PL/SQL

1. CURSOR EMPLOYEE

```
set serveroutput on;
DECLARE
    empid employee.id%type;
    empName employee.name%type;
    empSal employee.salary%type;
    CURSOR emp(given_sal number) is
    SELECT * FROM employee where salary < given_sal;
BEGIN
    OPEN emp(3200);
    LOOP
    FETCH emp into empid, empName, empSal;
    EXIT WHEN emp%notfound;
    dbms_output.put_line(empid || ' ' || empName || ' ' || empSal);
    END LOOP;
    CLOSE emp;
END;
```

2. FINE BORROWER BOOKS

```
SET SERVEROUTPUT ON;
DECLARE
    i_roll_no NUMBER;
    name_of_book VARCHAR2(25);
    no_of_days NUMBER;
    return_date DATE := TO_DATE(SYSDATE,'DD-MM-YYYY');
    temp NUMBER;
    doi DATE;
    fine NUMBER;
    NEG_DAYS exception;

BEGIN
    i_roll_no := &i_roll_no;
    name_of_book := '&nameofbook';
    SELECT to_date(borrower.dateofissue,'DD-MM-YYYY') INTO doi FROM borrower
WHERE borrower.roll_no = i_roll_no AND borrower.name_of_book = name_of_book;
    no_of_days := return_date-doi;
    IF (no_of_days<0) THEN
        raise NEG_DAYS;
    END IF;
    dbms_output.put_line(no_of_days);
    IF (no_of_days >15 AND no_of_days <=30) THEN
        fine := 5*no_of_days;

    ELSIF (no_of_days>30 ) THEN
        temp := no_of_days-30;
        fine := 150 + temp*50;
    END IF;
    dbms_output.put_line(fine);
    INSERT INTO fine VALUES(i_roll_no,return_date,fine);
    UPDATE borrower SET status = 'RETURNED' WHERE borrower.roll_no = i_roll_no;

    EXCEPTION
        WHEN NEG_DAYS THEN
        DBMS_OUTPUT.PUT_LINE('NEGATIVE DAYS NOT EXCEPTED');
        when NO_DATA_FOUND then
             dbms_output.put_line('no_data_found');
        when OTHERS then
             dbms_output.put_line('some_error_found');

END;
```

3. PROCEDURE PROC_GRADE

```
--CREATE TABLE STUDENT_MARKS_FINAL
--(
--FullName VARCHAR2(25),
--total_marks NUMBER
--);
--
--CREATE TABLE STUDENT_RESULTS
--(
--roll_number NUMBER ,
--FullName VARCHAR2(25),
--class VARCHAR2(30)
--);


CREATE OR REPLACE PROCEDURE proc_grade
(roll_no IN NUMBER, FullName IN VARCHAR2 ,marks IN NUMBER)
AS
BEGIN
    IF (marks<=1500 and marks>=990) THEN
        DBMS_OUTPUT.PUT_LINE (roll_no||' - '||FullName||' : DISTINCTION');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'DISTINCTION');
    ELSIF (marks<=989 and marks>=900) THEN
```

```
        DBMS_OUTPUT.PUT_LINE (roll_no||' - '||FullName||' : FIRST CLASS');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'FIRST CLASS');
    ELSIF (marks<=899 and marks>825) THEN
        DBMS_OUTPUT.PUT_LINE(roll_no||' - '||FullName||' : HIGHER SECOND CLASS');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'HIGHER SECOND
CLASS');
    ELSE
        DBMS_OUTPUT.PUT_LINE (roll_no||' - '||FullName||' : FAIL');
        INSERT INTO STUDENT_RESULTS VALUES (roll_no,FullName,'FAIL');
    END IF;
        INSERT INTO STUDENT_MARKS_FINAL VALUES (FullName,marks);
END proc_grade;

--set serveroutput on;
--BEGIN
--  proc_grade(1,'Garry',1000);
--  proc_grade(2,'Abbas ',720);
--  proc_grade(3,'Sohum ',650);
--  proc_grade(4,'Itachi ',570);
--END;
```

4. STUD ATTENDENCE

```
set serveroutput on;
declare
    s_rollno number;
    s_attend number;
    --s_status Stud.status%type;

begin
    s_rollno:=&s_rollno;
    select attend into s_attend from Stud where rollno=s_rollno;


    if s_attend < 75 then
        DBMS_OUTPUT.PUT_LINE('Term not granted');
        update Stud set status='D' where rollno=s_rollno;
    else
        DBMS_OUTPUT.PUT_LINE('Term granted');
        update Stud set status='ND' where rollno=s_rollno;
    end if;

    exception
        when no_data_found then
```

```
            dbms_output.put_line('No such student');
        when others then
             dbms_output.put_line('Error');
end;
```

5. TRIGGER LIBRARY

```
CREATE  TRIGGER trigger_1
AFTER UPDATE OR DELETE OR INSERT ON lib_tab FOR EACH ROW

    declare
    BEGIN
        IF UPDATING THEN
            dbms_output.put_line(:OLD.status);
            INSERT INTO library_audit VALUES
(SYSDATE,:OLD.book_name,:OLD.status,:NEW.status,'UPDATE');
        ELSIF INSERTING THEN
            dbms_output.put_line(:NEW.status);
            INSERT INTO library_audit VALUES
(SYSDATE,:NEW.book_name,:OLD.status,:NEW.status,'INSERT');
        ELSE
            dbms_output.put_line(:OLD.book_name||'deleting');
            INSERT INTO library_audit
VALUES(SYSDATE,:OLD.book_name,:OLD.status,:NEW.status,'DELETE');
        END IF;
    END;

--DELETE FROM lib_tab WHERE book_name = 'SILENT HILL';
--UPDATE lib_tab SET status = 'UNAVAILABLE' WHERE book_name =  'UNCHARTED';
--UPDATE lib_tab SET status = 'PRE-ORDER' WHERE book_name = 'GOD OF WAR';
--UPDATE lib_tab SET status = 'AVAILABLE' WHERE book_name = 'UNCHARTED';
--INSERT INTO lib_tab VALUES('SPM','UNAVAILABLE');
--Select * from library_audit;
--Select * from lib_tab;

--create table lib_tab(book_name varchar(20), status varchar(20));
```