Parent & Child & Ancestor:

```
create table parof(parent varchar(30),name varchar(30));

create table ancestor (parent varchar(30),name varchar(30), parent
varchar(30), child varchar(30));

insert into parof(parent,name) values('alice','carol');
insert into parof(parent,name) values('bob','carol');
insert into parof(parent,name) values('carol','dave');
insert into parof(parent,name) values('carol','george');
insert into parof(parent,name) values('dave','mary');
insert into parof(parent,name) values('eve','mary');
insert into parof(parent,name) values('mary','frank');
```

PARENT	NAME
alice	carol
bob	carol
carol	dave
carol	george
dave	mary
eve	mary
mary	frank

7 rows selected.

Finding parent of a child query:

```
WITH ancestor(parent) AS (SELECT parent FROM parof WHERE name =
'frank' UNION ALL SELECT parof.parent FROM ancestor, parof WHERE
ancestor.parent = parof.name)

SELECT * FROM ancestor;
PARENT
------------------------------
mary
dave
eve
carol
alice
bob

6 rows selected.
```

Query with Countup (dual is oracle based dummy table)

```
WITH countup (n) AS ( SELECT 1 FROM dual UNION ALL  SELECT n + 1 FROM
countup WHERE  n + 1 < 3)

SELECT n FROM   countup;
```

****

PL/SQL Programs

1. Leap Year

Program:

```
DECLARE
   yr NUMBER := &year;
BEGIN
   IF MOD(yr,400)=0 OR (MOD(yr,4)=0 AND MOD(yr,100)<>0) THEN
```

```
      DBMS_OUTPUT.PUT_LINE(yr || ' is a Leap Year');
   ELSE
      DBMS_OUTPUT.PUT_LINE(yr || ' is NOT a Leap Year');
   END IF;
END;
/
```

Input: 2000

Output:

```
2000 is a Leap Year
```

---

2. Arithmetic, Relational, Logical Operators

Program:

```
DECLARE
   a NUMBER := 20;
   b NUMBER := 10;
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));

   IF a > b THEN
      DBMS_OUTPUT.PUT_LINE('a is greater than b');
   END IF;

   IF (a > 10 AND b < 15) THEN
      DBMS_OUTPUT.PUT_LINE('Logical AND condition is TRUE');
   END IF;
END;
/
```

Output:

```
Addition: 30
Subtraction: 10
Multiplication: 200
Division: 2
a is greater than b
Logical AND condition is TRUE
```

---

3. Character, Number, Special Character

Program:

```
DECLARE
   ch CHAR(1) := '&input_char';
BEGIN
   IF ch BETWEEN '0' AND '9' THEN
      DBMS_OUTPUT.PUT_LINE(ch || ' is a Number');
   ELSIF ch BETWEEN 'A' AND 'Z' OR ch BETWEEN 'a' AND 'z' THEN
      DBMS_OUTPUT.PUT_LINE(ch || ' is a Character');
   ELSE
      DBMS_OUTPUT.PUT_LINE(ch || ' is a Special Character');
   END IF;
END;
/
```

Input: A

Output:

```
A is a Character
```

---

4. Largest of 2 Numbers

Program:

```
DECLARE
   a NUMBER := &a;
   b NUMBER := &b;
BEGIN
   IF a > b THEN
      DBMS_OUTPUT.PUT_LINE(a || ' is Larger');
   ELSE
      DBMS_OUTPUT.PUT_LINE(b || ' is Larger');
   END IF;
END;
/
```

Input: a=25, b=40

Output:

```
40 is Larger
```

---

5. Largest of 3 Numbers

Program:

```
DECLARE
   a NUMBER := &a;
   b NUMBER := &b;
   c NUMBER := &c;
BEGIN
   IF a > b AND a > c THEN
      DBMS_OUTPUT.PUT_LINE(a || ' is Largest');
   ELSIF b > c THEN
      DBMS_OUTPUT.PUT_LINE(b || ' is Largest');
   ELSE
      DBMS_OUTPUT.PUT_LINE(c || ' is Largest');
   END IF;
END;
/
```

Input: a=12, b=45, c=32

Output:

```
45 is Largest
```

---

6. Odd or Even

Program:

```
DECLARE
   n NUMBER := &num;
BEGIN
   IF MOD(n,2)=0 THEN
      DBMS_OUTPUT.PUT_LINE(n || ' is Even');
   ELSE
      DBMS_OUTPUT.PUT_LINE(n || ' is Odd');
   END IF;
END;
/
```

Input: 7

Output:

```
7 is Odd
```

---

7. Grade using Nested IF

Program:

```
DECLARE
   marks NUMBER := &marks;
BEGIN
   IF marks >= 90 THEN
      DBMS_OUTPUT.PUT_LINE('Grade A');
   ELSIF marks >= 75 THEN
      DBMS_OUTPUT.PUT_LINE('Grade B');
   ELSIF marks >= 50 THEN
      DBMS_OUTPUT.PUT_LINE('Grade C');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Fail');
   END IF;
END;
/
```

Input: 82

Output:

```
Grade B
```

---

8. Voting Eligibility

Program:

```
DECLARE
   age NUMBER := &age;
BEGIN
   IF age >= 18 THEN
      DBMS_OUTPUT.PUT_LINE('Eligible to Vote');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Not Eligible to Vote');
   END IF;
END;
/
```

Input: 16

Output:

```
Not Eligible to Vote
```

---

9. Sum of N Numbers (FOR Loop)

Program:

```
DECLARE
   n NUMBER := &n;
   sum NUMBER := 0;
BEGIN
   FOR i IN 1..n LOOP
      sum := sum + i;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Sum = ' || sum);
END;
/
```

Input: 5

Output:

```
Sum = 15
```

---

10. Multiplication Table

Program:

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

Input: n=5

Output:

```
5 x 1 = 5
5 x 2 = 10
...
5 x 10 = 50
```

---

11. Sum of Even, Odd, Prime Numbers

Program:

```
DECLARE
   n NUMBER := &n;
   sum_even NUMBER := 0;
   sum_odd NUMBER := 0;
   sum_prime NUMBER := 0;
   flag NUMBER;
BEGIN
   FOR i IN 1..n LOOP
      IF MOD(i,2)=0 THEN
         sum_even := sum_even + i;
      ELSE
         sum_odd := sum_odd + i;
      END IF;

      -- Prime check
      flag := 0;
      IF i > 1 THEN
         FOR j IN 2..i-1 LOOP
            IF MOD(i,j)=0 THEN
               flag := 1;
            END IF;
         END LOOP;
         IF flag=0 THEN
            sum_prime := sum_prime + i;
         END IF;
      END IF;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Sum Even = ' || sum_even);
   DBMS_OUTPUT.PUT_LINE('Sum Odd = ' || sum_odd);
   DBMS_OUTPUT.PUT_LINE('Sum Prime = ' || sum_prime);
END;
/
```

Input: 10

Output:

```
Sum Even = 30
Sum Odd = 25
Sum Prime = 17
```

---

12. Fibonacci Series

Program:

```
DECLARE
   n NUMBER := &n;
   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;
/
```

Input: 7

Output:

```
0
1
1
2
3
5
8
```

---

13. Factorial

Program:

```
DECLARE
   n NUMBER := &n;
   fact NUMBER := 1;
BEGIN
   FOR i IN 1..n LOOP
      fact := fact * i;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Factorial = ' || fact);
END;
/
```

Input: 5

Output:

```
Factorial = 120
```

---

14. Reverse a Number

Program:

```
DECLARE
   n NUMBER := &n;
   rev NUMBER := 0;
   rem NUMBER;
BEGIN
   WHILE n > 0 LOOP
      rem := MOD(n,10);
      rev := rev*10 + rem;
      n := TRUNC(n/10);
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Reverse = ' || rev);
END;
/
```

Input: 1234

Output:

```
Reverse = 4321
```
