7. Create a table **STOCK_DETAIL** with the columns PNO, PNAME and QTY_AVL to store stock details of computer accessories. Specify Primary Key and NOT NULL constraints on the table. QTY_AVL should be positive number.

Write a PL/SQL Program to define a user defined exception named "LOW_STOCK" to validate the transaction. The program facilitates the user to purchase the product by providing product number and quantity required.

It should display an error message "NO SUFFICIENT STOCK" when the user tries to purchase a product with quantity more than QTY_AVL, Otherwise the **STOCK_DETAIL** table should be updated for valid transaction.

7.1 Creating the Table and Describing its Structure

```
CREATE TABLE STOCK_DETAIL
(
    PNO INTEGER PRIMARY KEY,
    PNAME VARCHAR(20) NOT NULL,
    QTY_AVL INTEGER NOT NULL,
    CHECK (QTY_AVL >= 0)
);
```

Output

Table created.

Description of STOCK_DETAIL table.

```
DESC STOCK_DETAIL;      or      DESCRIBE STOCK_DETAIL;

Name                    Null?           Type
PNO                     NOT NULL        NUMBER(38)
PNAME                   NOT NULL        VARCHAR(20)
QTY_AVL                 NOT NULL        NUMBER(38)
```

7.2 Loading data into the table

```
INSERT INTO STOCK_DETAIL VALUES(101, 'KEYBOARD', 1500);
INSERT INTO STOCK_DETAIL VALUES(102, 'MOUSE', 500);
INSERT INTO STOCK_DETAIL VALUES(103, 'HDD', 150);
INSERT INTO STOCK_DETAIL VALUES(104, 'SSD', 100);
INSERT INTO STOCK_DETAIL VALUES(105, 'MONITOR', 75);
INSERT INTO STOCK_DETAIL VALUES(106, 'SPEAKER', 260);
INSERT INTO STOCK_DETAIL VALUES(107, 'POINTER', 100);
INSERT INTO STOCK_DETAIL VALUES(108, 'DVD-DRIVE', 60);
INSERT INTO STOCK_DETAIL VALUES(109, 'RAM-DDR3', 250);
INSERT INTO STOCK_DETAIL VALUES(110, 'RAM-DDR4', 300);
```

Output

5 rows created.

Displaying the values inserted into the STOCK_DETAIL table.

```
SELECT *
FROM STOCK_DETAIL;
```

```
       PNO PNAME                    QTY_AVL
---------- -------------------- ----------
       101 KEYBOARD                   1500
       102 MOUSE                       500
       103 HDD                         150
       104 SSD                         100
       105 MONITOR                      75
       106 SPEAKER                     260
       107 POINTER                     100
       108 DVD-DRIVE                    60
       109 RAM-DDR3                    250
       110 RAM-DDR4                    300

10 rows selected.
```

7.3 PL/SQL Program

```
SET SERVEROUTPUT ON;
DECLARE
avl_qty STOCK_DETAIL.QTY_AVL%type;
pnum STOCK_DETAIL.PNO%type:=&pnum;
qty number:=&qty;
LOW_STOCK EXCEPTION;

BEGIN
select QTY_AVL into avl_qty
from STOCK_DETAIL
where PNO=pnum;

if(qty<=0) then
    raise LOW_STOCK;
end if;

if(qty>avl_qty) then
dbms_output.put_line('NO SUFFICIENT STOCK..!'||chr(10)||'Available Quantity = '||avl_qty);
else
    avl_qty:=avl_qty-qty;
    UPDATE STOCK_DETAIL SET QTY_AVL=avl_qty WHERE PNO=pnum;
end if;

EXCEPTION
when LOW_STOCK then
dbms_output.put_line('Invalid Transaction..!');
when NO_DATA_FOUND then
dbms_output.put_line('Invalid product number..!');

END;
/
```

Output

Case 1: When the quantity required is more than the quantity available.

```
SQL> @"STOCK.SQL"

Enter value for pnum: 101
old   3: pnum STOCK_DETAIL.PNO%type:=&pnum;
new   3: pnum STOCK_DETAIL.PNO%type:=101;
Enter value for qty: 2000
old   4: qty number:=&qty;
new   4: qty number:=2000;

NO SUFFICIENT STOCK..!
Available Quantity = 1500

PL/SQL procedure successfully completed.
```

Case 2: Successful transaction.

```
SQL> @"STOCK.SQL"

Enter value for pnum: 102
old   3: pnum STOCK_DETAIL.PNO%type:=&pnum;
new   3: pnum STOCK_DETAIL.PNO%type:=102;
Enter value for qty: 100
old   4: qty number:=&qty;
new   4: qty number:=100;

PL/SQL procedure successfully completed.
```

Checking whether the STOCK_DETAIL table has been updated.

```
SELECT *
FROM STOCK_DETAIL
ORDER BY PNO;
```

PNO	PNAME	QTY_AVL
101	KEYBOARD	1000
102	MOUSE	500
103	HDD	150
104	SSD	100
105	MONITOR	75
106	SPEAKER	260
107	POINTER	100
108	DVD-DRIVE	60
109	RAM-DDR3	250
110	RAM-DDR4	300

```
10 rows selected.
```

Case 3: When the quantity required is not a positive integer.

```
SQL> @"STOCK.SQL"

Enter value for pnum: 101
old   3: pnum STOCK_DETAIL.PNO%type:=&pnum;
new   3: pnum STOCK_DETAIL.PNO%type:=101;
Enter value for qty: 0
```

```
old   4: qty number:=&qty;new   4: qty number:=0;

Invalid Transaction..!

PL/SQL procedure successfully completed.
```

Case 4: When the product number entered is not present in the STOCK_DETAIL table.

```
SQL> @"STOCK.SQL"

Enter value for pnum: 111
old   3: pnum STOCK_DETAIL.PNO%type:=&pnum;
new   3: pnum STOCK_DETAIL.PNO%type:=111;
Enter value for qty: 100
old   4: qty number:=&qty;
new   4: qty number:=100;

Invalid product number..!

PL/SQL procedure successfully completed.
```

Date of Submission	Remarks
Signature of the Lecturer	