Examples of

PL/SQL Programs

type slash then press Enter to execute the program

your output will be different

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> --PL/SQL program to display the current date
SQL> DECLARE
  2     TodaysDate DATE;
  3  BEGIN
  4     TodaysDate := SYSDATE;
  5     DBMS_OUTPUT.PUT_LINE('Today''s date is ');
  6     DBMS_OUTPUT.PUT_LINE(TodaysDate);
  7  END;
  8  /
Today's date is
28-NOV-01

PL/SQL procedure successfully completed.
```

Figure 4-6: PL/SQL program output

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> --PL/SQL program to display the current date
SQL> DECLARE
  2     TodaysDate DATE;
  3  BEGIN
  4     TodaysDate := SYSDATE;
  5     DBMS_OUTPUT.PUT_LINE('Today''s date is ' || TO_CHAR(TodaysDate));
  6  END;
  7  /
Today's date is 28-NOV-01

PL/SQL procedure successfully completed.
```

Figure 4-8: PL/SQL program with output concatenated on a single line

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2     TodaysDate DATE;
  3     Today VARCHAR2 (9);
  4     DayLength BINARY_INTEGER;
  5  BEGIN
  6     TodaysDate := SYSDATE;
  7     Today := TO_CHAR(SYSDATE, 'DAY');
  8     Today := RTRIM(Today);
  9     -- convert day display to lowercase
 10     Today := LOWER(Today);
 11     DayLength := LENGTH(Today);
 12     DBMS_OUTPUT.PUT_LINE('Today is ' || Today || ', ' || TO_CHAR(TodaysDa
 13     DBMS_OUTPUT.PUT_LINE('The length of the word ' || Today ||
 14     ' is ' || TO_CHAR(DayLength) || ' characters.');
 15  END;
 16  /
Today is wednesday, 28-NOV-01
The length of the word wednesday is 9 characters.

PL/SQL procedure successfully completed.
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> --PL/SQL program to display the current day
SQL> DECLARE
  2     Today VARCHAR2(9);
  3  BEGIN
  4     Today := TO_CHAR(SYSDATE, 'DAY');
  5     Today := RTRIM(Today);
  6    --add IF/THEN statement to determine if current day is Friday
  7    IF Today != 'FRIDAY' THEN
  8      DBMS_OUTPUT.PUT_LINE('Today is not Friday');
  9    END IF;
 10  END;
 11  /
Today is not Friday
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> --PL/SQL program to display the current day
SQL> DECLARE
  2     Today VARCHAR2(9);
  3  BEGIN
  4     Today := TO_CHAR(SYSDATE, 'DAY');
  5     Today := RTRIM(Today);
  6    --add IF/THEN statement to determine if current day is Friday
  7    IF Today = 'FRIDAY' THEN
  8      DBMS_OUTPUT.PUT_LINE('Today is Friday');
  9    ELSE
 10      DBMS_OUTPUT.PUT_LINE('Today is not Friday');
 11    END IF;
 12  END;
 13  /
Today is not Friday
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2     TermID BINARY_INTEGER;
  3     TermDesc VARCHAR2(20);
  4     TermStatus VARCHAR2(20);
  5   BEGIN
  6     TermID := 1;
  7     TermDesc := 'Fall 2000';
  8     TermStatus := 'CLOSED';
  9     -- insert the records
 10     INSERT INTO term VALUES (TermID, TermDesc, TermStatus);
 11     TermID := TermID + 1;
 12     TermDesc := 'Spring 2001';
 13     INSERT INTO term VALUES (TermID, TermDesc, TermStatus);
 14     TermID := TermID + 1;
 15     TermDesc := 'Summer 2001';
 16     INSERT INTO term VALUES (TermID, TermDesc, TermStatus);
 17     COMMIT;
 18   END;
 19   /
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2    LoopCount BINARY_INTEGER;
  3  BEGIN
  4    LoopCount := 0;
  5    LOOP
  6      LoopCount := LoopCount + 1;
  7      IF LoopCount = 6 THEN
  8        EXIT;
  9      END IF;
 10      INSERT INTO count_table VALUES(LoopCount);
 11    END LOOP;
 12  END;
 13  /

PL/SQL procedure successfully completed.

SQL> SELECT *
  2  FROM count_table;

    COUNTER
----------
         1
         2
         3
         4
         5
```

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> BEGIN
  2     FOR LoopCount IN 1..5
  3     LOOP
  4        INSERT INTO count_table VALUES(LoopCount);
  5     END LOOP;
  6  END;
  7  /

PL/SQL procedure successfully completed.

SQL> SELECT *
  2  FROM count_table;

   COUNTER
----------
         1
         2
         3
         4
         5
```

Exception Handling

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2     CurrentItemID BINARY_INTEGER;
  3     CurrentItemDesc  VARCHAR2(30);
  4   BEGIN
  5     CurrentItemID := 890;
  6     SELECT ItemDesc
  7     INTO CurrentItemDesc
  8     FROM item
  9     WHERE itemid = CurrentItemID;
 10     DBMS_OUTPUT.PUT_LINE('The description of Item ID ' ||
 11     TO_CHAR(CurrentItemID) || ' is ' || CurrentItemDesc || '.');
 12   END;
 13   /
DECLARE
*
ERROR at line 1:
ORA-01403: no data found
ORA-06512: at line 6
```

invalid item ID

type this code

variable to hold error message

query to return multiple rows

WHEN OTHERS exception code

error code and message

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2    CurrentItemID BINARY_INTEGER;
  3    CurrentItemDesc  VARCHAR2(30);
  4    ErrorMessage VARCHAR2(512);
  5  BEGIN
  6    CurrentItemID := 894;
  7    SELECT itemdesc
  8    INTO CurrentItemDesc
  9    FROM item, inventory
 10    WHERE item.itemid = CurrentItemID
 11    AND item.itemid = inventory.itemid;
 12    DBMS_OUTPUT.PUT_LINE('The description of Item ID ' ||
 13    TO_CHAR(CurrentItemID) || ' is ' || CurrentItemDesc || '.');
 14  EXCEPTION
 15    WHEN NO_DATA_FOUND THEN
 16     DBMS_OUTPUT.PUT_LINE('Item ID ' || CurrentItemID || ' is invalid.');
 17     DBMS_OUTPUT.PUT_LINE('Please enter a valid value.');
 18    WHEN OTHERS THEN
 19     ErrorMessage := SQLERRM;
 20     DBMS_OUTPUT.PUT_LINE('This program encountered the following error: ')
 21     DBMS_OUTPUT.PUT_LINE(ErrorMessage);
 22  END;
 23  /
This program encountered the following error:
ORA-01422: exact fetch returns more than requested number of rows
```

Figure 4-50: Displaying the Oracle error code and message in the WHEN OTHERS exception handler

exception declaration for undefined Oracle error

exception handler

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2    TermID  BINARY_INTEGER;
  3    TermDesc  VARCHAR2(20);
  4    e_NotNullInsert EXCEPTION;
  5    PRAGMA EXCEPTION_INIT (e_NotNullInsert, -1400);
  6  BEGIN
  7    TermID := 7;
  8    TermDesc := 'Fall 2002';
  9    -- insert the records
 10    INSERT INTO term VALUES (TermID, TermDesc, NULL);
 11    COMMIT;
 12  EXCEPTION
 13    WHEN e_NotNullInsert THEN
 14      DBMS_OUTPUT.PUT_LINE
 15      ('You must specify a data value for all TERM fields');
 16  END;
 17  /
You must specify a data value for all TERM fields

PL/SQL procedure successfully completed.
```

Figure 4-52: PL/SQL program with exception handler for undefined exception

exception declaration

record is deleted if grade is NULL

exception is raised if grade is not NULL

exception handler

exception message

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2     CurrentSID  BINARY_INTEGER;
  3     CurrentCSECID  BINARY_INTEGER;
  4     CurrentGrade  CHAR(1);
  5     e_NullGradeDelete EXCEPTION;
  6   BEGIN
  7     CurrentSID := 100;
  8     CurrentCSECID := 1000;
  9     SELECT grade INTO CurrentGrade
 10     FROM enrollment
 11     WHERE SID = CurrentSID AND CSECID = CurrentCSECID;
 12     IF CurrentGrade IS NULL THEN
 13       DELETE FROM enrollment
 14       WHERE SID = CurrentSID AND CSECID = CurrentCSECID;
 15       COMMIT;
 16     ELSE
 17       RAISE e_NullGradeDelete;
 18     END IF;
 19   EXCEPTION
 20     WHEN e_NullGradeDelete THEN
 21       DBMS_OUTPUT.PUT_LINE
 22       ('The GRADE field value for the current record is not NULL.');
 23       DBMS_OUTPUT.PUT_LINE ('  Delete not performed.');
 24   END;
 25   /
The GRADE field value for the current record is not NULL.
Delete not performed.
```

Figure 4-53: Creating and raising a user-defined exception

Examples of Cursors in PL/SQL

```
DECLARE
cursor payment_cursor is
   select cust_id, payment, total_due from payment_table;
cust_id      payment_table.cust_id%TYPE;
payment   payment_table.payment%TYPE;
total_due  payment_table.total_due%TYPE;

BEGIN
   open payment_cursor;
   WHILE payment < total_due
      LOOP
         FETCH payment_cursor into cust_id, payment, total_due;
         EXIT  WHEN payment_cursor%NOTFOUND;
          insert  into underpay_table values (cust_id, 'STILL
OWES');
      END LOOP;
    close payment_cursor;
END;
```

You can use the FOR-LOOP in the previous block to implicitly fetch the current row of the cursor into the defined variables

Next we rewrite the previous example using FOR-LOOP

```
DECLARE
cursor payment_cursor is
   select cust_id, payment, total_due from payment_table;
cust_id      payment_table.cust_id%TYPE;
payment    payment_table.payment%TYPE;
total_due   payment_table.total_due%TYPE;
Pay_rec    payment_cursor%rowtype;

BEGIN
   open payment_cursor;
   FOR pay_rec IN payment_cursor
     LOOP
       IF pay_rec.payment < pay_rec.total_due THEN
          insert into underpay_table values (pay_rec.cust_id, 'STILL
OWES');
      END IF;
     END LOOP;
  close payment_cursor;
```

This example uses the FOR-LOOP to scroll the cursor. The FOR-LOOP is performing an implicit FETCH, which is omitted this time. Also, notice that the %NOTFOUND attribute has been omitted. This attribute is implied with the FOR-LOOP; therefore, this and the previous example yield the same basic results.

cursor %FOUND attribute

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> DECLARE
  2    CurrentBldgCode VARCHAR2(5);
  3    CURSOR LocationCursor IS
  4      SELECT bldg_code, room, capacity
  5      FROM location
  6      WHERE bldg_code = CurrentBldgCode;
  7    LocationRow LocationCursor%ROWTYPE;
  8  BEGIN
  9    CurrentBldgCode := 'LIB';
 10    OPEN LocationCursor;
 11    FETCH LocationCursor
 12    INTO LocationRow;
 13    IF LocationCursor%FOUND THEN
 14      DBMS_OUTPUT.PUT_LINE('The building code ' || CurrentBldgCode ||
 15      ' returns at least one record.');
 16    ELSE
 17      DBMS_OUTPUT.PUT_LINE('The building code ' || CurrentBldgCode ||
 18      ' does not return any records.');
 19    END IF;
 20    CLOSE LocationCursor;
 21  END;
 22  /
The building code LIB returns at least one record.

PL/SQL procedure successfully completed.
```

Figure 4-43: Using the %FOUND attribute to signal whether or not a cursor returns any records

Procedures and Functions in PL/SQL

Procedures

```
Procedure <Procedure Name>
          ( <Parm1 Name> <Mode> <Data Type,
            <Parm2 Name> <Mode> <Data Type, …
          ) IS

<Variable Declarations>
Begin
   Procedure Body; -- PL/SQL statements
End;
```

Formal Parameters

----- Procedure Call ------

```
Procedure Name ( Actual Parameter List );
```

```
PROCEDURE UpdateInvValue IS
  CURSOR UpdateInvCursor IS
    SELECT * FROM inventory;
  InventoryRow UpdateInvCursor%ROWTYPE;
  InventoryValue NUMBER(11,2);
BEGIN
  FOR InventoryRow IN UpdateInvCursor LOOP
    InventoryValue := InventoryRow.curr_price * InventoryRow.qoh;
    UPDATE inventory
    SET inv_value := InventoryValue
    WHERE invid = InventoryRow.invid;
    COMMIT;
  END LOOP;
END;
```

Functions

```
Function <Function Name>
        ( <Parm1 Name> <Mode> <Data Type,
          <Parm2 Name> <Mode> <Data Type, …
        ) Return <Function Return Value Data Type> IS

<Variable Declarations>
Begin
   Function Body; -- PL/SQL statements
   Return <Return Value>
End;
```

Formal Parameters

----- Function Call ------

```
<Variable Name > := Function Name ( Actual Parameter List);
```

```
FUNCTION StudentAge (CurrentSID NUMBER)
  RETURN NUMBER
IS
  CurrentDate  DATE;
  StudentDOB   DATE;
  CurrentAge   NUMBER;
BEGIN
  CurrentDate := SYSDATE;
  --retrieve SDOB for SID
  SELECT sdob
  INTO StudentDOB
  FROM student
  WHERE sid = CurrentSID;
  CurrentAge := TRUNC((CurrentDate - StudentDOB)/365);
  RETURN CurrentAge;
END;
```