**PL/SQL Notes**

**🔹 What is PL/SQL?**

PL/SQL (Procedural Language/SQL) is Oracle's procedural extension to SQL. It allows for writing full programs to interact with databases.

**🔹 Features of PL/SQL**

* **Portable**: Runs on any Oracle-supported platform.
* **Efficient**: Reduces network traffic by bundling SQL statements.
* **Error Checking**: Built-in exception handling.
* **Development Tools**: Supported by IDEs like Oracle SQL Developer.
* **Exception Handling**: Gracefully manages runtime errors.

**🔹 Basic PL/SQL Block Structure**

Every PL/SQL program must have a BEGIN and END block.

BEGIN

dbms\_output.put\_line('Welcome to PL/SQL');

END;

/

**🔹 Sample Code: IF Statement**

Finds the maximum of two numbers using an IF condition.

DECLARE

a1 NUMBER := 10;

b NUMBER := 100;

c1 NUMBER;

BEGIN

IF a1 > b THEN

c1 := a1;

ELSE

c1 := b;

END IF;

dbms\_output.put\_line('Maximum number in 10 and 100: ' || c1);

END;

/

**🔹 Arithmetic Operations**

Performs basic arithmetic using user input.

DECLARE

n1 NUMBER := &n1;

n2 NUMBER := &n2;

sum NUMBER;

sub NUMBER;

div NUMBER;

mult NUMBER;

BEGIN

sum := n1 + n2;

sub := n1 - n2;

div := n1 / n2;

mult := n1 \* n2;

dbms\_output.put\_line('Addition: ' || sum);

dbms\_output.put\_line('Subtraction: ' || sub);

dbms\_output.put\_line('Multiplication: ' || mult);

dbms\_output.put\_line('Division: ' || div);

END;

/

**🔹 PL/SQL Data Types**

| **Type** | **Description** |
| --- | --- |
| **Scalar** | NUMBER, CHAR, VARCHAR2, BOOLEAN, DATE |
| **LOB** | CLOB, BLOB, NCLOB, BFILE |
| **Reference** | REF CURSOR, REFS |
| **Composite** | Arrays, Records, Tables |

**🔹 Constant Declaration**

Declares a constant string value.

DECLARE

school\_name CONSTANT VARCHAR2(20) := 'DYP';

BEGIN

dbms\_output.put\_line('I study in ' || school\_name);

END;

/

**🔹 Even or Odd Check**

Checks whether a number is even or odd using MOD.

DECLARE

n1 NUMBER := &n1;

BEGIN

IF MOD(n1, 2) = 0 THEN

dbms\_output.put\_line(n1 || ' is even');

ELSE

dbms\_output.put\_line(n1 || ' is odd');

END IF;

END;

/

**🔹 Switch Statement (CASE)**

Uses CASE to determine even or odd.

DECLARE

n1 INT := &n1;

n2 INT;

BEGIN

n2 := MOD(n1, 2);

CASE n2

WHEN 0 THEN dbms\_output.put\_line(n1 || ' is even');

ELSE dbms\_output.put\_line(n1 || ' is odd');

END CASE;

END;

/

**🔹 CASE Statement with Operation**

Performs arithmetic based on user-selected operation.

DECLARE

n1 INT := &n1;

n2 INT := &n2;

operation VARCHAR2(100) := '&operation';

BEGIN

CASE operation

WHEN 'sum' THEN dbms\_output.put\_line('Sum is: ' || (n1 + n2));

WHEN 'sub' THEN dbms\_output.put\_line('Subtraction is: ' || (n1 - n2));

ELSE dbms\_output.put\_line('No operation selected');

END CASE;

END;

/

**🔹 Looping: Print Numbers 1 to 10**

Demonstrates a simple loop.

DECLARE

i INT := 1;

BEGIN

LOOP

EXIT WHEN i > 10;

dbms\_output.put\_line(i);

i := i + 1;

END LOOP;

END;

/

**🔹 Looping: Table of 2**

Prints the multiplication table of 2.

DECLARE

i INT := 1;

num INT := 2;

BEGIN

LOOP

EXIT WHEN i > 10;

dbms\_output.put\_line(num \* i);

i := i + 1;

END LOOP;

END;

**🔹 While :**

**Priting even and odd numbers :**

DECLARE

num int := 1;

BEGIN

WHILE(num <= 10) LOOP

dbms\_output.put\_line('' || num);

num := num+2;

END LOOP;

END ;

**🔹 For Loop :**

DECLARE

i number(2);

BEGIN

FOR i IN 1..10 LOOP

dbms\_output.put\_line(i);

END LOOP;

END ;

**Reverse Loop -**

DECLARE

i number(2);

BEGIN

FOR i IN REVERSE 1..10 LOOP

dbms\_output.put\_line(i);

END LOOP;

END ;

* **FUNCTIONS**

1. **Find maximum number**

DECLARE

a1 number;

b1 number;

c1 number;

FUNCTION findMax(x IN number, y IN number)

RETURN number

IS

z number;

BEGIN

IF x >y THEN

z:=x;

ELSE

z:=y;

END IF;

RETURN z;

END ;

BEGIN

a1 :=10;

b1 :=100;

c1 := findMax(a1,b1);

dbms\_output.put\_line('Maximum number in 10 and 100 is : ' || c1);

END;

/

1. **Odd even**

**Difference between Stored procedure and Function**

|  |  |
| --- | --- |
| Return statement is there |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Create table -

create table user5(id number(10) primary key, name2 varchar2(100))

then **create procedure** -

create or replace procedure “INSERTUSER3”

(id IN NUMBER, name in VARCHAR2)

Is

Begin

insert into user5 values(id name);

end;

/

Insert data in procedure :

BEGIN

Insertuser3(101, “RAHUL”);

dbms\_output.put\_line(‘record inserted successfully’);

END;

Implicit cursor and explicity cursor

create table student(rollno number(10) primary key, sname varchar2(100), course varchar2(100))

BEGIN

INSERT INTO student (rollno, sname, course) VALUES (102, 'Amit', 'Python');

INSERT INTO student (rollno, sname, course) VALUES (103, 'Neha', 'C++');

INSERT INTO student (rollno, sname, course) VALUES (104, 'Ravi', 'JavaScript');

INSERT INTO student (rollno, sname, course) VALUES (105, 'Priya', 'SQL');

INSERT INTO student (rollno, sname, course) VALUES (106, 'Karan', 'Ruby');

INSERT INTO student (rollno, sname, course) VALUES (107, 'Sneha', 'Go');

INSERT INTO student (rollno, sname, course) VALUES (108, 'Vikas', 'Swift');

INSERT INTO student (rollno, sname, course) VALUES (109, 'Meera', 'Kotlin');

INSERT INTO student (rollno, sname, course) VALUES (110, 'Anil', 'PHP');

DBMS\_OUTPUT.PUT\_LINE('10 records inserted successfully');

END;

DECLARE

CURSOR showRec(sno student.rollno%type) IS SELECT sname, course FROM student WHERE rollno=sno;

a student.sname%type;

b student.course%type;

c student.rollno%type;

d number;

BEGIN

d := :rollno;

OPEN showRec(d);

IF showRec%Isopen = FALSE then

dbms\_output.put\_line('Cannot Open cursor');

ELSE

LOOP

FETCH showRec into a, b;

EXIT WHEN showRec%NOTFOUND;

dbms\_output.put\_line(a || ' ' || b);

END LOOP;

END IF;

CLOSE ShowRec;

END

* Second example :

CREATE table customer (

ID INT NOT NULL,

NAME VARCHAR2(40) NULL,

AGE INT NOT NULL,

ADDRESS VARCHAR2(40) NULL,

PRIMARY KEY(ID));

BEGIN

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (1, 'Amit Sharma', 34, 'Delhi', 55000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (2, 'Neha Verma', 28, 'Mumbai', 62000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (3, 'Ravi Mehta', 45, 'Bangalore', 75000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (4, 'Priya Kapoor', 39, 'Hyderabad', 68000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (5, 'Karan Singh', 31, 'Chennai', 53000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (6, 'Sneha Joshi', 42, 'Pune', 72000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (7, 'Vikas Desai', 36, 'Ahmedabad', 59000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (8, 'Meera Rao', 50, 'Kolkata', 81000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (9, 'Anil Nair', 33, 'Jaipur', 56000.00);

INSERT INTO customer (ID, NAME, AGE, ADDRESS, SALARY) VALUES (10, 'Arjun Malhotra', 26, 'Lucknow', 48000.00);

DBMS\_OUTPUT.PUT\_LINE('10 records with salary inserted successfully');

END;

**Creating trigger :**

CREATE OR REPLACE TRIGGER display\_salary\_changes

BEFORE DELETE OR INSERT OR UPDATE ON customer

FOR EACH ROW

WHEN(new.id>0)

DECLARE

sal\_diff number;

BEGIN

sal\_diff := :NEW.salary - :OLD.salary;

dbms\_output.put\_line('Old salary : ' || :OLD.salary);

dbms\_output.put\_line('New salary : ' || :NEW.salary);

dbms\_output.put\_line('salary Difference: ' || sal\_diff);

END;

/

* Update the data

BEGIN

UPDATE customer

SET SALARY = 72000.00,

ADDRESS = 'kolhapur'

WHERE ID = 4;

DBMS\_OUTPUT.PUT\_LINE(' records updated successfully');

END;

**Package creation**

CREATE OR REPLACE PACKAGE c\_package AS

PROCEDURE addCustomer(c\_id customer.id%type,

c\_name customer.Name%type,

c\_age customer.age%type,

c\_addr customer.address%type,

c\_sal customer.salary%type);

PROCEDURE delCustomer(c\_id customer.id%TYPE);

PROCEDURE listCustomer;

END c\_package;

/

**Package body**

CREATE OR REPLACE PACKAGE BODY c\_package AS

PROCEDURE addCustomer(c\_id customer.id%type,

c\_name customer.Name%type,

c\_age customer.age%type,

c\_addr customer.address%type,

c\_sal customer.salary%type)

IS

BEGIN

INSERT INTO customer(id,name,age,address,salary)

VALUES(c\_id, c\_name,c\_age,c\_addr,c\_sal);

END addCustomer;

PROCEDURE delCustomer(c\_id customer.id%type) IS

BEGIN

DELETE FROM customer

where id = c\_id;

END delCustomer;

END c\_package;  
  
**- package usage**

DECLARE

code customer.id%type := 8;

BEGIN

c\_package.addcustomer(17,'Rajnish',25,'chennai',3500);

c\_package.delcustomer(code);

END