1. Consider table Stud(Roll, Att, Status)
Write a PL/SQL block for following requirement and handle the exceptions.

INPUT:

```
CREATE TABLE STUD(ROLL INT PRIMARY KEY,ATT INT,STATUS
VARCHAR(255));
INSERT INTO STUD VALUES (11, 50, 'D');
INSERT INTO STUD VALUES (12, 80, 'ND');
INSERT INTO STUD VALUES (13, 90, 'ND');
INSERT INTO STUD VALUES (14, 60, 'D');
INSERT INTO STUD VALUES (15, 88, 'ND');
SELECT * FROM STUD;

Declare
    mroll number(10);
        matt number(10);
Begin
   mroll:= 11;
   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;
```

OUPUT:

```
Table created.

1 row(s) inserted.
```

Result Set 1

ROLL	ATT	STATUS
11	50	D
12	80	ND
13	90	ND
14	60	D
15	88	ND

11 is detained

2. Write a PL/SQL block for following requirement using user defined exception Handling.

INPUT:

```
CREATE TABLE ACCOUNT_MASTER(ACC_NO INT PRIMARY KEY,
                                                    BALANCE INT);
INSERT INTO ACCOUNT_MASTER VALUES(123011, 10000);
INSERT INTO ACCOUNT_MASTER VALUES(123012, 15000);
INSERT INTO ACCOUNT_MASTER VALUES(123013, 20000);
INSERT INTO ACCOUNT_MASTER VALUES(123014, 50000);
INSERT INTO ACCOUNT_MASTER VALUES(123015, 25000);
INSERT INTO ACCOUNT_MASTER VALUES(123016, 5000);
INSERT INTO ACCOUNT_MASTER VALUES(123017, 60000);
INSERT INTO ACCOUNT_MASTER VALUES(123018, 55000);
```

FOR WITHDRAWAL:

```
DECLARE
   MBAL NUMBER(10);
   MACC NUMBER(10);
   TRANS NUMBER(10);
   OPERATION VARCHAR2(10); -- Variable to store operation type
   No_sufficient_bal EXCEPTION;
```

```
BEGIN
   MACC := 123012;
   TRANS := 5000;
   OPERATION := 'withdraw'; -- Set operation type to 'withdraw' or 'deposit'

   IF OPERATION = 'withdraw' THEN
      SELECT BALANCE INTO MBAL FROM ACCOUNT_MASTER WHERE
ACC_NO = MACC;

      IF TRANS <= MBAL THEN
         UPDATE ACCOUNT_MASTER SET BALANCE = (BALANCE - TRANS)
WHERE ACC_NO = MACC;
         DBMS_OUTPUT.PUT_LINE('Withdrawal of ' || TRANS || ' successful.');
      ELSE
         RAISE No_sufficient_bal;
      END IF;

   ELSIF OPERATION = 'deposit' THEN
      UPDATE ACCOUNT_MASTER SET BALANCE = (BALANCE + TRANS)
WHERE ACC_NO = MACC;
      DBMS_OUTPUT.PUT_LINE('Deposit of ' || TRANS || ' successful.');

   ELSE
      DBMS_OUTPUT.PUT_LINE('Invalid operation.');
   END IF;

EXCEPTION
   WHEN No_sufficient_bal THEN
      DBMS_OUTPUT.PUT_LINE('Sufficient balance is not available in account');
END;
/
```

OUTPUT:

```
Statement processed.
Withdrawal of 5000 successful.
```

FOR DEPOSIT:

```
DECLARE
   MBAL NUMBER(10);
   MACC NUMBER(10);
```

```
   TRANS NUMBER(10);
   OPERATION VARCHAR2(10); -- Variable to store operation type
   No_sufficient_bal EXCEPTION;
BEGIN
   MACC := 123012;
   TRANS := 5000;
   OPERATION := 'deposit';

   IF OPERATION = 'withdraw' THEN
      SELECT BALANCE INTO MBAL FROM ACCOUNT_MASTER WHERE
ACC_NO = MACC;

      IF TRANS <= MBAL THEN
         UPDATE ACCOUNT_MASTER SET BALANCE = (BALANCE - TRANS)
WHERE ACC_NO = MACC;
         DBMS_OUTPUT.PUT_LINE('Withdrawal of ' || TRANS || ' successful.');
      ELSE
         RAISE No_sufficient_bal;
      END IF;

   ELSIF OPERATION = 'deposit' THEN
      UPDATE ACCOUNT_MASTER SET BALANCE = (BALANCE + TRANS)
WHERE ACC_NO = MACC;
      DBMS_OUTPUT.PUT_LINE('Deposit of ' || TRANS || ' successful.');

   ELSE
      DBMS_OUTPUT.PUT_LINE('Invalid operation.');
   END IF;

EXCEPTION
   WHEN No_sufficient_bal THEN
      DBMS_OUTPUT.PUT_LINE('Sufficient balance is not available in account');
END;
/
```

Output:

Statement processed.
Deposit of 5000 successful.

3.Write an SQL code block these raise a user defined exception where business rule is voilated. BR for client_master table specifies when the value of bal_due field is less than 0 handle the exception.

```
Declare
        v_bal_due client_master.bal_due%TYPE;
        v_client_id client_master.client_id%TYPE;
 Begin
      for client_rec in (select client_id,bal_due from client_master) loop
            if client_rec.bal_due <0 then
                  v_client_id := client_rec.client_id;
                  raise_application_error(-20001,'Business rule violated: Balance due
cannot be less than 0 for client_id ' || v_client_id);
            end if;
       end loop;
 exception
        when others then
              dbms_output.put_line('an error occurred: '|| sqlerrm);
 end;
 /
```

PL/SQL procedure successfully completed.

4.Consider below database schema:
Borrower(Roll_no, Name, DateofIssue, NameofBook, Status)
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 amount will be Rs 5per 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 condition of fine is true, then details will be stored into fine table.
Also handles the exception by named exception handler or user define exception handler.

```
select * from borrower;

  ROLL_NO DATEOFISS STATUS     NAMEOFBOOK
---------- --------- ---------- ----------
        71 20-JAN-04 issued     c++
        72 18-DEC-23 issued     java
        73 25-NOV-23 issued     python
        74 06-JAN-24 issued     c##
```

```
       75 10-FEB-24 issued     php

declare
  mroll number(10);
  mname varchar(20);
  di date;
  dor date;
  fine number(10);
  difference number(20);
begin
  mroll:=&mroll;
  select nameofbook,dateofissue into mname,di from borrower where roll_no=mroll;
  select sysdate into dor from dual;
  difference:=TO_DATE(dor)-TO_DATE(di);
  if difference<15 then
    dbms_output.put_line('Book is returned');
    insert into fine(roll_no,dateofreturn,amount) values(mroll,dor,0);
    update borrower set status='return' where roll_no =mroll;
  elsif difference<=30 then
    fine:=(difference-15)*5;
    dbms_output.put_line('Book is returned');
    insert into fine(roll_no,dateofreturn,amount) values(mroll,dor,fine);
    update borrower set status='return' where roll_no =mroll;
  else
    fine:=15*5+((difference-30)*50);
    dbms_output.put_line('Book is returned');
    insert into fine(roll_no,dateofreturn,amount) values(mroll,dor,fine);
    update borrower set status='return' where roll_no =mroll;
  end if;
end;
/
```

OUTPUT:

```
PL/SQL procedure successfully completed.

SQL> select * from fine;

   ROLL_NO DATEOFRET     AMOUNT
---------- --------- ----------
       72 15-FEB-24       1525
```

SQL> select * from borrower;

ROLL_NO	DATEOFISS	STATUS	NAMEOFBOOK
71	20-JAN-04	issued	c++
72	18-DEC-23	return	java
73	25-NOV-23	issued	python
74	06-JAN-24	issued	c##
75	10-FEB-24	issued	php