

CSE 4308
Database Management Systems Lab
Lab 10
Intro to PL/SQL
Anonymous Blocks, IF Statements, Functions



Department of Computer Science and Engineering
Islamic University of Technology, OIC

Mohammad Anas Jawad
Lecturer, IUT CSE

PL/SQL Anonymous Block & Syntax

```
SET SERVEROUTPUT ON;
```

```
--Declaration Section-- (Optional)
```

```
DECLARE
```

```
--variable_name datatype [NOT NULL] [:= initial_value];
```

```
Name VARCHAR2(30) := 'xyz';
```

```
-- Name VARCHAR2(30) DEFAULT 'xyz';
```

```
Amount NUMBER(10,3) NOT NULL := 5000;
```

```
Portion NUMBER(10,3) := Amount/3;
```

```
--Execution Section--
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, ' || Name);
```

```
DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is, ' || Portion);
```

```
--Exception Section (Optional)
```

```
EXCEPTION
```

```
WHEN ZERO_DIVIDE THEN
```

```
    DBMS_OUTPUT.PUT_LINE( SQLERRM );
```

```
END;
```

```
/
```

PL/SQL Anonymous Block Demonstration

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> --Declaration Section-- (Optional)
SQL>
SQL> DECLARE
2
3   --variable_name datatype [NOT NULL] [:= initial_value];
4
5   Name VARCHAR2(30) := 'xyz';   -- Name VARCHAR2(30) DEFAULT 'xyz';
6   Amount NUMBER(10,3) NOT NULL := 5000;
7   Portion NUMBER(10,3) := Amount/3;
8
9   --Execution Section--
10
11 BEGIN
12
13   DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, ' || Name);
14
15   DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is, ' || Portion);
16
17   --Exception Section (Optional)
18
19       EXCEPTION
20           WHEN ZERO_DIVIDE THEN
21               DBMS_OUTPUT.PUT_LINE( SQLERRM );
22 END;
23
24 /
Welcome to the Thunderdome, xyz
Your portion of the salary is, 1666.667

PL/SQL procedure successfully completed.

SQL>
```

PL/SQL Anonymous Block Demonstration

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> --Declaration Section-- (Optional)
SQL>
SQL> DECLARE
2
3  --variable_name datatype [NOT NULL] [:= initial_value]
4
5  Name VARCHAR2(30) := 'xyz';  -- Name VARCHAR2(30) DEF
6  Amount NUMBER(10,3) NOT NULL := 5000;
7  Portion NUMBER(10,3) := Amount/0;
8
9  --Execution Section--
10
11 BEGIN
12
13  DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, '
14
15  DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is,
16
17  --Exception Section (Optional)
18
19      EXCEPTION
20          WHEN ZERO_DIVIDE THEN
21              DBMS_OUTPUT.PUT_LINE( SQLERRM );
22  END;
23
24  /
DECLARE
*
ERROR at line 1:
ORA-01476: divisor is equal to zero
ORA-06512: at line 7

SQL>
SQL> edit
Wrote file afiedt.buf
```

```
*afiedt.buf - Notepad
File Edit Format View Help
DECLARE

--variable_name datatype [NOT NULL] [:= initial_value];

Name VARCHAR2(30) := 'xyz';  -- Name VARCHAR2(30) DEFAULT 'xyz';
Amount NUMBER(10,3) NOT NULL := 5000;
Portion NUMBER(10,3) := Amount/0;

--Execution Section--

BEGIN

DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, ' || Name);
|
DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is, ' || Portion);

--Exception Section (Optional)

EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE( SQLERRM );
END;

/

Ln 14, Col 1      100%  Windows (CRLF)  UTF-8
```

PL/SQL Anonymous Block Demonstration

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> --Declaration Section-- (Optional)
SQL>
SQL> DECLARE
2
3  --variable_name datatype [NOT NULL] [:= initial_value]
4
5  Name VARCHAR2(30) := 'xyz';  -- Name VARCHAR2(30) DEF
6  Amount NUMBER(10,3) NOT NULL := 5000;
7  Portion NUMBER(10,3) := Amount/0;
8
9  --Execution Section--
10
11 BEGIN
12
13  DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, '
14
15  DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is,
16
17  --Exception Section (Optional)
18
19      EXCEPTION
20          WHEN ZERO_DIVIDE THEN
21              DBMS_OUTPUT.PUT_LINE( SQLERRM );
22  END;
23
24  /
DECLARE
*
ERROR at line 1:
ORA-01476: divisor is equal to zero
ORA-06512: at line 7

SQL>
SQL> edit
Wrote file afiedt.buf
```

```
afiedt.buf - Notepad
File Edit Format View Help
DECLARE

--variable_name datatype [NOT NULL] [:= initial_value];

Name VARCHAR2(30) := 'xyz';  -- Name VARCHAR2(30) DEFAULT 'xyz';
Amount NUMBER(10,3) NOT NULL := 5000;
Portion NUMBER(10,3) := Amount/3;

--Execution Section--

BEGIN

DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, ' || Name);

DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is, ' || Portion);

--Exception Section (Optional)

    EXCEPTION
        WHEN ZERO_DIVIDE THEN
            DBMS_OUTPUT.PUT_LINE( SQLERRM );
END;

/

Ln 7, Col 33    100%    Windows (CRLF)    UTF-8
```

PL/SQL Anonymous Block Demonstration

SQL> edit

Wrote file afiedt.buf

```
1 DECLARE
2  --variable_name datatype [NOT NULL] [:= initial_value];
3  Name VARCHAR2(30) := 'xyz';  -- Name VARCHAR2(30) DEFAULT 'xyz';
4  Amount NUMBER(10,3) NOT NULL := 5000;
5  Portion NUMBER(10,3) := Amount/3;
6  --Execution Section--
7  BEGIN
8  DBMS_OUTPUT.PUT_LINE( 'Welcome to the Thunderdome, ' || Name);
9  DBMS_OUTPUT.PUT_LINE( 'Your portion of the salary is, ' || Portion);
10 --Exception Section (Optional)
11     EXCEPTION
12         WHEN ZERO_DIVIDE THEN
13             DBMS_OUTPUT.PUT_LINE( SQLERRM );
14* END;
15
```

PL/SQL Anchored Declarations

DECLARE

--variable_name table_name.column_name%TYPE

Var_Name Boys.Name%TYPE;

Var_Semester Boys.Semester%TYPE;

BEGIN

SELECT

name, semester

INTO

Var_Name, Var_Semester

FROM

Boys

WHERE

Boys.ID = 2;

DBMS_OUTPUT.PUT_LINE('Name of student: ' || Var_Name);

DBMS_OUTPUT.PUT_LINE('Semester of student: ' || Var_Semester);

END;

/

Boys

ID	Name	Semester
1	Aflan	7th
2	Saidul	8th
3	Anas	8th

Anchored Declaration Demonstration

```
SQL> SET SERVEROUTPUT ON;
SQL>
SQL> DECLARE
2
3   Var_Name Boys.Name%TYPE;
4   Var_Semester Boys.Semester%TYPE;
5
6 BEGIN
7
8   SELECT
9     name, semester
10  INTO
11     Var_Name, Var_Semester
12  FROM
13     Boys
14  WHERE
15     Boys.ID = 2;
16
17  DBMS_OUTPUT.PUT_LINE('Name of student: ' || Var_Name);
18  DBMS_OUTPUT.PUT_LINE('Semester of student: ' || Var_Semester);
19
20 END;
21
22 /
Name of student: Saidul
Semester of student: 8th
```


PL/SQL IF Statements

--General Syntax for IF statements

```
IF condition1 THEN
    statement1;
ELSEIF condition2 THEN
    statement2;
.
.
ELSE
    else_statement;
END IF;
```

PL/SQL IF Statements

DECLARE

```
Var_Name Boys_Marks.Name%TYPE;  
Var_Marks Boys_Marks.Marks%TYPE;  
Grade VARCHAR2(5);
```

BEGIN

```
SELECT name, marks INTO Var_Name, Var_Marks  
FROM Boys_Marks WHERE Boys_Marks.ID = 2;
```

```
IF Var_Marks > 79 THEN  
    Grade := 'A';  
ELSIF Var_Marks > 69 AND Var_Marks < 80 THEN  
    Grade := 'B';  
ELSE  
    Grade := 'C';  
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('Name of student: ' || Var_Name);  
DBMS_OUTPUT.PUT_LINE('Grade of the student: ' || Grade);
```

```
END;
```

```
/
```

Boys_Marks

ID	Name	Marks
1	Aflan	70
2	Saidul	85
3	Anas	60

IF Statement Demonstration

```
SQL> DECLARE
 2
 3 Var_Name Boys_Marks.Name%TYPE;
 4 Var_Marks Boys_Marks.Marks%TYPE;
 5 Grade VARCHAR2(5);
 6
 7 BEGIN
 8
 9 SELECT name, marks INTO Var_Name, Var_Marks
10 FROM Boys_Marks WHERE Boys_Marks.ID = 2;
11
12 IF Var_Marks > 79 THEN
13     Grade := 'A';
14 ELSIF Var_Marks > 69 AND Var_Marks < 80 THEN
15     Grade := 'B';
16 ELSE
17     Grade := 'C';
18 END IF;
19
20 DBMS_OUTPUT.PUT_LINE('Name of student: ' || Var_Name);
21 DBMS_OUTPUT.PUT_LINE('Grade of the student: ' || Grade);
22
23 END;
24
25 /
Name of student: Saidul
Grade of the student: A
```

PL/SQL Functions

--General Syntax for writing functions

```
CREATE [OR REPLACE] FUNCTION function_name (parameter_list)
    RETURN return_type
IS
```

```
/*Declaration Section*/
```

```
BEGIN
```

```
/*Execution Section (Function body)*/
```

```
--Exception Section (Optional)
```

```
END
```

```
/
```

--General Syntax for deleting or dropping functions

```
DROP FUNCTION function_name;
```

PL/SQL Functions

```
CREATE OR REPLACE FUNCTION get_total_sales (in_date DATE)
RETURN NUMBER
IS

total_sales NUMBER(10,3) := 0;

BEGIN

    -- get total sales
    SELECT SUM(Quantity * PPU)
    INTO total_sales
    FROM Orders, Order_Items
    WHERE Orders.Order_ID = Order_Items.Order_ID AND Order_Date = in_date;

    -- return the total sales
    RETURN total_sales;

END;
/

--Calling the function from an anonymous block
BEGIN
DBMS_OUTPUT.PUT_LINE(get_total_sales(to_date('15_07_2020','dd__mm_yyyy'))),
END;
/

--Calling the function in an SQL statement
SELECT get_total_sales(to_date('15_07_2020','dd__mm_yyyy')) FROM dual;
```

Orders

Order_ID	Client_ID	Order_Date
1	23	22-JUN-20
2	32	15-JUL-20
3	43	15-JUL-20

Order_Items

Order_ID	Name	Quantity	PPU
1	Bananas	6	9
2	Apples	10	5.5
2	Oranges	15	5
3	Coffee	50	6.5

Obj: Write a function that takes a date as parameter and returns the total money earned in sales on that date.
(10*5.5 + 15*5 + 50*6.5 = 455)

Function Demonstration

```
SQL> CREATE OR REPLACE FUNCTION get_total_sales (in_date DATE)
2  RETURN NUMBER
3  IS
4
5  total_sales NUMBER(10,3) := 0;
6
7  BEGIN
8
9      -- get total sales
10
11      SELECT SUM(Quantity * PPU)
12      INTO total_sales
13      FROM Orders, Order_Items
14      WHERE Orders.Order_ID = Order_Items.Order_ID AND Order_Date = in_date;
15
16      -- return the total sales
17
18      RETURN total_sales;
19
20  END;
21  /
```

Function created.

```
SQL> BEGIN
2  DBMS_OUTPUT.PUT_LINE(get_total_sales(to_date('15_07_2020','dd__mm_yyyy')));
3  END;
4  /
455
```

Thank You!