PL/SQL Assignment

Author: Krishna G

Created Date: 15/07/2024

1) **Create a Procedure to Insert Employee Data, Write a PL/SQL procedure named insert_employee to insert employee data into the EMPLOYEES table:**

   Table structure: EMPLOYEES (EMP_ID NUMBER, EMP_NAME VARCHAR2(100), DEPARTMENT VARCHAR2(50), SALARY NUMBER)

```
CREATE OR REPLACE PROCEDURE insert_employee (
p_emp_id     IN NUMBER,    p_emp_name    IN
VARCHAR2,    p_department  IN VARCHAR2,
p_salary     IN NUMBER
)  IS
BEGIN
   INSERT INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY)
   VALUES (p_emp_id, p_emp_name, p_department, p_salary);

   COMMIT;
END;
/
```

Calling Procedure:

```
BEGIN
   insert_employee(105, 'Ram', 'Software Developer', 75000);
END;
/
```

2) Create a Procedure to Update Employee Salary, Write a PL/SQL procedure named update_salary to update an employee's salary based on their current salary:

If the current salary is less than 5000, increase it by 10%.

If the current salary is between 5000 and 10000, increase it by 7.5%.

If the current salary is more than 10000, increase it by 5%.

```
CREATE OR REPLACE PROCEDURE update_salary (
p_emp_id IN NUMBER
) IS
   v_current_salary EMPLOYEES.SALARY%TYPE;
BEGIN
   SELECT SALARY INTO v_current_salary
   FROM EMPLOYEES
   WHERE EMP_ID = p_emp_id;
   IF v_current_salary < 5000 THEN
      UPDATE EMPLOYEES
      SET SALARY = SALARY * 1.10
      WHERE EMP_ID = p_emp_id;
   ELSIF v_current_salary BETWEEN 5000 AND 10000 THEN
      UPDATE EMPLOYEES
      SET SALARY = SALARY * 1.075
      WHERE EMP_ID = p_emp_id;
   ELSE
      UPDATE EMPLOYEES
      SET SALARY = SALARY * 1.05
      WHERE EMP_ID = p_emp_id;
   END IF;
   COMMIT;
END;
/
```

Calling Procedure:

```
 BEGIN
    update_salary(103);
END;
/
```

3) **Use a Cursor to Display Employee Names, Write a PL/SQL block using a cursor to fetch and display all employee names from the EMPLOYEES table.**

```
DECLARE
    CURSOR emp_cursor IS
        SELECT EMP_NAME
FROM EMPLOYEES;
    v_emp_name EMPLOYEES.EMP_NAME%TYPE;
BEGIN
    -- Open the cursor
    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 a View for Employees with High Salary, Write a SQL statement to create a view named high_salary_employees that displays employees earning more than 10000.**

```
CREATE VIEW high_salary_employees AS
SELECT EMP_ID, EMP_NAME, DEPARTMENT, SALARY
```

```
FROM EMPLOYEES
WHERE SALARY > 10000;
```

5) **Create a Function to Calculate Bonus, Write a PL/SQL function named calculate_bonus to calculate the bonus based on an employee's salary:**
Employees earning less than 5000 get a bonus of 10% of their salary.
Employees earning between 5000 and 10000 get a bonus of 7.5% of their salary.
Employees earning more than 10000 get a bonus of 5% of their salary.

```
CREATE OR REPLACE FUNCTION calculate_bonus (
p_salary IN NUMBER
)
RETURN NUMBER
IS
  v_bonus NUMBER;
BEGIN
  IF p_salary < 5000 THEN
v_bonus := p_salary * 0.10;
  ELSIF p_salary BETWEEN 5000 AND 10000 THEN
    v_bonus := p_salary * 0.075;
ELSE
    v_bonus := p_salary * 0.05;
  END IF;

  RETURN v_bonus;
END;
/
```

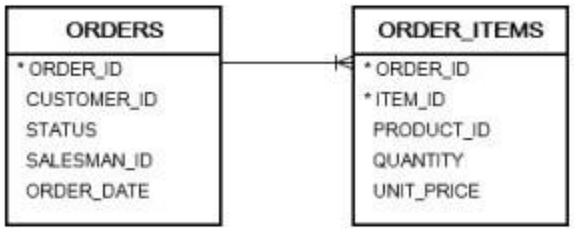
Running the Block:

```
SELECT EMP_ID, EMP_NAME, SALARY, calculate_bonus(SALARY) AS BONUS
FROM EMPLOYEES;
```

6) **Create a Trigger to Log Employee Insertions, Write a PL/SQL trigger named log_employee_insert to log whenever an employee is inserted into the EMPLOYEES table.**

```
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)
   VALUES (:NEW.EMP_ID, :NEW.EMP_NAME, :NEW.DEPARTMENT,
:NEW.SALARY);
END;
/
```

7) **Consider the orders and order_items tables from the sample database.**



A) **Create a view that returns the sales revenues by customers. The values of the credit column are 5% of the total sales revenues.**

```
CREATE VIEW CustomerSalesRevenues AS
SELECT
   o.CUSTOMER_ID,
   SUM(oi.QUANTITY * oi.UNIT_PRICE) AS Total_Revenue,
   SUM(oi.QUANTITY * oi.UNIT_PRICE) * 0.05 AS Credit
FROM
   ORDERS o
JOIN
```

```
    ORDER_ITEMS oi
ON
    o.ORDER_ID = oi.ORDER_ID
GROUP BY
    o.CUSTOMER_ID;
```

B) Write the PL/SQL query to develop an anonymous block which:

1. **Reset the credit limits of all customers to zero.**
2. **Fetch customers sorted by sales in descending order and give them new credit limits from a budget of 1 million.**

```
DECLARE
    v_budget NUMBER := 1000000;
v_total_sales NUMBER;
    v_customer_id ORDERS.CUSTOMER_ID%TYPE;
v_new_credit_limit NUMBER;
    CURSOR c_customers IS
        SELECT CUSTOMER_ID, SUM(oi.QUANTITY * oi.UNIT_PRICE) AS
Total_Sales
        FROM ORDERS o
        JOIN ORDER_ITEMS oi
        ON o.ORDER_ID = oi.ORDER_ID
        GROUP BY CUSTOMER_ID
        ORDER BY Total_Sales DESC;
BEGIN
    UPDATE CUSTOMERS
    SET CREDIT_LIMIT = 0;
    SELECT SUM(Total_Sales) INTO v_total_sales
    FROM (
        SELECT SUM(oi.QUANTITY * oi.UNIT_PRICE) AS Total_Sales
        FROM ORDERS o
        JOIN ORDER_ITEMS oi
        ON o.ORDER_ID = oi.ORDER_ID
        GROUP BY CUSTOMER_ID
    );
    FOR rec IN c_customers LOOP
        v_new_credit_limit := (rec.Total_Sales / v_total_sales) * v_budget;
        UPDATE CUSTOMERS
        SET CREDIT_LIMIT = v_new_credit_limit
        WHERE CUSTOMER_ID = rec.CUSTOMER_ID;
    END LOOP;
END;
```

8) Write a program in PL/SQL to show the uses of implicit cursor without using any attribute.

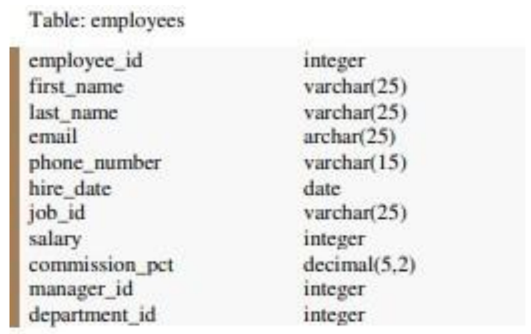


Table: employees

employee_id	integer
first_name	varchar(25)
last_name	varchar(25)
email	archar(25)
phone_number	varchar(15)
hire_date	date
job_id	varchar(25)
salary	integer
commission_pct	decimal(5,2)
manager_id	integer
department_id	integer

```
BEGIN
    FOR rec IN (SELECT * FROM employee) LOOP
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id);
        DBMS_OUTPUT.PUT_LINE('First Name: ' || rec.first_name);
        DBMS_OUTPUT.PUT_LINE('Last Name: ' || rec.last_name);
        DBMS_OUTPUT.PUT_LINE('Email: ' || rec.email);
        DBMS_OUTPUT.PUT_LINE('Phone Number: ' || rec.phone_number);
        DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(rec.hire_date, 'YYYY-
MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Job ID: ' || rec.job_id);
DBMS_OUTPUT.PUT_LINE('Salary: ' || rec.salary);
```

```
        DBMS_OUTPUT.PUT_LINE('Commission Pct: ' || rec.commission_pct);
        DBMS_OUTPUT.PUT_LINE('Manager ID: ' || rec.manager_id);
        DBMS_OUTPUT.PUT_LINE('Department ID: ' || rec.department_id);
    END LOOP;
END;
/
```

9) Write a program in PL/SQL to create a cursor displays the name and salary of each employee in the EMPLOYEES table whose salary is less than that specified by a passed in parameter value.

```
DECLARE
    v_salary_threshold INTEGER := 50000;
CURSOR employee_cursor IS
        SELECT first_name, last_name, salary
        FROM employee
        WHERE salary < v_salary_threshold;
employee_rec employee_cursor%ROWTYPE;
BEGIN
    OPEN employee_cursor;
    LOOP
        FETCH employee_cursor INTO employee_rec;
        EXIT WHEN employee_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Name: ' || employee_rec.first_name || ' ' ||
employee_rec.last_name);
        DBMS_OUTPUT.PUT_LINE('Salary: ' || employee_rec.salary);
    END LOOP;
    CLOSE employee_cursor;
END;
/
```

10) Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER check_duplicate_email
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW DECLARE
   v_count INTEGER;
BEGIN
   SELECT COUNT(*)
   INTO v_count
   FROM employee
   WHERE email = :NEW.email
   AND employee_id != :NEW.employee_id;
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Duplicate email detected: ' ||
:NEW.email);
   END IF;
END;
/
```

11) Write a PL/SQL procedure for selecting some records from the database using some parameters as filters. Consider that we are fetching details of employees from ib_employee table where salary is a parameter for filter.

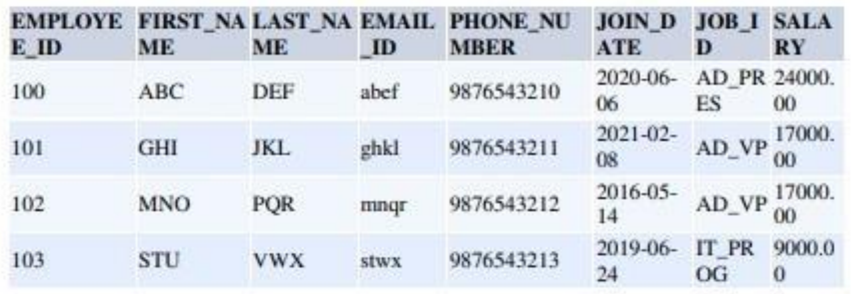
```
CREATE OR REPLACE PROCEDURE get_employees_by_salary(p_min_salary IN
NUMBER) IS
BEGIN
   FOR rec IN (
      SELECT employee_id, first_name, last_name, email, phone_number, hire_date,
job_id, salary, commission_pct, manager_id, department_id
      FROM employee
```

```
        WHERE salary > p_min_salary
    ) LOOP
        -- Display employee details
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id);
        DBMS_OUTPUT.PUT_LINE('First Name: ' || rec.first_name);
        DBMS_OUTPUT.PUT_LINE('Last Name: ' || rec.last_name);
        DBMS_OUTPUT.PUT_LINE('Email: ' || rec.email);
        DBMS_OUTPUT.PUT_LINE('Phone Number: ' || rec.phone_number);
        DBMS_OUTPUT.PUT_LINE('Hire Date: ' || TO_CHAR(rec.hire_date, 'YYYY-
MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Job ID: ' || rec.job_id);
        DBMS_OUTPUT.PUT_LINE('Salary: ' || rec.salary);
        DBMS_OUTPUT.PUT_LINE('Commission Pct: ' || rec.commission_pct);
        DBMS_OUTPUT.PUT_LINE('Manager ID: ' || rec.manager_id);
        DBMS_OUTPUT.PUT_LINE('Department ID: ' || rec.department_id);
        DBMS_OUTPUT.PUT_LINE('----------------------------');
    END LOOP;
    IF SQL%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No employees found with salary greater than ' ||
p_min_salary);
    END IF;
END;
/
```

Running the Block:

```
 BEGIN
    get_employees_by_salary(65000);
END;
/
```

12) Write PL/SQL code block to increment the employee's salary by 1000 whose employee_id is 102 from the given table below.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL_ID	PHONE_NUMBER	JOIN_DATE	JOB_ID	SALARY
100	ABC	DEF	abef	9876543210	2020-06-06	AD_PRES	24000.00
101	GHI	JKL	ghkl	9876543211	2021-02-08	AD_VP	17000.00
102	MNO	PQR	mnqr	9876543212	2016-05-14	AD_VP	17000.00
103	STU	VWX	stwx	9876543213	2019-06-24	IT_PROG	9000.00

```
BEGIN
   -- Increment the salary of the employee with employee_id 102 by 1000
   UPDATE employee
   SET salary = salary + 1000
   WHERE employee_id = 102;

   -- Display a message indicating the salary has been updated
   DBMS_OUTPUT.PUT_LINE('Salary of employee with ID 102 has been incremented
by 1000.');
END;
/
```