# Assignment No.4

**Q1 Consider table Stud(Roll, Att, Status) Write a PL/SQL block for following requirement and handle the exceptions.**

**➔ Roll no. of student will be entered by user.**

**➔ Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message "Term not granted" and set the status in stud table as "D". Otherwise display message "Term granted" and set the status in stud table as "ND"**

```
CREATE TABLE STUD (

ROLL INT,

ATT INT,

STATUS VARCHAR(2)

);

INSERT INTO STUD VALUES (1,75,NULL);

INSERT INTO STUD VALUES (2,5,NULL);

INSERT INTO STUD VALUES (3,6,NULL);

INSERT INTO STUD VALUES (4,100,NULL);

INSERT INTO STUD VALUES (5,81,NULL);

INSERT INTO STUD VALUES (6,78,NULL);

INSERT INTO STUD VALUES (7,0,NULL);
```

**SQL> -- For single user**

```
SQL> DECLARE
  2     mroll NUMBER(10);
  3     matt NUMBER(10);
  4  BEGIN
  5     mroll := &mroll;
  6     SELECT att INTO matt FROM stud WHERE roll = mroll;
  7     IF matt < 75 THEN
  8        DBMS_OUTPUT.PUT_LINE(mroll || ' is detained');
  9        UPDATE stud SET status = 'D' WHERE roll = mroll;
 10     ELSE
 11        DBMS_OUTPUT.PUT_LINE(mroll || ' is not detained');
 12        UPDATE stud SET status = 'ND' WHERE roll = mroll;
 13     END IF;
```

```
 14  EXCEPTION
 15      WHEN NO_DATA_FOUND THEN
 16          DBMS_OUTPUT.PUT_LINE(mroll || ' not found');
 17  END;
 18  /
Enter value for mroll: 1
old   5:     mroll := &mroll;
new   5:     mroll := 1;
1 is not detained
PL/SQL procedure successfully completed.
SQL> select * from stud;
     ROLL        ATT ST
---------- ---------- --
         1         75 ND
         2          5
         3          6
         4        100
         5         81
         6         78
         7          0
7 rows selected.
SQL> --All in single block
SQL> DECLARE
  2      mroll NUMBER(10);
  3      matt NUMBER(10);
  4  BEGIN
  5      FOR REC IN (SELECT roll, att FROM stud) loop
  6          mroll := REC.roll;
  7          matt := REC.att;
  8          IF matt < 75 THEN
  9              DBMS_OUTPUT.PUT_LINE(mroll || ' is detained');
 10              UPDATE stud SET status = 'D' WHERE roll = mroll;
```

```
 11        ELSE
 12           DBMS_OUTPUT.PUT_LINE(mroll || ' is not detained');
 13           UPDATE stud SET status = 'ND' WHERE roll = mroll;
 14        END IF;
 15     END loop;
 16  EXCEPTION
 17     WHEN NO_DATA_FOUND THEN
 18        DBMS_OUTPUT.PUT_LINE(mroll || ' not found');
 19  END;
 20  /
1 is not detained
2 is detained
3 is detained
4 is not detained
5 is not detained
6 is not detained
7 is detained
PL/SQL procedure successfully completed.
SQL> select * from stud;
     ROLL        ATT ST
---------- ---------- --
        1         75 ND
        2          5 D
        3          6 D
        4        100 ND
        5         81 ND
        6         78 ND
        7          0 D
7 rows selected.
```

**Q2. Write a PL/SQL block for following requirement using user defined exception handling.**

**→ The account_master table records the current balance for an account, which is updated**

**whenever, any deposits or withdrawals takes 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 above requirement using user defined exception handling.**

```
SQL> DECLARE
  2    insufficient_balance EXCEPTION;
  3    invalid_transaction_type EXCEPTION;
  4    v_account_id ACCOUNT_MASTER.ACCOUNT_ID%TYPE;
  5    v_current_balance ACCOUNT_MASTER.BAL%TYPE;
  6    v_amount INT := 0;
  7    v_transaction_type CHAR(1);
  8
  9  BEGIN
 10     v_account_id := &Enteraccountno;
 11     v_transaction_type := UPPER('&Entertransactiontype');
 12     IF v_transaction_type NOT IN ('W', 'D') THEN
 13        RAISE invalid_transaction_type;
 14     END IF;
 15     IF v_transaction_type IN ('W', 'D') THEN
 16        v_amount := &Enteramount;
 17     END IF;
 18     SELECT BAL INTO v_current_balance
 19     FROM ACCOUNT_MASTER
 20     WHERE ACCOUNT_ID = v_account_id;
 21     IF v_transaction_type = 'W' AND v_amount > v_current_balance THEN
 22        -- Raise custom exception if withdrawal amount exceeds balance
 23        RAISE insufficient_balance;
 24     ELSE
 25        IF v_transaction_type = 'W' THEN
 26           UPDATE ACCOUNT_MASTER
 27           SET BAL = BAL - v_amount
 28           WHERE ACCOUNT_ID = v_account_id;
```

```
 29            DBMS_OUTPUT.PUT_LINE('Withdrawal successful. New balance for account ' || v_account_id || ' is ' ||
(v_current_balance - v_amount));

 30        -- Update balance after deposit (for deposit transaction)

 31        ELSIF v_transaction_type = 'D' THEN

 32            UPDATE ACCOUNT_MASTER

 33            SET BAL = BAL + v_amount

 34            WHERE ACCOUNT_ID = v_account_id;

 35            DBMS_OUTPUT.PUT_LINE('Deposit successful. New balance for account ' || v_account_id || ' is ' ||
(v_current_balance + v_amount));

 36        END IF;

 37     END IF;

 39  EXCEPTION

 40     WHEN insufficient_balance THEN

 41        DBMS_OUTPUT.PUT_LINE('Insufficient balance for account ' || v_account_id || '. Withdrawal amount
exceeds current balance.');

 42     WHEN invalid_transaction_type THEN

 43        DBMS_OUTPUT.PUT_LINE('Invalid transaction type. Please enter "W" for withdrawal or "D" for deposit.');

 44     WHEN OTHERS THEN

 45        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);

 46  END;

 47  /
```

**CASE 1:**

```
Enter value for enteraccountno: 1

old  10:     v_account_id := &Enteraccountno;

new  10:     v_account_id := 1;

Enter value for entertransactiontype: w

old  11:     v_transaction_type := UPPER('&Entertransactiontype');

new  11:     v_transaction_type := UPPER('w');

Enter value for enteramount: 1000

old  16:        v_amount := &Enteramount;

new  16:        v_amount := 1000;

Insufficient balance for account 1. Withdrawal amount exceeds current balance.

PL/SQL procedure successfully completed.
```

**CASE 2:**

```
Enter value for enteraccountno: 1
old  10:     v_account_id := &Enteraccountno;
new  10:     v_account_id := 1;
Enter value for entertransactiontype: w
old  11:     v_transaction_type := UPPER('&Entertransactiontype');
new  11:     v_transaction_type := UPPER('w');
Enter value for enteramount: 50
old  16:        v_amount := &Enteramount;
new  16:        v_amount := 50;
Withdrawal successful. New balance for account 1 is 50
PL/SQL procedure successfully completed.
```

**CASE 3:**

```
Enter value for enteraccountno: 1
old  10:     v_account_id := &Enteraccountno;
new  10:     v_account_id := 1;
Enter value for entertransactiontype: d
old  11:     v_transaction_type := UPPER('&Entertransactiontype');
new  11:     v_transaction_type := UPPER('d');
Enter value for enteramount: 10000
old  16:        v_amount := &Enteramount;
new  16:        v_amount := 10000;
Deposit successful. New balance for account 1 is 10050
PL/SQL procedure successfully completed.
SQL> SELECT * FROM ACCOUNT_MASTER;
ACCOUNT_ID       BAL
---------- ----------
         1     10050
         2       800
         3     19000
         4      1009
         5      1002
```

```
      6     10011

6 rows selected.
```

**Q3. 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.**

```
CREATE TABLE CLIENT_MASTER(
CLIENT_CODE INT,
BAL_DUE INT
);
INSERT INTO CLIENT_MASTER VALUES (1,100);
INSERT INTO CLIENT_MASTER VALUES (2,10);
INSERT INTO CLIENT_MASTER VALUES (3,-100);
INSERT INTO CLIENT_MASTER VALUES (4,-600);
INSERT INTO CLIENT_MASTER VALUES (5,1000);
INSERT INTO CLIENT_MASTER VALUES (6,0);
SQL> SELECT * FROM CLIENT_MASTER;
CLIENT_CODE    BAL_DUE
---------- ----------
      1       100
      2        10
      3      -100
      4      -600
      5      1000
      6         0

6 rows selected.

SQL> DECLARE
  2    negative_bal EXCEPTION;
  3    temp_bal client_master.bal_due%TYPE;
  4    temp_acc client_master.client_code%TYPE;
  5  BEGIN
```

```
 6   temp_acc := '&accept_account';
 7
 8   BEGIN
 9    SELECT bal_due INTO temp_bal FROM client_master WHERE client_code = temp_acc;
10
11    IF temp_bal < 0 THEN
12     RAISE negative_bal;
13    ELSE
14     DBMS_OUTPUT.PUT_LINE('Balance of ' || temp_acc || ' is ' || temp_bal);
15    END IF;
16   EXCEPTION
17    WHEN NO_DATA_FOUND THEN
18     DBMS_OUTPUT.PUT_LINE('No data found for account ' || temp_acc);
19   END;
20
21  EXCEPTION
22   WHEN negative_bal THEN
23    DBMS_OUTPUT.PUT_LINE('Negative balance detected for account ' || temp_acc);
24  END;
25  /
```

**CASE 1 :**

```
Enter value for accept_account: 1
old   6:   temp_acc := '&accept_account';
new   6:   temp_acc := '1';
Balance of 1 is 100
PL/SQL procedure successfully completed.
```

**CASE 2 :**

```
Enter value for accept_account: 3
old   6:   temp_acc := '&accept_account';
new   6:   temp_acc := '3';
Negative balance detected for account 3
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 handle**

```
CREATE TABLE Borrower(Roll_no INT , Name varchar(20), Date_of_Issue DATE , NameofBook varchar(20), Status varchar(10));

CREATE TABLE Fine(

Roll_no INT,

Date_OF_FINE DATE ,

Amt INT

);

INSERT INTO Borrower VALUES (1, 'Soham', TO_DATE('2024-02-07', 'YYYY-MM-DD'), 'Time is money', 'I');

INSERT INTO Borrower VALUES (2,'Hari',TO_DATE('2024-01-01', 'YYYY-MM-DD'),'Time Series','I');

INSERT INTO Borrower VALUES (3,'Vedant',TO_DATE('2024-01-07', 'YYYY-MM-DD'),'School is better','I');

INSERT INTO Borrower VALUES (4,'Mrunalini',TO_DATE('2024-02-1', 'YYYY-MM-DD'),'Press','I');

SQL> DECLARE
  2     v_roll_no INT;
  3     v_NameofBook VARCHAR(20);
  4     v_Date_of_Issue DATE;
  5     v_DateDiff INT;
  6     v_FineAmt INT;
  7  BEGIN
  8     v_roll_no := &roll_no;
  9     v_NameofBook := '&name_of_book';
 10      SELECT Date_of_Issue INTO v_Date_of_Issue
 11      FROM Borrower
 12      WHERE Roll_no = v_roll_no AND NameofBook = v_NameofBook;
 13      v_DateDiff := TRUNC(SYSDATE) - TRUNC(v_Date_of_Issue);
```

```
14     IF v_DateDiff > 30 THEN
15         v_FineAmt := 150+(5 * (v_DateDiff-30));
16     ELSIF v_DateDiff >= 15 THEN
17         v_FineAmt := 5 * v_DateDiff;
18     ELSE
19         v_FineAmt := 0;
20     END IF;
21     UPDATE Borrower
22     SET Status = 'R'
23     WHERE Roll_no = v_roll_no AND NameofBook = v_NameofBook;
24     IF v_FineAmt > 0 THEN
25         INSERT INTO Fine VALUES (v_roll_no, SYSDATE, v_FineAmt);
26     END IF;
27     DBMS_OUTPUT.PUT_LINE('Fine Amount: Rs. ' || v_FineAmt);
28  EXCEPTION
29     WHEN NO_DATA_FOUND THEN
30         DBMS_OUTPUT.PUT_LINE('Book not found for the provided roll number.');
31     WHEN OTHERS THEN
32         DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
33  END;
34  /
```

**CASE 1:**

```
Enter value for roll_no: 3
old   8:     v_roll_no := &roll_no;
new   8:     v_roll_no := 3;
Enter value for name_of_book: School is better
old   9:     v_NameofBook := '&name_of_book';
new   9:     v_NameofBook := 'School is better';
Fine Amount: Rs. 180
```

**CASE 2:**

```
Enter value for roll_no: 4
old   8:     v_roll_no := &roll_no;
```

```
new   8:     v_roll_no := 4;
Enter value for name_of_book: Press
old   9:     v_NameofBook := '&name_of_book';
new   9:     v_NameofBook := 'Press';
Fine Amount: Rs. 0
PL/SQL procedure successfully completed.
SQL> SELECT * FROM FINE;

  ROLL_NO DATE_OF_F       AMT
    ---------- ---------           ----------
             3    12-FEB-24       180
```