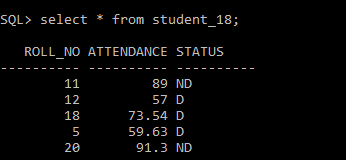
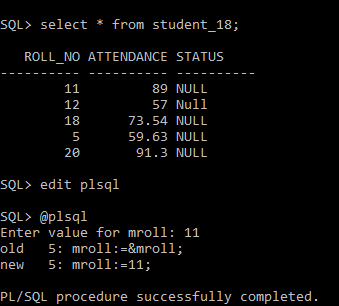
**Q1. Consider table Stud(Roll, Att,Status) Write a PL/SQL block for following requirement and handle the exceptions. Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message “Term not granted”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



**PL/SQL CODE:-**

Declare

mroll number(10);

matt number(10);

Begin

mroll:=&mroll;

select attendance into matt from student\_18 where roll\_no=mroll;

if matt<75then

dbms\_output.put\_line(mroll||'Term Not Granted');

update student\_18 set status='D'where roll\_no=mroll;

else

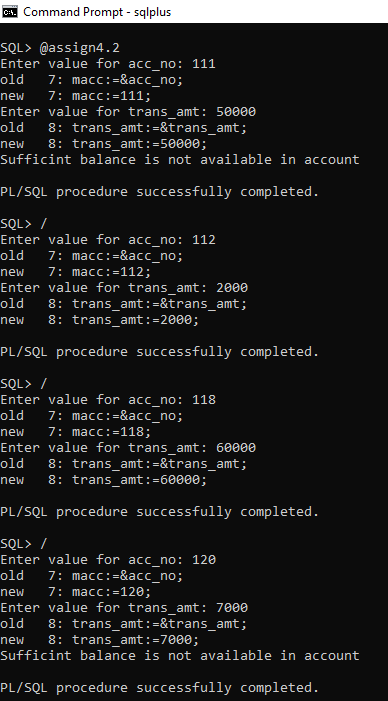
dbms\_output.put\_line(mroll||'Term Granted');

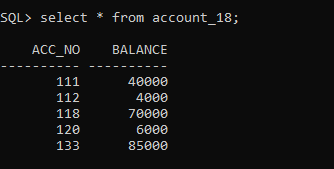
update student\_18 set status='ND'where roll\_no=mroll;

end if;

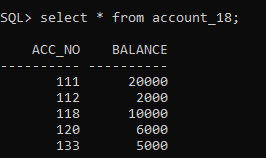
End;

/

**2. 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**



**AFTER PL/SQL PROCEDURE:-**



**PL/SQL CODE:-**

Declare

mbal number(10);

macc number(8);

trans\_amt number(10);

No\_sufficient\_bal exception;

Begin

macc:=&acc\_no;

trans\_amt:=&trans\_amt;

select balance into mbal from account\_18

where acc\_no = macc;

if (trans\_amt<=mbal) then

update account\_18 set balance=(balance-trans\_amt)

where acc\_no=macc;

else

raise No\_sufficient\_bal;

end if;

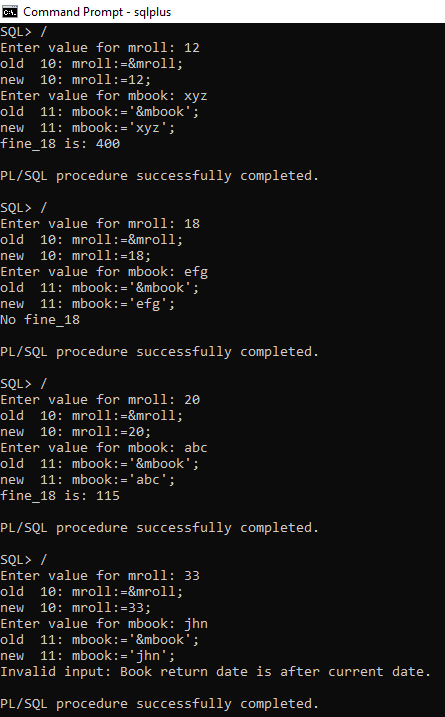
Exception

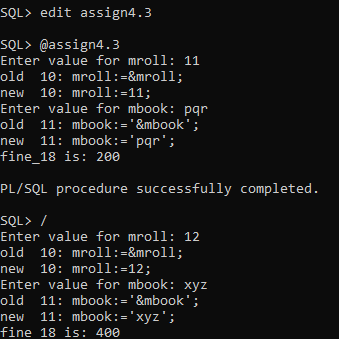
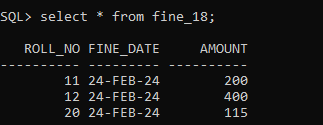
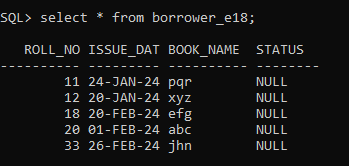
When No\_sufficient\_bal then

Dbms\_output.put\_line('Sufficint balance is not available in account');

End;

/

**3. 1. Borrower(Roll\_no, Name, Date\_of\_Issue, Name\_of\_Book, 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 amount will be Rs 5per day. If no. of days>30, 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.**



**PL/SQL CODE:-**

Declarel

mroll number(5);

mbook varchar(10);

mdate date;

mfine\_18 number(10);

datediff number(10);

-- User-define\_18d exceptions

invalid\_input EXCEPTION;

Begin

mroll:=&mroll;

mbook:='&mbook';

select issue\_date into mdate from borrower\_e18 where roll\_no=mroll and book\_name=mbook;

datediff:=to\_date(sysdate)-to\_date(mdate);

-- Check for overdue

IF datediff < 0 THEN

RAISE invalid\_input; -- Negative date difference indicates invalid input

end if;

IF datediff < 15 then

dbms\_output.put\_line('No fine\_18');

update borrower\_e18 set status='NF' where roll\_no=mroll and book\_name=mbook;

ELSIF datediff > 15 and datediff < 30 then

mfine\_18:=(datediff\*5);

dbms\_output.put\_line('fine\_18 is: '||mfine\_18);

update borrower\_e18 set status='F' where roll\_no=mroll and book\_name=mbook;

insert into fine\_18 values(mroll,sysdate,mfine\_18);

ELSIF datediff > 30 then

mfine\_18:=((datediff-30)\*50+(30\*5));

dbms\_output.put\_line('fine\_18 is: '||mfine\_18);

update borrower\_e18 set status='F' where roll\_no=mroll and book\_name=mbook;

insert into fine\_18 values(mroll,sysdate,mfine\_18);

End if;

Exception

WHEN invalid\_input THEN

dbms\_output.put\_line('Invalid input: Book return date is after current date.');

End;

/