
END LOOP;

The index is an implicitly declared integer which is automatically incremented by 1 on each iteration until the appropriate bound is reached. Notice that the index is not declared by the coder and can only be used in the loop specifications. Without the reverse clause, 1 is added starting from the lower_bound until the upper_bound is reached. With the reverse clause, everything is reverse and the upper_bound is decreased until the lower_bound is reached. The lower_bound and upper_bound can be literals or variable names or conditions that evaluate to an integer.

In the example below, the index is called I and the loop runs with the index starting at 1 and running through 5. The insert and incrementing of v_recno and v_ctr are strictly to supply data for the rows that are being inserted.

SQL CODE:

```
SQL> edit forloop1
```

PL/SQL CODE:

```
DECLARE
   v_recno    testloop.rec_no%TYPE :=1;
   v_ctr      testloop.ctr%TYPE :=100;
BEGIN
   FOR i IN 1..5 LOOP
```

Type here to search

10:29 AM
10/31/2019

Title: Oct 31-10:29 AM (14 of 16)

File Edit View History Bookmarks Tools Help

Pay raise examples

www.pgrocer.net/Cis50/payraise.html

I then enter the inner loop for this aspect of the processing which calls for processing until the v_LoopCnt is less than or equal to the count of the number of elements in the table. The IF test in this loop tests to see if the employee id matches one of the manager ids in the table. The loop continues until it has checked the employee id against all of the elements in the table or until a match is found. If a match is found, the flag is set to TRUE and the loop is exited. Note that each pass through the loop increments v_LoopCnt.

In summary. In the inner loop the employee id was checked against all manager ids in the table. If no match was found the flag stays at false, if a match was found the flag is set to true and the loop is terminated.

After the loop ends, the flag is checked. If the flag is true the salary is multiplied by 1.1 and if the flag is not true the salary is multiplied by 1.02. Then the table is updated setting the salary in the table to the newly calculated salary for the current record. At the end loop, the outer loop will be re-entered and the fetch will get a new employee id and salary and the processing will continue.

t_Manager Table

When the second cursor contains no more entries, the outer loop is terminated and the cursor is closed.

The output from the second program with the salary updates is shown below.

SQL CODE:

```
SQL> @ pay_raise_3
Input truncated to 1 characters

PL/SQL procedure successfully completed.

SQL> SELECT * FROM employeez;

EMPLOYEE_ID LAST_NAME       FIRST_NAME      MANAGER_ID     SALARY
----------- --------------- --------------- ---------- ----------
       7369 SMITH           JOHN                  7902        816
       7499 ALLEN           KEVIN                 7698       1632
       7505 DOYLE           JEAN                  7839       3135
       7506 DENNIS          LYNN                  7839       3025
       7507 BAKER           LESLIE                7839       2420
       7521 WARD            CYNTHIA               7698       1275
       7555 PETERS          DANIEL                7505       1275
       7557 SHAW            KAREN                 7505       1275
       7560 DUNCAN          SARAH                 7506       1275
       7564 LANGE           GREGORY               7506       1275
       7566 JONES           TERRY                 7839     3272.5
       7569 ALBERTS         CHRIS                 7839       3300
       7600 PORTER          RAYMOND               7505       1275
       7609 LEWIS           RICHARD               7507       1836
       7654 MARTIN          KENNETH               7698       1275
       7676 SOMMERS         DENISE                7507       1887
       7698 BLAKE           MARION                7839       3135
       7782 CLARK           CAROL                 7839       2695
       7788 SCOTT           DONALD                7566       3300
       7789 WEST            LIVIA                 7506       1530
       7799 FISHER          MATTHEW               7569       3300

EMPLOYEE_ID LAST_NAME       FIRST_NAME      MANAGER_ID     SALARY
----------- --------------- --------------- ---------- ----------
       7820 ROSS            PAUL                  7505       1326
       7839 KING            FRANCIS                          5500
       7844 TURNER          MARY                  7698       1530
       7876 ADAMS           DIANE                 7788       1122
       7900 JAMES           FRED                  7698        969
       7902 FORD            JENNIFER              7566       3300
       7916 ROBERTS         GRACE                 7569     2932.5
       7919 DOUGLAS         MICHAEL               7799        816
       7934 MILLER          BARBARA               7782       1326
       7950 JENSEN          ALICE                 7505        765
```

Introduction to next class.

Type here to search

10:39 AM
10/31/2019

```
7788
7799
7782

PL/SQL procedure successfully completed.
```

Second version of the program - this one actually gives pay raises:

This version of the program updates the file to give managers a 10% raise and non managers a 2% raise. The found flag is used to make this determination.

t_ManagerTable

7902 ← v_ManagerIdTable

7863 ← v_ManagerIdTable

7863

PL/SQL CODE:

```
DECLARE
        TYPE t_ManagerTable IS TABLE OF employeez.manager_id%TYPE
          INDEX BY BINARY_INTEGER;
        v_ManagerIdTable t_ManagerTable;
        CURSOR c_Employee1 IS
          SELECT manager_id FROM employeez;
        CURSOR c_Employee2 IS
          SELECT employee_id, salary FROM employeez
          FOR UPDATE OF salary;
        v_ManagerId employeez.manager_id%TYPE;
        v_EmployeeId employeez.employee_id%TYPE;
        v_Salary employeez.salary%TYPE;
        v_FoundFlag BOOLEAN;
        v_LoopCnt BINARY_INTEGER;
        v_TableIndx BINARY_INTEGER;

BEGIN
        OPEN c_Employee1;
        LOOP
          FETCH c_Employee1 INTO v_ManagerId;
          EXIT WHEN c_Employee1%NOTFOUND;
          v_FoundFlag := FALSE;
          v_LoopCnt := 1;
          WHILE v_LoopCnt <= v_ManagerIdTable.COUNT LOOP
            IF v_ManagerId = v_ManagerIdTable(v_LoopCnt) THEN
              v_FoundFlag := TRUE;
              EXIT;
            END IF;
            v_LoopCnt := v_LoopCnt + 1;
          END LOOP;
          IF v_FoundFlag = FALSE THEN
            v_ManagerIdTable(v_ManagerIdTable.COUNT + 1) := v_ManagerId;
          END IF;
        END LOOP;
        CLOSE c_Employee1;
        OPEN c_Employee2;
        LOOP
          FETCH c_Employee2 INTO v_EmployeeId, v_Salary;
          EXIT WHEN c_Employee2%NOTFOUND;
          v_FoundFlag := FALSE;
          v_LoopCnt := 1;
          WHILE v_LoopCnt <= v_ManagerIdTable.COUNT LOOP
            IF v_EmployeeId = v_ManagerIdTable(v_LoopCnt) THEN
              v_FoundFlag := TRUE;
              EXIT;
```