***PART 1***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL** | **Command Name** | **Command** | | | **Result** |
| 1 | **Variable**  **Declaration** | sales number(10, 2);  pi CONSTANT number := 3.1415;  name varchar2(25);  address varchar2(100); | | |  |
| 2 | **Assign Value to a Variable** | counter integer := 0;  greetings varchar2(20) DEFAULT 'Have a Good Day'; | | |  |
| 3 | **Initializing Variables in PL/SQL** | DECLARE  a integer := 10;  b integer := 20;  c integer;  f real;  BEGIN  c := a + b;  dbms\_output.put\_line('Value of c: ' || c);  f := 70.0/3.0;  dbms\_output.put\_line('Value of f: ' || f);  END;  / | | | Value of c: 30  Value of f: 23.33333333333333 |
| 4 | **Variable Scope in PL/SQL** | 1. Local variables 2. Global variables | | |  |
| DECLARE  -- Global variables  num1 number := 95;  num2 number := 85;  BEGIN  dbms\_output.put\_line('Outer Variable num1: ' || num1);  dbms\_output.put\_line('Outer Variable num2: ' || num2);  DECLARE  -- Local variables  num1 number := 195;  num2 number := 185;  BEGIN  dbms\_output.put\_line('Inner Variable num1: ' || num1);  dbms\_output.put\_line('Inner Variable num2: ' || num2);  END;  END; | | | Outer Variable num1: 95  Outer Variable num2: 85  Inner Variable num1: 195  Inner Variable num2: 185 |
| 5 | **Assigning SQL Query Results to PL/SQL Variables** | DECLARE  c\_id emp.empno%type := 7369;  c\_name emp.ename%type;  c\_sal emp.sal%type;  BEGIN  SELECT ename, sal INTO c\_name, c\_sal  FROM emp  WHERE empno = c\_id  and :job = 'CLERK';  ***dbms\_output.put\_line***  ('Employee ' ||c\_name || ' earns ' || c\_sal);  END; | | | Employee SMITH earns 800 |
| 6 | **Declaring a Constant** | DECLARE  -- constant declaration  pi constant number := 3.141592654;  -- other declarations  radius number(5,2);  dia number(5,2);  circumference number(7, 2);  area number (10, 2);  BEGIN  -- processing  radius := 9.5;  dia := radius \* 2;  circumference := 2.0 \* pi \* radius;  area := pi \* radius \* radius;  -- output  dbms\_output.put\_line('Radius: ' || radius);  dbms\_output.put\_line('Diameter: ' || dia);  dbms\_output.put\_line('Circumference: ' || circumference);  dbms\_output.put\_line('Area: ' || area);  END; | | | Radius: 9.5  Diameter: 19  Circumference: 59.69  Area: 283.53 |
| 7 | **The PL/SQL Literals** | **Different Kinds of Literals** | | | |
| 1. Numeric Literals | 1. -14 0 | | |
| 1. Character Literals | 'A' '%' '9' ' ' 'z' '(' | | |
| 1. String Literals | 'Hello, world!'  '19-NOV-12' | | |
| 1. BOOLEAN Literals | T RUE, FALSE, and NULL | | |
| 1. Date and T ime Literals | DAT E '1978-12-25';  T IMEST AMP '2012-10-29 12:01:01'; | | |
| DECLARE  message varchar2(30):= ''That''s tutorialspoint.com!'';  BEGIN  dbms\_output.put\_line(message);  END;  / | | | That's tutorialspoint.com! |
| 8 | **PL/SQL - OPERATORS** | **Different Types of Operators** | | | |
| 1. Arithmetic operators | | + , - , \* , / , \*\* | |
| 1. Relational operators | | = , < , > , >= , <= | |
| 1. Comparison operators | | LIKE , BETWEEN , IN , IS NULL | |
| 1. Log ical operators | | AND , OR , NOT | |
| 9 | **IF – ELSE Statement** | DECLARE  name varchar2(20);  company varchar2(30);  introduction clob;  choice char(1);  BEGIN  name := 'John Smith';  company := 'Infotech';  introduction := ' Hello! I''m John Smith from Infotech.';  choice := 'y';  IF choice = 'y' THEN  dbms\_output.put\_line(name);  dbms\_output.put\_line(company);  dbms\_output.put\_line(introduction);  END IF;  END; | | | John Smith  Infotech  Hello! I'm John Smith from Infotech. |
| 10 | **LOOP** | DECLARE  i number(1);  j number(1);  BEGIN  << outer\_loop >>  FOR i IN 1..3 LOOP  << inner\_loop >>  FOR j IN 1..3 LOOP  dbms\_output.put\_line('i is: '|| i || ' and j is: ' || j);  END loop inner\_loop;  END loop outer\_loop;  END; | | |  |

***PART 2***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SL** | **Command**  **Name** | **Command** | | | **Result** |
| 1 | **PL/SQL - CURSORS** | 1. **Implicit Cursor** | | Oracle automatically (implicit) controls or processes the information of SQL statement executed. In this process, the user is unaware of implicit cursor. Oracle automatically performs the OPEN, FETCH, and CLOSE operations. | |
| 1. **Explicit Cursor** | | Explicit cursor is used for the query that returns more than one row of data. These cursors are explicitly declared in the DECLARE section of the PL/SQL block. This declaration allows to sequentially process each row of data as the cursor returns it. In explicit cursor DECLARE, OPEN, FETCH, and CLOSE operations are done by the programmer. | |
| **Attribute of Cursor** | | | | | |
|  | %FOUND | Returns TRUE if an INSERT, UPDAT E, or DELET E statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE. | | | |
| %NOT FOUND | The logical opposite of %FOUND. It returns T RUE if an INSERT , UPDAT E, or DELET E statement affected no rows, or a SELECT INT O statement returned no rows. Otherwise, it returns FALSE. | | | |
| %ISOPEN | Always returns FALSE for implicit cursors, because Oracle closes the SQL  cursor automatically after executing its associated SQL statement. | | | |
| %ROWCOUNT | Returns the number of rows affected by an INSERT, UPDAT E, or DELETE statement, or returned by a SELECT INTO statement. | | | |
| 2 | **Implicit Cursor** | DECLARE  total\_rows number(2);  BEGIN  UPDATE employee  SET sal = sal + 500  WHERE job='SALESMAN' ;  IF sql%notfound THEN  ***dbms\_output.put\_line***('no customers selected');  ELSIF sql%found THEN  total\_rows := sql%rowcount;  ***dbms\_output.put\_line***( total\_rows || ' customers selected ');  END IF;  END; | | | 4 customers selected |
| **Steps for Explicit Cursor** | | | | | |
|  | 1. Declaring the Cursor | | CURSOR c\_employee IS  SELECT empno, ename, job FROM employee; | | |
| 2. Opening the Cursor | | OPEN c\_employee; | | |
| 3. Fetching the Cursor | | FETCH c\_employee INTO c\_empno, c\_ename, c\_job; | | |
| 4. Closing the Cursor | | CLOSE c\_employee; | | |
| 3 | **Explicit Cursors** | DECLARE  c\_empno employee.empno%type;  c\_ename employee.ename%type;  c\_job employee.job%type;  CURSOR c\_employee IS  SELECT empno, ename, job  FROM employee  WHERE JOB='SALESMAN';  BEGIN  OPEN c\_employee;  LOOP  FETCH c\_employee into c\_empno, c\_ename, c\_job;  ***dbms\_output.put\_line***(c\_empno || ' ' || c\_ename || ' ' || c\_job);  EXIT WHEN c\_employee%notfound;  END LOOP;  CLOSE c\_employee;  END; | | | 7499 ALLEN SALESMAN  7521 WARD SALESMAN  7654 MARTIN SALESMAN  7844 TURNER SALESMAN  7844 TURNER SALESMAN |
| 4 | **Explicit Cursors**  **(Cont.)**  **Programmer defined record** | DECLARE  CURSOR c\_emp\_detail IS  SELECT empno,ename,sal  FROM employee;  */\*declaring a record datatype, with same datatype of tables of database using %TYPE attribute,*  *with same order of corresponding cursor \*/*  TYPE type\_rectype IS RECORD  (emp\_id employee.empno%TYPE,  f\_name employee.ename%TYPE,  s\_salary employee.sal%TYPE  );  rec\_type type\_rectype; *--variable of record datatype.*  BEGIN  OPEN c\_emp\_detail;  LOOP  FETCH c\_emp\_detail INTO rec\_type; *-- Fetches the cursor into record variable.*  EXIT WHEN c\_emp\_detail%NOTFOUND;  *-- variable is part of each record datatype,so to reference it use dot notation in DBMS\_OUTPUT.*  ***DBMS\_OUTPUT.PUT\_LINE***('Employees Details : '||' '||rec\_type.emp\_id  ||' '||rec\_type.f\_name||' '||rec\_type.s\_salary);  END LOOP;  ***DBMS\_OUTPUT.PUT\_LINE***('Total number of Employees : '||c\_emp\_detail%ROWCOUNT);  CLOSE c\_emp\_detail;  END; | | | Employees Details : 7369 SMITH 800  Employees Details : 7499 ALLEN 1600  Employees Details : 7521 WARD 1250  Employees Details : 7566 JONES 2975  Employees Details : 7654 MARTIN 1250  Employees Details : 7698 BLAKE 2850  Employees Details : 7782 CLARK 2450  Employees Details : 7788 SCOTT 3000  Employees Details : 7839 KING 5000  Employees Details : 7844 TURNER 1500  Employees Details : 7876 ADAMS 1100  Employees Details : 7900 JAMES 950  Employees Details : 7902 FORD 3000  Employees Details : 7934 MILLER 1300  Total number of Employees : 14 |
| 5 | **Explicit Cursors**  **(Cont.)**  **Cursors in Nested Loops** | DECLARE  CURSOR c\_dept IS  SELECT \*  FROM department  WHERE deptno in (select distinct deptno from emp)  ORDER BY dname;  r\_dept c\_dept%ROWTYPE;  *-- Declaration of department cursor and record variable.*  CURSOR c\_emp (c\_dept\_no department.deptno%TYPE) IS  SELECT \*  FROM employee  WHERE deptno = c\_dept\_no;  r\_emp c\_emp%ROWTYPE;  *-- Declaration of employees cursor and record variable.*  BEGIN  OPEN c\_dept;  LOOP  FETCH c\_dept INTO r\_dept;  EXIT WHEN c\_dept%NOTFOUND;  ***DBMS\_OUTPUT.PUT\_LINE***('----------------------------------');  ***DBMS\_OUTPUT.PUT\_LINE***('Department Name : '||r\_dept.dname);  ***DBMS\_OUTPUT.PUT\_LINE***('----------------------------------');  OPEN c\_emp(r\_dept.deptno);  LOOP  FETCH c\_emp INTO r\_emp;  EXIT WHEN c\_emp%NOTFOUND;  ***DBMS\_OUTPUT.PUT\_LINE***('Employees Details : '||r\_emp.empno  ||' '||r\_emp.ename||' '||r\_emp.job||' '||r\_emp.sal);  END LOOP;  CLOSE c\_emp;  END LOOP;  CLOSE c\_dept;  END; | | | ----------------------------------  Department Name : ACCOUNTING  ----------------------------------  Employees Details : 7782 CLARK MANAGER 2450  Employees Details : 7839 KING PRESIDENT 5000  Employees Details : 7934 MILLER CLERK 1300  ----------------------------------  Department Name : RESEARCH  ----------------------------------  Employees Details : 7369 SMITH CLERK 800  Employees Details : 7566 JONES MANAGER 2975  Employees Details : 7788 SCOTT ANALYST 3000  Employees Details : 7876 ADAMS CLERK 1100  Employees Details : 7902 FORD ANALYST 3000  ----------------------------------  Department Name : SALES  ----------------------------------  Employees Details : 7499 ALLEN SALESMAN 1600  Employees Details : 7521 WARD SALESMAN 1250  Employees Details : 7654 MARTIN SALESMAN 1250  Employees Details : 7698 BLAKE MANAGER 2850  Employees Details : 7844 TURNER SALESMAN 1500  Employees Details : 7900 JAMES CLERK 950 |
| 6 | **For Loop** | FOR *loop\_counter* IN [REVERSE] *lowest\_number*..*highest\_number*  LOOP  {...statements...}  END LOOP; | | |  |
| 7 | **Implicit cursor for loop** | BEGIN  FOR item IN(SELECT d.dname, d.deptno, e.ename, e.job, e.sal  FROM department d JOIN employee e  ON e.deptno = d.deptno  WHERE JOB = 'SALESMAN'  AND sal > 1200)  LOOP  ***DBMS\_OUTPUT.PUT\_LINE***(item.ename||' '||item.dname  ||' '||item.deptno||' '||item.job||' '||item.sal);  END LOOP;  END; | | | ALLEN SALES 30 SALESMAN 1600  WARD SALES 30 SALESMAN 1250  MARTIN SALES 30 SALESMAN 1250  TURNER SALES 30 SALESMAN 1500 |
| 8 | **Nested Cursors Using Cursor FOR Loops** | DECLARE  v\_dept\_id department.deptno%TYPE;  CURSOR c\_dept IS  SELECT distinct b.\*  FROM employee a, dept b  WHERE a.mgr IS NOT NULL  AND a.deptno=b.deptno  AND a.sal > 2500;  CURSOR c\_emp IS  SELECT \*  FROM employee  WHERE deptno = v\_dept\_id;  BEGIN  FOR r\_dept IN c\_dept  LOOP  v\_dept\_id := r\_dept.deptno;  ***DBMS\_OUTPUT.PUT\_LINE***('----------------------------------');  ***DBMS\_OUTPUT.PUT\_LINE***('Department Name : '||r\_dept.dname);  ***DBMS\_OUTPUT.PUT\_LINE***('----------------------------------');  FOR r\_emp IN c\_emp  LOOP  ***DBMS\_OUTPUT.PUT\_LINE***('Employee Name : '||r\_emp.ename);  END LOOP;  END LOOP;  END; | | | ----------------------------------  Department Name : RESEARCH  ----------------------------------  Employee Name : SMITH  Employee Name : JONES  Employee Name : SCOTT  Employee Name : ADAMS  Employee Name : FORD  ----------------------------------  Department Name : SALES  ----------------------------------  Employee Name : ALLEN  Employee Name : WARD  Employee Name : MARTIN  Employee Name : BLAKE  Employee Name : TURNER  Employee Name : JAMES |
| 9 | **For UPDATE and WHERE CURRENT CLAUSE**  **In FOR LOOP** | DECLARE  CURSOR c\_sal\_update IS  SELECT empno,ename,job,dname,e.deptno,sal  FROM dept1 d , emp1 e  WHERE e.deptno = 30  FOR UPDATE OF sal NOWAIT;  rec\_sal c\_sal\_update%ROWTYPE;  BEGIN  OPEN c\_sal\_update; *-- rows are locked.*  LOOP  FETCH c\_sal\_update INTO rec\_sal;  EXIT WHEN c\_sal\_update%NOTFOUND;  IF rec\_sal.job = 'SALESMAN' THEN  UPDATE emp1  SET sal = rec\_sal.sal + 1000  WHERE CURRENT OF c\_sal\_update;  END IF;  END LOOP;  COMMIT; *-- rows are unlocked.*  CLOSE c\_sal\_update;  END; | | |  |
| 10 | **Cursor attributes with explicit cursor** | DECLARE  CURSOR c\_high\_sal IS  SELECT \*  FROM (SELECT empno,ename,sal  FROM employee ORDER BY sal DESC)  WHERE ROWNUM < 11;  high\_sal c\_high\_sal%ROWTYPE;  BEGIN  IF NOT c\_high\_sal%ISOPEN THEN  ***DBMS\_OUTPUT.PUT\_LINE***('Cursor is Closed');  END IF;  OPEN c\_high\_sal;  IF c\_high\_sal%ISOPEN THEN  ***DBMS\_OUTPUT.PUT\_LINE***('Cursor is open');  END IF;  LOOP  FETCH c\_high\_sal INTO high\_sal;  IF c\_high\_sal%FOUND THEN  ***DBMS\_OUTPUT.PUT\_LINE***(high\_sal.empno||' '||high\_sal.ename  ||' '||high\_sal.sal);  ELSE  EXIT; *-- the same as exit when c\_high\_sal%NOTFOUND;*  END IF;  END LOOP;  ***DBMS\_OUTPUT.PUT\_LINE***(' Number of rows fetched : '||c\_high\_sal%ROWCOUNT);  CLOSE c\_high\_sal;  IF NOT c\_high\_sal%ISOPEN THEN  ***DBMS\_OUTPUT.PUT\_LINE***('Cursor is closed ');  END IF;  END; | | | Cursor is Closed  Cursor is open  7839 KING 5000  7788 SCOTT 3000  7902 FORD 3000  7566 JONES 2975  7698 BLAKE 2850  7782 CLARK 2450  7499 ALLEN 1600  7844 TURNER 1500  7934 MILLER 1300  7521 WARD 1250  Number of rows fetched : 10  Cursor is closed |
| 11 | **Cursor attributes with implicit cursor** | DECLARE  v\_dept department.deptno%TYPE := 10;  v\_dept\_name department.dname%TYPE;  BEGIN  DELETE FROM dept1 WHERE deptno = v\_dept;  IF SQL%FOUND THEN  INSERT INTO dept1 VALUES(70,'Personnel','VEGAS');  END IF;  ***DBMS\_OUTPUT.PUT\_LINE***('Number of rows inserted : '||SQL%ROWCOUNT);  SELECT dname INTO v\_dept\_name  FROM dept1  WHERE deptno = 70;  ***DBMS\_OUTPUT.PUT\_LINE***('Department Name : '||v\_dept\_name);  END; | | | Number of rows inserted : 1  Department Name : Personnel |

***PART 3***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL** | **Command Name** | **Command** | | **Result** |
| 1 | **Sub-Program** | A **subprogram** is a program unit/module that performs a particular task.  When a subprogram is created in schema level, then it is a **standalone subprogram**. If the subprogram created inside a package is a **packaged subprogram**. | | |
| 2 | **Parameter Modes in PL/SQL Subprograms** | **IN** | An IN parameter lets you pass a value to the subprogram. **It is a read-only parameter**. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a  Constant, literal, initialized variable, or expression as an IN parameter. | |
| **OUT** | An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. | |
| **IN OUT** | An IN OUT parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and its value can be read. | |
| 3 | **Procedures** | These subprograms do not return a value directly, mainly used to perform an action. | | |
| 4 | **Creating**  **a Procedure** | CREATE OR REPLACE PROCEDURE GREETINGS  IS  BEGIN  ***DBMS\_OUTPUT.PUT\_LINE***('Hello World');  END; | |  |
| 5 | **Executing a Procedure** | EXECUTE greetings;  Or,  Greetings;  Or,  BEGIN  greetings;  END; | |  |
| 6 | **Example of Procedure** | DECLARE  a number;  b number;  c number;  PROCEDURE findMin(x IN number, y IN number, z OUT number) IS  BEGIN  IF x < y THEN  z:= x;  ELSE  z:= y;  END IF;  END;  BEGIN  a:= 23;  b:= 45;  findMin(a, b, c);  dbms\_output.put\_line(' Minimum of (23, 45) : ' || c);  END; | | Minimum of (23, 45) : 23 |
| 7 | **Example of Procedure**  **(Cont.)** | DECLARE  a number;  PROCEDURE squareNum(x IN OUT number) IS  BEGIN  x := x \* x;  END;  BEGIN  a:= 23;  squareNum(a);  ***dbms\_output.put\_line***(' Square of (23): ' || a);  END; | | Square of (23): 529 |
| 8 | **Example of Procedure**  **(Cont.)** | CREATE OR REPLACE PROCEDURE employer\_details  IS  DECLARE  CURSOR C1 IS  SELECT ename, job , sal FROM emp;  BEGIN  FOR r in c1  LOOP  ***dbms\_output.put\_line***(r.ename || ' ' ||r.job  || ' ' ||r.sal);  END LOOP;  END; | |  |
| 9 | **Functions** | These subprograms return a single value, mainly used to compute and return a value. | | |
| 10 | **Creating a Function** | CREATE OR REPLACE FUNCTION totalCustomers  RETURN number IS  total number(2) := 0;  BEGIN  SELECT count(\*) into total  FROM emp;  RETURN total;  END; | |  |
| 11 | **Calling a Function** | DECLARE  c number(2);  BEGIN  c := totalCustomers();  dbms\_output.put\_line('Total no. of Customers: ' || c);  END; | |  |
| 12 | **Example of Function** | DECLARE  a number;  b number;  c 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  a:= 23;  b:= 45;  c := findMax(a, b);  dbms\_output.put\_line(' Maximum of (23,45): ' || c);  END; | | Maximum of (23,45): 45 |
| 13 | **Exception** | DECLARE  c\_id emp.empno%type := 7126;  c\_name emp.ename%type;  c\_job emp.job%type;  BEGIN  SELECT ename, job INTO c\_name, c\_job  FROM emp  WHERE  empno = c\_id;  ***DBMS\_OUTPUT.PUT\_LINE*** ('Name: '|| c\_name);  ***DBMS\_OUTPUT.PUT\_LINE*** ('Address: ' || c\_job);  EXCEPTION  WHEN no\_data\_found THEN  ***dbms\_output.put\_line***('No such employee');  WHEN others THEN  ***dbms\_output.put\_line***('Error!');  END; | |  |
| 14 | **User-defined Exceptions** | DECLARE  c\_id emp.empno%type := :cc\_id;  c\_name emp.ename%type;  c\_job emp.job%type;  *-- user defined exception*  ex\_invalid\_id EXCEPTION;  BEGIN  IF c\_id <= 0 THEN  RAISE ex\_invalid\_id;  ELSE  SELECT ename, job INTO c\_name, c\_job  FROM emp  WHERE empno = c\_id;  ***DBMS\_OUTPUT.PUT\_LINE*** ('Name: '|| c\_name);  ***DBMS\_OUTPUT.PUT\_LINE*** ('Address: ' || c\_job);  END IF;  EXCEPTION  WHEN ex\_invalid\_id THEN  ***dbms\_output.put\_line***('ID must be greater than zero!');  WHEN no\_data\_found THEN  ***dbms\_output.put\_line***('No such customer!');  WHEN others THEN  ***dbms\_output.put\_line***('Error!');  END; | |  |
|  |  |  | |  |