1. B. NOT VALID AS MULTIPLE VARIABLES DECLARED IN SAME LINE.
   D. NOT VALID AS BOOLEAN CAN ONLY TAKE TRUE, FALSE AND NULL.

2. A. VALID, DATE
   B. VALID, VARCHAR
   C. VALID, CURRENCY
   D. VALID, BOOLEAN
   E. VALID, BOOLEAN
   F. VALID, BOOLEAN

3.
```
DECLARE
MESSAGE VARCHAR(50):= 'MY PL/SQL BLOCK WORKS';

BEGIN
   DBMS_OUTPUT.PUT_LINE(MESSAGE);
END;
```

4.
```
 DECLARE
V_CHAR VARCHAR(50) ;
V_NUM VARCHAR(50);

BEGIN
V_CHAR := '42 IS THE ANSWER';
V_NUM := SUBSTR(V_CHAR,1,2);
DBMS_OUTPUT.PUT_LINE(V_CHAR);
DBMS_OUTPUT.PUT_LINE(V_NUM);

END;
```

5.

a. 2, NUMBER
b. WESTERN EUROPE, VARCHAR
c. 601, NUMBER
d. PRODUCT 10012, VARCHAR
e. WESTERN, VARCHAR

6.

a. 201, NUMBER
b. UNISPORTS, VARCHAR
c. EXCELLENT, VARCHAR
d. WOMANSPORT, VARCHAR
e. NULL
f. EXCELLENT, VARCHAR

7.

```
DECLARE
P_NUM1 NUMBER(3):=2;
P_NUM2 NUMBER(3):=4;
RES NUMBER(4);
BEGIN
RES:=P_NUM2/P_NUM1;
DBMS_OUTPUT.PUT_LINE(RES);
END;
```

8.

```
DECLARE
V_SAL NUMBER(7):=50000;
V_ANSAL NUMBER(7);
V_BONUS NUMBER(7):=10;
RES NUMBER(8);
```

```
BEGIN
IF V_SAL IS NULL
THEN
V_BONUS:=0;
END IF;
V_ANSAL:=12*V_SAL;

RES := V_ANSAL+(V_ANSAL*(V_BONUS/100));
DBMS_OUTPUT.PUT_LINE(RES);
END;
```

9.

```
DECLARE
V_MAX DEPT.DEPTNO%TYPE;
BEGIN
SELECT MAX(DEPTNO)
INTO V_MAX
FROM DEPT;
DBMS_OUTPUT.PUT_LINE(V_MAX);
END;
```

10.

```
DECLARE
V_DNAME DEPT.DNAME%TYPE:= 'EDUCATION';
V_DEPTNO DEPT.DEPTNO%TYPE:=60;
BEGIN

INSERT INTO DEPT (DNAME,DEPTNO) VALUES
(V_DNAME,V_DEPTNO);

DBMS_OUTPUT.PUT_LINE(V_DEPTNO);
DBMS_OUTPUT.PUT_LINE(V_DNAME);
```

```
END;
```

11.

```
DEFINE P_DEPTNO=280
DEFINE P_LOC=1700
DECLARE
VAR1 NUMBER:=&P_DEPTNO ;
VAR2 NUMBER:=&P_LOC ;
DEPTNO NUMBER;
LOC NUMBER;
BEGIN
UPDATE DEPT SET LOC=VAR2
WHERE DEPTNO=VAR1;
DBMS_OUTPUT.PUT_LINE(DEPTNO || LOC);
END;
```

12.

```
DEFINE P_DEPTNO=280
DEFINE P_LOC=1700
DECLARE
VAR1 NUMBER:=&P_DEPTNO ;
VAR2 NUMBER:=&P_LOC ;
DEPTNO NUMBER;
LOC NUMBER;
BEGIN

DELETE FROM DEPT
WHERE DEPTNO=VAR1;
END;
```

13.

```
BEGIN
FOR I IN 1..10 LOOP
```

```
IF I NOT IN (6,8) THEN
DBMS_OUTPUT.PUT_LINE(I);
END IF;
END LOOP;
END;
```

14.

```
DEFINE P_EMPNO=7839
DECLARE
VAR1 NUMBER:=&P_EMPNO ;
V_EMPNO EMP.EMPNO%TYPE;
BEGIN
SELECT EMP.SAL
INTO V_EMPNO
FROM EMP
WHERE EMPNO= VAR1;
IF EMP.SAL<5000
THEN DBMS_OUTPUT.PUT_LINE(EMP.SAL*0.10);
ELSE IF EMP.SAL>5000 AND EMP.SAL<10000
THEN DBMS_OUTPUT.PUT_LINE(EMP.SAL*0.15);
ELSE IF EMP.SAL>10000
THEN DBMS_OUTPUT.PUT_LINE(EMP.SAL*0.20);
ELSE IF EMP.SAL IS NULL
THEN  DBMS_OUTPUT.PUT_LINE('0');
END IF;
END;
```

15.

```
CREATE TABLE employees AS SELECT * FROM emp;
ALTER TABLE employees ADD stars VARCHAR2(50);
```

16.

```
DEFINE P_EMPNO= 104;
```

```
DECLARE
V_SAL EMP.SAL%TYPE;
V_ASTERISK EMP.STARS%TYPE: = NULL;
BEGIN
SELECT EMP.SAL INTO V_SAL
FROM EMP WHERE EMP.EMPNO = &P_EMPNO;

FOR counter IN 1..TRUNC(V_SAL / 1000) LOOP
V_ASTERISK := V_ASTERISK || '*';
END LOOP;

UPDATE EMP SET STARS = V_ASTERISK
WHERE EMPNO = &P_EMPNO;
END;
/
SELECT EMPNO, SAL, STARS
FROM EMP
WHERE EMPNO IN (104, 174, 176);
```

17.

```
SET SERVEROUTPUT ON;
DECLARE
V_SAL NUMBER(8,2);

CURSOR EMPLOYEE
IS
SELECT SAL
INTO V_SAL
FROM EMP;

BEGIN
OPEN EMPLOYEE;
LOOP
FETCH EMPLOYEE INTO V_SAL;
INSERT INTO TOP_DOGS VALUES V_SAL;
EXIT WHEN EMPLOYEE%NOTFOUND;
END LOOP;
CLOSE EMPLOYEE;
END;
```

18. A.

```
DECLARE
V_NO NUMBER(5);
V_SAL EMP.SAL%TYPE;
COUNTER NUMBER(4):=0;
CURSOR TOP_EMPLOYEE
IS
SELECT SAL
FROM EMP
ORDER BY SAL DESC;

BEGIN
OPEN TOP_EMPLOYEE;
WHILE(COUNTER<&V_NO) LOOP
FETCH TOP_EMPLOYEE INTO V_SAL;
DBMS_OUTPUT.PUT_LINE(V_SAL);
COUNTER:=COUNTER+1;
END LOOP;
CLOSE TOP_EMPLOYEE;
END;
```

B.

```
DECLARE
V_NO NUMBER(5);
V_SAL EMP.SAL%TYPE;
COUNTER NUMBER(4):=0;
CURSOR TOP_EMPLOYEE
IS
```

```
SELECT DISTINCT(SAL)
FROM EMP
ORDER BY SAL DESC;

BEGIN
OPEN TOP_EMPLOYEE;
WHILE(COUNTER<&V_NO) LOOP
FETCH TOP_EMPLOYEE INTO V_SAL;
DBMS_OUTPUT.PUT_LINE(V_SAL);
COUNTER:=COUNTER+1;
END LOOP;
CLOSE TOP_EMPLOYEE;
END;
```

C.

```
DECLARE
V_NO NUMBER(5);
V_SAL EMP.SAL%TYPE;
COUNTER NUMBER(4):=0;
CURSOR TOP_EMPLOYEE
IS
SELECT DISTINCT(SAL)
FROM EMP
ORDER BY SAL DESC;

BEGIN
OPEN TOP_EMPLOYEE;
WHILE(COUNTER<&V_NO) LOOP
FETCH TOP_EMPLOYEE INTO V_SAL;
INSERT INTO TOP_DOGS VALUES(V_SAL);
DBMS_OUTPUT.PUT_LINE(V_SAL);
```

```
COUNTER:=COUNTER+1;
END LOOP;
CLOSE TOP_EMPLOYEE;
END;
```

19.

```
DECLARE
V_DEPTNO NUMBER(5);
V_ENAME EMP.ENAME%TYPE;
V_MGR EMP.MGR%TYPE;
V_SAL EMP.SAL%TYPE;

CURSOR EMPLOYEE
IS
SELECT ENAME,SAL,MGR
FROM EMP
WHERE DEPTNO=&V_DEPTNO;

BEGIN
OPEN EMPLOYEE;
LOOP
FETCH EMPLOYEE INTO V_ENAME,V_SAL,V_MGR;
DBMS_OUTPUT.PUT_LINE(V_ENAME||' '||V_SAL||' '||V_MGR);
EXIT WHEN EMPLOYEE%NOTFOUND;
END LOOP;
CLOSE EMPLOYEE;
END;
```

C.

```
DECLARE
V_DEPTNO NUMBER(5);
V_ENAME EMP.ENAME%TYPE;
```

```
V_MGR EMP.MGR%TYPE;
V_SAL EMP.SAL%TYPE;

CURSOR EMPLOYEE
IS
SELECT ENAME,SAL,MGR
FROM EMP
WHERE DEPTNO=&V_DEPTNO;

BEGIN
OPEN EMPLOYEE;
LOOP
FETCH EMPLOYEE INTO V_ENAME,V_SAL,V_MGR;
IF V_SAL<5000 AND V_MGR IN (101,124) THEN
DBMS_OUTPUT.PUT_LINE(V_ENAME||' Due for a raise.');
ELSE
DBMS_OUTPUT.PUT_LINE(V_ENAME||' Not due for a raise.');
END IF;
--DBMS_OUTPUT.PUT_LINE(V_ENAME||' '||V_SAL||' '||V_MGR);
EXIT WHEN EMPLOYEE%NOTFOUND;
END LOOP;
CLOSE EMPLOYEE;
END;
```

20.

```
DECLARE
V_ENAME EMP.ENAME%TYPE;
V_SAL EMP.SAL%TYPE;
CURSOR EMPLOYEE
IS
SELECT ENAME INTO V_ENAME
FROM EMP
```

```
WHERE SAL=&V_SAL;
BEGIN
OPEN EMPLOYEE;
LOOP
FETCH EMPLOYEE INTO V_ENAME;
EXIT WHEN EMPLOYEE%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(V_ENAME);
END LOOP;
CLOSE EMPLOYEE;
END;
```

C.

```
DECLARE
V_ENAME EMP.ENAME%TYPE;
V_SAL EMP.SAL%TYPE;

BEGIN
SELECT ENAME INTO V_ENAME
FROM EMP
WHERE SAL=&V_SAL;

DBMS_OUTPUT.PUT_LINE(V_ENAME);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No employee with a salary of '||V_SAL);
END;
```

E.

```
DECLARE
V_ENAME EMP.ENAME%TYPE;
V_SAL EMP.SAL%TYPE;
```

```
BEGIN
SELECT ENAME INTO V_ENAME
FROM EMP
WHERE SAL=&V_SAL;

DBMS_OUTPUT.PUT_LINE(V_ENAME);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No employee with a salary of '||V_SAL);
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE ('Some other exception occurred');
END;
```

21.

```
DECLARE
V_DEPTNO DEPT.DEPTNO%TYPE;
V_LOC DEPT.LOC%TYPE;

BEGIN
SELECT LOC INTO V_LOC
FROM DEPT
WHERE DEPTNO=&V_DEPTNO;

DBMS_OUTPUT.PUT_LINE(V_LOC);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No DEPARTMENT with a DEPTID of
'||V_DEPTNO);
--WHEN OTHERS THEN
--DBMS_OUTPUT.PUT_LINE ('Some other exception occurred');
```

```
END;
C.
DECLARE
V_DEPTNO DEPT.DEPTNO%TYPE;
V_LOC DEPT.LOC%TYPE;
V_REF DEPT.DEPTNO%TYPE;
BEGIN
V_REF := &V_DEPTNO;
SELECT LOC, DEPTNO INTO V_LOC, V_REF
FROM DEPT
WHERE DEPTNO=V_REF;

DBMS_OUTPUT.PUT_LINE(V_LOC);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('No DEPARTMENT with a DEPTID of
'||V_REF);
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE ('Department'||V_REF||' is an invalid department');
END;
```

22.

```
DECLARE
V1_SAL EMP.SAL%TYPE;
V_SAL NUMBER(5);
COUNTER NUMBER(5);
NO_EMP EXCEPTION;

BEGIN
V_SAL:=&V1_SAL;
```

```
SELECT COUNT(EMPNO)INTO COUNTER
FROM EMP
WHERE SAL BETWEEN (V_SAL-100) AND (V_SAL+100);
DBMS_OUTPUT.PUT_LINE(COUNTER);
IF COUNTER=0 THEN
RAISE NO_EMP;
END IF;
EXCEPTION
WHEN NO_EMP THEN
DBMS_OUTPUT.PUT_LINE('NO EMPLOYEE IN GIVEN RANGE');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('SOME ERROR OCCOURED.');

END;
```

23.

```
CREATE OR REPLACE PROCEDURE ADD_JOB
(J_ID IN JOBS.JOB_ID%TYPE,TITLE IN JOBS.JOB_TITLE%TYPE) IS
BEGIN
INSERT INTO JOBS VALUES (J_ID,TITLE,'87','13');
END;

EXECUTE ADD_JOB('IT_DBA','Database Administrator');
```

C.

```
EXECUTE ADD_JOB('ST_MAN','Stock Manager');
```

24.

```
CREATE OR REPLACE PROCEDURE upd_job
(JOBID JOBS.JOB_ID%TYPE,
TITLE JOBS.JOB_TITLE%TYPE)
```

```
IS
COUNTER JOBS.MAX_SALARY%TYPE := 0;
ID_NOT_FOUND EXCEPTION;
CURSOR c1 IS
SELECT JOB_ID
FROM JOBS;

BEGIN
FOR JOBSID IN c1 LOOP
IF JOBSID.JOB_ID = JOBID THEN
COUNTER := COUNTER + 1;
END IF;
END LOOP;

IF COUNTER = 1 THEN
UPDATE JOBS
SET
JOB_TITLE = TITLE
WHERE
JOB_ID = JOBID;
ELSE
RAISE ID_NOT_FOUND;
END IF;

EXCEPTION
WHEN ID_NOT_FOUND THEN
dbms_output.put_line('JOB ID NOT IN TABLE');
END;
```

26.

```
CREATE OR REPLACE  PROCEDURE QUERY_EMP
```

```
( EID IN EMP.EMPNO%TYPE,
ESAL OUT EMP.SAL%TYPE,
EJOB OUT EMP.JOB%TYPE)
IS
BEGIN
SELECT SAL,JOB
INTO ESAL,EJOB
FROM EMP
WHERE EMPNO=EID;
DBMS_OUTPUT.PUT_LINE(ESAL||' '||EJOB);
END QUERY_EMP;
```

B.

```
DECLARE
ID EMP.EMPNO%TYPE:=7499;
SALA EMP.SAL%TYPE;
JOBT EMP.JOB%TYPE;
BEGIN
QUERY_EMP(ID,SALA,JOBT);
END;
```

C.

```
NO DATA FOUND
```

28.

```
CREATE OR REPLACE PACKAGE BODY EMP_PACK AS

PROCEDURE NEW_EMP IS
VALE BOOLEAN;
INVALID_DEPTID EXCEPTION;
BEGIN
VALE :=VALID_DEPTID(15);
```

```
IF VALE=FALSE THEN
RAISE INVALID_DEPTID;
ELSE
DBMS_OUTPUT.PUT_LINE('true');
INSERT INTO EMP values('arpit','SA_REP',7316,sysdate,4000,145,30);
DBMS_OUTPUT.PUT_LINE('true');
END IF;
EXCEPTION
WHEN INVALID_DEPTID THEN
DBMS_OUTPUT.PUT_LINE('Invalid department id');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('SOME OTHER ERRORS');
END;


FUNCTION VALID_DEPTID(DEPT EMP.DEPTNO%TYPE)
RETURN BOOLEAN
IS
DEPTNO EMP.DEPTNO%TYPE;
BEGIN
SELECT COUNT(*)
INTO DEPTNO
FROM  EMP
WHERE DEPTNO=DEPT;
IF
DEPTNO =0 THEN
RETURN FALSE;
ELSE
RETURN TRUE;
END IF;
```

```
END;

END EMP_PACK;



CREATE OR REPLACE PACKAGE EMP_PACK AS

PROCEDURE NEW_EMP;

FUNCTION VALID_DEPTID(DEPT EMO.DEPTNO%TYPE) RETURN
BOOLEAN;

END EMP_PACK;
```

29.

```
CREATE OR REPLACE PACKAGE BODY CHK_PACK AS


PROCEDURE CHK_HIREDATE(P_DATE VARCHAR2) as

CUSTOM_NULL_ERROR EXCEPTION;

TYPE DATECHAR IS TABLE OF DATA;

COUNTVAL INTEGER(5,2);

DATECHARS DATECHAR;

RETVAL VARCHAR2(50);

BEGIN

RETVAL :=TEST_DATE(P_DATE);

IF RETVAL ='FALSE' then

RAISE CUSTOM_NULL_ERROR;

END IF;

SELECT HIRE_DATE BULK COLLECT INTO DATECHARS FROM EMP
WHERE P_DATE BETWEEN (SYSDATE-18000) AND (SYSDATE+90)
GROUP BY HIRE_DATE;

SELECT COUNT(*) INTO COUNTVAL FROM EMP WHERE P_DATE
BETWEEN (SYSDATE-90) AND (SYSDATE+90);

FOR I in 1..COUNTVAL

LOOP
```

```
DBMS_OUTPUT.PUT_LINE('Work MAadi '||COUNTVAL||'
'||DATECHARS(I));

END LOOP;

EXCEPTION

WHEN CUSTOM_NULL_ERROR THEN

DBMS_OUTPUT.PUT_LINE('Either Null data or incorrect date format has been
passed,Please supply input date in dd-mon-yy format');

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Date format is not correct');

END;


FUNCTION TEST_DATE(P_DATE VARCHAR2) RETURN VARCHAR2 IS

L_DATE DATE;

BEGIN

IF P_DATE IS NULL THEN

RETURN 'FALSE';

END IF;

L_DATE := TO_DATE(P_DATE,'dd-MON-YY');

RETURN 'TRUE';

EXCEPTION

WHEN OTHERS THEN

RETURN 'FALSE';

END;


PROCEDURE CHK_DEPT_MGR(P_EMPID EMP.EMPNO%TYPE,P_MGR
EMP.MGR%TYPE) AS

INVALID EXCEPTION;

P_deptid EMP.EMPNO%TYPE;

M_DEPTID EMP.DEPTNO%TYPE;

TITLE JOBS.JOB_TITLE%TYPE;

BEGIN
```

```
SELECT DEPARTMENT_ID INTO P_DEPTID FROM EMP WHERE
EMPNO=P_EMPID;
SELECT DEPARTMENT_ID INTO M_DEPTID FROM EMP WHERE
EMPNO=P_MGR;
IF M_DEPTID=P_DEPTID THEN
DBMS_OUTPUT.PUT_LINE('Manager ID and Employee ID belong to same
department');
SELECT JOB_ID INTO TITLE FROM EMP WHERE EMPNO=P_MGR and
JOB_ID LIKE '%MAN';
ELSE
RAISE INVALID;
END IF;
EXCEPTION
WHEN INVALID THEN
DBMS_OUTPUT.PUT_LINE('Manager ID and Employee ID doesnt belong to
same department');
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Manager ID doesnt have a manager role');
END;
END CHK_PACK;
```

30.

```
CREATE OR REPLACE PROCEDURE SECURE_DML
IS
BEGIN
IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:45' AND '17:30'
OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN')
THEN
RAISE_APPLICATION_ERROR (-20001,
'You may only make changes during normal office hours.');
END IF;
END SECURE_DML;
```

32. A. CREATE OR REPLACE TRIGGER SEC_JOB

```
BEFORE INSERT OR UPDATE OR DELETE ON JOBHIST
BEGIN
SECURE_DML;
END;
/
```

B. INSERT INTO JOBHIST(EMPNO,JOB) VALUES(152, 'MARKETING');