1. PL/SQL Block to Calculate Grade of Minimum 10 Students

sql

Copy

Edit

```
BEGIN
  FOR i IN 1..10 LOOP
    DECLARE
      marks NUMBER := DBMS_RANDOM.VALUE(0, 100);
      grade CHAR(1);
    BEGIN
      IF marks >= 90 THEN
        grade := 'A';
      ELSIF marks >= 75 THEN
        grade := 'B';
      ELSIF marks >= 60 THEN
        grade := 'C';
      ELSIF marks >= 40 THEN
        grade := 'D';
      ELSE
        grade := 'F';
      END IF;
      DBMS_OUTPUT.PUT_LINE('Student ' || i || ': Marks=' || ROUND(marks) || ', Grade=' || grade);
    END;
  END LOOP;
END;
/
```

----

2. Procedure to Add Employee and Return Count in Department

sql

Copy

Edit

```
CREATE OR REPLACE PROCEDURE AddEmployee(
  p_empno NUMBER,
  p_ename VARCHAR2,
  p_sal NUMBER,
  p_deptno NUMBER,
  emp_count OUT NUMBER
)
IS
BEGIN
  INSERT INTO emp(empno, ename, sal, deptno)
  VALUES (p_empno, p_ename, p_sal, p_deptno);

  SELECT COUNT(*) INTO emp_count
  FROM emp
  WHERE deptno = p_deptno;
END;
/
```

----

3. Function to Return Salary Status
sql
Copy
Edit
```
CREATE OR REPLACE FUNCTION GetSalaryStatus(p_empno NUMBER)
RETURN VARCHAR2
IS
  v_sal NUMBER;
BEGIN
  SELECT sal INTO v_sal FROM emp WHERE empno = p_empno;

  IF v_sal < 30000 THEN
    RETURN 'Low';
  ELSIF v_sal BETWEEN 30000 AND 60000 THEN
    RETURN 'Medium';
  ELSE
    RETURN 'High';
  END IF;
END;
/
```
----
4. Function to Return Number of Employees in Department
sql
Copy
Edit
```
CREATE OR REPLACE FUNCTION GetEmpCount(p_deptno NUMBER)
RETURN NUMBER
IS
  v_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO v_count
  FROM emp
  WHERE deptno = p_deptno;
  RETURN v_count;
END;
/
```
----
5. Function for Customer Level
sql
Copy
Edit
```
CREATE OR REPLACE FUNCTION CustomerLevel(p_total_purchase NUMBER)
RETURN VARCHAR2
IS
BEGIN
  IF p_total_purchase > 100000 THEN
    RETURN 'Platinum';
  ELSIF p_total_purchase >= 50000 THEN
    RETURN 'Gold';
  ELSE
    RETURN 'Silver';
  END IF;
END;
/
```

----

6. Function for Even/Odd and Largest of Three

(a) Even or Odd

sql

Copy

Edit

```
CREATE OR REPLACE FUNCTION IsEven(p_number NUMBER)
RETURN VARCHAR2
IS
BEGIN
  IF MOD(p_number, 2) = 0 THEN
    RETURN 'Even';
  ELSE
    RETURN 'Odd';
  END IF;
END;
/
```

(b) Largest of Three Numbers

sql

Copy

Edit

```
CREATE OR REPLACE FUNCTION LargestOfThree(a NUMBER, b NUMBER, c NUMBER)
RETURN NUMBER
IS
BEGIN
  RETURN GREATEST(a, b, c);
END;
/
```

----

7. Cursor Without and With Handler

Without Handler

sql

Copy

Edit

```
DECLARE
  CURSOR c_emp IS SELECT ename FROM emp;
  v_name emp.ename%TYPE;
BEGIN
  OPEN c_emp;
  LOOP
```

```
    FETCH c_emp INTO v_name;
    EXIT WHEN c_emp%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_name);
  END LOOP;
  CLOSE c_emp;
END;
/
```
With Exception Handler

sql

Copy

Edit

```
DECLARE
  CURSOR c_emp IS SELECT ename FROM emp;
  v_name emp.ename%TYPE;
BEGIN
  OPEN c_emp;
  LOOP
    FETCH c_emp INTO v_name;
    EXIT WHEN c_emp%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_name);
  END LOOP;
  CLOSE c_emp;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
/
```
----

8. Procedure for Arithmetic Operations

sql

Copy

Edit

```
CREATE OR REPLACE PROCEDURE DoArithmetic(a NUMBER, b NUMBER)
IS
BEGIN
  DBMS_OUTPUT.PUT_LINE('Addition: ' || (a + b));
  DBMS_OUTPUT.PUT_LINE('Subtraction: ' || (a - b));
  DBMS_OUTPUT.PUT_LINE('Multiplication: ' || (a * b));
  DBMS_OUTPUT.PUT_LINE('Division: ' || (a / b));
END;
/
```
----

9. Trigger Before Insert / After Delete

Before Insert

sql

Copy

Edit

```
CREATE OR REPLACE TRIGGER BeforeInsertEmp
BEFORE INSERT ON emp
```

```
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Inserting new employee: ' || :NEW.ename);
END;
/
```

After Delete

sql

Copy

Edit

```
CREATE OR REPLACE TRIGGER AfterDeleteEmp
AFTER DELETE ON emp
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('Deleted employee: ' || :OLD.ename);
END;
/
```