Q1. Consider table Stud(Roll, Att,Status) Write a PL/SQL block for the following requirement and handle the exceptions. Roll no. of students will be entered by the user. Attendance of roll no. entered by the user will be checked in the Stud table. If attendance is less than 75% then display the message “Term not granted” and set the status in the stud table as “D”. Otherwise display message “Term granted” and set the status in the stud table as “ND”.

And:-

```
CREATE TABLE Stud(Roll Number(3), Att NUMBER(2), Status VARCHAR(3));

INSERT INTO Stud(Roll,Att) VALUES(101,55);
INSERT INTO Stud(Roll,Att) VALUES(102,75);
INSERT INTO Stud(Roll,Att) VALUES(103,99);
INSERT INTO Stud(Roll,Att) VALUES(104,85);
INSERT INTO Stud(Roll,Att) VALUES(105,35);
```

```
ROLL       ATT        STA
---------- ---------- ---
101        55
102        75
102        75
103        99
104        85
105        35
```

```
Declare
mroll number(10);
matt number(10);
Begin
mroll:= &mroll;
select att into matt from stud where roll = mroll;
if matt<75 then
dbms_output.put_line(mroll || 'is detained');
update stud set status='D' where roll=mroll;
else
dbms_output.put_line(mroll || 'is Not detained');
update stud set status='ND' where roll=mroll;
end if;
Exception
```

```
when no_data_found then
dbms_output.put_line(mroll||'Not found');
End;
```

OUTPUT:-

```
Enter value for mroll: 105
old   5: mroll:= &mroll;
new   5: mroll:= 105;

PL/SQL procedure successfully completed.
```

Q2. Write a PL/SQL block for following requirements using user defined exception handling.
The account_master table records the current balance for an account, which is updated whenever any deposits or withdrawals take place. If the withdrawal attempted is more than the current balance held in the account. The user defined exception is raised, displaying an appropriate message. Write a PL/SQL block for the above requirement using user defined exception handling.

Ans:-

```
CREATE TABLE account_mstr(acc_no NUMBER(12), bal NUMBER(8));

INSERT INTO account_mstr VALUES(100001,10000);
INSERT INTO account_mstr VALUES(100001,10000);
INSERT INTO account_mstr VALUES(100003,35000);
INSERT INTO account_mstr VALUES(100004,100000);
INSERT INTO account_mstr VALUES(100005,1000);
```

ACC_NO	BAL
100001	10000
100002	28000
100003	35000
100004	100000
100005	1000

```
Declare
```

```
Amount NUMBER(9);
Operation NUMBER(1);
Balance NUMBER(8);
Acc NUMBER(12);
Insufficient EXCEPTION;
Begin
DBMS_OUTPUT.PUT_LINE('1.Withdraw    2.Deposite');
Acc := &Account_Number;
Operation := &Operation;
Amount := &Amount;
SELECT bal INTO Balance FROM account_mstr WHERE acc_no=Acc;
IF Operation=1 THEN
  IF Balance < Amount THEN
    RAISE Insufficient;
  Else
    Update account_mstr SET bal=(bal-Amount) WHERE acc_no=Acc;
    DBMS_OUTPUT.PUT_LINE('Collect your cash');
  END IF;
Else
  Update account_mstr SET bal=(bal+Amount) WHERE acc_no=Acc;
  Amount := Amount + Balance;
  DBMS_OUTPUT.PUT_LINE('Updated Balance= ' || Amount);
END IF;
EXCEPTION
WHEN Insufficient THEN
DBMS_OUTPUT.PUT_LINE('Insufficient Balance!');
END;
```

OUTPUT:-

```
Enter value for account_number: 100001
old   9: Acc := &Account_Number;
new   9: Acc := 100001;
Enter value for operation: 2
old  10: Operation := &Operation;
new  10: Operation := 2;
Enter value for amount: 5000
old  11: Amount := &Amount;
new  11: Amount := 5000;
1.Withdraw    2.Deposite
Updated Balance= 15000
```

PL/SQL procedure successfully completed.

Q3.Write an PL/SQL code block that raises a user defined exception where business rule is violated. BR for client_master table specifies when the value of bal_due field is less than 0 handle the exception.

Ans:-

```
CREATE TABLE client_mstr(client_id NUMBER(9), due_bal NUMBER(8,2));

INSERT INTO client_mstr VALUES(15420,5000);
INSERT INTO client_mstr VALUES(15421,25000);
INSERT INTO client_mstr VALUES(15422,-5000);
INSERT INTO client_mstr VALUES(15423,8000);
INSERT INTO client_mstr VALUES(15424,21000);
```

CLIENT_ID	DUE_BAL
15420	5000
15421	25000
15422	-5000
15423	8000
15424	21000

```
DECLARE
  id NUMBER(9);
  bal_due NUMBER(8,2);
  bal_due_error EXCEPTION;
BEGIN
  id := &client_id;
  SELECT due_bal INTO bal_due FROM client_mstr WHERE client_id=id;
  IF bal_due < 0 THEN
    DBMS_OUTPUT.PUT_LINE('Balance Due cannot be negative');
    RAISE bal_due_error;
  Else
    DBMS_OUTPUT.PUT_LINE('Balance due: ' || bal_due);
  END IF;
EXCEPTION
```

```
  WHEN bal_due_error THEN
  DBMS_OUTPUT.PUT_LINE('The balance due for client ' || id || ' violates the business
rule');
END;
```

OUTPUT:-

```
Enter value for client_id: 15422
old   6: id := &client_id;
new   6: id := 15422;
Balance Due cannot be negative
The balance due for client 15422 violates the business rule

PL/SQL procedure successfully completed.
```

Q4.1.Borrower(Roll_no, Name, DateofIssue, NameofBook, Status)
2.Fine(Roll_no,Date,Amt)
Accept roll_no & name of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine will be Rs 5 per day. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If the condition of fine is true, then details will be stored into the fine table.
Also handles the exception by named exception handler or user defined exception handler.

Ans:-

```
CREATE TABLE Borrower (Roll_no NUMBER(2) PRIMARY KEY,Name
VARCHAR2(10),DateofIssue DATE,NameofBook VARCHAR2(25),Status CHAR(1));

INSERT INTO Borrower VALUES (1, 'John', TO_DATE('2023-01-01', 'YYYY-MM-DD'),
'Intro to Programming', 'I');
INSERT INTO Borrower VALUES (2, 'Alice', TO_DATE('2023-02-15', 'YYYY-MM-DD'),
'DSA', 'I');
INSERT INTO Borrower VALUES (3, 'Bob', TO_DATE('2023-03-01', 'YYYY-MM-DD'),
'Database Systems', 'I');
INSERT INTO Borrower VALUES (4, 'Jane', TO_DATE('2023-04-10', 'YYYY-MM-DD'),
'AI', 'I');
```

```
INSERT INTO Borrower VALUES (5, 'Mike', TO_DATE('2023-05-20', 'YYYY-MM-DD'),
'OS', 'I');
```

Roll No	Name	Date of Issue	Name of Book	Status
1	John	01-JAN-23	Intro to Programming	I
2	Alice	15-FEB-23	DSA	I
3	Bob	01-MAR-23	Database Systems	I
4	Jane	10-APR-23	AI	I
5	Mike	20-MAY-23	OS	I

```
CREATE TABLE Fine (Roll_no NUMBER(10),Dat DATE,Amt NUMBER(10,2));

INSERT INTO Fine (Roll_no, Dat, Amt) VALUES (12, '01-APR-2023', 15.00);
INSERT INTO Fine (Roll_no, Dat, Amt) VALUES (24, '15-APR-2023', 30.00);
INSERT INTO Fine (Roll_no, Dat, Amt) VALUES (24, SYSDATE, 30.00);
```

ROLL_NO	DAT	AMT
12	01-APR-23	15
24	15-APR-23	30

```
DECLARE
  roll_no NUMBER(10);
  name_of_book VARCHAR2(100);
  date_of_issue DATE;
  fine_amount NUMBER(10,2);
  status CHAR(1);
  days_overdue NUMBER(10);
  fine_calculation_error EXCEPTION;
BEGIN
  Fine_amount := 0;
  roll_no := &roll_no;
  name_of_book := '&name_of_book';
  SELECT DateofIssue, Status INTO date_of_issue, status FROM Borrower WHERE
Roll_no =   roll_no AND NameofBook = name_of_book;
    days_overdue := SYSDATE - date_of_issue;
    DBMS_OUTPUT.PUT_LINE('Number of Days= ' || days_overdue);
    IF days_overdue <= 15 THEN
      fine_amount := 0;
```

```
    ELSIF days_overdue <= 30 THEN
      fine_amount := 5 * days_overdue;
    ELSE
      fine_amount := 50 * days_overdue;
    END IF;
    UPDATE Borrower SET Status = 'R' WHERE Roll_no = roll_no AND
NameofBook=name_of_book;
    IF fine_amount > 0 THEN
      INSERT INTO Fine VALUES (roll_no, SYSDATE, fine_amount);
    END IF;
    DBMS_OUTPUT.PUT_LINE('Fine= ' || fine_amount);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: Could not calculate the fine amount for the book');
END;
```

OUTPUT:-

```
Enter value for roll_no: 1
old  11:   roll_no := &roll_no;
new  11:   roll_no := 1;
Enter value for name_of_book: Intro to Programming
old  12:   name_of_book := '&name_of_book';
new  12:   name_of_book := 'Intro to Programming';
Number of Days= 117
Fine= 5850

PL/SQL procedure successfully completed.
```