PL SQL Assignment

Author: Kathiravan A

IFET College of Engineering

Date: 23.07.2024

1.
```
CREATE TABLE EMPLOYEES (
   EMP_ID      NUMBER PRIMARY KEY,
   EMP_NAME    VARCHAR2(100),
   DEPARTMENT  VARCHAR2(50),
   SALARY      NUMBER
);
CREATE OR REPLACE PROCEDURE insert_employee (
  p_emp_id IN EMPLOYEES.EMP_ID%TYPE,
  p_emp_name IN EMPLOYEES.EMP_NAME%TYPE,
  p_department IN EMPLOYEES.DEPARTMENT%TYPE,
  p_salary IN EMPLOYEES.SALARY%TYPE
)
IS
BEGIN
  INSERT INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY)
  VALUES (p_emp_id, p_emp_name, p_department, p_salary);
END;
/
INSERT ALL
  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (100, 'kathir',
'Development', 30000)
  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (101, 'karthi',
'Development', 30000)
  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (102, 'bharath',
'Development', 40000)
  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (103, 'balaji', 'manager',
33000)
  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (103,  'Karan',
'teamLead', 31000)
```

```
  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (103, 'hari', 'teamLead',
45000)

  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (103, 'naveen',
'marketing', 25000)

  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (103, 'arun', 'marketing',
25000)

  INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY) VALUES (103, 'akash', 'Sales',
27000)

SELECT * FROM EMPLOYEES;


2. CREATE OR REPLACE PROCEDURE update_salary(p_emp_id IN EMPLOYEES.EMP_ID%TYPE)

IS

  v_current_salary EMPLOYEES.SALARY%TYPE;

  v_salary_increase EMPLOYEES.SALARY%TYPE;

BEGIN

  SELECT SALARY INTO v_current_salary FROM EMPLOYEES WHERE EMP_ID = p_emp_id;


  IF v_current_salary < 5000 THEN

    v_salary_increase := v_current_salary * 0.1;

  ELSIF v_current_salary BETWEEN 5000 AND 10000 THEN

    v_salary_increase := v_current_salary * 0.075;

  ELSE

    v_salary_increase := v_current_salary * 0.05;

  END IF;


  UPDATE EMPLOYEES

  SET SALARY = SALARY + v_salary_increase

  WHERE EMP_ID = p_emp_id;


  COMMIT;

EXCEPTION

  WHEN NO_DATA_FOUND THEN

    DBMS_OUTPUT.PUT_LINE('Employee not found.');

  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);

    ROLLBACK;
```

```
END;
/

3. DECLARE
  CURSOR emp_cursor IS
    SELECT EMP_NAME
    FROM EMPLOYEES;

  v_emp_name EMPLOYEES.EMP_NAME%TYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_emp_name;
    EXIT WHEN emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emp_name);
  END LOOP;
  CLOSE emp_cursor;
END;
/

4. CREATE VIEW high_salary_employees AS
SELECT EMP_ID, EMP_NAME, DEPARTMENT, SALARY
FROM EMPLOYEES
WHERE SALARY > 30000;

5. CREATE OR REPLACE FUNCTION calculate_bonus(p_salary IN EMPLOYEES.SALARY%TYPE)
RETURN EMPLOYEES.SALARY%TYPE
IS
  v_bonus EMPLOYEES.SALARY%TYPE;
BEGIN
  IF p_salary < 3000 THEN
    v_bonus := p_salary * 0.1;
  ELSIF p_salary BETWEEN 25000 AND 40000 THEN
    v_bonus := p_salary * 0.075;
```

```
  ELSE
    v_bonus := p_salary * 0.05;
  END IF;
  RETURN v_bonus;
END;
/
```

6.
```
CREATE OR REPLACE TRIGGER log_employee_insert
AFTER INSERT ON EMPLOYEES
FOR EACH ROW
BEGIN
  INSERT INTO employee_log (emp_id, emp_name, department, salary, action_date)
  VALUES (:new.emp_id, :new.emp_name, :new.department, :new.salary, SYSDATE);
END;
/
```

7.

A)
```
CREATE OR REPLACE VIEW customer_sales_revenue AS
SELECT
    o.customer_id,
    SUM(oi.quantity * oi.unit_price) AS total_sales,
    SUM(oi.quantity * oi.unit_price) * 0.05 AS credit
FROM
    orders o
INNER JOIN order_items oi ON o.order_id = oi.order_id
GROUP BY
    o.customer_id;
```

B)
```
DECLARE
  CURSOR cust_cursor IS
    SELECT customer_id, total_sales
    FROM customer_sales_revenue
    ORDER BY total_sales DESC;

  v_customer_id customers.customer_id%TYPE;
```

```
  v_total_sales customer_sales_revenue.total_sales%TYPE;
  v_credit_limit NUMBER := 10000;
  v_allocated_credit NUMBER := 0;
BEGIN
  -- Reset credit limits
  UPDATE customers SET credit_limit = 0;

  OPEN cust_cursor;
  LOOP
    FETCH cust_cursor INTO v_customer_id, v_total_sales;
    EXIT WHEN cust_cursor%NOTFOUND;

    -- Calculate credit limit based on available budget
    IF v_credit_limit >= v_total_sales * 0.1 THEN
      UPDATE customers SET credit_limit = v_total_sales * 0.1 WHERE customer_id = v_customer_id;
      v_credit_limit := v_credit_limit - v_total_sales * 0.1;
    ELSE
      UPDATE customers SET credit_limit = v_credit_limit WHERE customer_id = v_customer_id;
      v_credit_limit := 0;
      EXIT; -- No more budget
    END IF;
  END LOOP;
  CLOSE cust_cursor;
END;
/
```

8.
```
CREATE TABLE EMPLOYEES_DETAILS (
    EMPLOYEE_ID NUMBER,
    FIRST_NAME VARCHAR2(50),
    LAST_NAME VARCHAR2(50),
    EMAIL VARCHAR2(100),
    PHONE_NUMBER VARCHAR2(20),
    HIRE_DATE DATE,
    JOB_ID VARCHAR2(10),
```

```
    SALARY NUMBER,
    COMMISSION_PCT NUMBER,
    MANAGER_ID NUMBER,
    DEPARTMENT_ID NUMBER
);
```

```
INSERT INTO EMPLOYEES_DETAILS (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID,
DEPARTMENT_ID)
VALUES (1, 'Kathiravan', 'A', 'ak@gmail.com', '9876543210', TO_DATE('2002-12-29', 'YYYY-MM-DD'),
'DEVOPS, 60000, 0.10, 1, 101);
```

```
INSERT INTO EMPLOYEES_DETAILS (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID,
DEPARTMENT_ID)
VALUES (2, 'Bharath', 'S', 'bh@gmail.com', '9345234535', TO_DATE('2002-06-12', 'YYYY-MM-DD'),
'Leader', 60000, 0.08, 1, 102);
```

```
INSERT INTO EMPLOYEES_DETAILS (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL,
PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID,
DEPARTMENT_ID)
VALUES (3, 'Balaji', 'K', 'bk@gmail.com', '9887654634', TO_DATE('2003-09-12', 'YYYY-MM-DD'), 'Testing',
40000, 0.02, 2, 103);
```

```
COMMIT;
```

```
DROP TABLE EMPLOYEES_DETAILS;
SELECT * FROM EMPLOYEES_DETAILS;
```

```
DECLARE
CURSOR C2
IS
SELECT EMPLOYEE_ID,FIRST_NAME,LAST_NAME FROM EMPLOYEES_DETAILS;
v_employee_id EMPLOYEES_DETAILS.EMPLOYEE_ID%TYPE;
```

```
v_first_name EMPLOYEES_DETAILS.FIRST_NAME%TYPE;
v_last_name EMPLOYEES_DETAILS.LAST_NAME%TYPE;
BEGIN
OPEN C2;
LOOP
FETCH C2 INTO v_employee_id,v_first_name,v_last_name;
EXIT WHEN C2%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Employee_ID: ' || v_employee_id);
DBMS_OUTPUT.PUT_LINE('First_Name: ' || v_first_name);
DBMS_OUTPUT.PUT_LINE('Last_Name: ' || v_last_name);
END LOOP;
CLOSE C2;
END;
/
```

9.
```
DECLARE
  CURSOR emp_cursor (p_max_salary NUMBER) IS
    SELECT first_name, salary
    FROM employees
    WHERE salary < p_max_salary;

  v_first_name employees.first_name%TYPE;
  v_salary employees.salary%TYPE;
BEGIN
  OPEN emp_cursor(50000);

  LOOP
    FETCH emp_cursor INTO v_first_name, v_salary;
    EXIT WHEN emp_cursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Name: ' || v_first_name || ', Salary: ' || v_salary);
  END LOOP;

  CLOSE emp_cursor;
```

```
END;
/

10. CREATE OR REPLACE TRIGGER check_duplicate_values
BEFORE INSERT OR UPDATE ON your_table
FOR EACH ROW
DECLARE
  v_count NUMBER;
BEGIN
  SELECT COUNT(*)
  INTO v_count
  FROM your_table
  WHERE your_column = :new.your_column;

  IF v_count > 0 THEN
    RAISE_APPLICATION_ERROR(-20001, 'Duplicate value found for column your_column');
  END IF;
END;
/

11. CREATE OR REPLACE PROCEDURE select_records (
  p_column1 IN your_table.column1%TYPE,
  p_column2 IN your_table.column2%TYPE,
  p_filter_value IN VARCHAR2
)
IS
  CURSOR emp_cursor IS
    SELECT column1, column2, ...
    FROM your_table
    WHERE column1 = p_column1
      AND column2 = p_column2
      AND some_column LIKE '%' || p_filter_value || '%';

  v_column1 your_table.column1%TYPE;
```

```
  v_column2 your_table.column2%TYPE;
  -- ... other columns
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO v_column1, v_column2, ...;
    EXIT WHEN emp_cursor%NOTFOUND;

    -- Process fetched data
    DBMS_OUTPUT.PUT_LINE('Column1: ' || v_column1 || ', Column2: ' || v_column2 || ' ...');
  END LOOP;
  CLOSE emp_cursor;
END;
/
```

12.
```
DECLARE
  v_salary NUMBER;
BEGIN
  SELECT salary
  INTO v_salary
  FROM employees
  WHERE employee_id = 102;

  UPDATE employees
  SET salary = v_salary + 1000
  WHERE employee_id = 102;

  COMMIT;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
    ROLLBACK;
```

```
END;
/
```