Name: Hamayoon jan
F/Name: Mohammad jan
Semester: 5th-semester
Year: 3rd
Department: Software Engineering
Faculty: Computer science
subject: Database

1. Annual Salary

```
DECLARE
 basic_salary NUMBER := 30000;
 bonus NUMBER := 5000;
 annual_salary NUMBER;
BEGIN
 annual_salary := basic_salary + bonus;
 DBMS_OUTPUT.PUT_LINE('Annual Salary is: ' || annual_salary);
END;
/
```

2. Average Marks

```
DECLARE
 m1 NUMBER := 80;
 m2 NUMBER := 70;
 m3 NUMBER := 90;
 avg_marks NUMBER;
BEGIN
 avg_marks := (m1 + m2 + m3) / 3;
 DBMS_OUTPUT.PUT_LINE('Average Marks: ' || avg_marks);
END;
/
```

3. Bank Balance

```
DECLARE
 balance NUMBER := 3500;
BEGIN
 IF balance < 1000 THEN
 DBMS_OUTPUT.PUT_LINE('Low Balance');
```

```
 ELSIF balance BETWEEN 1000 AND 5000 THEN
 DBMS_OUTPUT.PUT_LINE('Sufficient Balance');
 ELSE
 DBMS_OUTPUT.PUT_LINE('High Balance');
 END IF;
END;
/
```

4. Grading System

```
DECLARE
 percentage NUMBER := 76;
BEGIN
 CASE
     WHEN percentage BETWEEN 90 AND 100 THEN
       DBMS_OUTPUT.PUT_LINE('A Grade');
     WHEN percentage BETWEEN 75 AND 89 THEN
       DBMS_OUTPUT.PUT_LINE('B Grade');
     WHEN percentage BETWEEN 50 AND 74 THEN
       DBMS_OUTPUT.PUT_LINE('C Grade');
     ELSE
       DBMS_OUTPUT.PUT_LINE('Fail');
  END CASE;
END;
/
```

5. Shopping Discount

```
DECLARE
  bill NUMBER := 2500;
  final_bill NUMBER;
BEGIN
  IF bill > 5000 THEN
    final_bill := bill - (bill * 0.20);
  ELSIF bill >= 2000 AND bill <= 5000 THEN
    final_bill := bill - (bill * 0.10);
  ELSE
    final_bill := bill;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Final Bill: ' || final_bill);
END;
/
```

6. Multiplication Table

```
DECLARE
  n NUMBER := 5;
BEGIN
  FOR i IN 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE(n || ' x ' || i || ' = ' || (n*i));
  END LOOP;
END;
/
```

7. Employee IDs

```
BEGIN
  FOR id IN 100..120 LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || id);
  END LOOP;
END;
/
```

8. Factorial

```
DECLARE
  n NUMBER := 5;
  fact NUMBER := 1;
  i NUMBER := 1;
BEGIN
  WHILE i <= n LOOP
    fact := fact * i;
    i := i + 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Factorial of ' || n || ' = ' || fact);
END;
/
```

9. Countdown

```
BEGIN
  FOR i IN REVERSE 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE(i);
  END LOOP;
END;
/
```

10. Employees in IT Department

```
BEGIN
  FOR rec IN (SELECT emp_name FROM employees WHERE dept_id = 'IT') LOOP
    DBMS_OUTPUT.PUT_LINE('Employee: ' || rec.emp_name);
  END LOOP;
END;
/
```

11. Salary Increase

```
BEGIN
  FOR rec IN (SELECT emp_id FROM employees WHERE salary < 3000) LOOP
    UPDATE employees
    SET salary = salary + (salary*0.10)
    WHERE emp_id = rec.emp_id;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Salaries updated successfully');
END;
/
```

12. Above Average Salary

```
DECLARE
  avg_sal NUMBER;
BEGIN
  SELECT AVG(salary) INTO avg_sal FROM employees;

  FOR rec IN (SELECT emp_name, salary FROM employees WHERE salary > avg_sal) LOOP
    DBMS_OUTPUT.PUT_LINE(rec.emp_name || ' → ' || rec.salary);
  END LOOP;
END;
/
```

13. Salary Category

```
DECLARE
  sal NUMBER := 5000;
BEGIN
  IF sal > 8000 THEN
    DBMS_OUTPUT.PUT_LINE('High Earner');
  ELSIF sal BETWEEN 4000 AND 8000 THEN
    DBMS_OUTPUT.PUT_LINE('Mid Earner');
```

```
  ELSE
    DBMS_OUTPUT.PUT_LINE('Low Earner');
  END IF;
END;
/
```

14. Salary by Department

```
BEGIN
  FOR rec IN (SELECT dept_id, SUM(salary) AS total FROM employees GROUP BY dept_id) LOOP
    DBMS_OUTPUT.PUT_LINE('Dept: ' || rec.dept_id || ' → ' || rec.total);
  END LOOP;
END;
/
```

15. Fibonacci Series

```
DECLARE
  n NUMBER := 10;
  a NUMBER := 0;
  b NUMBER := 1;
  c NUMBER;
BEGIN
  DBMS_OUTPUT.PUT_LINE(a);
  DBMS_OUTPUT.PUT_LINE(b);

  FOR i IN 3..n LOOP
    c := a + b;
    DBMS_OUTPUT.PUT_LINE(c);
    a := b;
    b := c;
  END LOOP;
END;
/
```

16. Bank Transactions

```
DECLARE
  balance NUMBER := 0;
BEGIN
  FOR rec IN (SELECT amount, type FROM transactions) LOOP
    IF rec.type = 'CREDIT' THEN
      balance := balance + rec.amount;
    ELSE
```

```
        balance := balance - rec.amount;
      END IF;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Final Balance = ' || balance);
END;
/
```

17. Employee Details Procedure

```
CREATE OR REPLACE PROCEDURE get_employee(p_id NUMBER) IS
   v_name employees.emp_name%TYPE;
   v_dept employees.dept_id%TYPE;
   v_salary employees.salary%TYPE;
BEGIN
   SELECT emp_name, dept_id, salary
   INTO v_name, v_dept, v_salary
   FROM employees
   WHERE emp_id = p_id;

   DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
   DBMS_OUTPUT.PUT_LINE('Dept: ' || v_dept);
   DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
END;
/
```