PL/SQL - Notes Part 1 for students
References
----------=----------

The PL/SQL programming language was developed by Oracle Corporation as procedural extension language for SQL and the Oracle relational database.

Following are notable facts about PL/SQL:
****

PL/SQL is a completely portable, high-performance transaction-processing language.

PL/SQL provides a built-in interpreted and OS independent programming environment.

PL/SQL can also directly be called from the command-line SQL*Plus interface.

Direct call can also be made from external

programming language calls to database.

Features of PL/SQL

****

PL/SQL is tightly integrated with SQL.
It offers extensive error checking.
It offers numerous data types.
It offers a variety of programming structures.
It supports structured programming through functions and procedures.
It supports developing web applications and server pages.

Advantages of PL/SQL

****

SQL is the standard database language and PL/SQL is strongly integrated with SQL.

PL/SQL supports both static and dynamic SQL.

Static SQL supports DML operations and

transaction control from PL/SQL block.

Dynamic SQL is SQL allows embedding DDL statements in PL/SQL blocks.

PL/SQL allows sending an entire block of statements to the database at one time.

PL/SQL give high productivity to programmers as it can query, transform, and update data in a database.

PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.

Applications written in PL/SQL are fully portable.

PL/SQL provides high security level.

PL/SQL provides access to predefined SQL packages.
PL/SQL provides support for

Object-Oriented Programming.
PL/SQL provides support for Developing Web Applications and Server Pages

---

PL/SQL is not a stand-alone programming language;
it is a tool within the Oracle programming environment.

SQL* Plus is an interactive tool that allows you to type SQL and PL/SQL statements at the command prompt.

These commands are then sent to the database for processing.
Once the statements are processed, the results are sent back and displayed on screen.

==========

1st Program in PL/SQL Block -- First.sql

```
**********************************************
/*
==============================
1) Declare Section
It is a optional section.
we declare variables, cursors, procedures,
functions and

2)Execution Section
THis section is the second section and the
compulsary section. it has logic of the
program.

Begin
.......
.............
End;
3) exceptional handling.
*/

set serveroutput on;
DECLARE

 messg  varchar2(40) := 'Good Morning
India.';
```

```
 name2 varchar2(89) := 'I will get
Peformance bonus of
                $333535 every year.';

BEGIN

   dbms_output.put_line(messg);
   dbms_output.put_line(name2);
END;
/
================================
========================
================================
========================
```

Program using if conditions.

```
*****************************
/* write a pl/sql block to intialize the rating
and if
 it is less than 215 flash a message YOU
ARE selected in the process.
*/

DECLARE
   a number(4) := 33;
```

```
BEGIN
  -- check the boolean condition using if
statement
IF( a < 215 )
   THEN
      dbms_output.put_line('you are selected
in the process ' );
END IF;

   dbms_output.put_line('value of a is : ' ||
a);
END;
/
```

==========

==========

==========

==========

PL/SQL Another program on accepting values using if conditons.

****

****

/* write a pl/sql block where you will intalize a salesman no and if found and current

```
insenitve less than 25 k increase the salary
by 1000 rupees
*/

set serveroutput on;
DECLARE
   tempno salespeople.snum%type := 1019;
   tsal  salespeople.comm%type;

BEGIN
   SELECT  comm INTO  tsal
   FROM salespeople
   WHERE snum = tempno;

   IF (tsal <= 25000)
     THEN

        UPDATE salespeople
         SET comm =  comm + 1000
           WHERE snum = tempno;
         commit;
        dbms_output.put_line ('Salary
updated');
   END IF;
```

```
END;
/
==============================
==========================
==============================
==========================
```

Program using while loops
****

```
DECLARE
   a number(2) := 1;

BEGIN

   WHILE a < 20
      LOOP
         dbms_output.put_line('value of a: ' ||
a);
         a := a + 1;
   END LOOP;

END;
/
==============================
```

==========
==========
==========

A program using for loops
****

```
DECLARE
   a number(2);

BEGIN

   FOR a in 10 .. 20
      LOOP
         dbms_output.put_line('value of a: ' ||
a);
   END LOOP;

END;
/
```

==========
==========
==========
==========

A program using %type
****

```
set serveroutput on ;
DECLARE

   tempno employee.empno%type ;
   tempname  employee.sname%type;
   tdoj employee.doj%type;
   tdesig employee.desig%type;
   tbasic  employee.basic%type;

 BEGIN
   tempno := &snum;

   SELECT empno, sname, doj, desig,
basic into
   tempno, tempname,tdoj, tdesig, tbasic
   FROM employee
   WHERE empno = tempno;

dbms_output.put_line('employee ' ||
tempname || ' Posted as  ' || tdesig || ' earns
' || tbasic);
END;
/
```

==========
==========
==========
==========

Program using % Rowtypes
****

```
DECLARE
customer_rec customers%rowtype;

BEGIN
   SELECT * into customer_rec
   FROM customers
   WHERE cnum = 2001;

   dbms_output.put_line('Customer Cnum: '
|| customer_rec.cnum);
   dbms_output.put_line('Customer Name: '
|| customer_rec.cname);
   dbms_output.put_line('Customer City: ' ||
customer_rec.city);
   dbms_output.put_line('Customer
Salesman: ' || customer_rec.snum);
```

```
END;
/
==========
==========
==========
==========
```