Name: Vignesh RG

College: Sri Krishna College of Technology

Answers

1.
```
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
) AS
BEGIN
INSERT INTO EMPLOYEES (EMP_ID, EMP_NAME, DEPARTMENT, SALARY)
VALUES (p_emp_id, p_emp_name, p_department, p_salary);
END;
```

2.
```
CREATE OR REPLACE PROCEDURE update_salary (
p_emp_id IN EMPLOYEES.EMP_ID%TYPE
) AS
v_salary EMPLOYEES.SALARY%TYPE;
BEGIN
SELECT SALARY INTO v_salary FROM EMPLOYEES WHERE EMP_ID =
p_emp_id;
IF v_salary < 5000 THEN
v_salary := v_salary * 1.10;
ELSIF v_salary BETWEEN 5000 AND 10000 THEN
v_salary := v_salary * 1.075; ELSE
v_salary := v_salary * 1.05;
END IF;
UPDATE EMPLOYEES SET SALARY = v_salary WHERE EMP_ID = p_emp_id;
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 * FROM EMPLOYEES WHERE SALARY > 10000;
5. CREATE OR REPLACE FUNCTION calculate_bonus (
p_salary IN EMPLOYEES.SALARY%TYPE
) 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;
```

```
6. CREATE OR REPLACE TRIGGER log_employee_insert
AFTER INSERT ON EMPLOYEES
FOR EACH ROW
BEGIN
INSERT INTO EMPLOYEE_LOG (LOG_DATE, EMP_ID, EMP_NAME, ACTION)
VALUES (SYSDATE, :NEW.EMP_ID, :NEW.EMP_NAME, 'INSERT');
END;
```

```
7. A) CREATE VIEW customer_sales_revenue AS
SELECT
c.customer_id,
c.customer_name,
SUM(oi.quantity * oi.unit_price) AS total_sales,
SUM(oi.quantity * oi.unit_price) * 0.05 AS credit
FROM customers c
JOIN
orders o ON c.customer_id = o.customer_id
JOIN
order_items oi ON o.order_id = oi.order_id
GROUP BY
c.customer_id, c.customer_name;
B) DECLARE
CURSOR customer_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 NUMBER;
v_budget NUMBER := 1000000;
BEGIN
UPDATE customers SET credit_limit = 0;
OPEN customer_cursor;
LOOP
FETCH customer_cursor INTO v_customer_id, v_total_sales;
EXIT WHEN customer_cursor%NOTFOUND;
```

```
IF v_budget > 0 THEN
UPDATE customers
SET credit_limit = LEAST(v_total_sales * 0.05, v_budget)
WHERE customer_id = v_customer_id;
v_budget := v_budget - LEAST(v_total_sales * 0.05, v_budget);
END IF; END LOOP;
CLOSE customer_cursor;
END;
```

```
8) DECLARE
v_employee_id employees.employee_id%TYPE;
v_first_name employees.first_name%TYPE;
BEGIN
FOR emp IN (SELECT employee_id, first_name FROM employees) LOOP
v_employee_id := emp.employee_id;
v_first_name := emp.first_name;
DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id || ', First Name: ' ||
v_first_name);
END LOOP;
END;
```

```
9) CREATE OR REPLACE PROCEDURE display_employees_below_salary (
p_salary IN EMPLOYEES.SALARY%TYPE
) AS
CURSOR emp_cursor IS
SELECT first_name, last_name, salary FROM employees WHERE salary < p_salary;
v_first_name employees.first_name%TYPE;
v_last_name employees.last_name%TYPE;
v_salary employees.salary%TYPE;
BEGIN
OPEN emp_cursor;
LOOP
FETCH emp_cursor INTO v_first_name, v_last_name, v_salary; EXIT WHEN
emp_cursor%NOTFOUND;
```

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

```
10) CREATE OR REPLACE TRIGGER check_duplicate_employee_email
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
DECLARE
v_count NUMBER;
BEGIN
SELECT COUNT(*) INTO v_count FROM employees WHERE email
= :NEW.email;
IF v_count > 0 THEN
RAISE_APPLICATION_ERROR(-20001, 'Duplicate email found: ' || :NEW.email);
END IF;
END;
```

```
11) CREATE OR REPLACE PROCEDURE select_employees_by_salary (
p_min_salary IN employees.salary%TYPE
) AS
BEGIN
FOR emp IN (SELECT * FROM employees WHERE salary >= p_min_salary) LOOP
DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id || ', Name: ' ||
emp.first_name
|| ' ' || emp.last_name || ', Salary: ' || emp.salary);
END LOOP;
END;
```

12) BEGIN

```
UPDATE employees
SET salary = salary + 1000
WHERE employee_id = 102;
END;
```