

NAME : KAVI BALAJI E

College : BIT

PL/ANS: Assignments

Question 1: Create a Procedure to Insert Employee Data

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

Question 2: Create a Procedure to Update Employee Salary

```
CREATE OR REPLACE PROCEDURE update_salary (  
    p_emp_id NUMBER  
) 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;

/
```

Question 3: Use a Cursor to Display Employee Names

```
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;

/
```

Question 4: Create a View for Employees with High Salary

Ans:

```
CREATE OR REPLACE VIEW high_salary_employees AS

SELECT *

FROM EMPLOYEES

WHERE SALARY > 10000;
```

Question 5: Create a Function to Calculate Bonus

Ans:

```
CREATE OR REPLACE FUNCTION calculate_bonus (  
    p_salary 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;  
/
```

Question 6: Create a Trigger to Log Employee Insertions

Ans:

```
CREATE OR REPLACE TRIGGER log_employee_insert  
AFTER INSERT ON EMPLOYEES  
FOR EACH ROW  
BEGIN  
    INSERT INTO EMPLOYEE_LOG (LOG_ID, EMP_ID, LOG_DATE)  
    VALUES (LOG_SEQ.NEXTVAL, :NEW.EMP_ID, SYSDATE);  
END;  
/
```

Question 7: Orders and Order_Items Tables

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

Ans:

```
CREATE OR REPLACE VIEW sales_revenues_by_customers 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) Write the PL/ANS: query to develop an anonymous block

Ans:

```
DECLARE
```

```
    v_budget NUMBER := 1000000;
```

```
    CURSOR cust_cursor IS
```

```
        SELECT customer_id FROM sales_revenues_by_customers ORDER BY total_sales DESC;
```

```
    v_customer_id sales_revenues_by_customers.customer_id%TYPE;
```

```
BEGIN
```

```
    -- Reset credit limits
```

```
    UPDATE customers SET credit_limit = 0;
```

```
    OPEN cust_cursor;
```

```
    LOOP
```

```
        FETCH cust_cursor INTO v_customer_id;
```

```
        EXIT WHEN cust_cursor%NOTFOUND;
```

```
        -- Update new credit limit
```

```
        UPDATE customers
```

```

        SET credit_limit = credit_limit + (v_budget / (SELECT COUNT(*) FROM
sales_revenues_by_customers))

        WHERE customer_id = v_customer_id;

        v_budget := v_budget - (v_budget / (SELECT COUNT(*) FROM sales_revenues_by_customers));

    END LOOP;

    CLOSE cust_cursor;

END;

/

```

Question 8: Show the Uses of Implicit Cursor

Ans:

```

DECLARE

    v_count INTEGER;

BEGIN

    SELECT COUNT(*) INTO v_count FROM employees;

    DBMS_OUTPUT.PUT_LINE('Total number of employees: ' || v_count);

END;

/

```

Question 9: Create a Cursor to Display Name and Salary

Ans:

```

DECLARE

    CURSOR emp_cursor (p_salary NUMBER) 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(10000);

    LOOP

        FETCH emp_cursor INTO v_first_name, v_last_name, v_salary;
    
```

```

        EXIT WHEN emp_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(v_first_name || ' ' || v_last_name || ': ' || v_salary);
    END LOOP;

    CLOSE emp_cursor;

END;

/

```

Question 10: Create a Trigger to Check for Duplicate Values

Ans:

```

CREATE OR REPLACE TRIGGER check_duplicate_emp_id
BEFORE INSERT OR UPDATE ON employees
FOR EACH ROW
DECLARE
    v_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM employees
    WHERE employee_id = :NEW.employee_id;

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Duplicate employee_id found.');
```

```

    END IF;
END;

/

```

Question 11: Procedure for Selecting Records with Filters

Ans:

```

CREATE OR REPLACE PROCEDURE select_employees_by_salary (
    p_salary NUMBER
) AS

```

BEGIN

FOR emp IN (SELECT * FROM ib_employee WHERE salary = p_salary) LOOP

DBMS_OUTPUT.PUT_LINE(emp.first_name || ' ' || emp.last_name || ': ' || emp.salary);

END LOOP;

END;

/

Question 12: Increment Employee's Salary

Ans:

BEGIN

UPDATE EMPLOYEES

SET SALARY = SALARY + 1000

WHERE EMPLOYEE_ID = 102;

END;