DATABASE MANAGEMENT SYSTEMS LABORATORY

Lab Journal

NAME :- OJUS P. JAISWAL

YEAR & DIV :- TE A

ROLL NO. :- TACO19108

SEAT NO. :- S191094290

INDEX

Sr. No.		Title of the Experiment	Page No.
Group A			
1	A1	Study and Draw ER Modelling diagram along with normalization using ERD win/ERD plus for selected problem statement.	2-3
2	A2	a. Design and develop SQL DDL statements which demonstrates use of SQL objects such as Table, View, Index, Sequence, and Synonym. b. Design at least 10 subqueries for suitable database application using DML statements: Insert, select, Update and Delete with operators, functions and set operators	4-13
3	A3	Design at least 10 subqueries for suitable database application using DML statements: all types of joins, subquery and Views.	14-17
4	A4	Write Unnamed PL/SQL code block: use of control structures and exception handling	18-23
5	A6	Write Named PL/SQL stored procedure and stored function.	24-30
6	A7	Write a PL/SQL block of code using Implicit, Explicit, for loop and parameterized cursor that will merge the data available in the newly created table.	31-36
7	A8	Write a database trigger on a library table. The system should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in newly created table.	37-39
8	A9	Implement MYSQL/ORACLE database connectivity with PHP/PYTHON/JAVA implement database navigation operations using JDBC/ODBC.	40-42
Group B			
9	B1	Design and develop MongoDB queries using CRUD operations, SAVE method and logical operators.	43-45
10	B2	Implement Indexing and Aggregation using MongoDB	46-47
11	B3	Implement Map-reduce operation with suitable using MongoDB.	48-49
12	B4	Write a program to implement MongoDB database connectivity with PHP/PYTHON/JAVA implement database navigation operations using JDBC/ODBC.	50-52
Group C			
13	C1	According to DBMS concept covered in Group A and D develop and application using provided guidelines.	53-56

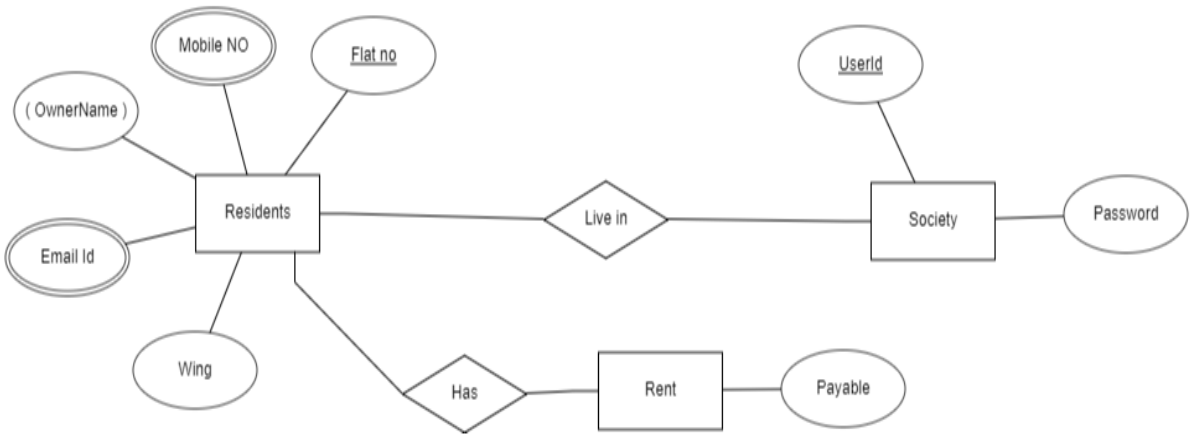
Assignment No. A1

Problem Statement :-

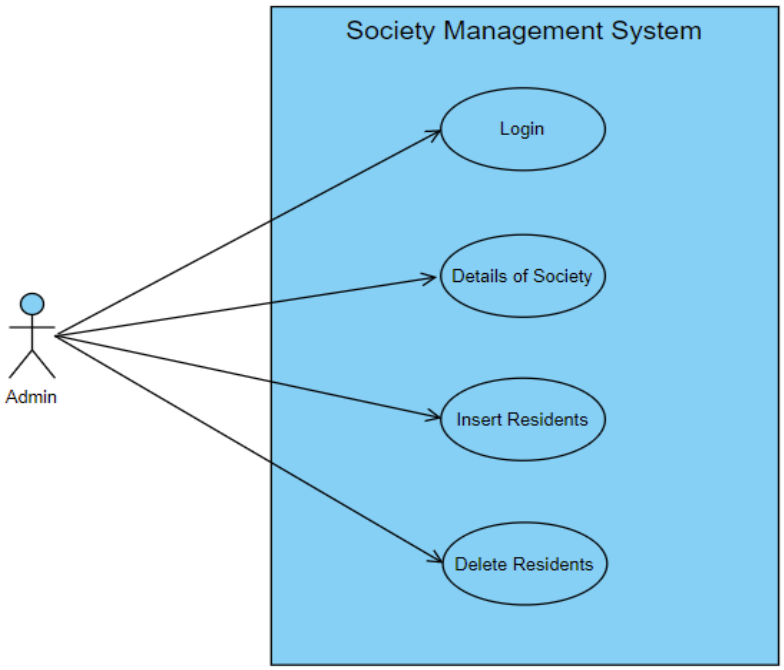
Study and Draw ER Modelling diagram along with normalization using ERD win / ERD plus for selected problem statement.

Solution :-

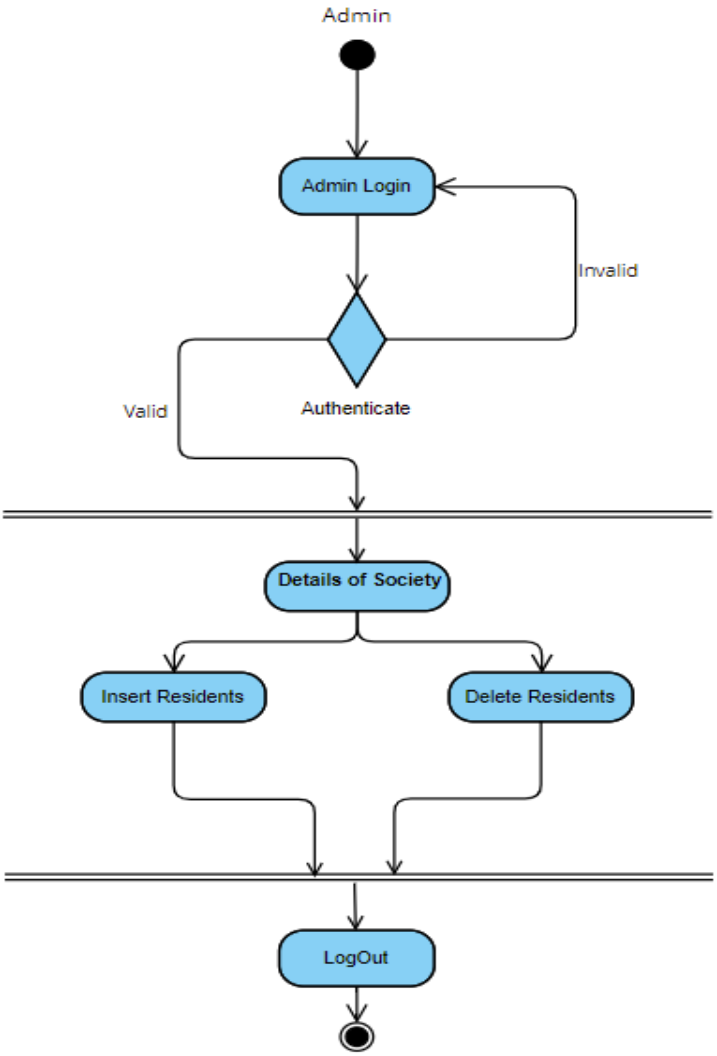
ER Diagram of Society Management System :



Use Case Diagram of Society Management System :



Activity Diagram of Society Management System :



Assignment No. A2

A. Problem Statement :-

An employee management system needs to record following data about employees – ID, Name, Age, Department, Salary, Experience, AreaOfExperties.

1. Identify columns, their data types and write create statement. Define primary key.
2. Create a view that will display all details of the employee except Salary and AreaOfExperties.
3. Create a sequence to generate employee id.
4. Create an index for the column ID.
5. Create a synonym for the generated table as “EMP” and demonstrate its use.

Solution :-

Program :

```
----CREATE A TABLE EMPLOYEE----
CREATE TABLE EMPLOYEE(
ID NUMBER PRIMARY KEY,
NAME VARCHAR2(20),
AGE NUMBER,
DEPARTMENT VARCHAR2(20),
SALARY NUMBER,
EXPERIENCE VARCHAR2(20),
AREAOFEXPERTIES VARCHAR2(20)
);
```

----CREATE A SEQUENCE TO GENERATE EMPLOYEE ID----

```
CREATE SEQUENCE SEQ

START WITH 1

INCREMENT BY 1;
```

----INSERTING ROWS IN TABLE EMPLOYEE----

```
INSERT INTO EMPLOYEE VALUES(SEQ.NEXTVAL,'AAA',30,'COMP',10000,'3 YEARS','TRAINER');

INSERT INTO EMPLOYEE VALUES(SEQ.NEXTVAL,'BBB',31,'CIVIL',20000,'4 YEARS','MANAGER') ;

INSERT INTO EMPLOYEE VALUES(SEQ.NEXTVAL,'CCC',31,'IT',30000,'5 YEARS','TEAMLEAD');
```

----CREATE A VIEW THAT WILL DISPLAY ALL DETAILS OF EMPLOYEE EXCEPT SALARY AND AREAOFEXPERTISE----

```
CREATE VIEW EMPLOYEE1 AS

SELECT ID,NAME,AGE,DEPARTMENT,EXPERIENCE FROM EMPLOYEE;
```

-----CRAETE AN INDEX FOR COLUMN_ID-----

```
CREATE INDEX INDEX_ID ON EMPLOYEE(ID);
```

-----CREATE SYNONYM FOR GENERATED TABLE AS 'EMP' AND DEMONSTRATE ITS USE------

```
CREATE SYNONYM EMP FOR EMPLOYEE;

SELECT * FROM EMP;
```

Output :

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.004 seconds

	ID	NAME	AGE	DEPARTMENT	SALARY	EXPERIENCE	AREAOFEXPERTIES
1	1	AAA	30	COMP	10000	3 YEARS	TRAINER
2	2	BBB	31	CIVIL	20000	4 YEARS	MANAGER
3	3	CCC	31	IT	30000	5 YEARS	TEAMLEAD

B. Problem Statement :-

For the following relation schema:
Account(Acc_no, branch_name,balance)
branch(branch_id, branch_name,branch_city,assets)
customer(cust_id, cust_name, cust_street,cust_city)
Depositor(cust_id, acc_no)
Loan(loan_no, branch_id, amount)
Borrower(cust_id, loan_no)

Create above tables and insert few rows in each table. Solve following query:

1. Find the branches where average account balance > 12000.
2. Find all customers who have an account or loan or both at bank.
3. Find all customers who have both account but not loan at bank.
4. Delete all tuples at every branch located in ‘Nigdi’.
5. Find Maximum loan amount in branch ‘Nigdi’
6. Find no. of depositors at each branch.
7. For all accounts in Akurdi branch increase the balance by 10%.

Solution :-

Program :

```
CREATE TABLE ACCOUNT
(
  ACC_NO INTEGER,
  BRANCH_NAME VARCHAR(30),
  BALANCE INTEGER
);

INSERT INTO ACCOUNT VALUES('10','AKURDI','1000');
INSERT INTO ACCOUNT VALUES('11','RAVET','2000');
INSERT INTO ACCOUNT VALUES('12','CHINCHWAD','3000');
```

```
CREATE TABLE BRANCH
(
  BRANCH_ID INTEGER,
  BRANCH_NAME VARCHAR(30),
  BRANCH_CITY VARCHAR(20),
  ASSETS VARCHAR(10)
);

INSERT INTO BRANCH VALUES('1','AKURDI','PUNE','HOUSE');
INSERT INTO BRANCH VALUES('2','RAVET','NASHIK','JEWELLERY');
INSERT INTO BRANCH VALUES('3','CHINCHWAD','AMRAVATI','FLAT');
INSERT INTO BRANCH VALUES('4','AKURDI','AMRAVAT','LAT');
INSERT INTO BRANCH VALUES('5','AKURDI','AMRAVA','AT');
INSERT INTO BRANCH VALUES('6','NIGDI','AMRAV','T');

CREATE TABLE CUSTOMER
(
  CUST_ID INTEGER,
  CUST_NAME VARCHAR(30),
  CUST_STREET VARCHAR(20),
  CUST_CITY VARCHAR(10)
);

INSERT INTO CUSTOMER VALUES('20','ABC','LINK ROAD','PUNE');
INSERT INTO CUSTOMER VALUES('21','BCD','LPRO ROAD','NASHIK');
INSERT INTO CUSTOMER VALUES('22','CDE','SHAGUN ROAD','AMRAVATI');
```

```
CREATE TABLE DEPOSITOR
(
  CUST_ID INTEGER,
  ACC_NO INTEGER
);

INSERT INTO DEPOSITOR VALUES('20','10');
INSERT INTO DEPOSITOR VALUES('21','11');
INSERT INTO DEPOSITOR VALUES('22','12');

CREATE TABLE LOAN
(
  LOAN_NO INTEGER,
  BRANCH_ID INTEGER,
  AMOUNT INTEGER
);

INSERT INTO LOAN VALUES('100','31','10000');
INSERT INTO LOAN VALUES('101','32','20000');
INSERT INTO LOAN VALUES('102','33','30000');
INSERT INTO LOAN VALUES('103','6','90000');

CREATE TABLE BORROWERR
(
  CUST_ID INTEGER,
  LOAN_NO INTEGER
);
```

```
INSERT INTO BORROWERR VALUES('41','1');
INSERT INTO BORROWERR VALUES('42','2');
INSERT INTO BORROWERR VALUES('43','3');


----FIND ALL BRANCHES WHERE AVERAGE BALANCE IS GREATER THAN 12000---
select BRANCH_NAME, avg (balance) from account
group by branch_name
having avg (balance) > 12000;

------FIND ALL CUSTOMERS WHO HAVE ACCOUNT BUT NOT LOAN ----
SELECT CUST_NAME FROM CUSTOMER,DEPOSITOR,BORROWERR
WHERE CUSTOMER.CUST_ID=DEPOSITOR.CUST_ID AND
BORROWERR.CUST_ID!=CUSTOMER.CUST_ID;

----DELETE ALL TUPLES AT EVERY BRANCH LOCATED IN NIGDI-----
DELETE FROM ACCOUNT
WHERE BRANCH_NAME='NIGDI';

------FIND MAX LOAN AMOUNT IN NIGDI BRANCH-----
SELECT MAX(LOAN.AMOUNT) AS "MAXIMUM AMOUNT" FROM LOAN,BRANCH
WHERE LOAN.BRANCH_ID = BRANCH.BRANCH_ID AND
BRANCH.BRANCH_NAME='NIGDI';

-------FIND NO. OF DEPOSITORS AT EACH BRANCH----
SELECT COUNT(DEPOSITOR.CUST_ID) AS "NO OF
CUSTOMERS",ACCOUNT.BRANCH_NAME FROM DEPOSITOR,ACCOUNT
WHERE DEPOSITOR.ACC_NO=ACCOUNT.ACC_NO
GROUP BY ACCOUNT.BRANCH_NAME;
```

```
-----FIND ALL ACCOUNTS IN AKURDI BRANCH INCREASE THE BALANCE BY10%----
UPDATE ACCOUNT
SET BALANCE=BALANCE*1.1
WHERE BRANCH_NAME='AKURDI';

select * from Account;
select * from branch;
select * from customer;
select * from Depositor;
select * from Loan;
select * from Borrower;
```

Output :

```
CUST_NAME
------------------------------
ABC
ABC
ABC
BCD
BCD
BCD
CDE
CDE
CDE

9 rows selected.


0 rows deleted.


MAXIMUM AMOUNT
--------------
         90000
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 3 in 0.006 seconds

	NO OF CUSTOMERS	BRANCH_NAME
1	1	RAVET
2	1	CHINCHWAD
3	1	AKURDI

Query Result × | Query Result 1 × | Query R

SQL | All Rows Fetched: 3 in 0.054 s

	ACC_NO	BRANCH_NAME	BALANCE
1	10	AKURDI	1100
2	11	RAVET	2000
3	12	CHINCHWAD	3000

Query Result × | Query Result 1 × | Query Result 2 × | Query

SQL | All Rows Fetched: 6 in 0.005 seconds

	BRANCH_ID	BRANCH_NAME	BRANCH_CITY	ASSETS
1	1	AKURDI	PUNE	HOUSE
2	2	RAVET	NASHIK	JEWELLERY
3	3	CHINCHWAD	AMRAVATI	FLAT
4	4	AKURDI	AMRAVAT	LAT
5	5	AKURDI	AMRAVA	AT
6	6	NIGDI	AMRAV	T

Query Result × | Query Result 1 × | Query Result 2 × | Query

SQL | All Rows Fetched: 3 in 0.005 seconds

	CUST_ID	CUST_NAME	CUST_STREET	CUST_CITY
1	20	ABC	LINK ROAD	PUNE
2	21	BCD	LPRO ROAD	NASHIK
3	22	CDE	SHAGUN ROAD	AMRAVATI

Query Result × | Query Result 1 × | Query

SQL | All Rows Fetched: 3 in 0.00

	CUST_ID	ACC_NO
1	20	10
2	21	11
3	22	12

Query Result × | Query Result 1 × | Query Result

SQL | All Rows Fetched: 4 in 0.004 second

	LOAN_NO	BRANCH_ID	AMOUNT
1	100	31	10000
2	101	32	20000
3	102	33	30000
4	103	6	90000

Query Result × | Query Result 1 × | Query Result 2 × | Query Result 3

SQL | All Rows Fetched: 3 in 0.004 seconds

	ROLLNO	B_NAME	DATEOFISSUE	NAMEOFBOOK	STATUS
1	101	PQR	09-OCT-21	TOC	Issued
2	102	ABC	03-AUG-21	DM	Issued
3	103	XYZ	01-SEP-21	CN	Issued

Assignment No. A3

Problem Statement :-

For the following relation schema:
employee(employee-name, street, city)
works(employee-name, company-name, salary)
company(company-name, city)
manages(employee-name, manager-name)

Create above tables and insert 5 rows in each table. Give an expression in SQL for each of the following queries:

1. Find the names, street address, and cities of residence for all employees who work for 'First Bank Corporation' and earn more than $10,000.
2. Find the names of all employees in the database who live in the same cities as the companies for which they work.
3. Display employee details that live in cities Pune, Mumbai, and Nasik
4. List employees from ‘First Bank Corporation’ that earn salary more than all employees of ‘Small Bank Corporation’
5. Create a view that will display employee details along with name of his/her manager.
6. Find average salary of employees of ‘First Bank Corporation’.
7. Give employees of ‘First Bank Corporation’ 15% rise if salary is less than 20000.

Solution :-

Program :

```
create table employees(emp_name VARCHAR(100),street VARCHAR(100) ,city
VARCHAR(100));
create table work(name VARCHAR(100),company VARCHAR(100),salary int);
create table company(cname VARCHAR(100),city VARCHAR(100));
create table manages(name VARCHAR(100),manager VARCHAR(100));

insert into employees values('Rohit','Pimpri','Pune');
insert into work values('Rohit','First Bank Corporation',20000);
INSERT INTO COMPANY VALUES('First Bank Corporation','Pune');
insert into manages values('Rohit','Tejas');
insert into employees values('Rahul','akurdi','Mumbai');
```

```
insert into work values('Rahul','First Bank Corporation',20500);
INSERT INTO COMPANY VALUES('First Bank Corporation','Mumbai');
insert into manages values('Rahul','Rohit');
insert into employees values('Pittu','AKURDI','Pune');
insert into work values('Pittu','Small Bank Corporation',5000);
INSERT INTO COMPANY VALUES('Small Bank Corporation','Pune');
insert into manages values('Pittu','Raj');


--1
SELECT a.emp_name,a.street,a.city FROM employees a,work b WHERE a.emp_name=b.name
and b.company='First Bank Corporation' AND b.salary>10000;

--2
select distinct a.emp_name from employees a,company b where a.city=b.city;

--3
select * from employees where city='Pune' or city='Mumbai' or city='Nashik';

--4
select name from work where COMPANY='First Bank Corporation' and salary > (select
max(salary) from work where company='Small Bank Corporation');

--5
create view my as select employees.emp_name,street,city,manager from employees FULL join
manages on employees.emp_name = manages.name;
SELECT * FROM MY;

--6
select avg(salary) from work where company='Small Bank Corporation';

--7
update work SET salary=(1.15*salary) where company='First Bank Corporation' and
salary<20000;
```

Output :

Script Output × | Query Result × | Query

Task completed in 0.735 se

```
View MY created.

>>Query Run In:Query Result 4
>>Query Run In:Query Result 5

0 rows updated.
```

Script Output × | Query Result × | Query Res

SQL | All Rows Fetched: 4 in 0.015 s

	EMP_NAME	STREET	CITY
1	Rohit	Pimpri	Pune
2	Rahul	akurdi	Mumbai
3	Rohit	Pimpri	Pune
4	Rahul	akurdi	Mumbai

Script Output × | Query Result × | Query

SQL | All Rows Fetched: 3 in 0.00

	EMP_NAME
1	Rohit
2	Pittu
3	Rahul

Script Output × | Query Result × | Query Result

SQL | All Rows Fetched: 3 in 0.001 secc

	EMP_NAME	STREET	CITY
1	Rohit	Pimpri	Pune
2	Rahul	akurdi	Mumbai
3	Pittu	AKURDI	Pune

Script Output × | Query Result × | Query R

SQL | All Rows Fetched: 4 in 0.001

	NAME
1	Rohit
2	Rahul
3	Rohit
4	Rahul

Script Output × | Query Result × | Query Result 1 × | Query

SQL | All Rows Fetched: 6 in 0.004 seconds

	EMP_NAME	STREET	CITY	MANAGER
1	Rohit	Pimpri	Pune	Tejas
2	Rahul	akurdi	Mumbai	Rohit
3	Pittu	AKURDI	Pune	Raj
4	Rohit	Pimpri	Pune	Tejas
5	Rahul	akurdi	Mumbai	Rohit
6	Pittu	AKURDI	Pune	Raj

Script Output × | Query Result × | Query Res

SQL | All Rows Fetched: 1 in 0.001 s

	AVG(SALARY)
1	5000

Assignment No. A4

Problem Statement :-

A) Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. Write a PL/SQL block of code for the following requirements:-
Schema:
1. Borrower(Rollin, Name, DateofIssue, NameofBook, Status)
2. Fine(Roll_no,Date,Amt)
   a. Accept roll_no & name of book from user.
   b. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
   c. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
   d. After submitting the book, status will change from I to R.
   e. If condition of fine is true, then details will be stored into fine table.

B) 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”

Solution :-

Program :

A)

```
set serveroutput on;

DECLARE

   Roll_No NUMBER;

   BookName varchar2(20);

   IssueDate DATE;

   CurrentDate DATE;

   NoOfDays Number;
```

```
    FineAmt Number;
BEGIN
    CurrentDate := SYSDATE;
    Roll_No := &ROLLNO;
    DBMS_OUTPUT.PUT_LINE('Enter Student Roll Number : '|| Roll_No);
    BookName := '&NAMEOFBOOK';
    DBMS_OUTPUT.PUT_LINE('Enter Book Name : '|| BookName);
    SELECT DateOfIssue into IssueDate FROM borrower WHERE RollNo = Roll_No AND
NameOfBook = BookName;
    DBMS_OUTPUT.PUT_LINE('Issue Date : '||IssueDate);
    NoOfDays := SYSDATE - IssueDate;
    DBMS_OUTPUT.PUT_LINE('No of Days : '|| NoOfDays);
    IF (NoOfDays> 30) THEN
        FineAmt :=NoOfDays * 50;
    ELSIF  (NoOfDays>= 15 AND NoOfDays<=30)  THEN
        FineAmt :=NoOfDays * 5;
    END IF;
    IF FINEAMT > 0 THEN
        INSERT INTO fine values (Roll_No, currentdate, FineAmt);
    END IF;
    UPDATE borrower SET Status = 'RETURNED' WHERE RollNo=Roll_No;
    Exception
        When no_data_found then
        DBMS_OUTPUT.PUT_LINE(Roll_No||'Not found');
END;
/
```

B)

```
Declare
    mroll number;
    matt number;
Begin
    mroll := &mroll;
    select studatt into matt from student where studroll = mroll;
    if matt<75 then
        dbms_output.put_line(mroll||' is detained');
        update student set status='D' where studroll=mroll;
    else
        dbms_output.put_line('Roll No. '||mroll||' is not detained');
        update student set status='ND' where studroll=mroll;
    end if;
    Exception
        when no_data_found then
        dbms_output.put_line(mroll||'Not found');
End;
/
```

Output :

A)

```
Script Output
Task completed in 8.152 seconds
Enter Student Roll Number : 101
Enter Book Name : TOC
Issue Date : 09-OCT-21
No of Days : 54.893217592592592592592592592592592592 59

PL/SQL procedure successfully completed.
```

```
Script Output
Task completed in 7.355 seconds
Enter Student Roll Number : 102
Enter Book Name : DM
Issue Date : 03-AUG-21
No of Days : 121.89520833333333333333333333333333333

PL/SQL procedure successfully completed.
```

```
Script Output
Task completed in 8.444 seconds
Enter Student Roll Number : 103
Enter Book Name : CN
Issue Date : 01-SEP-21
No of Days : 92.89564814814814814814814814814814814815

PL/SQL procedure successfully completed.
```

Welcome Page | DBMS | BORROWER

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Depende

Sort.. | Filter:

	ROLLNO	B_NAME	DATEOFISSUE	NAMEOFBOOK	STATUS
1	101	PQR	09-OCT-21	TOC	RETURNED
2	102	ABC	03-AUG-21	DM	RETURNED
3	103	XYZ	01-SEP-21	CN	RETURNED

Welcome Page | DBMS | FINE

Columns | Data | Model | Constraints | Grants | Statistics | Trigge

Sort.. | Filter:

	ROLLNO	CURR_DATE	AMT
1	101	02-DEC-21	2745
2	102	02-DEC-21	6095
3	103	02-DEC-21	4645

B)

Script Output | Task completed in 1.204 seconds

```
Roll No. 1 is not detained


PL/SQL procedure successfully completed.
```

Script Output | Task completed in 1.978 seconds

```
2 is detained


PL/SQL procedure successfully completed.
```

Script Output | Task completed in 2.151 seconds

```
Roll No. 3 is not detained


PL/SQL procedure successfully completed.
```

Script Output | Task completed in 2.429 seconds

```
Roll No. 4 is not detained


PL/SQL procedure successfully completed.
```

Welcome Page | DBMS | STUDENT

Columns | Data | Model | Constraints | Grants | Statistics | Trigger

Sort.. | Filter:

	STUDROLL	STUDATT	STATUS
1	1	75	ND
2	2	46	D
3	3	95	ND
4	4	78	ND

Assignment No. A6

Problem Statement :-

A) Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class
Write a PL/SQL block for using procedure created with above requirement.
Stud_Marks(name, total_marks)
Result(Roll,Name, Class)

B) Write a function namely func_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class
Write a PL/SQL block for using function created with above requirement.
Stud_Marks(name, total_marks)
Result(Roll,Name, Class)

Solution :-

Program :

A)

```
create or replace procedure proc_grade

(temp in number,

 p_roll_no out stud_marks.roll_no%type,

 p_name out stud_marks.name%type,

 p_total out stud_marks.total_marks%type)

as

begin

   select name,total_marks,roll_no into p_name,p_total,p_roll_no from stud_marks where
roll_no=temp;
```

```
    if p_total <=1500 and p_total >= 990 then
        insert into result values(p_roll_no,p_name,'distinction');
    elsif p_total <=989 and p_total >= 900 then
        insert into result values(p_roll_no,p_name,'first class');
    elsif p_total <=899 and p_total >= 825 then
        insert into result values(p_roll_no,p_name,'HSC');
    else
        insert into result values(p_roll_no,p_name,'fail');
    end if;

    Exception
        when no_data_found then
        dbms_output.put_line('Roll no ' || temp ||'  not found');
end;
/

Declare
   temp number(20);
   p_roll_no stud_marks.roll_no%type;
   p_name stud_marks.name%type;
   p_total stud_marks.total_marks%type;
Begin
   temp:=&temp;
   Proc_grade(temp,p_roll_no,p_name,p_total);
End;
/
```

```
select * from stud_marks;
select * from result;
```

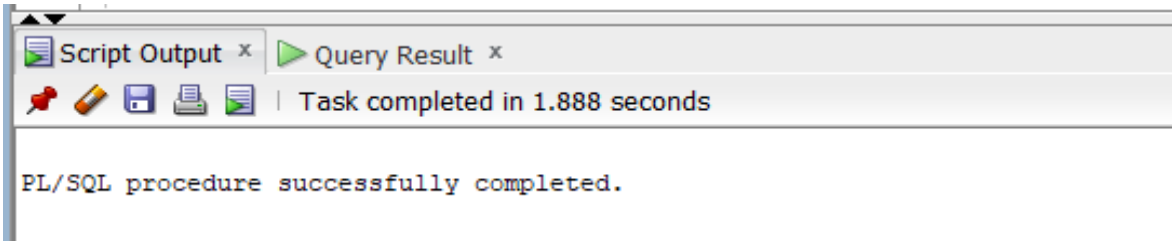
B)

```
create or replace function fun_grade
(temp in number)
return number
as
    p_roll_no stud_marks.roll_no%type;
    p_name stud_marks.name%type;
    p_total stud_marks.total_marks%type;
begin
    select name,total_marks,roll_no into p_name,p_total,p_roll_no from stud_marks where
roll_no=temp;

    if p_total <=1500 and p_total >= 990 then
        insert into result values(p_roll_no,p_name,'distinction');
    elsif p_total <=989 and p_total >= 900 then
        insert into result values(p_roll_no,p_name,'first class');
    elsif p_total <=899 and p_total >= 825 then
        insert into result values(p_roll_no,p_name,'HSC');
    else
        insert into result values(p_roll_no,p_name,'fail');
    end if;

    return p_roll_no;
```

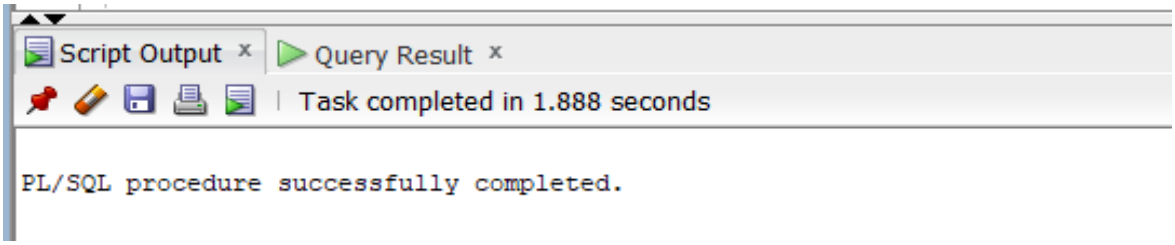
```
 Exception
        when no_data_found then
        dbms_output.put_line('Roll no ' || temp ||'  not found');
end;
/

Declare
    temp number(20):=&temp;
    p_roll_no varchar2(20);
Begin
    p_roll_no :=fun_grade(temp);
End;
/

select * from stud_marks;
select * from result;
```

Output :

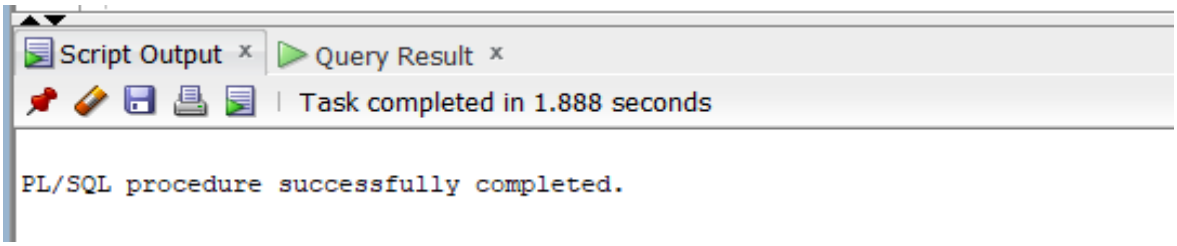
A)



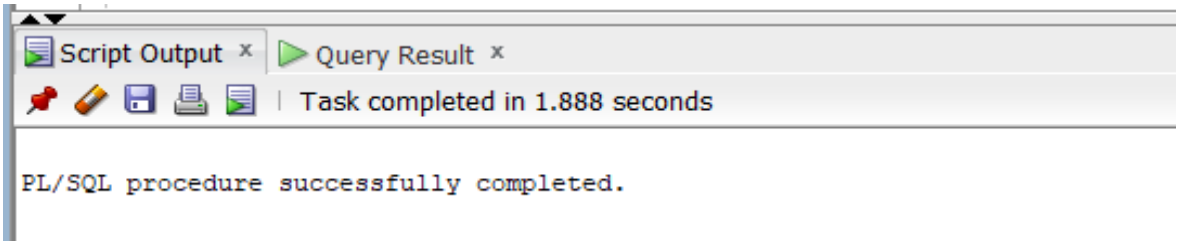
```
Task completed in 1.888 seconds

PL/SQL procedure successfully completed.
```



```
Task completed in 1.888 seconds

PL/SQL procedure successfully completed.
```



```
Task completed in 1.888 seconds

PL/SQL procedure successfully completed.
```



```
Task completed in 1.888 seconds

PL/SQL procedure successfully completed.
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 4 in 0.002 seconds

	ROLL_NO	NAME	TOTAL_MARKS
1	1	ABC	1000
2	2	XYZ	960
3	3	PQR	850
4	4	LMN	820

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 4 in 0.002 seconds

	ROLL_NO	NAME	CLASS
1	1	ABC	distinction
2	2	XYZ	first class
3	3	PQR	HSC
4	4	LMN	fail

B)

Script Output ×

Task completed in 1.837 seconds

```
PL/SQL procedure successfully completed.
```

Script Output ×

Task completed in 1.837 seconds

```
PL/SQL procedure successfully completed.
```

Script Output ×

Task completed in 1.837 seconds

```
PL/SQL procedure successfully completed.
```

Script Output

Task completed in 1.837 seconds

```
PL/SQL procedure successfully completed.
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 4 in 0.003 seconds

	ROLL_NO	NAME	TOTAL_MARKS
1	1	ABC	1000
2	2	XYZ	960
3	3	PQR	850
4	4	LMN	820

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 4 in 0.001 seconds

	ROLL_NO	NAME	CLASS
1	1	ABC	distinction
2	2	XYZ	first class
3	3	PQR	HSC
4	4	LMN	fail

Assignment No. A7

Problem Statement :-

A) Write PL/SQL block using explicit cursor for following requirements:
College has decided to mark all those students detained (D) who are having attendance less than 75%. Whenever such update takes place, a record for the same is maintained in the d_stud table.

B) Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table new_class with the data available in the table old_class. If the data in the first table already exist in the second table then that data should be skipped.

C) An explicit cursor FOR LOOP statement prints the last name and job ID of every clerk whose manager has an ID greater than 120.

Solution :-

Program :

A)

```
Declare
   Cursor crsr_att is select roll, att,status from stud where att<75;
   mroll stud.roll%type;
   matt stud.att%type;
   mstatus stud.status%type;
Begin
   open crsr_att;
   if crsr_att%isopen then
      loop
         fetch crsr_att into mroll,matt,mstatus;
```

```
            exit when crsr_att%notfound;
            if crsr_att%found then
                update stud set status='D' where roll=mroll;
                insert into d_stud values(mroll,matt);
            end if;
        end loop;
    end if;
end;

select * from stud;
select * from d_stud;
```

B)

```
Declare
    cursor crsr_class is select * from old_class;
    cursor crsr_chk(str_name varchar) is select roll from new_class where name = str_name;
    str_roll new_class.roll%type;
    str_name new_class.name%type;
    v varchar(10);
Begin
    Open crsr_class;
    Loop
        fetch crsr_class into str_roll,str_name;
        Exit When crsr_class%NOTFOUND;
        Open crsr_chk(str_name);
        Fetch crsr_chk into v;
        if crsr_chk%FOUND Then
            dbms_output.put_line('brach '|| str_name || ' exist');
```

```
        Else
            dbms_output.put_line('brach '|| str_name || ' not exist. Inserting in New_class table');
            insert into new_class values(str_roll,str_name);
        End if;
        Close crsr_chk;
    End loop;
    Close crsr_class;
End;

select * from old_class;
select * from new_class;
```

C)

```
DECLARE
  CURSOR c1 IS
    SELECT last_name, job_id FROM employees1
    WHERE job_id LIKE '%CLERK%' AND manager_id > 120
    ORDER BY last_name;
BEGIN
  FOR item IN c1
  LOOP
    DBMS_OUTPUT.PUT_LINE
      ('Name = ' || item.last_name || ', Job = ' || item.job_id);
  END LOOP;
END;

select * from employees1;
```

Output :

A)

Script Output × | Query Result × | Query Result 1 ×

Task completed in 0.092 seconds

```
Table STUD created.

Table D_STUD created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

PL/SQL procedure successfully completed.
```

Script Output × | Query Result × | Query Result 1 ×

SQL | All Rows Fetched: 4 in 0.003 seconds

	ROLL	ATT	STATUS
1	1	80	(null)
2	2	56	D
3	3	90	(null)
4	4	75	(null)

Script Output × | Query Result × | Query R

SQL | All Rows Fetched: 1 in 0.002

	ROLL	ATT
1	2	56

B)

Script Output

Task completed in 0.025 seconds

```
brach ABC exist
brach DEF not exist. Inserting in New_class table
brach GHI not exist. Inserting in New_class table


PL/SQL procedure successfully completed.
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 3 in 0.002 seconds

	ROLL	NAME
1	1	ABC
2	2	DEF
3	3	GHI

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 4 in 0.001 seconds

	ROLL	NAME
1	1	ABC
2	4	JKL
3	2	DEF
4	3	GHI

C)

Script Output

Task completed in 0.026 seconds

```
Name = ABC, Job = PT_CLERK
Name = JKL, Job = SB_CLERK


PL/SQL procedure successfully completed.
```

Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.002 seconds

	MANAGER_ID	LAST_NAME	JOB_ID
1	121	ABC	PT_CLERK
2	215	DEF	Manager
3	110	GHI	SB_CLERK
4	140	JKL	SB_CLERK

Assignment No. A8

Problem Statement :-

Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in LibraryAudit table.

Solution :-

Program :

```
create or replace trigger Library_insert
after insert on Library
for each row
begin
insert into LibraryAudit values
(:new.Book_ID,:new.Book_Name,sysdate,'insert');
end;
/

create or replace trigger Library_update
before update on Library
for each row
begin
insert into LibraryAudit values
(:old.Book_ID,:old.Book_Name,sysdate,'update');
end;
/
```

```
create or replace trigger Library_delete
before delete on Library
for each row
begin
insert into LibraryAudit values
(:old.Book_ID,:old.Book_Name,sysdate,'delete');
end;
/

update Library set Status='Returned' where Book_ID=1;

insert into Library values(4,'JKL','26-NOV-2021','Issued');

delete from Library where Book_ID=2;

select * from Library;
select * from LibraryAudit;
```

Output :

Script Output

Task completed in 0.031 seconds

```
Trigger LIBRARY_INSERT compiled

Trigger LIBRARY_UPDATE compiled

Trigger LIBRARY_DELETE compiled
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 3 in 0.003 seconds

	BOOK_ID	BOOK_NAME	ISSUEDATE	STATUS
1	1	ABC	01-OCT-21	Returned
2	3	GHI	31-OCT-21	Issued
3	4	JKL	26-NOV-21	Issued

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 3 in 0.003 seconds

	BOOK_ID	BOOK_NAME	ACTIONDATE	ACTION
1	1	ABC	03-DEC-21	update
2	4	JKL	03-DEC-21	insert
3	2	DEF	03-DEC-21	delete

Assignment No. A9

Problem Statement :-

Implement MYSQL/ORACLE database connectivity with PHP/PYTHON/JAVA implement database navigation operations using JDBC/ODBC.

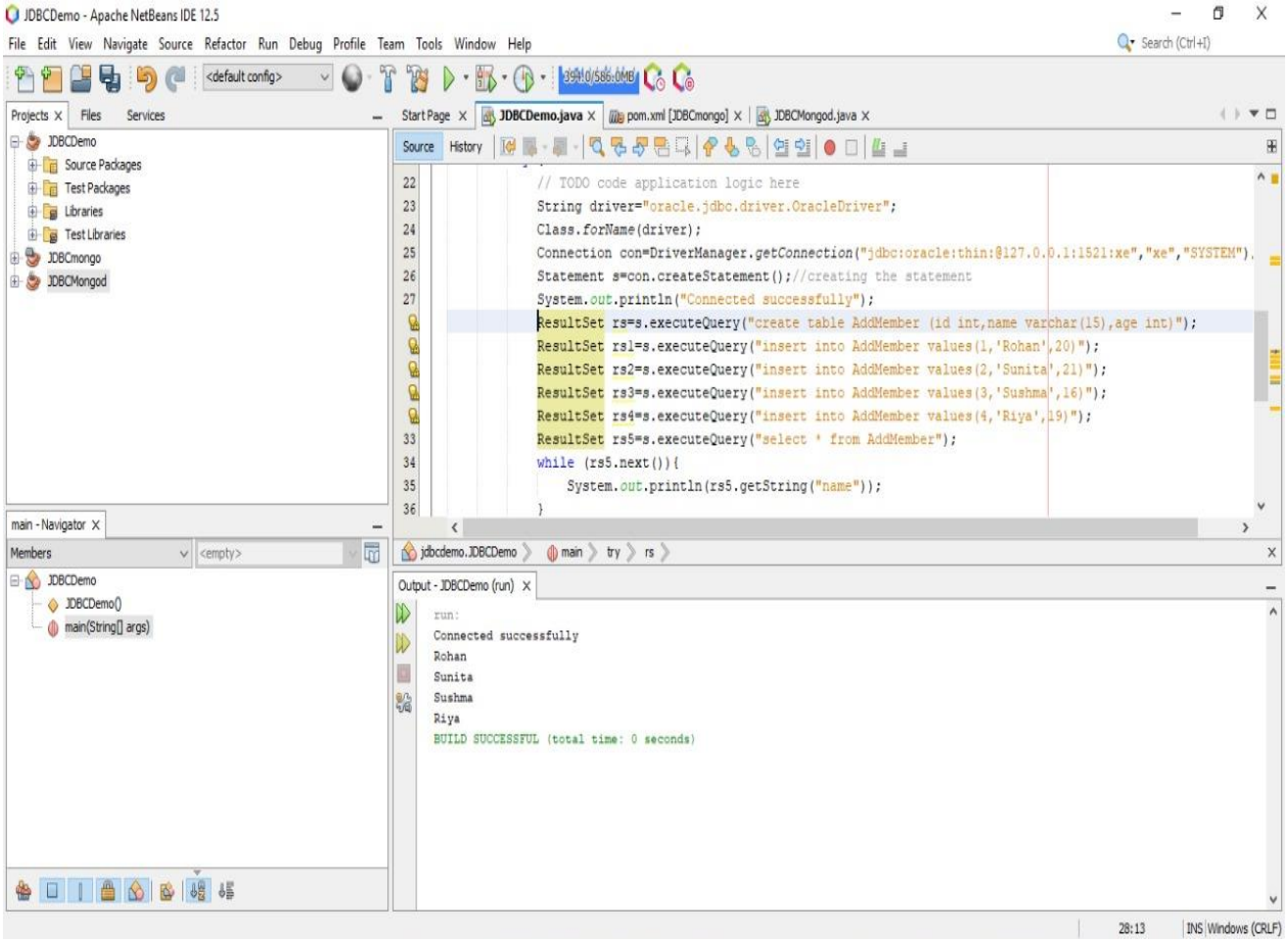
Solution :-

Program :

```
package A9;

import  java.sql.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JDBCDemo {

    public static void main(String[] args) {

        try {
                String driver="oracle.jdbc.driver.OracleDriver";
                Class.forName(driver);
                Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@127.0.0.1:1521:xe","system","paramojus");

                //creating the statement
                Statement s=con.createStatement();
                System.out.println("Connected successfully");

                ResultSet rs=s.executeQuery("create table AddMember (id int, name varchar(15),
age int)");
                ResultSet rs1=s.executeQuery("insert into AddMember values(1, 'Rohan', 20)");
                ResultSet rs2=s.executeQuery("insert into AddMember values(2, 'Sunita', 21)");
                ResultSet rs3=s.executeQuery("insert into AddMember values(3, 'Sushma', 16)");
                ResultSet rs4=s.executeQuery("insert into AddMember values(4, 'Riya', 19)");
                ResultSet rs5=s.executeQuery("select * from AddMember");

                while (rs5.next()){
                System.out.println(rs5.getString("name"));
                }
```

```
        } catch (Exception ex) {
            System.out.println("Error:"+ex);
        }

    }

}
```

Output :


JDBCDemo - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
JDBCDemo
Source Packages
Test Packages
Libraries
Test Libraries
JDBCmongo
JDBCMongod
Start Page
JDBCDemo.java
pom.xml [JDBCmongo]
JDBCMongod.java
Source History
// TODO code application logic here
String driver="oracle.jdbc.driver.OracleDriver";
Class.forName(driver);
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@127.0.0.1:1521:xe","xe","SYSTEM");
Statement s=con.createStatement();//creating the statement
System.out.println("Connected successfully");
ResultSet rs=s.executeQuery("create table AddMember (id int,name varchar(15),age int)");
ResultSet rs1=s.executeQuery("insert into AddMember values(1,'Rohan',20)");
ResultSet rs2=s.executeQuery("insert into AddMember values(2,'Sunita',21)");
ResultSet rs3=s.executeQuery("insert into AddMember values(3,'Sushma',16)");
ResultSet rs4=s.executeQuery("insert into AddMember values(4,'Riya',19)");
ResultSet rs5=s.executeQuery("select * from AddMember");
while (rs5.next()){
System.out.println(rs5.getString("name"));
}
main - Navigator
Members
JDBCDemo
JDBCDemo()
main(String[] args)
jdbcdemo.JDBCDemo main try rs
Output - JDBCDemo (run)
run:
Connected successfully
Rohan
Sunita
Sushma
Riya
BUILD SUCCESSFUL (total time: 0 seconds)
28:13
INS Windows (CRLF)

Assignment No. B1

Problem Statement :-

Create a collection **employee** in mongodb and insert few documents with fields (emp_id, name, dept, salary)

1. Find employees having salary greater than 50000
2. Find employees having salary between 50000 and 80000
3. Find employees having salary more than 60000 from ‘hr’ department
4. Update salary of all employees from ‘comp’ department. Set salary to 40000
5. Delete employees from ‘hr’ department having salary less than 45000

Solution :-

Program :

```
db.employee.insert([{"emp_id":"1","name":"Shubh","dept":"comp","salary":50005},
{"emp_id":"2","name":"Shubhash","dept":"hr","salary":40000},
{"emp_id":"3","name":"Baji","dept":"stack","salary":101200},
{"emp_id":"4","name":"Ram","dept":"comp","salary":10120}])
```

```
db.employee.find().pretty()
```

1. db.employee.find({"salary":{$gt:50000}})
2. db.employee.find({"salary":{$gte:50000,$lte:80000}})
3. db.employee.find({$and:[{"salary":{$gt:60000}},{"dept":"hr"}]})
4. db.employee.updateMany({"dept":"comp"},{$set:{"salary":40000}})
5. db.employee.remove({$and:[{"dept":"hr"},{"salary":{$lt:45000}}]})

Output :

```
>
> db.employee.insert([{"emp_id":"1","name":"Shubh","dept":"comp","salary":50005},{"emp_id":"2","name":"Shubhash","dept":"
hr","salary":40000}, {"emp_id":"3","name":"Baji","dept":"stack","salary":101200}, {"emp_id":"4","name":"Ram","dept":"comp
","salary":10120}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.employee.find().pretty()
{
        "_id" : ObjectId("61a9b88c61c184cc858062e9"),
        "emp_id" : "1",
        "name" : "Shubh",
        "dept" : "comp",
        "salary" : 50005
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062ea"),
        "emp_id" : "2",
        "name" : "Shubhash",
        "dept" : "hr",
        "salary" : 40000
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062eb"),
        "emp_id" : "3",
        "name" : "Baji",
        "dept" : "stack",
        "salary" : 101200
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062ec"),
        "emp_id" : "4",
        "name" : "Ram",
        "dept" : "comp",
        "salary" : 10120
}
> db.employee.find({"salary":{$gt:50000}})
{ "_id" : ObjectId("61a9b88c61c184cc858062e9"), "emp_id" : "1", "name" : "Shubh", "dept" : "comp", "salary" : 50005 }
{ "_id" : ObjectId("61a9b88c61c184cc858062eb"), "emp_id" : "3", "name" : "Baji", "dept" : "stack", "salary" : 101200 }
>
> db.employee.find({"salary":{$gte:50000,$lte:80000}})
{ "_id" : ObjectId("61a9b88c61c184cc858062e9"), "emp_id" : "1", "name" : "Shubh", "dept" : "comp", "salary" : 50005 }
>
> db.employee.find({$and:[{"salary":{$gt:60000}},{"dept":"hr"}]})
```

```
>
> db.employee.updateMany({"dept":"comp"},{$set:{"salary":40000}})
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 0 }
>
> db.employee.find().pretty()
{
        "_id" : ObjectId("61a9b88c61c184cc858062e9"),
        "emp_id" : "1",
        "name" : "Shubh",
        "dept" : "comp",
        "salary" : 40000
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062ea"),
        "emp_id" : "2",
        "name" : "Shubhash",
        "dept" : "hr",
        "salary" : 40000
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062eb"),
        "emp_id" : "3",
        "name" : "Baji",
        "dept" : "stack",
        "salary" : 101200
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062ec"),
        "emp_id" : "4",
        "name" : "Ram",
        "dept" : "comp",
        "salary" : 40000
}
>
```

```
>
> db.employee.remove({$and:[{"dept":"hr"},{"salary":{$lt:45000}}]})
WriteResult({ "nRemoved" : 1 })
>
> db.employee.find().pretty()
{
        "_id" : ObjectId("61a9b88c61c184cc858062e9"),
        "emp_id" : "1",
        "name" : "Shubh",
        "dept" : "comp",
        "salary" : 40000
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062eb"),
        "emp_id" : "3",
        "name" : "Baji",
        "dept" : "stack",
        "salary" : 101200
}
{
        "_id" : ObjectId("61a9b88c61c184cc858062ec"),
        "emp_id" : "4",
        "name" : "Ram",
        "dept" : "comp",
        "salary" : 40000
}
>
```

Assignment No. B2

Problem Statement :-

Create a collection **employees** in mongodb and insert few documents with fields (emp_id, emp_name, dept, salary)

1. Display maximum salary in each department
2. Display minimum salary in each department
3. Display average salary in each department
4. Display number of employees in each department

Solution :-

Program :

```
db.employees.insert([{"emp_id":1,"emp_name":"ABC","dept":"HR","salary":20000},
{"emp_id":2,"emp_name":"BCD","dept":"Developer","salary":25000},
{"emp_id":3,"emp_name":"CDE","dept":"Testing","salary":10000},
{"emp_id":4,"emp_name":"DEF","dept":"Developer","salary":20000},
{"emp_id":5,"emp_name":"EFG","dept":"Testing","salary":40000},
{"emp_id":6,"emp_name":"FGH","dept":"HR","salary":33000}])

db.employees.find().pretty()

1. db.employees.aggregate([{$group:{_id:"$dept",max_salary:{$max:"$salary"}}}])
2. db.employees.aggregate([{$group:{_id:"$dept",min_salary:{$min:"$salary"}}}])
3. db.employees.aggregate([{$group:{_id:"$dept",avg_salary:{$avg:"$salary"}}}])
4. db.employees.aggregate([{$group:{_id:"$dept",no_of_emp:{$sum:1}}}])
```

Output :

```
> db.employees.find().pretty()
{
        "_id" : ObjectId("61a9c28d61c184cc858062ed"),
        "emp_id" : 1,
        "emp_name" : "ABC",
        "dept" : "HR",
        "salary" : 20000
}
{
        "_id" : ObjectId("61a9c28d61c184cc858062ee"),
        "emp_id" : 2,
        "emp_name" : "BCD",
        "dept" : "Developer",
        "salary" : 25000
}
{
        "_id" : ObjectId("61a9c28d61c184cc858062ef"),
        "emp_id" : 3,
        "emp_name" : "CDE",
        "dept" : "Testing",
        "salary" : 10000
}
{
        "_id" : ObjectId("61a9c28d61c184cc858062f0"),
        "emp_id" : 4,
        "emp_name" : "DEF",
        "dept" : "Developer",
        "salary" : 20000
}
{
        "_id" : ObjectId("61a9c28d61c184cc858062f1"),
        "emp_id" : 5,
        "emp_name" : "EFG",
        "dept" : "Testing",
        "salary" : 40000
}
{
        "_id" : ObjectId("61a9c28d61c184cc858062f2"),
        "emp_id" : 6,
        "emp_name" : "FGH",
        "dept" : "HR",
        "salary" : 33000
}
> db.employees.aggregate([{$group:{_id:"$dept",max_salary:{$max:"$salary"}}}])
{ "_id" : "Testing", "max_salary" : 40000 }
{ "_id" : "HR", "max_salary" : 33000 }
{ "_id" : "Developer", "max_salary" : 25000 }
>
```

```
>
> db.employees.aggregate([{$group:{_id:"$dept",min_salary:{$min:"$salary"}}}])
{ "_id" : "Testing", "min_salary" : 10000 }
{ "_id" : "HR", "min_salary" : 20000 }
{ "_id" : "Developer", "min_salary" : 20000 }
>
> db.employees.aggregate([{$group:{_id:"$dept",avg_salary:{$avg:"$salary"}}}])
{ "_id" : "Testing", "avg_salary" : 25000 }
{ "_id" : "HR", "avg_salary" : 26500 }
{ "_id" : "Developer", "avg_salary" : 22500 }
>
> db.employees.aggregate([{$group:{_id:"$dept",no_of_emp:{$sum:1}}}])
{ "_id" : "Testing", "no_of_emp" : 2 }
{ "_id" : "HR", "no_of_emp" : 2 }
{ "_id" : "Developer", "no_of_emp" : 2 }
>
```

Assignment No. B3

Problem Statement :-

Create a collection books in mongodb and insert few documents with fields (book_id, title, author, type)
Write a MapReduce function to display number of books of each type.

Solution :-

Program :

```
db.books.insert([{book_id:1,title:"My",author:"Rajesh",type:"songs"},
{book_id:2,title:"Jack",author:"Raj",type:"Poem"},
{book_id:3,title:"What",author:"John",type:"Story"},
{book_id:4,title:"Real",author:"Warner",type:"Real Stories"},
{book_id:5,title:"Ram",author:"Raj",type:"Poem"},
{book_id:6,title:"Temperature",author:"Tejas",type:"Story"}])

var Mapfunction = function(){emit(this.type,1)}
var Reducefunction = function(key,values){return Array.sum(values)}
db.books.mapReduce(Mapfunction,Reducefunction,{'out':'typeofbooks'})

db.typeofbooks.find()
```

Output :

```
> db.books.insert([{book_id:1,title:"My",author:"Rajesh",type:"songs"},
... {book_id:2,title:"Jack",author:"Raj",type:"Poem"},
... {book_id:3,title:"What",author:"John",type:"Story"},
... {book_id:4,title:"Real",author:"Warner",type:"Real Stories"},
... {book_id:5,title:"Ram",author:"Raj",type:"Poem"},
... {book_id:6,title:"Temperature",author:"Tejas",type:"Story"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

```
>
> db.books.find().pretty()
{
        "_id" : ObjectId("61aa4e4f8dbb58a5e29bf6b5"),
        "book_id" : 1,
        "title" : "My",
        "author" : "Rajesh",
        "type" : "songs"
}
{
        "_id" : ObjectId("61aa4e4f8dbb58a5e29bf6b6"),
        "book_id" : 2,
        "title" : "Jack",
        "author" : "Raj",
        "type" : "Poem"
}
{
        "_id" : ObjectId("61aa4e4f8dbb58a5e29bf6b7"),
        "book_id" : 3,
        "title" : "What",
        "author" : "John",
        "type" : "Story"
}
{
        "_id" : ObjectId("61aa4e4f8dbb58a5e29bf6b8"),
        "book_id" : 4,
        "title" : "Real",
        "author" : "Warner",
        "type" : "Real Stories"
}
{
        "_id" : ObjectId("61aa4e4f8dbb58a5e29bf6b9"),
        "book_id" : 5,
        "title" : "Ram",
        "author" : "Raj",
        "type" : "Poem"
}
{
        "_id" : ObjectId("61aa4e4f8dbb58a5e29bf6ba"),
        "book_id" : 6,
        "title" : "Temperature",
        "author" : "Tejas",
        "type" : "Story"
}
>
```

```
>
> var Mapfunction = function(){emit(this.type,1)}
> var Reducefunction = function(key,values){return Array.sum(values)}
> db.books.mapReduce(Mapfunction,Reducefunction,{'out':'typeofbooks'})
{ "result" : "typeofbooks", "ok" : 1 }
>
> db.typeofbooks.find()
{ "_id" : "Poem", "value" : 2 }
{ "_id" : "songs", "value" : 1 }
{ "_id" : "Real Stories", "value" : 1 }
{ "_id" : "Story", "value" : 2 }
>
```

Assignment No. B4

Problem Statement :-

Write a program to implement MongoDB database connectivity with PHP/PYTHON/JAVA implement database navigation operations using JDBC/ODBC.

Solution :-

Program :

```
package B4;

import com.mongodb.*;

public class MongoDB {
 public static void main( String args[] ) {
        try{
                        //create connection
                        MongoClient mongo = new MongoClient( "localhost" , 27017 );

                        //create database
                        DB db = mongo.getDB( "dbms" );
                        System.out.println("Connect to database successfully");

                        //create collection
                        DBCollection col1=db.getCollection("jdbc");
                        System.out.println("collection created");
```

```
                    //insert document
                    //creating object
                    BasicDBObject doc1 = new BasicDBObject();
                    doc1.put("name", "shraddha");
                    doc1.put("website", "google.com");
                    //creating object
                    BasicDBObject doc2 = new BasicDBObject();
                    doc2.put("addressLine1", "Sweet Home");
                    doc2.put("addressLine2", "Karol Bagh");
                    doc2.put("addressLine3", "New Delhi, India");

                    //inserting objects in collection
                    col1.insert(new BasicDBObject[] {doc1,doc2});



          }catch(Exception e){
                    System.err.println( e.getClass().getName() + ": " + e.getMessage() );
          }
    }
}
```

Output :

```
Console  MongoDB.java
<terminated> MongoDB [Java Application] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (Dec 10, 2021, 1:31:02 PM)
Dec 10, 2021 1:31:02 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Connect to database successfully
collection created
Dec 10, 2021 1:31:02 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Dec 10, 2021 1:31:02 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:9}] to localhost:27017
Dec 10, 2021 1:31:02 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, version=Serv
Dec 10, 2021 1:31:02 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:10}] to localhost:27017
```

```
>
> use dbms
switched to db dbms
>
> show collections
books
employee
employees
jdbc
typeofbooks
>
> db.jdbc.find().pretty()
{
        "_id" : ObjectId("61b3069d1f971860fd31469b"),
        "name" : "shraddha",
        "website" : "google.com"
}
{
        "_id" : ObjectId("61b3069d1f971860fd31469c"),
        "addressLine1" : "Sweet Home",
        "addressLine2" : "Karol Bagh",
        "addressLine3" : "New Delhi, India"
}
>
```

Assignment No. C1

Problem Statement :-

According to DBMS concept covered in Group A and D develop and application using provided guidelines.

Solution :-

Title of the Project :

Society Management System using JAVA and ORACLE with JDBC (Database Connectivity).

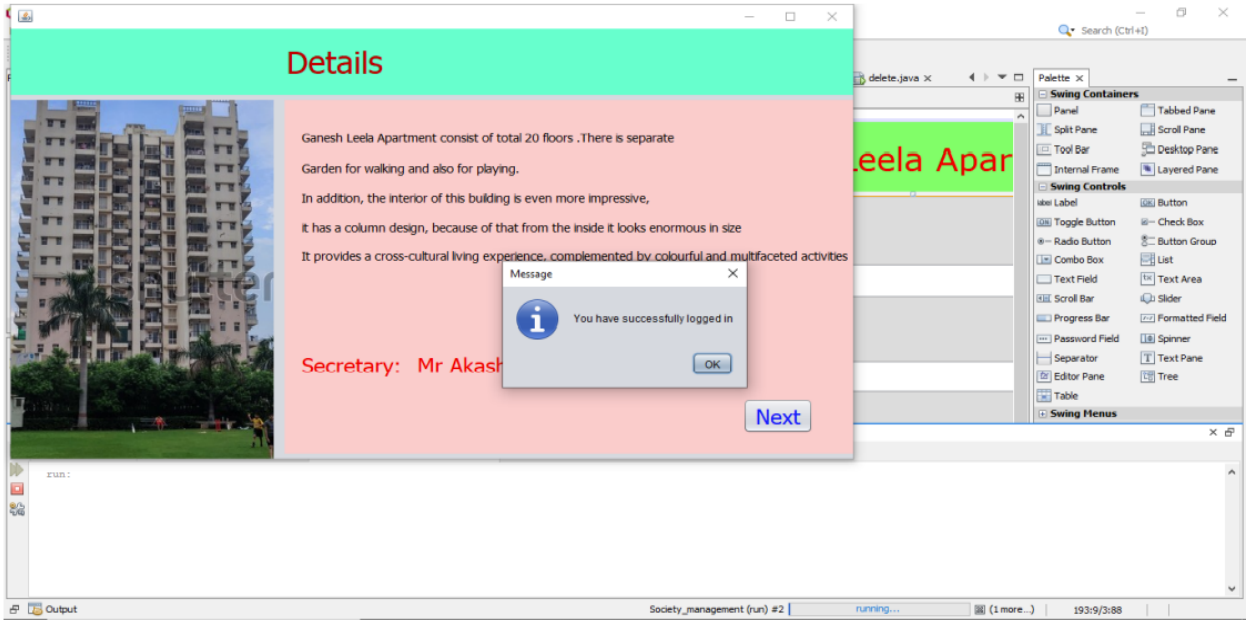
Introduction :

This project presents a solution for housing/residential societies to manage their residents with more ease through a computer-based approach. Any more features depending on the needs of a society can be easily added. The project gives easy access to CRUD operations and makes it easier for the user to maintain records.

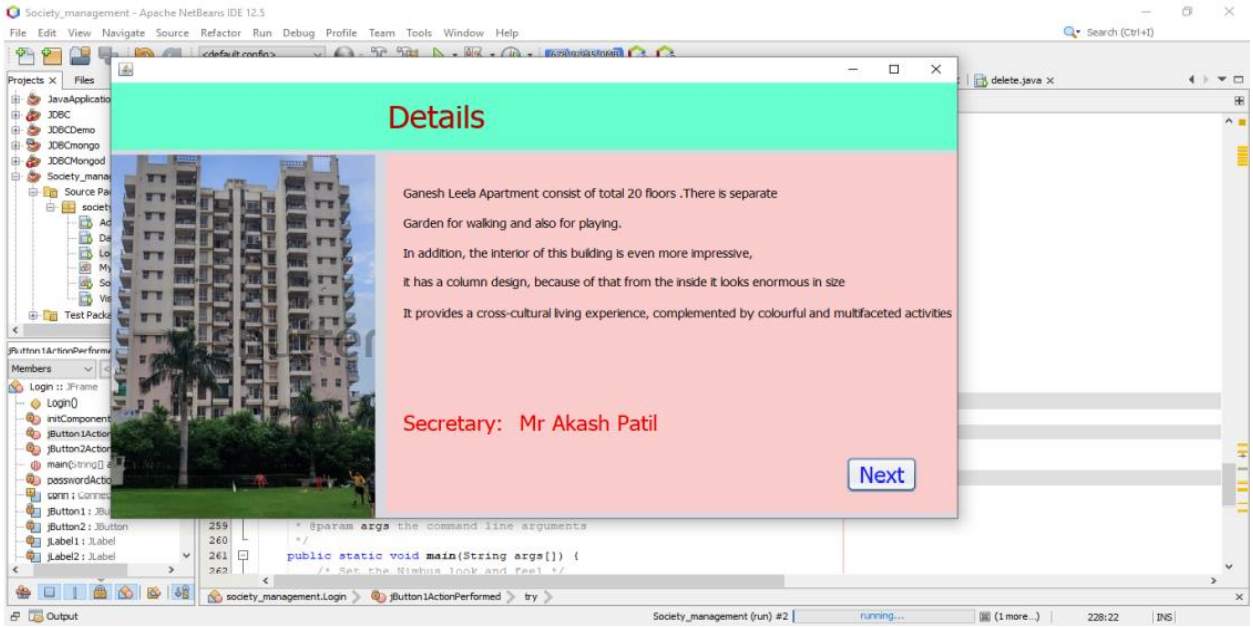
Output :

A) Login Page =>

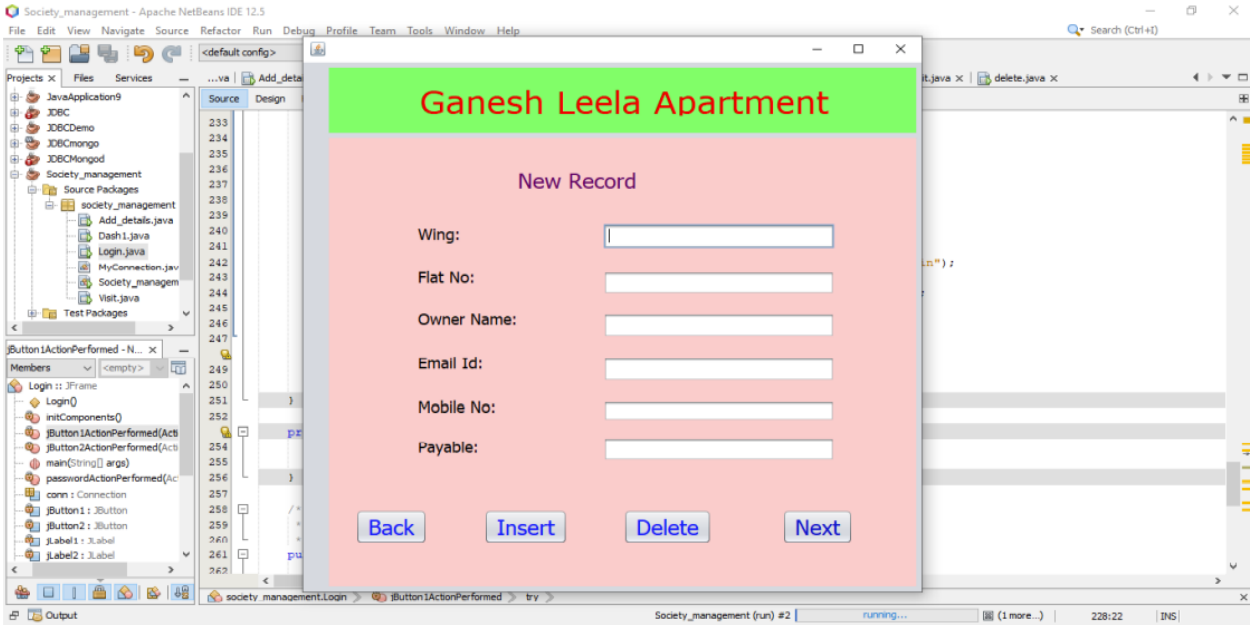
B) Once user has entered correct username and password, user will be successfully logged in =>



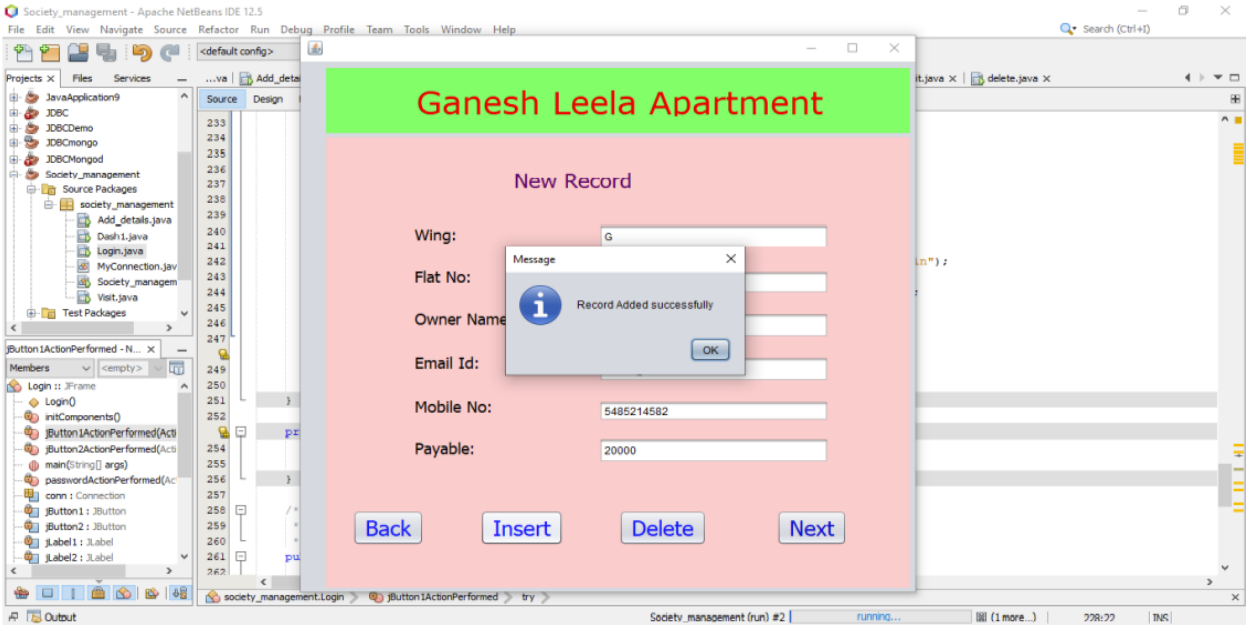
C) Details of Society =>



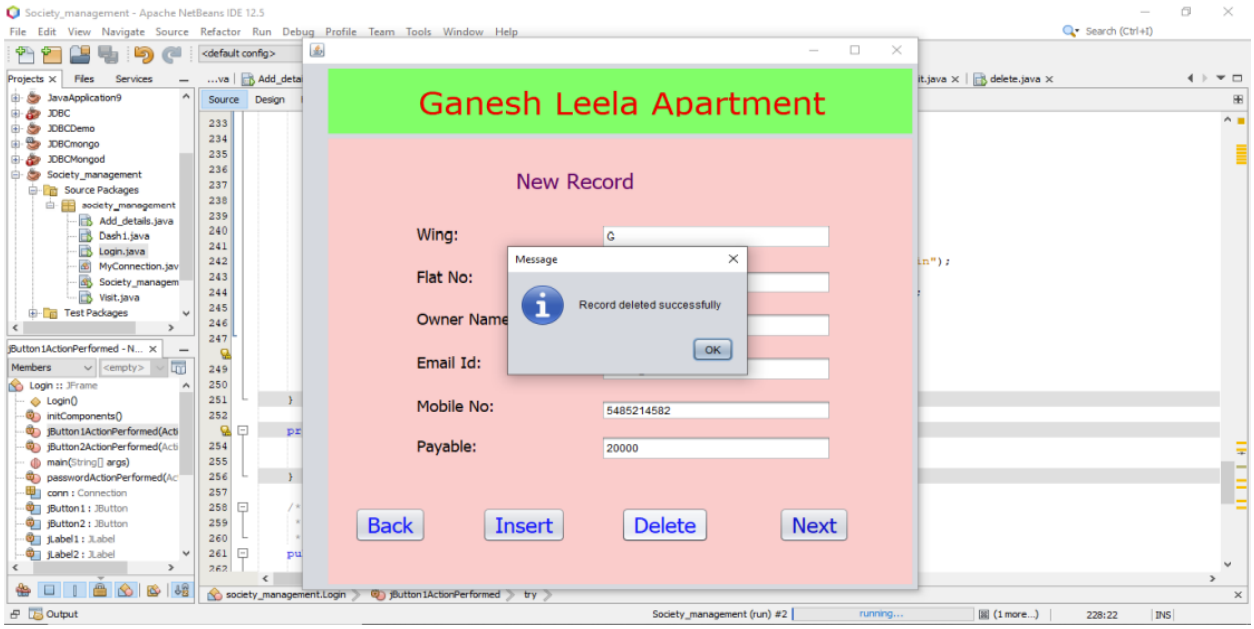
D) User can insert details of new members and delete details of member =>



E) Record inserted successfully =>



F) Record deleted successfully =>



G) Exit =>