```
--------------------------------------------------------------------------------
------------Execute Immediate Statement (Code Samples)--------------------------
--------------------------------------------------------------------------------

BEGIN
    EXECUTE IMMEDIATE 'GRANT SELECT ON EMPLOYEES TO SYS';
END;
/
BEGIN
    EXECUTE IMMEDIATE 'GRANT SELECT ON EMPLOYEES TO SYS;';
END;
/
CREATE OR REPLACE PROCEDURE prc_create_table_dynamic
    (p_table_name IN VARCHAR2, p_col_specs IN VARCHAR2) IS
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE '||p_table_name||' ('||p_col_specs||')';
END;
/
EXEC prc_create_table_dynamic('dynamic_temp_table', 'id NUMBER PRIMARY KEY, name
VARCHAR2(100)');
/
SELECT * FROM dynamic_temp_table;
/
CREATE OR REPLACE PROCEDURE prc_generic (p_dynamic_sql IN VARCHAR2) IS
BEGIN
    EXECUTE IMMEDIATE p_dynamic_sql;
END;
/
EXEC prc_generic('drop table dynamic_temp_table');
/
EXEC prc_generic('drop procedure PRC_CREATE_TABLE_DYNAMIC');
/
DROP PROCEDURE prc_generic;
--------------------------------------------------------------------------------
----------------
----------EXECUTE IMMEDIATE STATEMENT with the USING Clause (Code
Samples)----------------
--------------------------------------------------------------------------------
-------------

CREATE TABLE names (ID NUMBER PRIMARY KEY, NAME VARCHAR2(100));
/
CREATE OR REPLACE FUNCTION insert_values (ID IN VARCHAR2, NAME IN VARCHAR2) RETURN
PLS_INTEGER IS
BEGIN
    EXECUTE IMMEDIATE 'INSERT INTO names VALUES(:a, :b)' USING ID,NAME;
    RETURN SQL%rowcount;
END;
/
SET SERVEROUTPUT ON;
DECLARE
    v_affected_rows PLS_INTEGER;
BEGIN
    v_affected_rows := insert_values(2,'John');
    dbms_output.put_line(v_affected_rows|| ' row inserted!');
END;
/
SELECT * FROM names;
/
```

```
ALTER TABLE names ADD (last_name VARCHAR2(100));
/
CREATE OR REPLACE FUNCTION update_names (ID IN VARCHAR2, last_name IN VARCHAR2)
RETURN PLS_INTEGER IS
    v_dynamic_sql VARCHAR2(200);
BEGIN
    v_dynamic_sql := 'UPDATE names SET last_name = :1 WHERE id = :2' ;
    EXECUTE IMMEDIATE v_dynamic_sql USING last_name, ID;
    RETURN SQL%rowcount;
END;
/
DECLARE
    v_affected_rows PLS_INTEGER;
BEGIN
    v_affected_rows := update_names(2,'Brown');
    dbms_output.put_line(v_affected_rows|| ' row updated!');
END;
/
CREATE OR REPLACE FUNCTION update_names (ID IN VARCHAR2, last_name IN OUT VARCHAR2)
RETURN PLS_INTEGER IS
    v_dynamic_sql VARCHAR2(200);
BEGIN
    v_dynamic_sql := 'UPDATE names SET last_name = :1 WHERE id = :2' ;
    EXECUTE IMMEDIATE v_dynamic_sql USING IN OUT last_name, ID;
    RETURN SQL%rowcount;
END;
/
CREATE OR REPLACE FUNCTION update_names (ID IN VARCHAR2, last_name IN VARCHAR2,
first_name OUT VARCHAR2) RETURN PLS_INTEGER IS
    v_dynamic_sql VARCHAR2(200);
BEGIN
    v_dynamic_sql := 'UPDATE names SET last_name = :1 WHERE id = :2 :3' ;
    EXECUTE IMMEDIATE v_dynamic_sql USING last_name, ID, OUT first_name;
    RETURN SQL%rowcount;
END;
/
DECLARE
    v_affected_rows PLS_INTEGER;
    v_first_name VARCHAR2(100);
BEGIN
    v_affected_rows := update_names(2,'KING',v_first_name);
    dbms_output.put_line(v_affected_rows|| ' row updated!');
    dbms_output.put_line(v_first_name);
END;
/
CREATE OR REPLACE FUNCTION update_names (ID IN VARCHAR2, last_name IN VARCHAR2,
first_name OUT VARCHAR2) RETURN PLS_INTEGER IS
    v_dynamic_sql VARCHAR2(200);
BEGIN
    v_dynamic_sql := 'UPDATE names SET last_name = :1 WHERE id = :2 RETURNING name
INTO :3' ;
    EXECUTE IMMEDIATE v_dynamic_sql USING last_name, ID RETURNING INTO first_name;
    RETURN SQL%rowcount;
END;
/
DROP TABLE names;
DROP FUNCTION insert_values;
DROP FUNCTION update_names;
--------------------------------------------------------------------------------
```

```
-------------------------
---------EXECUTE IMMEDIATE STATEMENT with the USING and INTO Clauses (Code
Samples)----------------------
--------------------------------------------------------------------------------
----------------------
CREATE OR REPLACE FUNCTION get_count (table_name IN VARCHAR2) RETURN PLS_INTEGER IS
    v_count PLS_INTEGER;
BEGIN
    EXECUTE IMMEDIATE 'SELECT COUNT(*) FROM ' || table_name INTO v_count;
    RETURN v_count;
END;
/
SET SERVEROUTPUT ON;
BEGIN
    dbms_output.put_line('There are '||get_count('employees')||' rows in the
employees table!');
END;
/
DECLARE
    v_table_name VARCHAR2(50);
BEGIN
    FOR r_table IN (SELECT table_name FROM user_tables) LOOP
        dbms_output.put_line('There are '||get_count(r_table.table_name)||' rows in
the '||r_table.table_name||' table!');
    END LOOP;
END;
/
DECLARE
    v_table_name VARCHAR2(50);
BEGIN
    FOR r_table IN (SELECT table_name FROM user_tables) LOOP
        IF get_count(r_table.table_name) > 100 THEN
            dbms_output.put_line('There are '||get_count(r_table.table_name)||'
rows in the '||r_table.table_name||' table!');
            dbms_output.put_line('It should be considered for partitioning');
        END IF;
    END LOOP;
END;
/

CREATE TABLE stock_managers AS SELECT * FROM employees WHERE job_id = 'ST_MAN';
/
CREATE TABLE stock_clerks AS SELECT * FROM employees WHERE job_id = 'ST_CLERK';
/
CREATE OR REPLACE FUNCTION get_avg_sals (p_table IN VARCHAR2, p_dept_id IN NUMBER)
RETURN PLS_INTEGER IS
    v_average PLS_INTEGER;
BEGIN
    EXECUTE IMMEDIATE 'SELECT AVG(salary) FROM :1 WHERE department_id = :2' INTO
v_average USING p_table, p_dept_id;
    RETURN v_average;
END;
/
SELECT get_avg_sals('stock_clerks','50') FROM dual;
/
CREATE OR REPLACE FUNCTION get_avg_sals (p_table IN VARCHAR2, p_dept_id IN NUMBER)
RETURN PLS_INTEGER IS
    v_average PLS_INTEGER;
BEGIN
```

```
    EXECUTE IMMEDIATE 'SELECT AVG(salary) FROM '||p_table||' WHERE department_id
= :2' INTO v_average USING p_dept_id;
    RETURN v_average;
END;
/
SELECT get_avg_sals('stock_managers','50') FROM dual;
/
DROP FUNCTION get_count;
DROP FUNCTION get_avg_sals;
DROP TABLE stock_clerks;
DROP TABLE stock_managers;
--------------------------------------------------------------------------------
------------------------------
---------------Execute Immediate with Bulk Collect (Code
Samples)--------------------
--------------------------------------------------------------------------------
------------------------------
DECLARE
   TYPE t_name IS TABLE OF VARCHAR2(20);
   names   t_name;
BEGIN
    EXECUTE IMMEDIATE 'SELECT distinct first_name FROM employees'
        BULK COLLECT INTO names;
    FOR i IN 1..names.COUNT LOOP
        dbms_output.put_line(names(i));
    END LOOP;
END;
/
CREATE TABLE employees_copy AS SELECT * FROM employees;
/
DECLARE
   TYPE t_name IS TABLE OF VARCHAR2(20);
   names   t_name;
BEGIN
    EXECUTE IMMEDIATE 'UPDATE employees_copy SET salary = salary + 1000 WHERE
department_id = 30 RETURNING first_name INTO :a'
        RETURNING BULK COLLECT INTO names;
    FOR i IN 1..names.COUNT LOOP
        dbms_output.put_line(names(i));
    END LOOP;
END;
/
DROP TABLE employees_copy;
--------------------------------------------------------------------------------
------------------------------
---------------Dynamic PL/SQL Blocks (Code Sample)--------------------------
--------------------------------------------------------------------------------
------------
BEGIN
    FOR r_emp in (SELECT * FROM employees) LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
    END LOOP;
END;
/
DECLARE
    v_dynamic_text varchar2(1000);
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employees) LOOP
```

```
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
    END LOOP;
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text;
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    v_department_id PLS_INTEGER := 30;
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employees WHERE department_id = v_department_id)
LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
    END LOOP;
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text;
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    --v_department_id pls_integer := 30;
BEGIN
    v_dynamic_text := q'[DECLARE
    v_department_id pls_integer := 30;
    BEGIN
    FOR r_emp in (SELECT * FROM employees WHERE department_id = v_department_id)
LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
    END LOOP;
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text;
END;
/
CREATE OR REPLACE PACKAGE pkg_temp AS
v_department_id_pkg PLS_INTEGER := 50;
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    --v_department_id pls_integer := 30;
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employees WHERE department_id =
pkg_temp.v_department_id_pkg) LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
    END LOOP;
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text;
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    v_department_id PLS_INTEGER := 30;
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employees WHERE department_id = :1) LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
    END LOOP;
```

```
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text USING v_department_id;
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    v_department_id PLS_INTEGER := 30;
    v_max_salary PLS_INTEGER := 0;
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employees WHERE department_id = :1) LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
        if r_emp.salary > :sal then
            :sal := r_emp.salary;
        end if;
    END LOOP;
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text USING v_department_id, IN OUT v_max_salary;
    dbms_output.put_line('The maximum salary of this department is : '||
v_max_salary);
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    v_department_id PLS_INTEGER := 30;
    v_max_salary PLS_INTEGER := 0;
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employeese WHERE department_id = :1) LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
        if r_emp.salary > :sal then
            :sal := r_emp.salary;
        end if;
    END LOOP;
    END;]';
    EXECUTE IMMEDIATE v_dynamic_text USING v_department_id, IN OUT v_max_salary;
    dbms_output.put_line('The maximum salary of this department is : '||
v_max_salary);
EXCEPTION
    WHEN OTHERS THEN
    dbms_output.put_line('The error is : '||sqlerrm);
END;
/
DECLARE
    v_dynamic_text VARCHAR2(1000);
    v_department_id PLS_INTEGER := 30;
    v_max_salary PLS_INTEGER := 0;
BEGIN
    v_dynamic_text := q'[BEGIN
    FOR r_emp in (SELECT * FROM employeese WHERE department_id = :1) LOOP
        dbms_output.put_line(r_emp.first_name||' '||r_emp.last_name);
        if r_emp.salary > :sal then
            :sal := r_emp.salary;
        end if;
    END LOOP;
    EXCEPTION
    WHEN OTHERS THEN
    dbms_output.put_line('The error is : '||SQLERRM);
    END;]';
```

```
    EXECUTE IMMEDIATE v_dynamic_text USING v_department_id, IN OUT v_max_salary;
    dbms_output.put_line('The maximum salary of this department is : '||
v_max_salary);
END;
/
DROP PACKAGE pkg_temp;
--------------------------------------------------------------------------------
------------------------------
--------------OPEN - FOR, FETCH Statements (Code
Sample)-----------------------------------------
--------------------------------------------------------------------------------
------------------------------
DECLARE
  TYPE emp_cur_type  IS REF CURSOR;
  emp_cursor      emp_cur_type;
  emp_record      employees%rowtype;
BEGIN
  OPEN emp_cursor FOR 'SELECT * FROM employees WHERE job_id = ''IT_PROG''';
    FETCH emp_cursor INTO emp_record;
    dbms_output.put_line(emp_record.first_name||emp_record.last_name);
  CLOSE emp_cursor;
END;
/
DECLARE
  TYPE emp_cur_type  IS REF CURSOR;
  emp_cursor      emp_cur_type;
  emp_record      employees%rowtype;
BEGIN
  OPEN emp_cursor FOR 'SELECT * FROM employees WHERE job_id = :job' USING
'IT_PROG';
    FETCH emp_cursor INTO emp_record;
    dbms_output.put_line(emp_record.first_name||emp_record.last_name);
  CLOSE emp_cursor;
END;
/
DECLARE
  TYPE emp_cur_type  IS REF CURSOR;
  emp_cursor      emp_cur_type;
  emp_record      employees%rowtype;
BEGIN
  OPEN emp_cursor FOR 'SELECT * FROM employees WHERE job_id = :job' USING
'IT_PROG';
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%notfound;
    dbms_output.put_line(emp_record.first_name||emp_record.last_name);
  END LOOP;
  CLOSE emp_cursor;
END;
/
DECLARE
  TYPE emp_cur_type  IS REF CURSOR;
  emp_cursor      emp_cur_type;
  emp_record      employees%rowtype;
  v_table_name    VARCHAR(20);
BEGIN
  v_table_name := 'employees';
  OPEN emp_cursor FOR 'SELECT * FROM '||v_table_name||' WHERE job_id = :job' USING
'IT_PROG';
```

```
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%notfound;
    dbms_output.put_line(emp_record.first_name||emp_record.last_name);
  END LOOP;
  CLOSE emp_cursor;
END;
--------------------------------------------------------------------------------
--------------------------------
-------------------------Using the DBMS_SQL Package (Code
Samples)----------------------------
--------------------------------------------------------------------------------
--------------------------------
CREATE TABLE employees_copy AS SELECT * FROM employees;

/
set serveroutput on;
DECLARE
    v_table_name VARCHAR2(20) := 'employees_copy';
    v_cursor_id PLS_INTEGER;
    v_affected_rows PLS_INTEGER;
BEGIN
    v_cursor_id := dbms_sql.open_cursor;
    dbms_sql.parse(v_cursor_id, 'update '||v_table_name||' set
salary=salary+500',dbms_sql.NATIVE);
    v_affected_rows := dbms_sql.EXECUTE(v_cursor_id);
    dbms_output.put_line(v_affected_rows|| ' rows are updated by dbms_sql!');
    dbms_sql.close_cursor(v_cursor_id);
END;

/

select * from employees_copy;

/

DECLARE
    v_table_name varchar2(20) := 'employees_copy';
    v_cursor_id pls_integer;
    v_affected_rows pls_integer;
BEGIN
    v_cursor_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(v_cursor_id, 'update '||v_table_name||' set salary=salary+500
WHERE job_id = :jid',DBMS_SQL.NATIVE);
    DBMS_SQL.BIND_VARIABLE(v_cursor_id, ':jid','IT_PROG');
    v_affected_rows := DBMS_SQL.EXECUTE(v_cursor_id);
    dbms_output.put_line(v_affected_rows|| ' rows are updated by dbms_sql!');
    DBMS_SQL.CLOSE_CURSOR(v_cursor_id);
END;

/

DECLARE
    v_table_name varchar2(20) := 'employees_copy';
    v_cursor_id pls_integer;
    v_affected_rows pls_integer;
BEGIN
    v_cursor_id := DBMS_SQL.OPEN_CURSOR;
    DBMS_SQL.PARSE(v_cursor_id, 'update '||v_table_name||' set salary=salary+:inc
```

```
WHERE job_id = :jid',DBMS_SQL.NATIVE);
    DBMS_SQL.BIND_VARIABLE(v_cursor_id, ':jid','IT_PROG');
    DBMS_SQL.BIND_VARIABLE(v_cursor_id, ':inc','5');
    v_affected_rows := DBMS_SQL.EXECUTE(v_cursor_id);
    dbms_output.put_line(v_affected_rows|| ' rows are updated by dbms_sql!');
    DBMS_SQL.CLOSE_CURSOR(v_cursor_id);
END;

/

SELECT * FROM user_tab_columns;
EXEC prc_method4_example('employees');
EXEC prc_method4_example('departments');
EXEC prc_method4_example('countries');
EXEC prc_method4_example('locations');
/

create or replace PROCEDURE prc_method4_example (p_table_name IN VARCHAR2) IS
    TYPE t_columns IS TABLE OF user_tab_columns%rowtype INDEX BY PLS_INTEGER;
    v_columns               t_columns;
    v_columns_with_commas   VARCHAR2(32767);
    v_number_value          NUMBER;
    v_string_value          VARCHAR2(32767);
    v_date_value            DATE;
    v_output_string         VARCHAR2(32767);
    cur_dynamic             INTEGER;
BEGIN
    SELECT * BULK COLLECT INTO v_columns FROM user_tab_columns WHERE table_name =
upper(p_table_name);
    v_columns_with_commas:=v_columns(1).column_name;
    FOR i IN 2..v_columns.COUNT LOOP
        v_columns_with_commas:=v_columns_with_commas||','||
v_columns(i).column_name;
    END LOOP;
    cur_dynamic := dbms_sql.open_cursor;
    dbms_sql.parse(cur_dynamic,'SELECT '||v_columns_with_commas||' FROM '||
p_table_name,dbms_sql.NATIVE);
     FOR idx IN 1..v_columns.COUNT LOOP
        IF v_columns(idx).data_type = 'NUMBER' THEN
            dbms_sql.define_column(cur_dynamic,idx,1);
        ELSIF v_columns(idx).data_type IN ('VARCHAR2','VARCHAR','CHAR') THEN
            dbms_sql.define_column(cur_dynamic,idx,'dummy
text',v_columns(idx).char_length);
        ELSIF v_columns(idx).data_type = 'DATE' THEN
            dbms_sql.define_column(cur_dynamic,idx,sysdate);
        END IF;
        v_output_string:=v_output_string||'  '||
rpad(v_columns(idx).column_name,20);
     END LOOP;
     dbms_output.put_line(v_output_string);
     v_number_value:=dbms_sql.execute(cur_dynamic);
     WHILE dbms_sql.fetch_rows(cur_dynamic) > 0 LOOP
        v_output_string:=NULL;
        FOR t IN 1..v_columns.COUNT LOOP
            IF v_columns(T).data_type = 'NUMBER' THEN
                dbms_sql.column_value(cur_dynamic,t,v_number_value);
                v_output_string := v_output_string||'  '||
rpad(nvl(to_char(v_number_value),' '),20);
            ELSIF v_columns(T).data_type IN ('VARCHAR2','VARCHAR','CHAR') THEN
```

```
                dbms_sql.column_value(cur_dynamic,t,v_string_value);
                v_output_string := v_output_string||'  '||
rpad(nvl(to_char(v_string_value),' '),20);
            ELSIF v_columns(T).data_type = 'DATE' THEN
                dbms_sql.column_value(cur_dynamic,t,v_date_value);
                v_output_string := v_output_string||'  '||
rpad(nvl(to_char(v_date_value),' '),20);
            END IF;
        END LOOP;
        dbms_output.put_line(v_output_string);
     END LOOP;
END;
```