# CO3 ASSIGNMENT QUESTIONS

1. **Course registration application has a department table with dept id, dept name and head of department fields. Write a pl/sql block to declare variables for the above fields and display the same.**

**%Type :**

The `%TYPE` attribute in PL/SQL lets you declare variables with the same data type as a column, parameter, or another variable in the database, ensuring consistency and adaptability in your code.

**Query :**

```
DECLARE
   v_dept_id department.dept_id%TYPE;
   v_dept_name department.dept_name%TYPE;
   v_head_of_dept department.head_of_department%TYPE;
BEGIN
   v_dept_id := 1;
   v_dept_name := 'Computer Science';
   v_head_of_dept := 'John Doe';
   DBMS_OUTPUT.PUT_LINE('Department ID: ' || v_dept_id);
   DBMS_OUTPUT.PUT_LINE('Department Name: ' || v_dept_name);
   DBMS_OUTPUT.PUT_LINE('Head of Department: ' || v_head_of_dept);
END;
```

2. **Create a pl/sql procedure to update the salaries of all employees 10% in their basic pay. Verify whether the row is updated and display the number of records affected by the operation. Write a program in pl/sql to display a cursor based detail information of employees from employees table.**

**Procedure :**

The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.

The procedure contains a header and a body.

- **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.

- **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

**Query :**

```
CREATE OR REPLACE PROCEDURE update_salaries AS
BEGIN
   UPDATE employees SET salary = salary * 0.1;
   DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' records updated.');
   COMMIT;
END;

BEGIN
Update_salaries;
END;
```

**Cursor :**
When an SQL statement is processed, Oracle creates a memory area known as context area. A cursor is a pointer to this context area. It contains all information needed for processing the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information ona select statement and the rows of data accessed by it.

A cursor is used to referred to a program to fetch and process the rows returned by the SQLstatement, one at a time. There are two types of cursors:

- Implicit Cursors
- Explicit Cursors

**Query :**

```
DECLARE
  CURSOR employee_cursor IS
    SELECT employee_id, first_name, last_name, salary,
    hire_dateFROM employees;

 v_employee_id employees.employee_id%TYPE;
 v_first_name employees.first_name%TYPE;
 v_last_name employees.last_name%TYPE;
 v_salary employees.salary%TYPE;
 v_hire_date employees.hire_date%TYPE;
 BEGIN
     OPEN employee_cursor;
   LOOP
     FETCH employee_cursor INTO v_employee_id, v_first_name, v_last_name, v_salary, v_hire_date;
    EXIT WHEN employee_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_first_name || ' ' || v_last_name);
    DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
    DBMS_OUTPUT.PUT_LINE('Hire Date: ' || v_hire_date);
    DBMS_OUTPUT.PUT_LINE('-------------------------');
  END LOOP;

  CLOSE employee_cursor;
END;
```

3. **Develop a stored function sf_get_emp_id_by_deptid that accepts department id and returns the employee id working under that department. In case of invalid department id return -1, more than one row is fetched return -2 and for any other error return -3. Invoke the function from anonymous block for department ids 10, 20 and 25.**

**Function :**
The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value. Except this, all the other things of PL/SQL procedure are true for PL/SQL function too.

**Query :**

```
CREATE OR REPLACE FUNCTION sf_get_emp_id_by_deptid(p_dept_id IN
employees.department_id%TYPE)
RETURN NUMBER IS
    v_emp_id employees.employee_id%TYPE;
BEGIN
    SELECT employee_id INTO v_emp_id FROM employees
    WHERE department_id = p_dept_id;

    IF v_emp_id IS NULL THEN
        RETURN -1;
    ELSIF SQL%ROWCOUNT > 1 THEN
        RETURN -2;
    END IF;

    RETURN v_emp_id;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN -1;
    WHEN TOO_MANY_ROWS THEN
        RETURN -2;
    WHEN OTHERS THEN
        RETURN -3;
END;
DECLARE
    v_result1   NUMBER;
    v_result2   NUMBER;
    v_result3  NUMBER;
BEGIN
    v_result1 := sf_get_emp_id_by_deptid(10);
    DBMS_OUTPUT.PUT_LINE('Result for department ID 10: ' || v_result1);
    v_result2 := sf_get_emp_id_by_deptid(1);
    DBMS_OUTPUT.PUT_LINE('Result for department ID 20: ' || v_result2);
    v_result3 := sf_get_emp_id_by_deptid(25);
    DBMS_OUTPUT.PUT_LINE('Result for department ID 25: ' || v_result3);
END;
```

4. **Hr manager wants to see the details (employee id, first_name, job_title, department_id and hire date) of all the employees who report to a specific manager. Develop a pl/sql program to display the details in below format for each employee who reports to alexander, employee id 103.**
   **Cursor :**

When an SQL statement is processed, Oracle creates a memory area known as context area. A cursor is a pointer to this context area. It contains all information needed for processing the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information ona select statement and the rows of data accessed by it.

A cursor is used to referred to a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors:

- Implicit Cursors
- Explicit Cursors

**Query :**

```
DECLARE
  v_employee_id employees.employee_id%TYPE;
  v_first_name employees.first_name%TYPE;
  v_job_title employees.job_title%TYPE;
  v_department_id employees.department_id%TYPE;
  v_hire_date employees.hire_date%TYPE;

  CURSOR employee_cursor IS
SELECT e.employee_id, e.first_name, e.job_title, e.department_id, e.hire_date
FROM employees e JOIN employees m ON e.manager_id = m.employee_id
WHERE m.employee_id = 103;
BEGIN
  OPEN employee_cursor;
  LOOP
    FETCH employee_cursor INTO v_employee_id, v_first_name, v_job_title, v_department_id,
v_hire_date;
    EXIT WHEN employee_cursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_first_name);
    DBMS_OUTPUT.PUT_LINE('Job Title: ' || v_job_title);
    DBMS_OUTPUT.PUT_LINE('Department ID: ' || v_department_id);
    DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(v_hire_date, 'DD-MON-YYYY'));
    DBMS_OUTPUT.PUT_LINE(' -------------------------');
  END LOOP;
  CLOSE employee_cursor;
END;
```

5. **Hr manager needs a report that shows how many employees are working on a particular job. Write a pl/sql code to get report for 'programmer' job. Before generating report, given job should be checked whether it is valid or not. If not, display appropriate message.**

```
DECLARE
v_employee_count NUMBER;
BEGIN
SELECT COUNT(*)
INTO v_employee_count FROM employees75
WHERE job_title = 'Programmer';

IF v_employee_count = 0 THEN
DBMS_OUTPUT.PUT_LINE('No employees found for job: Programmer'); ELSE
DBMS_OUTPUT.PUT_LINE('Number of employees working as Programmer : ' || v_employee_count);
END IF;
END;
```

6. **Every employee salary should be greater than or equal to the minimum salary of his job. Before adding any employee this requirement has to be checked. If the salary is less than the minimum salary, then employee's salary should be inserted with the minimum salary of his job. write a triggerfor this.**

```
CREATE OR REPLACE TRIGGER check_employee_salary
BEFORE INSERT ON employees
FOR EACH ROW
DECLARE
    v_min_salary NUMBER;
    v_job_title VARCHAR2(100);
BEGIN
    SELECT MIN(salary), job_title INTO v_min_salary, v_job_title
    FROM employees
    WHERE job_title = :NEW.job_title;

    IF v_min_salary IS NULL THEN
        v_min_salary := 5000;
    END IF;

    IF :NEW.salary < v_min_salary THEN
        :NEW.salary := v_min_salary;
    END IF;
END;
```