DAY 15

```
PRN : 200243020003
```

PL/SQL FUNCTION

1.Write a function which returns basic salary of an employee (DA+HRA+ALLOWANCE). Later use the above function to calculate total salary of employees

```
CREATE OR REPLACE FUNCTION total_salary RETURN number
AS HRA
    number: = 100;
DA number: = 100;
allowance number: = 100;
basic_sal number;
BEGIN
    basic_sal: =
HRA + DA + allowance;

RETURN basic_sal;

END;

/
DECLARE
    sal NUMBER: = 1000;
    total number;
BEGIN
    total: = sal + annual_salary;
    dbms_output.put_line (total);
END;

/
```

2.Write a function which returns max salary of user entered department_id. Use this function to display max salary and its department details

```
CREATE OR REPLACE FUNCTION max_salary (id IN NUMBER) RETURN NUMBER
AS sal
    NUMBER;
BEGIN
    SELECT
        max(SALARY) INTO
sal
FROM
```

```
    EMPLOYEES
WHERE
    DEPARTMENT_ID = id;

RETURN sal;

END;

/
DECLARE
    dept_id number: = & id;
    sal NUMBER;
BEGIN
    sal: = max_salary (dept_id);
    dbms_output.put_line (sal);
END;

/
```

3.Create and compile a function called GET_JOB to return a job title. Create a VARCHAR2 host variable called TITLE, allowing a length of 35 characters. Invoke the function with SA_REP job ID to return the value in the host variable. Print the host variable to view the result.

```
CREATE OR REPLACE FUNCTION get_job (id NUMBER) RETURN VARCHAR2
AS job
    VARCHAR2 (30);
BEGIN
    SELECT
        JOB_ID INTO
job
FROM
    EMPLOYEES
WHERE
    EMPLOYEE_ID = id;

RETURN job;

END;

/
DECLARE
    job_title varchar2 (30);
    id NUMBER: = & id;
BEGIN
    job_title: = get_job (id);
    dbms_output.put_line (job_title);
END;

/
```

4.Create a function called GET_ANNUAL_COMP to return the annual salary computed from an employee?s monthly salary and commission passed as parameters.Develop and store the GET_ANNUAL_COMP function, accepting parameter values for monthly salary and commission. Either or both values passed can be NULL, but the function should still return a non-NULL annual salary. Use the following basic formula to calculate the annual salary:(salary*12) + (commission_pct*salary*12) Use the function in a SELECT statement against the EMPLOYEES table for employees in department 30.

```
CREATE OR REPLACE FUNCTION GET_ANNUAL_COMP (sal IN number, comm IN number)
RETURN number
AS annual_salary
    number;
BEGIN
    annual_salary: = (nvl (sal,
            1) * 12) + (nvl (comm,
            1) * nvl (sal,
            1) * 12);
    RETURN
annual_salary;

END;

DECLARE
    sal number: = & sal;
    comm number: = & comm;
    Annual_salary number;
BEGIN
    Annual_salary: = GET_ANNUAL_COMP (sal,
        comm);
    DBMS_OUTPUT.PUT_LINE ('Annual Salary::' || Annual_salary);
END;

/
SELECT
    GET_ANNUAL_COMP (SALARY,
        COMMISSION_PCT)
FROM
    EMPLOYEES
WHERE
    DEPARTMENT_ID = 30;
```

PL/SQL PROCEDURE

1.Create a procedure called ADD_JOB to insert a new job into the JOBS table. Provide the ID and title of the job using two parameters.

```
CREATE OR REPLACE PROCEDURE ADD_JOB (id IN VARCHAR2, job_title IN
VARCHAR2)
AS BEGIN
    INSERT INTO JOBS (JOB_ID, JOB_TITLE)
```

```
        VALUES(id, job_title);
    dbms_output.put_line (sql % rowcount || ' row inserted');
        COMMIT;
END;
DECLARE
    id VARCHAR2 (30) := & id;
    job VARCHAR2 (30) := & job;
BEGIN ADD_JOB (id,
    job);
END;
```

2.Create a procedure called UPD_JOB to update the job title. Provide the job ID and a new title using two parameters. Include the necessary exception handling if no update

```
    CREATE OR REPLACE PROCEDURE UPD_JOB (id IN VARCHAR2, new_title IN
VARCHAR2)
    AS BEGIN
    UPDATE
        JOBS
    SET
        JOB_TITLE = new_title
    WHERE
        JOB_ID = id;
    dbms_output.put_line (sql % rowcount || ' row updated');
    COMMIT;
END;
    EXECUTE UPD_JOB ('AC_MGR',
        'CODER');
```

3.Create a procedure, ADD_EMPLOYEE, to insert a new employee into the EMPLOYEE table. The procedure should call a VALID_DEPTID function to check whether the department ID specified for the new employee exists in the DEPARTMENTS table.

```
    CREATE OR REPLACE PROCEDURE add_emp AS
    BEGIN
        INSERT INTO emp100
            values(25, 'rayn', 35000, 122,.90);
        dbms_output.put_line (sql % rowcount || ' row inserted');
    END;
        / CREATE OR REPLACE PROCEDURE check_dept (did IN out number ) AS d
number;
        CURSOR check1 IS
        SELECT
            dept_id
        FROM
            dpt
        WHERE
            dept_id = did;
```

```
        BEGIN
            OPEN check1;
        LOOP
            FETCH check1 INTO d;
            exit
            WHEN check1 % notfound;
        END LOOP;
        IF check1 % rowcount = 0 THEN
            did: = 0;
        ELSE
            did: = 1;
        END IF;
        END;
        /
DECLARE
    dept number: = & did;
    flag number;
BEGIN
    flag: = dept;
    check_dept (flag);
    IF flag = 1 THEN
        add_emp;
    ELSE
        dbms_output.put_line ('enter valid department number');
    END IF;
END;
```

4.Create a procedure called GET_EMPLOYEE to query the EMPLOYEES table, retrieving the salary and job ID for an employee when provided with the employee ID. Execute the procedure using host variables for the two OUT parameters?one for the salary and the other for the job ID. Display the salary and job ID for employee ID 120.

```
CREATE OR REPLACE PROCEDURE GET_EMPLOYEE (id IN NUMBER, sal out VARCHAR2,
jobid out VARCHAR2) AS
BEGIN
SELECT
    SALARY,
    JOB_ID INTO sal,
    jobid
FROM
    EMPLOYEES
WHERE
    EMPLOYEE_ID = id;

COMMIT;

END;

DECLARE
    job employees.job_id % TYPE;
    sal number;
```

```
BEGIN
    get_employee (120,
        sal,
        job);
    dbms_output.put_line ('employee_id:120' || ' job_id:' || job || '
salary:' || sal);
END;

/
```