

Database management systems laboratory

Lab Journal



NAME :- OJUS P. JAISWAL

YEAR & DIV :- TE A

ROLL NO. :- TACO19108

SEAT NO. :- S191094290

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 1. Design and develop SQL DDL statements which demonstrates use of SQL objects such as Table, View, Index, Sequence, and Synonym. 2. 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 |

**INDEX**

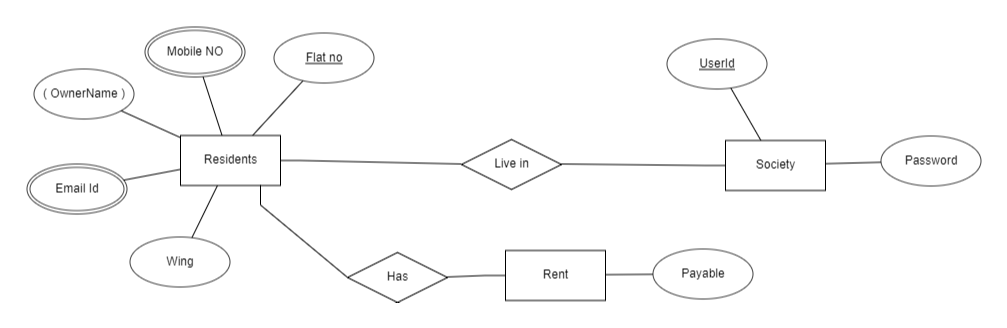
**Assignment No. A1**

**Problem Statement :-**

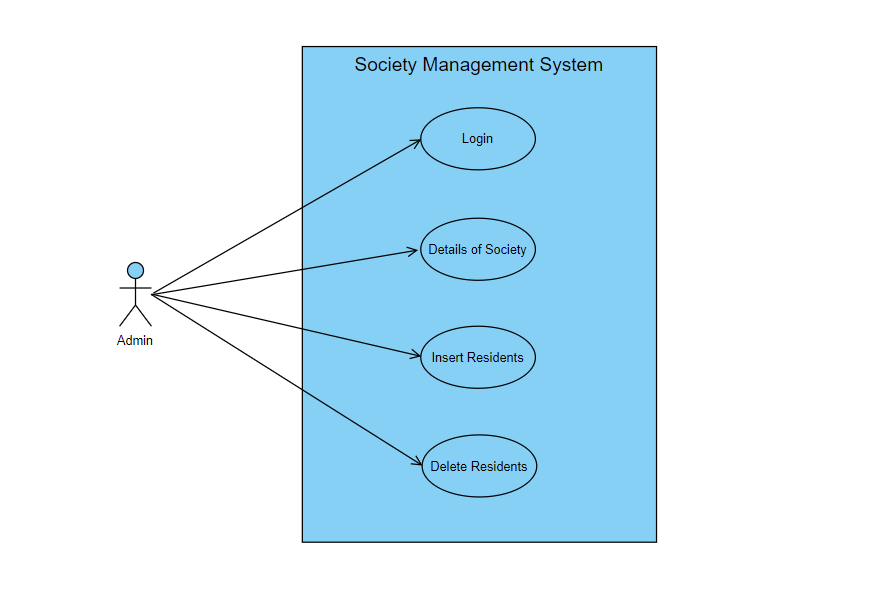
Study and Draw ER Modelling diagram along with normalization using ERD win / ERD plus for selected problem statement.

**Solution :-**

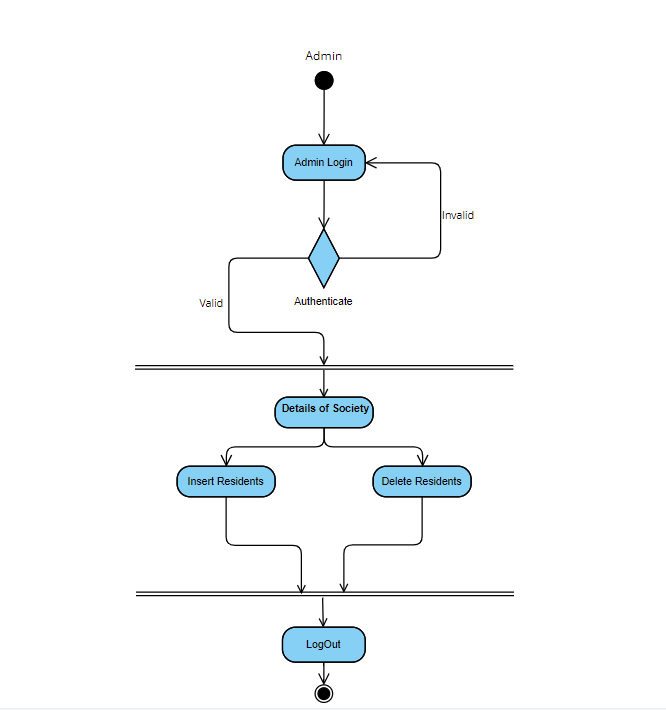
ER Diagram of Society Management System :



Use Case Diagram of Society Management System :



Activity Diagram of Society Management System :



**Assignment No. A2**

1. **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-----

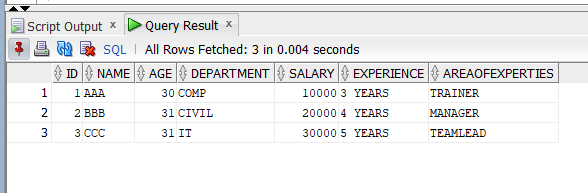
CREATE INDEX INDEX\_ID ON EMPLOYEE(ID);

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

CREATE SYNONYM EMP FOR EMPLOYEE;

SELECT \* FROM EMP;

Output :



1. **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;

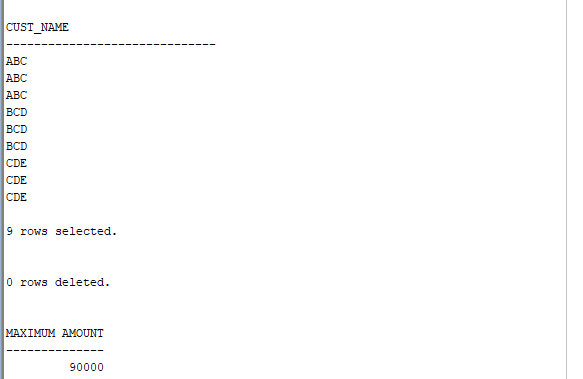
select \* from customer;

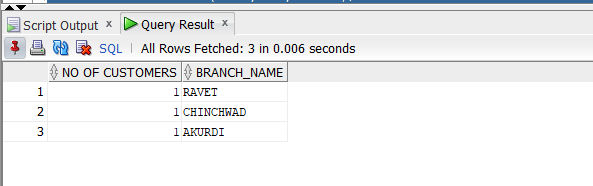
select \* from Depositor;

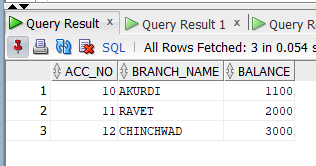
select \* from Loan;

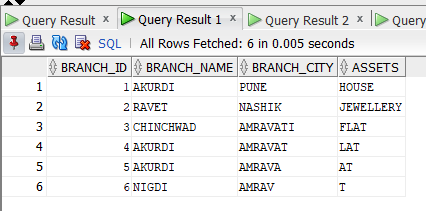
select \* from Borrower;

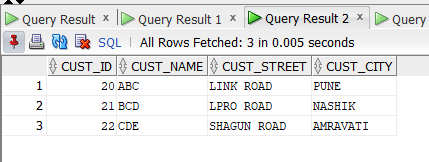
Output :

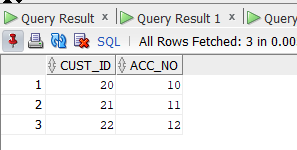


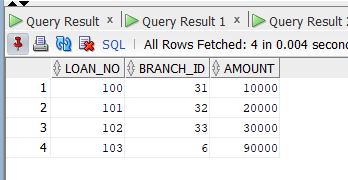


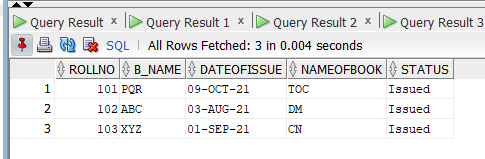












**Assignment No. A3**

**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;

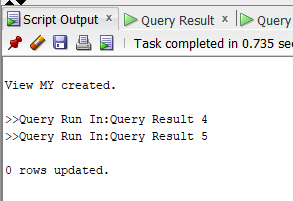
--6

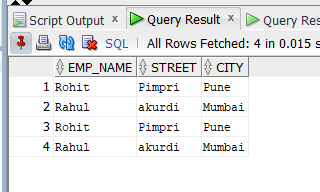
select avg(salary) from work where company='Small Bank Corporation';

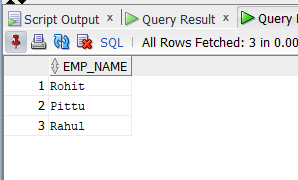
--7

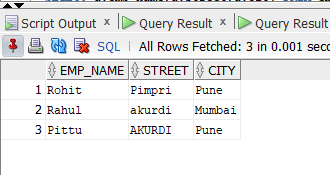
update work SET salary=(1.15\*salary) where company='First Bank Corporation' and salary<20000;

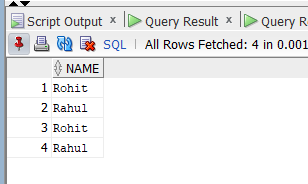
Output :

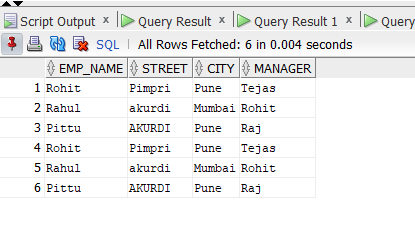


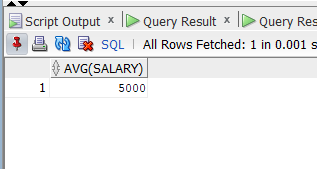












**Assignment No. A4**

**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)

1. Accept roll\_no & name of book from user.
2. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
3. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
4. After submitting the book, status will change from I to R.
5. 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

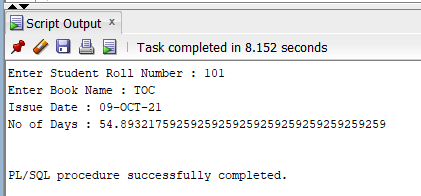
dbms\_output.put\_line(mroll||'Not found');

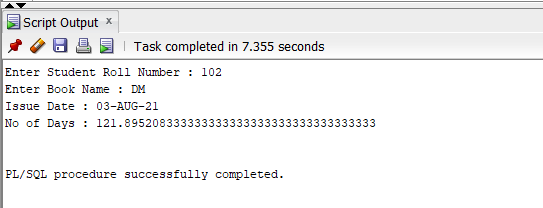
End;

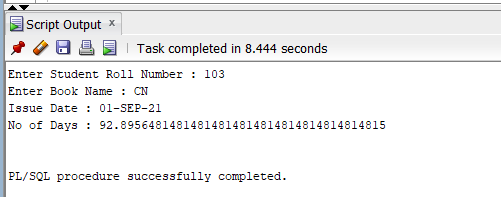
/

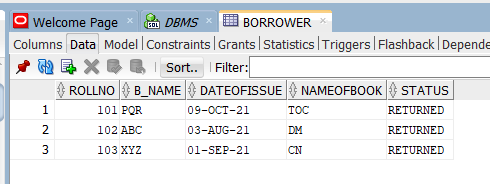
Output :

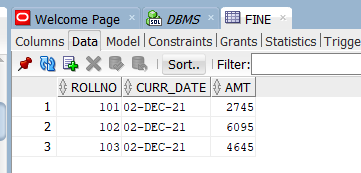
A)



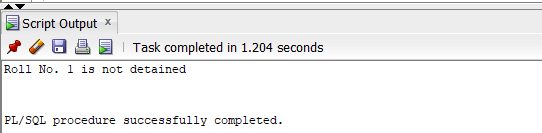


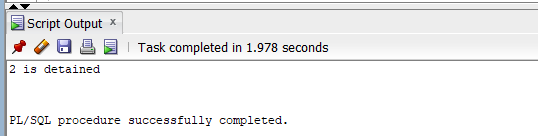


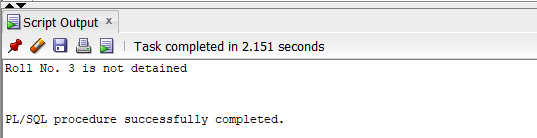


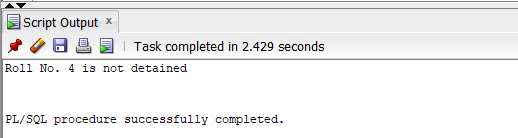


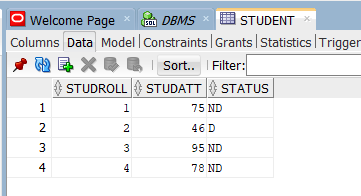
B)











**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;

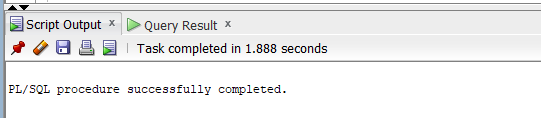
/

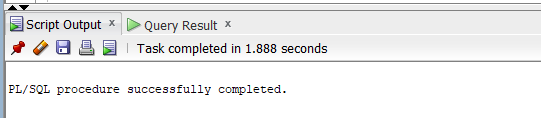
select \* from stud\_marks;

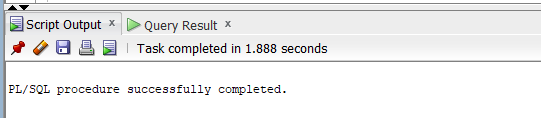
select \* from result;

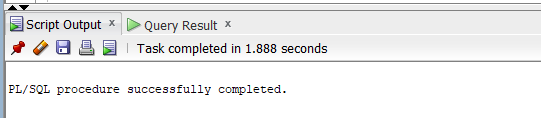
Output :

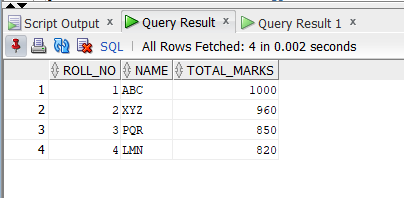
A)

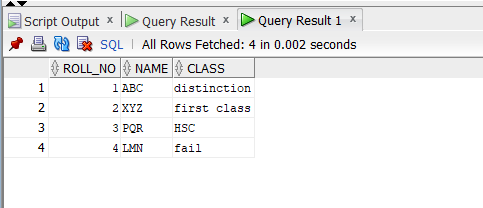




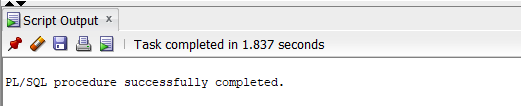


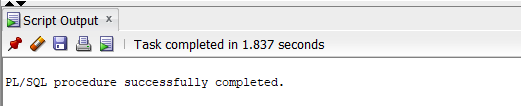


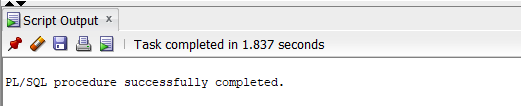


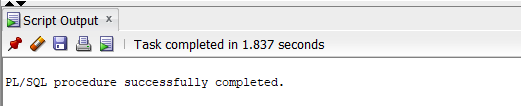


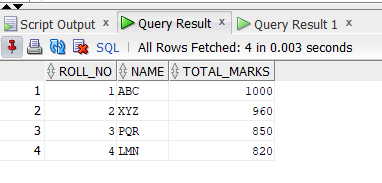
B)

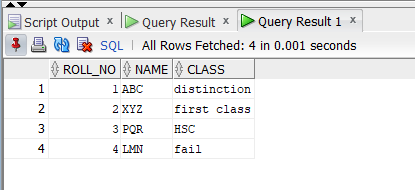












**Assignment No. A7**

**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);

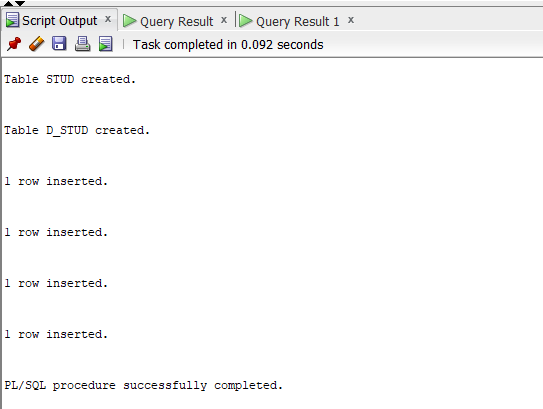
END LOOP;

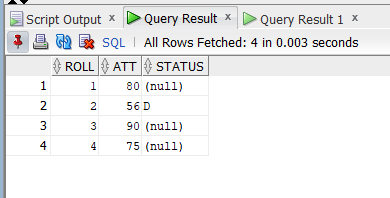
END;

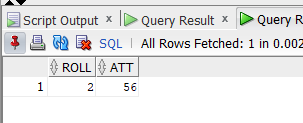
select \* from employees1;

Output :

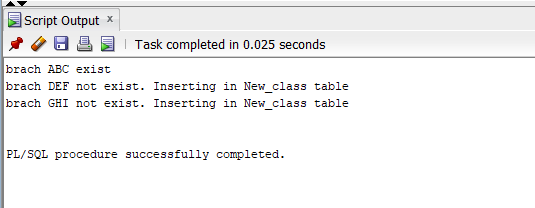
A)

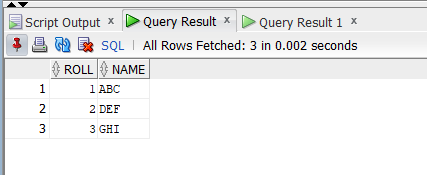


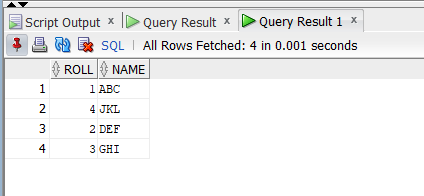




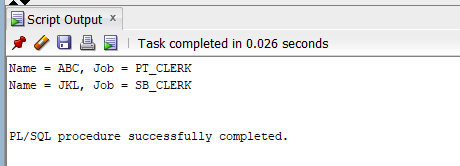
B)

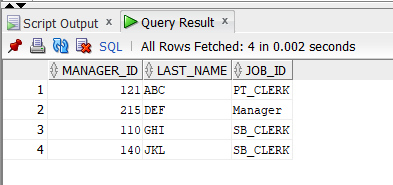






C)





**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;

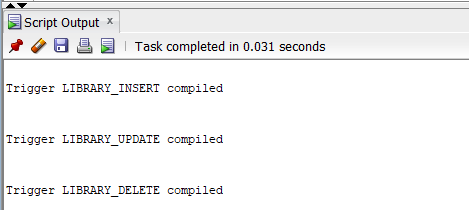
insert into Library values(4,'JKL','26-NOV-2021','Issued');

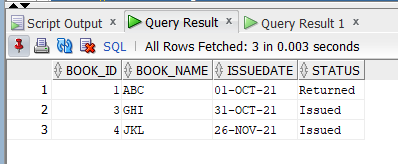
delete from Library where Book\_ID=2;

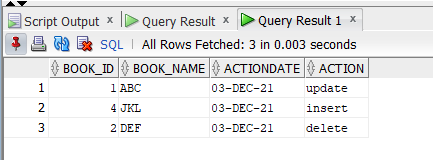
select \* from Library;

select \* from LibraryAudit;

Output :







**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) {

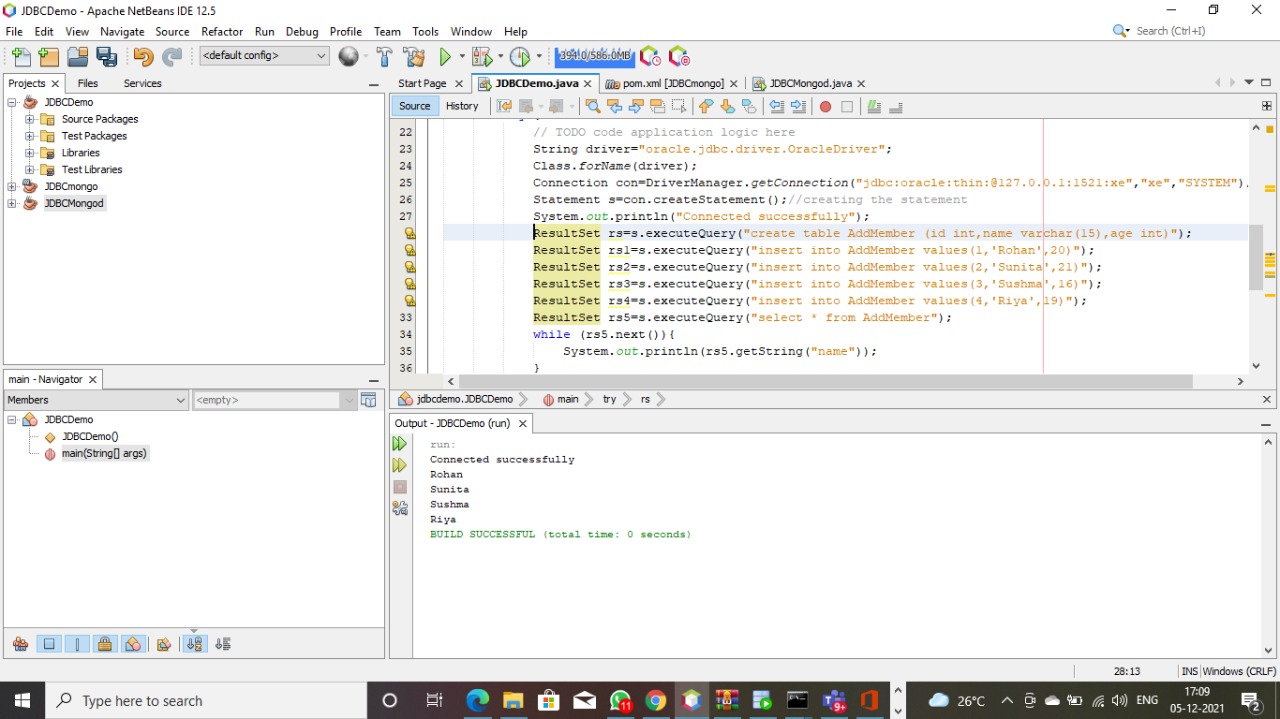
System.out.println("Error:"+ex);

}

}

}

Output :



**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}})

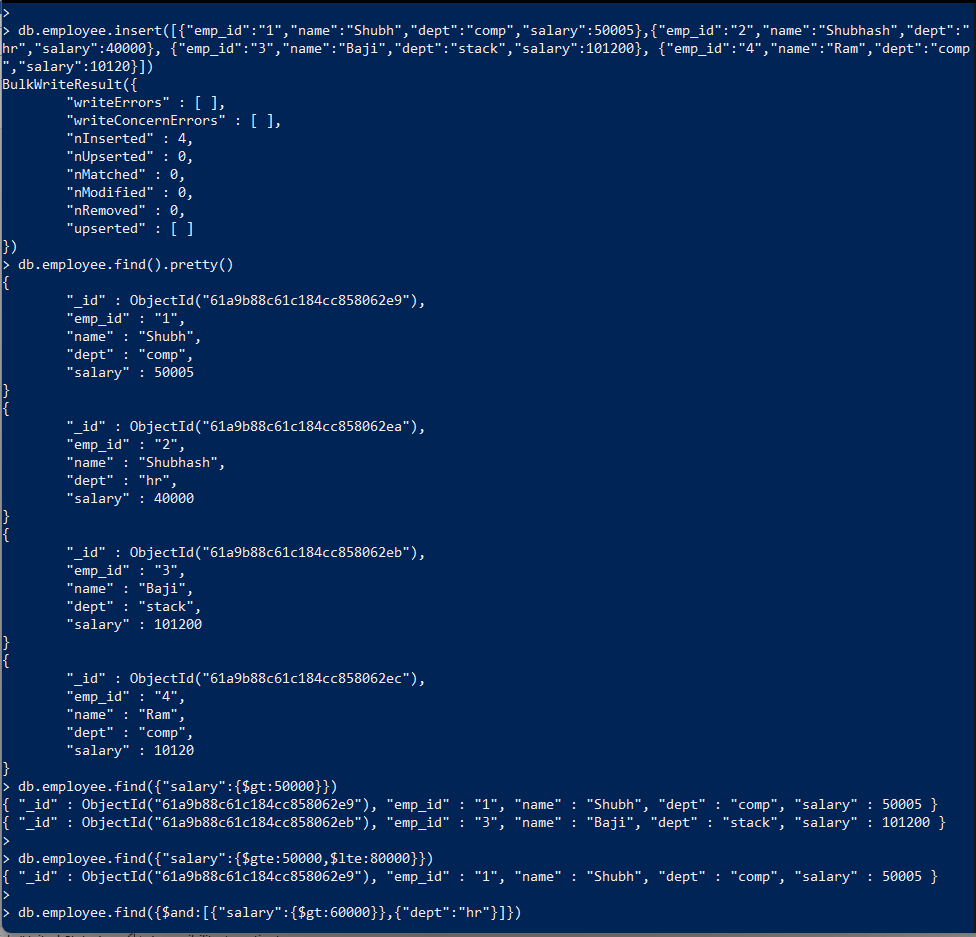
2. db.employee.find({"salary":{$gte:50000,$lte:80000}})

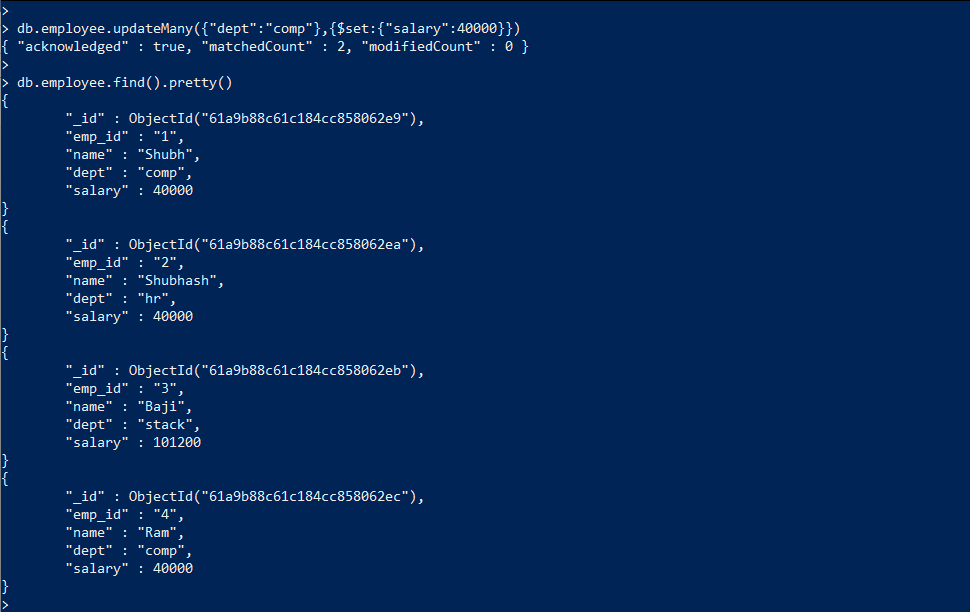
3. db.employee.find({$and:[{"salary":{$gt:60000}},{"dept":"hr"}]})

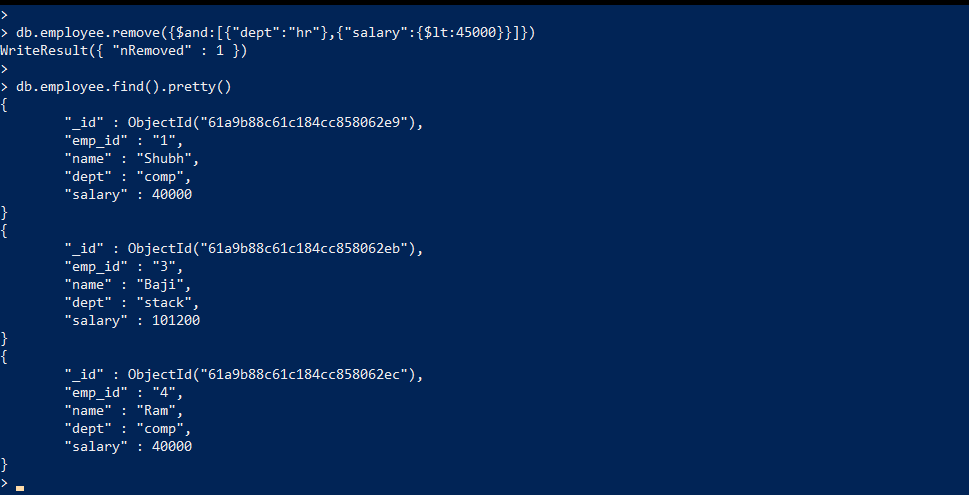
4. db.employee.updateMany({"dept":"comp"},{$set:{"salary":40000}})

5. db.employee.remove({$and:[{"dept":"hr"},{"salary":{$lt:45000}}]})

Output :







**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()

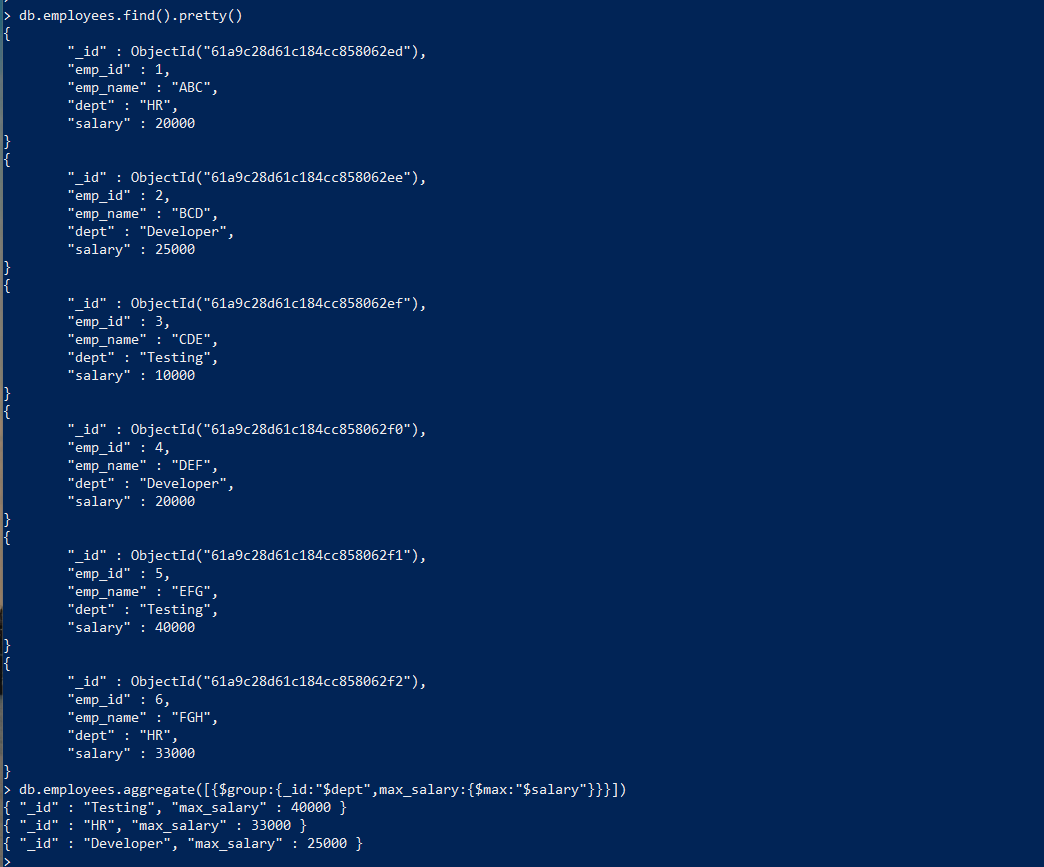
1. db.employees.aggregate([{$group:{\_id:"$dept",max\_salary:{$max:"$salary"}}}])

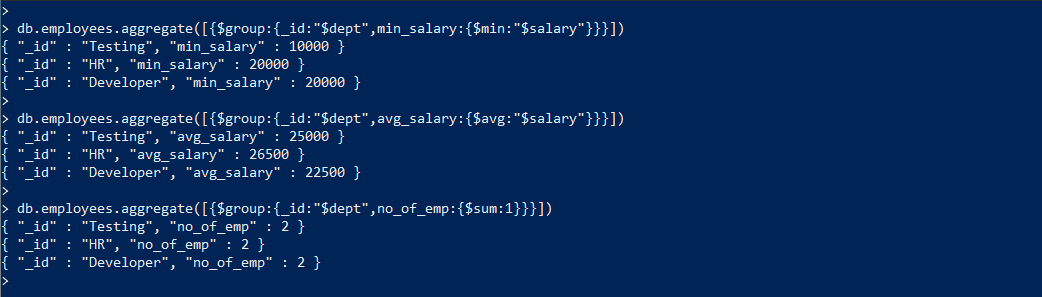
2. db.employees.aggregate([{$group:{\_id:"$dept",min\_salary:{$min:"$salary"}}}])

3. db.employees.aggregate([{$group:{\_id:"$dept",avg\_salary:{$avg:"$salary"}}}])

4. db.employees.aggregate([{$group:{\_id:"$dept",no\_of\_emp:{$sum:1}}}])

Output :





**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"}])

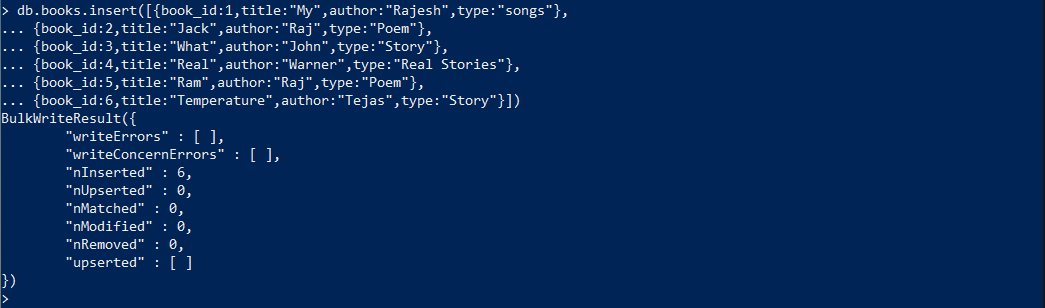
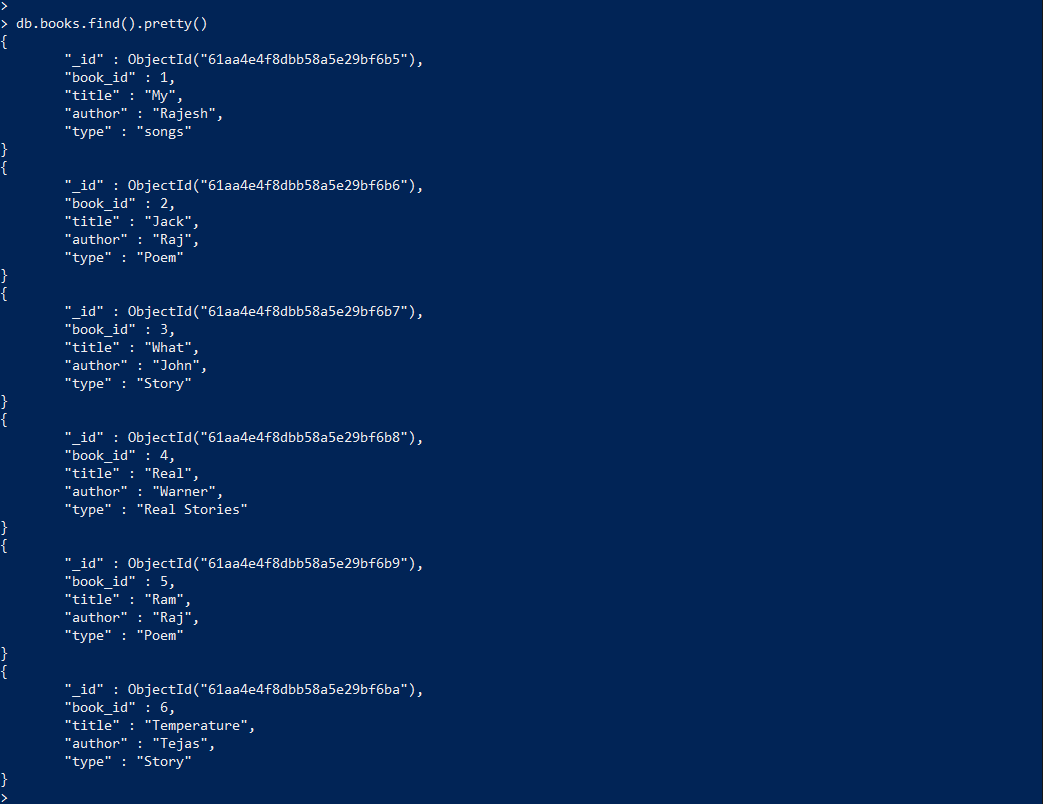
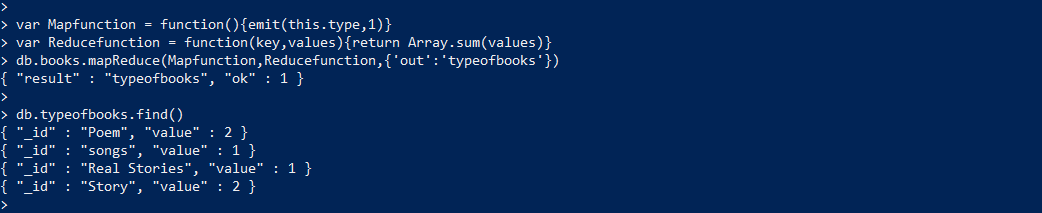
var Mapfunction = function(){emit(this.type,1)}

var Reducefunction = function(key,values){return Array.sum(values)}

db.books.mapReduce(Mapfunction,Reducefunction,{'out':'typeofbooks'})

db.typeofbooks.find()

Output :

**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){

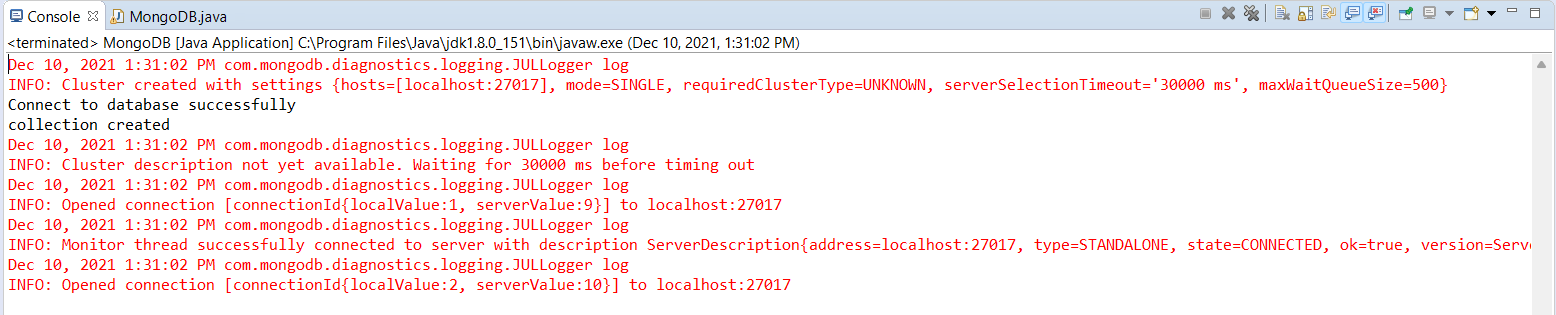
System.err.println( e.getClass().getName() + ": " + e.getMessage() );

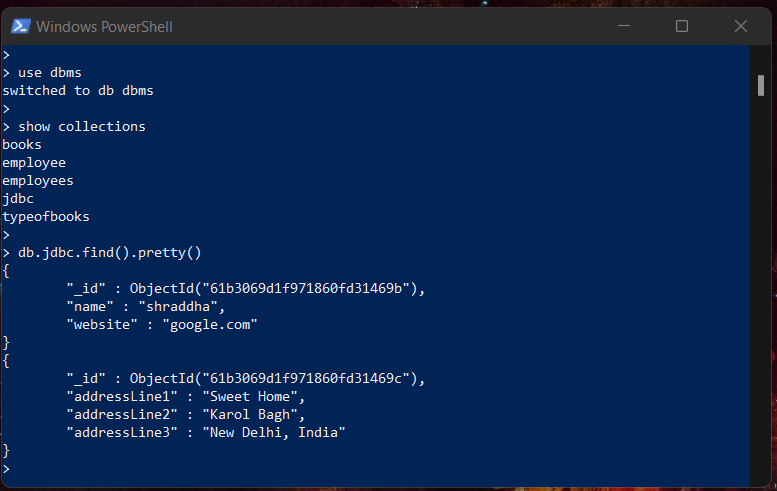
}

}

}

Output :





**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 :

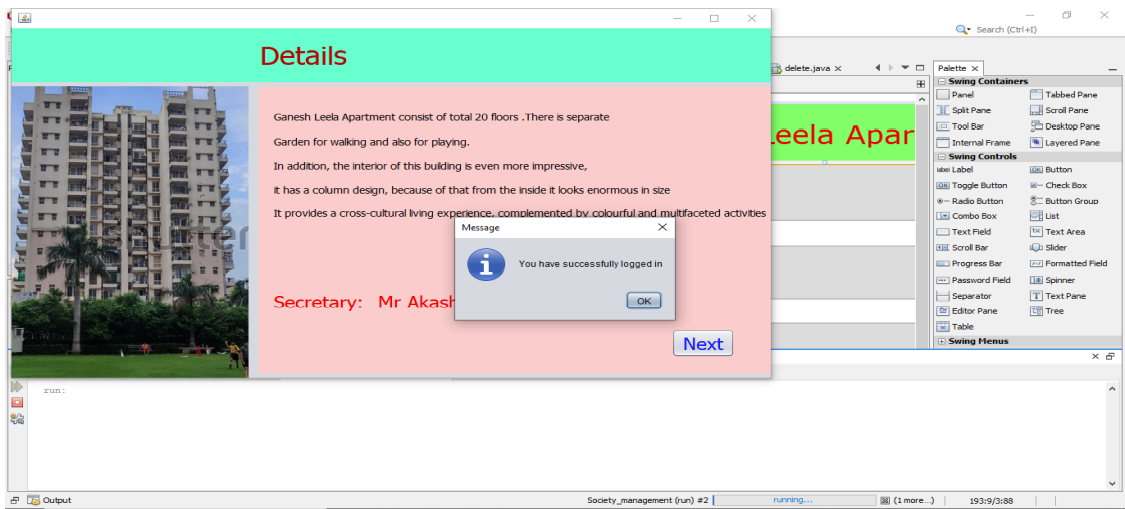
This project presents a solution for housing/residential societies to manage their residents with more ease through a computer-based approach. Any more features depending on the needs of a society can be easily added. The project gives easy access to CRUD operations and makes it easier for the user to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

Output :

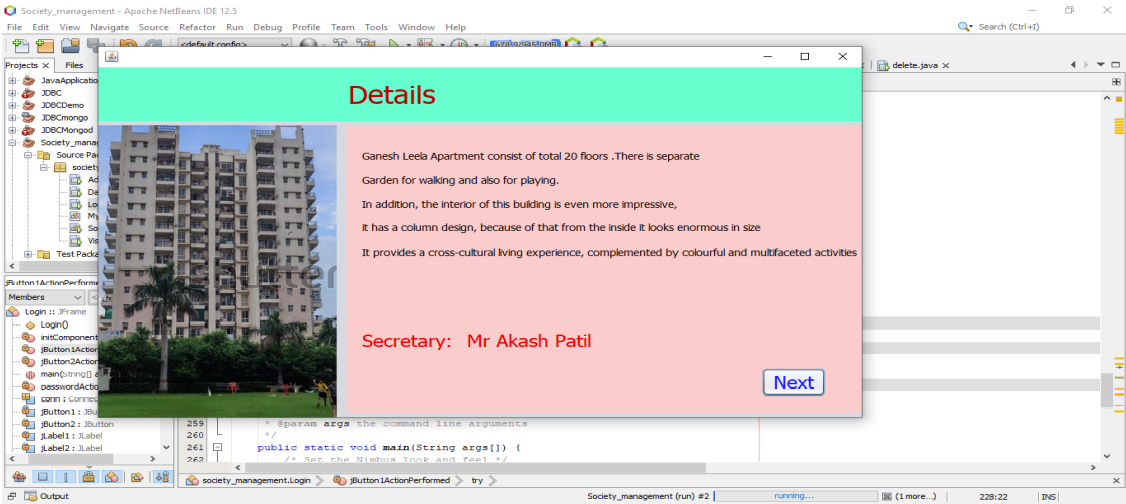
A) Login Page =>



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



C) Details of Society =>



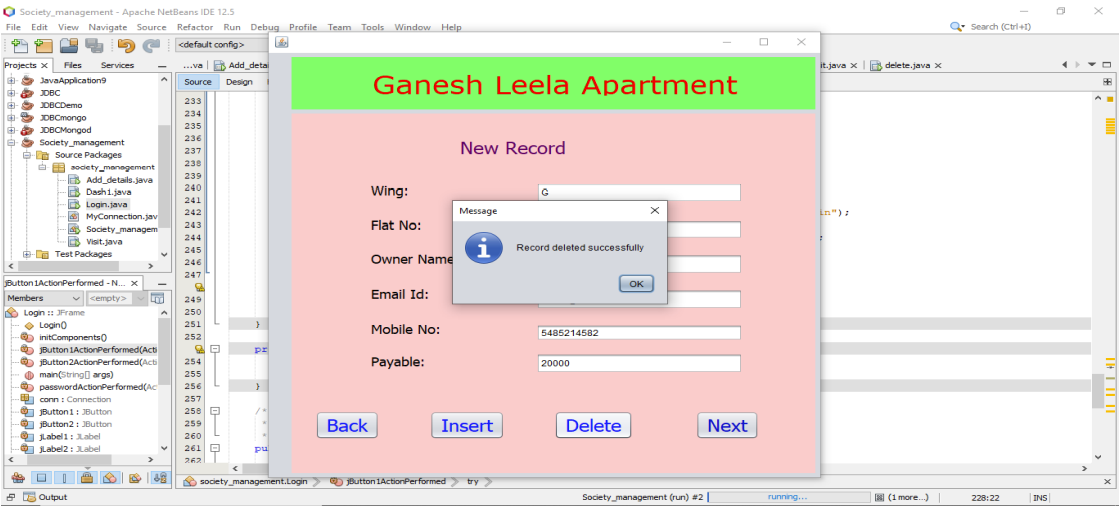
D) User can insert details of new members and delete details of member =>



E) Record inserted successfully =>



F) Record deleted successfully =>



G) Exit =>

