JEPPIAAR INSTITUTE OF TECHNOLOGY

Self-Belief|Self-Discipline|Self-Respect

Kunnam, Sunguvarchatram, Sriperumbudur, Tamilnadu-631 604

www.jeppiaarinstitute.org | 044-2715 9000.

DEPARTMENT OF INFORMATION TECHNOLOGY

DATABASE MANAGEMENT SYSTEMS LAB

(CS3481)

NAME	: ____________________
REG. NUMBER	: ____________________
SEMESTER	: ____________________

JEPPIAAR INSTITUTE OF TECHNOLOGY

Self-Belief | Self-Discipline | Self-Respect

Kunnam, Sunguvarchatram, Sriperumbudur, Tamilnadu-631 604

www.jeppiaarinstitute.org | 044-2715 9000.

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is a certified bonafide record work of Mr./Ms_______________ Reg.No.___________________________ submitted for the anna university practical examination held on _______________ in **CS3481 Database Management System Laboratory as** during the year of 2022-2023.

Signature of the Lab In-charge **Head of the Department**

Internal Examiner **External Examiner**

INSTITUTE VISION :

Jeppiaar Institute of Technology aspires to provide technical education in futuristic technologies with the perspective of innovative, industrial and social application for the betterment of humanity.

INSTITUTE MISSION :

- **M1:** To produce competent and disciplined high-quality professionals with the practical skills necessary to excel as innovative professionals and entrepreneurs for the benefit of the society.
- **M2:** To improve the quality of education through excellence in teaching and learning, research, leadership and by promoting the principles of scientific analysis, and creative thinking.
- **M3:** To provide excellent infrastructure, serene and stimulating environment that is most conducive to learning.
- **M4:** To strive for productive partnership between the Industry and the Institute for research and development in the emerging fields and creating opportunities for employability.
- **M5:** To serve the global community by instilling ethics, values and life skills among the students needed to enrich their lives.

Department Vision

The department will be an excellent centre to impart futuristic and innovative technological education to facilitate the evolution of problem-solving skills along with knowledge application in the field of Information Technology, understanding industrial and global requirements and societal needs for the benefit of humanity.

Department Mission

- **M1:** Produce competent and high-quality professional computing graduates in software development considering global requirements and societal needs thereby maximizing employability.
- **M2:** Enhance evolution of professional skills and development of leadership traits among the students by providing favourable infrastructure and environment to grow into successfulentrepreneurs.
- **M3:** Training in multidisciplinary skills needed by Industries, higher educational institutions, research establishments and Entrepreneurship.
- **M4:** Impart Human Values and Ethical Responsibilities in professional activities.

PEO's OF THE DEPARTMENT

- Provided with a fundamental knowledge in Science, mathematics and computing skills for creative and innovative application.
- Enabled students competent and employable by providing excellent Infrastructure to learn and contribute for the welfare of the society.
- To channelize the potentials of the students by offering state of the art amenities to undergo research and higher education.
- To evolve computing engineers with multi-disciplinary understanding and maximize Job Opportunities.
- To facilitate students, obtain profound understanding nature and social requirements and grow as professionals with values and integrity

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

- **Engineering knowledge:** Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **Design/development of solutions:** Design solutions for complex engineering problemsand design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, andenvironmental considerations.
- **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **Individual and team work:** Function effectively as an individual, and as a member orleader in diverse teams, and in multidisciplinary settings.

LIST OF EXPERIMENTS

EX.NO.	DATE	NAME OF THE EXPERIMENT	PAGE NO	SIGN

EX.NO:1 IMPLEMENTATION OF DDL COMMANDS

AIM:

To execute and verify the Data Definition Language commands.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Execute different Commands and extract information from

the table.

STEP 4: Stop

DDL COMMANDS:

1. The Create Table Command: - It defines each column of the table uniquely. Each column hasminimum of three attributes, a name, data type and size.

Syntax:

Create table <table name> (<col1> <datatype>(<size>),<col2> <datatype>(<size>));

Ex:create table emp(empno number(4) primary key, ename char(10));

Table created.

2. Modifying the structure of tables.

a) Add new columns

Syntax:

Alter table <tablename> add(<new col><datatype>(size),<new col><datatype>(size));

Ex: alter table emp add(sal number(7,2));

Table altered.

```
SQL>      desc
emp

 Name                          Null?   Type

 -------------------------------------------- --------- --------------------------------


 EMPNO                           NOT              NULL
```

```
NUMBER(4)ENAME              CHAR(10)

SAL                         NUMBER(7,2)
```

3. Dropping a column from a table.

Syntax:

Alter table <tablename> drop column <col>;

Ex : alter table emp drop column sal;

```
Table altered.
SQL>    desc
emp;

Name                            Null?           Type
----------------------------------------- --------- -------------------------------

EMPNO                           NOTNULL     NUMBER(4)
ENAME                                   CHAR(10)
```

4. Modifying existing columns.

Syntax:

Alter table <tablename> modify(<col><newdatatype>(<newsize>));

Ex:alter table emp modify(ename varchar2(15));

```
Table altered.

SQL>    desc
emp

Name                            Null?   Type
----------------------------------------- --------- -------------------------------

EMPNO                           NOT             NULL
NUMBER(4)ENAME                          VARCHAR2(15)
```

5. Renaming the tables

Syntax:

Rename <oldtable> to <new

table>;Ex: rename emp to emp1;

Table renamed.

SQL> desc emp1

```
Name                              Null?   Type
--------------------------------------------- --------- --------------------------------

EMPNO                              NOT     NULL     NUMBER(4)
ENAME                                      VARCHAR2(15)
```

6. Truncating the tables.

Syntax:
Truncate table <tablename>;

Ex: truncate table emp1;

Table truncated.

SQL> desc emp1

```
Name                              Null?   Type
--------------------------------------------- --------- --------------------------------

EMPNO                              NOT NULL NUMBER(4)
ENAME                                      VARCHAR2(15)
```

7. Destroying tables.

Syntax:

Drop table <tablename>;

Ex: drop table emp1;

Table dropped.

SQL> desc emp1

ERROR:
ORA-04043: object emp1 does not exist

CONSTRAINTS:

Create table tablename (column_name1 data_ type constraints, column_name2 data_ typeconstraints …)

Example:

Create table stud1(sname varchar2(20) not null, rollno number(10) not null,dob date not null);

DOMAIN INTEGRITY

Example: Create table cust(custid number(6) not null, name char(10));

Alter table cust modify (name not null);

ENTITY INTEGRITY

Primary Key Constraint:
Example: Create table stud2(regno number(6) primary key, name char(20));

RESULT:

Thus the DDL commands have been executed successfully.

EX.NO:2 IMPLEMENTATION OF DML COMMANDS

AIM:

To execute and verify the DML and commands.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Insert the record into table

STEP 4: Update the existing records into the table

STEP 5: Delete the records in to the table

STEP 6: Stop

DML COMMANDS

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are Insert, Select, Update, Delete.

Insert Command: This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.

Select Commands: It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.

Update Command: It is used to alter the column values in a table. A single column may be updatedor more than one column could be updated.

Delete command: After inserting row in a table we can also delete them if required. The deletecommand consists of a from clause followed by an optional where clause.

Q1: Insert a single record into dept table.

```
SQL> insert into dept values (1,'IT','Tholudur');

1 row created.
SQL> create table emp(empno number(6) primary key,ename varchar2(20),job
varchar2(13),deptnonumber(3),sal number(7,2));

Table created.
```

Q2: Insert more than a record into emp table using a single insert command.

```
SQL> insert into emp values(&empno,'&ename','&job',&deptno,&sal);

Enter value for empno: 1


Enter value for ename:

Mathi Enter value for job:

AP

Enter value for deptno:

1 Enter value for sal:

10000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(1,'Mathi','AP',1,10000)

1 row created.

SQL> / Enter value for empno: 2

Enter value for ename: Arjun

Enter value for job: ASP

Enter value for deptno:

2 Enter value for sal:

12000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(2,'Arjun','ASP',2,12000)

1 row created.
SQL> / Enter value for

empno: 3 Enter value for

ename: Gugan Enter value for

job: ASP

Enter value for deptno: 1

Enter value for sal: 12000

old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)

new 1: insert into emp values(3,'Gugan','ASP',1,12000)
```

1 row created.

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working asASP

```
SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL

----------- ---------------------- --------------- ----------- -----------

1 Mathi AP 1 10000

2 Arjun ASP 2 12000

3 Gugan ASP 1 12000

SQL> update emp set sal=15000 where
job='ASP';2 rows updated.

SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL

----------- ---------------------- --------------- ----------- -----------


1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gugan ASP 1 15000

SQL> insert into emp values(&empno,'&ename','&job',&deptno,&sal);
Enter value for empno: 4
Enter value for ename:
Karthik Enter value for job:
Prof
Enter value for deptno:
2 Enter value for sal:
30000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(4,'Karthik','Prof',2,30000)

1 row

created.

SQL> /
Enter value for empno: 5
Enter value for ename:
Akalya Enter value for job:
AP
Enter value for deptno: 1
Enter value for sal: 10000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
```

```
new  1: insert into emp values(5,'Akalya','AP',1,10000)

1           row
created.
SQL> /
Enter  value  for  empno:  6
Enter   value   for   ename:
suresh Enter  value  for  job:
lect
Enter value for deptno:
1 Enter  value  for  sal:
8000
old              1:              insert              into              emp
values(&empno,'&ename','&job',&deptno,&sal)  new   1:  insert
into emp values(6,'suresh','lect',1,8000)

1 row created.

SQL> select * from emp;

    EMPNO ENAME      JOB        DEPTNO     SAL
------- ---------------------- --------------- -----------       -----------

        1       Mathi            AP          1          10000
        2       Arjun            ASP         2          15000
        3       Gugan            ASP         1          15000
        4       Karthik          Prof        2          30000
        5       Akalya           AP          1          10000
        6       suresh           lect        1          8000

6 rows selected.
```

Q4: Create a pseudo table employee with the same structure as the table emp and insert rows into thetable using select clauses.

```
SQL> create table employee as select * from

emp;Table created.

SQL>            desc

employee;      Name
--------------------------------------------- --------- --------------------------------

Null? Type

EMPNO NUMBER(6)
ENAME           NOT           NULL
VARCHAR2(20) JOB  NOT  NULL
VARCHAR2(13)              DEPTNO
NUMBER(3)
SAL NUMBER(7,2)
```

Q5: select employee name, job from the emp table

```
SQL> select ename, job from emp;
ENAME    JOB
-------------  ---------------

Mathi       AP
Arjun      ASP
Gugan      ASP
Karthik    Prof
Akalya     AP
suresh     lect
6          rows
selected.
```

Q6: Delete only those who are working as lecturer

```
SQL> select * from emp;
EMPNO   ENAME  JOB    DEPTNO    SAL
-------------------------    ---------  ---------
1         Mathi      AP          1          10000
2         Arjun      ASP        2           15000
3        Gugan      ASP        1            15000
4         Karthik    Prof        2            30000
5         Akalya   AP          1            10000
6        suresh    lect          1            8000
6 rows selected.

SQL>    delete    from    emp    where
job='lect';1 row deleted.

SQL> select * from emp;
EMPNO    ENAME   JOB    DEPTNO  SAL
-------------------  -------------------------
1            Mathi      AP         1      10000
2            Arjun      ASP        2      15000
3            Gugan      ASP       1      15000
4            Karthik    Prof       2      30000
5            Akalya     AP        1      10000
```

Q7: List the records in the emp table orderby salary in ascending order.

```
SQL> select * from emp order by sal;
EMPNO   ENAME   JOB   DEPTNO    SAL
------------------      ------    --------------------
1          Mathi          AP       1        10000
5         Akalya         AP       1        10000
2          Arjun          ASP      2        15000
3         Gugan          ASP    1          15000
4         Karthik         Prof     2         30000
```

Q8: List the records in the emp table orderby salary in descending order.

```
SQL> select * from emp order by sal desc;

EMPNO  ENAME   JOB   DEPTNO  SAL
------------------------  --------  ---------
4       Karthik     Prof      2           30000
2       Arjun       ASP       2           15000
3       Gugan       ASP       1           15000
1       Mathi        AP       1            10000
5       Akalya       AP       1            10000
```

Q9: Display only those employees whose deptno is 1.

```
SQL> select * from emp where deptno=1;
EMPNO ENAME  JOB  DEPTNO  SAL
----------- -------- -------------- ----------- -----------


1       Mathi        AP         1 10000
3        Gugan      ASP       1 15000
5        Akalya     AP         1 10000
```

Q10: Display deptno from the table employee avoiding the duplicated values.

```
SQL> select distinct deptno from emp;
DEPTNO
-----------

1
2
```

IMPLEMENTATION OF DATA AND BUILT IN FUNCTIONS IN SQL

CHARACTER/STRING FUNCTION:

```
SQL> select upper('hai') from dual;

UPP
---
HA
I

SQL>  select  lower('HAI')  from

dual;LOW
---
hai

SQL>   select   initcap(‘hello   world')   from   dual;

INITCAP('Hello’)
---------------
Hello World

SQL> select ltrim('    hai') from dual;
```

```
LTR
---
hai

SQL> select rtrim('hai     ')from   dual;

RTR
---
hai

SQL> select rtrim('    hai     ')from       dual;

RTRIM('
--------
hai

SQL> select concat('SRM',' university')from dual;
---------------------------

SRM university

SQL>  select  length('SRM')from  dual;

LENGTH('SRM')
-------------------------
                12

SQL> select replace('SRM university', 'SRM','Anna')from dual;
------------------

Anna university

SQL> select substr('SRM', 7,6)from dual;

SUBSTR
-------

lingam

SQL>   select   rpad('hai',3,'*')from   dual;
RPAD('
-------

hai***
SQL>    select    lpad('hai',3,'*')from    dual;

LPAD('
-------
***hai

SQL>   select   replace('Dany','y','ie')from   dual;

REPLACE
--------
Danie
```

```
SQL> select translate('cold','ld','ol')from dual;

TRANSL
-------
Cool
```

DATE & TIME FUNCTION

```
SQL> select sysdate from dual;

SYSDATE
----------
07-APR-10

SQL> select round(sysdate)from dual;

ROUND(SYS
----------
07-APR-10

SQL> select add_months(sysdate,3)from dual;

ADD_MONTH
----------
07-JUL-10

SQL> select last_day(sysdate)from dual;

LAST_DAY(
----------
30-APR-10

SQL> select sysdate+20 from dual;

SYSDATE+2

----------
27-APR-10

SQL> select next_day(sysdate,'tuesday')from dual;

NEXT_DAY(
----------
13-APR-10
```

NUMERIC FUNCTION

```
SQL> select round(15.6789) from dual;

ROUND(15.6789)
----------------
          16
```

```
SQL>   select   ceil(23.20)from   dual;

CEIL(23.20)
-----------
        24

SQL>      select      floor(34.56)from      dual;

FLOOR(34.56)
------------
         34

SQL>   select   trunc(15.56743)from   dual;

TRUNC(15.56743)
---------------
           15

SQL>   select   sign(-345)from   dual;

SIGN(-345)
----------
       -1

SQL>   select   abs(-70)from   dual;

 ABS(-70)
---------
      70
```

MATH FUNCTION:

```
SQL>  select  abs(45)  from  dual;

  ABS(45)
---------
      45

SQL>   select   power(10,12)   from   dual;

POWER(10,12)
------------
   1.000E+12

SQL>   select   mod(11,5)   from   dual;
MOD(11,5)
---------

       1
SQL>   select   exp(10)   from   dual;

  EXP(10)
---------
22026.466

SQL>   select   sqrt(225)   from   dual;
```

```
SQRT(225)
    15
```

SET OPERATORS

QUERIES:

```
SQL> create table dept(dno number(10),dname varchar(10),loc varchar(10));
Table created.
SQL> insert into dept values(10,'inventory','hyd');
1 row created.
SQL> insert into dept values(20,'finance','bglr');
1 row created.
SQL> insert into dept values(30,'HR','mumbai');
1 row created.
SQL> select * from dept;
    DNO DNAME LOC
   -------- ----------- -----------
     10 inventory hyd
     20 finance   bglr
     30 HR       mumbai
```

Q1: Display all the dept numbers available with the dept and emp tables avoiding duplicates.

Solution:

```
SQL> select deptno from emp union select deptno from dept;
DEPTNO
-----------
1
2
12
30
40
```

Q2: Display all the dept numbers available with the dept and emp tables. SQL> select deptno from emp union all select deptno from dept;

```
DEPTNO

1
2
2
1
12
1
2
30
40

9 rows selected.
```

Q3: Display all the dept numbers available in emp and not in dept tables and vice versa.SQL> select deptno from emp minus select deptno from dept;

```
DEPTNO
-----------

12
```

SQL> select deptno from dept minus select deptno from emp;

```
DEPTNO
-----------
30
40
```

RESULT

Thus the DML commands was performed successfully and executed.

EX.NO:3 IMPLEMENTATION OF TCL COMMANDS

AIM:

To execute and verify the TCL and commands.

PROCEDURE

STEP 1: Start
STEP 2: Create the table with its essential attributes.
STEP 3: Insert the record into table
STEP 4: Update the existing records into the table
STEP 5: Delete the records in to the table
STEP 6: use save point if any changes occur in any portion of the record to undo its original state.
STEP 7: use rollback for completely undo the records
STEP 8:use commit for permanently save the records.

TCL COMMANDS:

COMMIT: command is used to save the

Records. ROLL BACK: command is used to

undo the Records.

SAVE POINT command is used to undo the Records in a particular transaction.

Queries:

Tables Used: Consider the following tables namely "DEPARTMENTS" and"EMPLOYEES"

Their schemas are as follows , Departments (dept _no , dept_ name , dept_location); Employees (emp_id , emp_name , emp_salary);

Q1: Develop a query to grant all privileges of employees table into departments table

```
SQL> Grant all on employees to departments;
Grant succeeded.
```

Q2: Develop a query to grant some privileges of employees table into departments table

```
SQL> Grant select, update , insert on departments to departments with grant option;
Grant succeeded.
```

Q3: Develop a query to revoke all privileges of employees table from departments table

```
SQL> Revoke all on employees from departments;
Revoke succeeded.
```

Q4: Develop a query to revoke some privileges of employees table from departments table

```
SQL> Revoke select, update , insert on departments from departments;
Revoke succeeded.
```

Q5: Write a query to implement the save point

```
SQL> SAVEPOINT S1;

Savepoint created.

SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL
----------- ---------------------- --------------- ----------- -----------


1 Mathi AP 1 10000

2 Arjun ASP 2 15000

3 Gugan ASP 1 15000

4 Karthik Prof 2 30000

SQL> INSERT INTO EMP VALUES(5,'Akalya','AP',1,10000);

1 row created.

SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL
----------- ---------------------- --------------- ----------- -----------

1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
5 Akalya AP 1 10000
```

Q6: Write a query to implement the rollback

```
SQL> rollback s1;
SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL
 ----------- ---------------------- --------------- ----------- -----------

 1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
```

Q6: Write a query to implement the commit

```
 SQL> COMMIT;
Commit complete.
```

RESULT

Thus the TCL commands was performed successfully and executed.

EX.NO:4 IMPLEMENTATION OF NESTED QUERIES AND JOIN QUERIES

AIM

To execute and verify the SQL commands for Nested &join Queries.

PROCEDURE

STEP 1: Start
STEP 2: Create two different tables with its essential attributes.
STEP 3: Insert attribute values into the table.
STEP 4: Create the Nested &join query from the above created table.
STEP 5: Execute Command and extract information from the tables.
STEP 6: Stop

NESTED QUERIES:

Q1: Display all employee names and salary whose salary is greater than minimum salary of thecompany and job title starts with ‗M‘.

Solution:

```
SQL> select ename,sal from emp where sal > (select min(sal) from emp where job like

'A%');ENAME SAL
---------------------- ------------
Arjun 12000

Gugan 20000

Karthik 15000
```

Q2: Issue a query to find all the employees who work in the same job as Arjun.

```
SQL> select ename from emp where job = (select job from emp where
ename='Arjun');ENAME

 ----------------

 Arjun

Gugan
```

Q3: Issue a query to display information about employees who earn more than any employeein dept 1.

```
 SQL> select * from emp where sal > (select max(sal) from emp where empno=1);
```

```
EMPNO ENAME JOB DEPTNO SAL
----------- ---------------------- ----------- ----------- -----------

2 Arjun ASP 2 12000
3 Gugan ASP 2 20000
4 Karthik AP 1 15000
```

JOIN QUERIES:

INNER JOIN/ NATURAL JOIN/ JOIN: It is a binary operation that allows us to combine certainselections and a Cartesian product into one operation.

OUTER JOIN: It is an extension of join operation to deal with missing information.

Left Outer Join: It takes tuples in the left relation that did not match with any tuple in the right relation, pads the tuples with null values for all other attributes from the right relation and adds them to the result of the natural join.

Right Outer Join: It takes tuples in the right relation that did not match with any tuple in the left relation, pads the tuples with null values for all other attributes from the left relation and adds them to the result of the natural join.

Full Outer Join: It combines tuples from both the left and the right relation and pads the tuples with null values for the missing attributes and hem to the result of the natural join.

Creating Dept table:

```
SQL> create table dept(dno number(10),dname varchar(10),loc varchar(10));

Table created.

SQL> insert into dept values(10,'inventory','hyd');

1 row created.

SQL> insert into dept values(20,'finance','bglr');

1 row created.

SQL> insert into dept values(30,'HR','mumbai');

1 row created.

SQL> select * from dept;

    DNO DNAME LOC
   -------- ----------- -----------

     10 inventory hyd

     20 finance   bglr

     30 HR        Mumbai
```

Creating emp2 table:

```
SQL> create table emp2(eno number(10),ename varchar(10),job varchar(10),MGR number(10),dnonumber(10));

Table created.

SQL> insert into emp2 values(111,'saketh','analyst',444,10);

 1 row created.

SQL>insert into emp2 values(222,'sandeep','clerk',333,20);

1 row created.

SQL>insert into emp2  values(333,'jagan','manager',111,10);

 1 row created.

SQL>insert into emp2 values(444,'madhu','engineer',222,40);

1 row created.

SQL> select * from emp2;

ENO    ENAME JOB              MGR          DNO

      -------- - ----------- - ----------- - ---- -    - -
                                 --
111 saketh             Analyst     444        10

222 sandeep           Clerk        333        20

333 jagan              Manager     111        10

444 madhu            Engineer      222        40
```

1.**Equijoin:**

A join which contains an equal to '=' operator in this joins condition.

```
ENO    ENAME        JOB             DNAME  LOC

-------      - -- -          ------        ------       - -- -
111    saketh           analyst          inventory  hyd
222    sandeep              Clerk        finance      bglr
333    jagan                Manager    Inventory   hyd
```

Using Clause:

```
SQL> select eno,ename,job,dname,loc from emp2 e join dept d using(dno);

ENO ENAME JOB        DNAME  LOC
     -----  ----------- - ----------- - ----------- - ------------
111 saketh       Analyst   inventory hyd
222 sandeep     Clerk      finance  bglr
```

```
333 jagan        manager inventory hyd
```

On Clause:

SQL> select eno,ename,job,dname,loc from emp2 e join dept d on(e.dno=d.dno);

```
ENO ENAMEJOB        DNAMELOC
-------------- - ----------- - ----------- - -----------
111 saketh       Analyst  inventory hyd
222 sandeep      Clerk    finance  bglr
333 jagan        manager  inventory hyd
```

2.Non-Equijoin:

A join which contains an operator other than equal to '=' in the join condition.

SQL> select eno,ename,job,dname,loc from emp2 e,dept d where e.dno>d.dno;

```
ENO ENAME     JOB      DNAME  LOC
     ------- ----------- --------   ----------- -----------
222 sandeep       Clerk      inventory hyd
444 madhu         enginee inventory hyd
                  r
444 madhu         enginee Finance   Bglr
                  r
444 madhu         enginee HR         Mumbai
                  r
```

3.Self Join:

Joining the table itself is called self join.

SQL> select e1.eno,e2.ename,e1.job,e2.dno from emp2 e1,emp2 e2 where e1.eno=e2.mgr;

```
ENO    ENAME    JOB          DN
                             O
-----------------------------------
444     saketh       engineer     10
333     sandeep      manage       20
111     jagan        analyst      10
222     madhu        clerk         40
```

4.Natural Join:

It compares all the common columns.

SQL> select eno, ename, job, dname, loc from emp2 natural join dept;

```
ENO ENAME JOB        DNAME  LOC
      ------------------- -------------------------- -----
111 saketh         analyst  inventory hyd
222 sandeep        Clerk    finance  bglr
333 jagan          manage   inventory hyd
                   r
```

5. Cross Join:

This will give the cross product.

SQL> select eno,ename,job,dname,loc from emp2 cross join dept;

```
ENO ENAME          JOB         DNAME              LOC
     ------ -----------  -----------  -----------    ----------
111 saketh           analyst      inventory  Hyd
222 sandeep            clerk       inventory hyd
333 jagan            manager     inventory hyd
444 madhu              engineer   inventory hyd
111 saketh           analyst      finance     Bglr
222 sandeep            clerk       finance     Bglr
333 jagan            manager     finance     Bglr
444 madhu               engineer    finance     Bglr
111 saketh            analyst       HR          Mumbai
222 sandeep             clerk        HR          Mumbai
333 jagan             manager      HR           Mumbai
11rows selected.
```

6. Outer Join:

It gives the non matching records along with matching records.

6.1 Left Outer Join:

This will display the all matching records and the records which are in left hand side table thosethat are in right hand side table.

```
SQL> select eno,ename,job,dname,loc from emp2 e left outer join dept d on(e.dno= d.dno);
(OR)
SQL> select eno,ename,job,dname,loc from emp2 e,dept d where e.dno=d.dno(+);
ENO  ENAME   JOB DNAME LOC
-------------             --------
333 jagan      manager  inventory hyd
111 saketh     analyst  inventory Hyd
222 sandeep  Clerk    finance    Bglr
444 madhu    Engineer
```

6.2 Right Outer Join:

This will display the all matching records and the records which are in right hand side table thosethat are not in left hand side table.

```
SQL> select eno,ename,job,dname,loc from emp2 e right outer join dept d on(e.dno =d.dno);
(OR)
SQL> select eno,ename,job,dname,loc from emp2 e,dept d where e.dno(+)=d.dno;
```

```
ENO ENAME JOB           DNAME  LOC
    -----  ----------  ----------  ----------  ----------
111 saketh      Analyst  inventory hyd
222 sandeep     Clerk    Finance   Bglr
333 jagan       Manager  inventory hyd
                         HR        Mumbai
```

6.3 Full Outer Join:

This will display the all matching records and the non matching records from both tables.

```
SQL> select eno,ename,job,dname,loc from emp2 e full outer join dept d on(e.dno= d.dno);

ENO ENAME JOB       DNAME   LOC
    ----------------------------------------------------
333 jagan      Manager   inventory hyd
111 saketh     Analyst   inventory hyd
222 sandeep    Clerk     Finance   Bglr
444 madhu      Engineer
                         HR        Mumbai
```

RESULT

Thus the relationship between databases has been implemented using join operation.

EX.NO:5 IMPLEMENTATION OF VIEWS

AIM

To execute and verify the SQL commands for Views.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential

attributes. STEP 3: Insert attribute values into

the table.

STEP 4: Create the view from the above created table.

STEP 5: Execute different Commands and extract information from the

View. STEP 6: Stop

QUERIES:

Q1: The organization wants to display only the details of the employees those who

are ASP. SQL> create view empview as select * from emp where job='ASP';

View created.

SQL> select * from empview;

```
EMPNO ENAME JOB   DEPTNO SAL

----------- ---------- --------    --------      ------

2        Arjun     ASP     2    12000
3        Gugan     ASP     2    20000
```

Q2: The organization wants to display only the details like empno, empname, deptno,deptname of the employees. (Vertical portioning)

SQL> create view empview1 as select ename, sal from emp;

View created.

Q3: Display all the views generated.

SQL> select * from tab;

```
TNAME TABTYPE CLUSTERID
-------------------------------- -------- -----------
DEPT TABLE
EMP       TABLE
EMPVIEW   VIEW
EMPVIEW1 VIEW
```

Q4: Execute the DML commands on the view created.

```
SQL> select * from empview;
EMPNO  ENAME JOB  DEPTNO  SAL
----------- -----------   -------- ---------   -------
2          Arjun   ASP      2    12000
3          Gugan   ASP      2    20000
```

Q5: Drop a view.

```
SQL>  drop  view  empview1;
View dropped.
```

RESULT

Thus the view commands were performed successfully and executed.

EX.NO:6 IMPLEMENTATION OF SYNONYMS

AIM

To execute and verify the SQL commands for Synonyms.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Insert attribute values into the table.

STEP 4: Create the synonyms from the above created table.

STEP 5: Execute different Commands .

STEP 6: Stop

SYNONYMS

- A *synonym* is an *alias*, that is, a form of shorthand used to simplify the task of referencing adatabase object.
- There are two categories of synonyms, *public* and *private*.
- A public synonym can be accessed by any system user.
- Private synonyms, on the other hand, belong to the system user that creates them and residein that user's schema.
- A system user can grant the privilege to use private synonyms that they own to other systemusers.

Examples:

```
SQL> select * from class;


NAME      ID
----------------
Anu        1
Brindha    2
Chinthiya 3
Divya      4
Ezhil      5
Fairoz     7
Hema       9
7 rows selected.
```

Create synonym:

In order to create synonyms, we will need to have the CREATE SYNONYM privilege.This privilege will be granted to us by the DBA.

We must have the CREATE PUBLIC SYNONYM privilege in order to create public synonyms.

```
SQL> create synonym c1 for class;
Synonym created.
SQL> insert into c1 values('kalai',20);
1 row created.

SQL> select * from class;


NAME       ID
---------- ----------
Anu        1
brindha    2
chinthiya  3
divya      4
ezhil      5
fairoz     7
hema       9
kalai      20

8 rows selected.

SQL> select * from c1;


NAME       ID
---------- ----------
anu        1
brindha    2
chinthiya  3
divya      4
ezhil      5
fairoz     7
hema       9
kalai      20

8 rows selected.
SQL> insert into class values('Manu',21);

1 row created.

SQL> select * from c1;

NAME       ID
---------- ----------

anu        1
brindha    2
chinthiya  3
divya      4
```

```
ezhil       5
fairoz      7
hema        9
kalai       20
Manu        21

9 rows selected.
```

Drop Synonym:

- In order to drop a public synonym we must include the PUBLIC keyword in the DROPSYNONYM command.
- In order to drop a public synonym, we must have the DROP PUBLIC SYNONYM privilege.
- DROP PUBLIC SYNONYM synonym_name;

```
SQL>  drop  synonym

c1;Synonym dropped.

SQL>  select  *  from
c1;select * from c1
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

RESULT

Thus the synonyms commands were performed successfully and executed.

EX.NO:7 IMPLEMENTATION OF SEQUENCES

AIM

To execute and verify the SQL commands for Sequences.

PROCEDURE

STEP 1: Start
STEP 2: Create the table with its essential attributes.
STEP 3: Insert attribute values into the table.
STEP 4: Create the sequences from the above created table.
STEP 5: Execute different Commands.
STEP 6: Stop

SEQUENCES

- Oracle provides the capability to generate sequences of unique numbers, and they are called **sequences.**
- Just like tables, views, indexes, and synonyms, a sequence is a type of database object.
- Sequences are used to generate unique, sequential integer values that are used as primarykey values in database tables.
- The sequence of numbers can be generated in either ascending or descending order.

Creation of table:

```
SQL> create table class(name varchar(10),id
number(10));Table created.
```

Insert values into table:

```
SQL> insert into class values('&name',&id);
 Enter value for name: anu
 Enter value for id: 1

 Old          1: insert into class values('&name',&id)
 new          1: insert into class values('anu',1)
 1      row
 created.

 SQL> /
 Enter value for name:
 brindha Enter value for id:
 02
 old1: insert into class
 values('&name',&id) new1: insert into
 class values('brindha',02) 1 row
 created.

 SQL> /
 Enter value for name:
 chinthiya Enter value for id:
```

```
03
old1:   insert   into   class   values('&name',&id)
new1:  insert  into  class  values('chinthiya',03) 1
row created.
SQL> select * from class;
NAME     ID
```

```
-----------     -----------
Anu         1
brindha     2
chinthiya   3
```

Create Sequence:

```
SQL> create sequence s_1
    2   start with 4
    3   increment by 1
    4          maxvalue 100
    5          cycle;
Sequence created.

SQL> insert into class values('divya',s_1.nextval);
             1 row created.

SQL>   select   *   from
class;NAME         ID
-----------          -----------

anu                        1
brindha                  2
chinthiya                3
divya                    4
```

Alter Sequence:

```
SQL>  alter  sequence  s_1  increment
by 2;Sequence altered.
SQL>insert into class values('fairoz',s_1.nextval);

1 row created.

SQL> select * from class;
           NAME      ID

           ----------- -----------
           anu           1
           brindha       2
           chinthiya     3
           divya         4
           ezhil         5
                         7
           fairoz
```

Drop Sequence:

```
SQL> drop sequence
s_1;
Sequence dropped.
```

RESULT

Thus the sequence commands were performed successfully and executed.

EX.NO:8 IMPLEMENTATION OF CURSORS

AIM:

To implement the cursor program for electricity bill calculation.

ALGORITHM:

STEP1:Start

STEP2:Create a table with table name bill.

STEP3:Insert the values into the table .

STEP4: Execute the procedure function for the bill calculation.

STEP5: Display the total amount.

STEP6: End

CURSOR PROGRAM FOR ELECTRICITY BILL CALCULATION:

```
SQL> create table bill(name varchar2(10), address varchar2(20), city varchar2(20),
unitnumber(10));

Table created.

SQL>   insert   into   bill   values('&name','&addess','&city','&unit');

Enter value for name: yuva

Enter  value  for  addess:  srivi

Enter      value      for      city:

srivilliputur  Enter   value   for

unit: 100

old  1: insert into bill values('&name','&addess','&city','&unit')

new            1:            insert            into            bill
values('yuva','srivi','srivilliputur','100') 1 row created.

SQL> /

Enter value for name: nithya

Enter  value  for  addess:  Lakshmi

nagarEnter value for city: sivakasi

Enter value for unit: 200

old  1: insert into bill values('&name','&addess','&city','&unit')
```

```
new  1: insert into bill values('nithya','Lakshmi nagar','sivakasi','200')

1 row created.

SQL> /
Enter value for name: maya

Enter  value  for  addess:  housing

boardEnter value for city: sivakasi

Enter value for unit: 300

old  1: insert into bill values('&name','&addess','&city','&unit')

new     1:     insert     into     bill     values('maya','housing
board','sivakasi','300')1 row created.

SQL> /

Enter value for name: jeeva

Enter  value  for  addess:  RRR

nagar  Enter   value   for   city:

sivaganagai   Enter   value   for

unit: 400

old  1: insert into bill values('&name','&addess','&city','&unit')

new      1:      insert      into      bill      values('jeeva','RRR

nagar','sivaganagai','400')1 row created.

SQL> select * from bill;
NAME      ADDRESS            CITY                 UNIT

----------- ---------------------- ---------------------- ----------

yuva    srivi                    srivilliputur          100

nithya    Lakshmi nagar       sivakasi                  200

maya  housing board       sivakasi                 300

jeeva    RRR nagar          sivaganagai            400
```

```
SQL> declare
 2  cursor c is select * from bill;

 3  b bill %ROWTYPE;

 4  begin

 5  open c;

 6 dbms_output.put_line('Name  Address      city  Unit      Amount');
 7  loop
 8  fetch c into b;
 9  if(c    %    notfound)
then10 exit;
11 else
12 if(b.unit<=100) then
13 dbms_output.put_line(b.name||' '||b.address||' '||b.city||' '||b.unit||' '||b.uni t*1);
14 elsif(b.unit>100 and b.unit<=200) then
15 dbms_output.put_line(b.name||' '||b.address||' '||b.city||' '||b.unit||' '||b. unit*2);
16 elsif(b.unit>200 and b.unit<=300) then
17 dbms_output.put_line(b.name||'  '||b.address||'  '||b.city||'  '||b.unit||'  '||b.
unit*3);18 elsif(b.unit>300 and b.unit<=400) then
19   dbms_output.put_line(b.name||'       '||b.address||'     '||b.city||'      '||b.unit||'       '||b.unit*
);
20   Else
21   dbms_output.put_line(b.name||'       '||b.address||'     '||b.city||'      '||b.unit||'       '||b.unit*
5);
22   end if;



 23  end if;
 24  end loop;
 25  close c;
 26  end;
```

```
27 /

Name       Address          city              Unit    Amount

Yuva       srivi            srivilliputur      100      100

Nithya     Lakshmi nagar              sivakasi        200      400

Maya       housing board              sivakasi        300      900

Jeeva      RRR nagar                  sivaganagai      400  1600


PL/SQL procedure successfully completed.
```

RESULT:

Thus the program to implement cursors was executed and output was verified successfully.

PROGRAM FOR STUDENT GRADE CALCULATION

AIM

To write a PL/SQL block to display the student name, marks whose average mark is above 60%.

ALGORITHM

STEP1:Start

STEP2:Create a table with table name stud_exam

STEP3:Insert the values into the table and Calculate total and average of each student

STEP4: Execute the procedure function the student who get above 60%.

STEP5: Display the total and average of student

STEP6: End

```
SQL> create table std(name varchar(10), rollno number(3),mark1 number(3),
mark2number(3), mark3 number(3));
Table created.

SQL> insert into std values('&name','&rollno','&mark1','&mark2','&mark3');

Enter value for name: gowri

Enter value for rollno:

101 Enter value for

mark1: 78 Enter value

for mark2: 89 Enter

value for mark3: 99

old 1: insert into std values('&name','&rollno','&mark1','&mark2','&mark3')

new 1: insert into std values('gowri','101','78','89','99')
1 row created.

SQL> /

Enter value for name:

prem Enter value for

rollno: 102 Enter value

for mark1: 88 Enter
```

```
value for mark2: 99

Enter value for mark3:

90

old 1: insert into std values('&name','&rollno','&mark1','&mark2','&mark3')

new 1: insert into std values('prem','102','88','99','90')
1 row created.

SQL> /

Enter value for name:

ravathi Enter value for

rollno: 103 Enter value for

mark1: 67 Enter value for

mark2: 89 Enter value for

mark3: 99

old 1: insert into std values('&name','&rollno','&mark1','&mark2','&mark3')

new 1: insert into std values('ravathi','103','67','89','99')

1 row created.

SQL> /

Enter value for name:

arun Enter value for

rollno: 104 Enter value

for mark1: 56 Enter

value for mark2: 66

Enter value for mark3:

77

old 1: insert into std values('&name','&rollno','&mark1','&mark2','&mark3')

new 1: insert into std values('arun','104','56','66','77')

1 row created.
```

```
SQL> set serveroutput on;
SQL> declare
  2 tot number;
  3 average number;
  4 cursor c is select * from std;
  5 s std %ROWTYPE;
  6 begin
  7 open c;
  8 dbms_output.put_line('Name Rollno Mark1 Mark2 Mark3 Total Average Grade');
  9 loop
 10 fetch c into s;
 11 tot:=s.mark1+s.mark2+s.mark3;
 12 average:=floor(tot/3);
 13 if(c % notfound)then
 14 exit;
 15 else
 16 if(s.mark1<50 or s.mark2<50 or s.mark3<50)then
 17 dbms_output.put_line(s.name||'   '||s.rollno||'   '||s.mark1||'   '||s.mark2||'
    '||s.mark3||'   '||tot||'           '||average||'   '||'F');
 18 elsif(average>=90 and average<=100)then
 19 dbms_output.put_line(s.name||'   '||s.rollno||'   '||s.mark1||'   '||s.mark2||'
    '||s.mark3||'   '||tot||'           '||average||'   '||'S');
 20 elsif(average>=80 and average<90)then
 21 dbms_output.put_line(s.name||'   '||s.rollno||'   '||s.mark1||'   '||s.mark2||'
    '||s.mark3||'   '||tot||'           '||average||'   '||'A+');
 22 elsif(average>=70 and average<80)then
 23 dbms_output.put_line(s.name||'   '||s.rollno||'   '||s.mark1||'   '||s.mark2||'
'||s.mark3||'   '||tot||'    '||average||'   '||'B');
 24 elsif(average>=60 and average<70)then
 25  dbms_output.put_line(s.name||'   '||s.rollno||'   '||s.mark1||'   '||s.mark2||'
     '||s.mark3||'   '||tot||'            '||average||'   '||'C');
 26  else
 27dbms_output.put_line(s.name||'    '||s.rollno||'   '||s.mark1||'   '||s.mark2||'   '||s.mark3|
''||tot||'  '||average||'   '||'D');   28                                                 |
 end if;
 29 end if;
 30 end loop;
 31 close c;
 32 end;
 33 /
```

```
Name     Rollno  Mark1   Mark2   Mark3   Total   Average   Grade

Gowri    101     78      89      99      266     88         A+

Prem     102     88      99      90      277     92        S

ravath   103     67      89      99      255     85        A+
i
Arun     104     56      66      77      199     66        C
```

PL/SQL procedure successfully completed.

RESULT:

Thus the program to implement cursors was executed and output was verified successfully.

EX.NO:9 IMPLEMENTATION OF TRIGGERS

AIM

To develop and execute a Trigger for before and after update, Delete, Insert operations on a table.

PROCEDURE

STEP 1: Start

STEP 2: Initialize the trigger with specific table id.

STEP 3:Specify the operations (update, delete, insert) for which the trigger has to be executed.

STEP 4: Execute the Trigger procedure for both Before and After sequences

STEP 5: Carryout the operation on the table to check for Trigger execution.

STEP 6: Stop

<u>TRIGGER FOR DISPLAYING GRADE OF THE STUDENT</u>

```
SQL> create table stdn(rollno number(3),name varchar(2),m1 number(3),m2 number(3),m3
number(3),tot number(3),avrg number(3),result varchar(10));

Table created.

SQL> create or replace trigger t1 before insert on stdn
  2  for each row
  3  begin
  4  :new.tot:=:new.m1+:new.m2+:new.m3;
  5  :new.avrg:=:new.tot/3;
  6  if(:new.m1>=50 and :new.m2>=50 and :new.m3>=50) then
  7  :new.result:='pass';
  8  else
  9  :new.result:='Fail';
 3   end if;
 4   end;
 5   /
Trigger created.
SQL>             insert             into             stdn
values(101,'SM',67,89,99,",",");1 row created.
SQL> select * from stdn;

   ROLLNO NA          M1        M2          M3            TOT      AVRG RESULT
---------------------------------------------------------------------------
101              SM      67          89            9 9            255        85pass
```

PROGRAM TO INDICATE INVALID CONDITION USING TRIGGER

```
SQL> create table emp (name varchar(10),empno number(3),age number(3));

Table created.
SQL>
  1  create or replace trigger t2 before insert on emp
  2  for each row
  3  when(new.age>100)
  4  begin
  5  RAISE_APPLICATION_ERROR(-20998,'INVALID   ERROR');
  6 end;
SQL> /

Trigger created.

SQL>  insert  into  emp  values('nithya',101,24);

1 row created.

SQL>   insert   into   emp   values('nithya',101,103);
insert into emp values('nithya',101,103)
        *
```

```
ERROR at line 1:

ORA-20998: INVALID ERROR

ORA-06512:    at    "SCOTT.T2",

line 2

ORA-04088: error during execution of trigger 'SCOTT.T2'
```

RESULT:

Thus triggers were implemented successfully.

EXNO:10 PROCEDURES AND FUNCTIONS

AIM

To write a Functional procedure to insert a number into a table.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with essential attributes.

STEP 3: Initialize the procedure to insert a number.

STEP 5: Execute the procedure.

STEP 6: Stop

PROCEDURE TO INSERT NUMBER

```
SQL> create table emp1(id number(3),First_name varchar2(20));

Table created.

SQL> insert into emp1 values(101,'Nithya');
1 row created.

SQL> insert into emp1 values(102,'Maya');
1 row created.

SQL> select * from emp1;
    ID FIRST_NAME

---------- ----------------------


    101  Nithya

    102  Maya

SQL> set serveroutput on;

SQL> create or replace

 2 procedure insert_num(p_num number)is
 3  begin

 4  insert into emp1(id,First_name) values(p_num,user);

 5  end

 insert_num;  6   /
```

```
Procedure created.

SQL> exec insert_num(3);
PL/SQL procedure successfully completed.

SQL> select * from emp1;
      ID FIRST_NAME

---------- ----------------------

     101   Nithya

     102   Maya

     103   SCOTT
```

FUNCTION TO FIND FACTORIAL

AIM

To write a Function to find factorial of given number.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with essential attributes.

STEP 3: Initialize the Function to find the factorial a given number.

STEP 5: Execute the Function .

STEP 6: Stop

```
SQL> create or replace function fact(n number)
 6   return number is
 7   i number(10);
 8   f number:=1;
 9   begin
 10 for   i   in   1..N
 loop11 f:=f*i;
 12 end loop;
 13 return f;
 10
 end;

 11  /


Function created.

SQL>   select   fact(2)   from

  dual;FACT(2)
----------
     2
```

RESULT:

Thus procedures and functions were implemented successfully.

EX.NO:11 IMPLEMENTATION OF XML SCHEMA

AIM:

To create an XML database and validate it using an XML schema using SQL.

ALGORITHM

1. Create a table to store the XML data, specifying the columns for the book ID and the book XML data, using the CREATE TABLE statement.
2. Insert sample XML data into the table using the INSERT statement.
3. Create an XML schema for the book data using the CREATE TABLE statement, specifying the columns for the schema ID and the schema XML data.
4. Insert the XML schema into the book schema table using the INSERT statement.
5. Use the XMLVALIDATE function to validate the XML data against the XML schema using a SELECTstatement.

PROGRAM:

```
-- Step 1: Create a table to store the XML data
CREATE TABLE books (
    book_id NUMBER,
    book_xml XMLTYPE
);

-- Step 2: Insert some sample XML data
INSERT INTO books VALUES (
    1,
    XMLTYPE('<book id="1">
        <title>The Catcher in the Rye</title>
        <author>J.D. Salinger</author>
        <published>1951</published>
    </book>')
);

INSERT INTO books VALUES (
    2,
    XMLTYPE('<book id="2">
        <title>To Kill a Mockingbird</title>
        <author>Harper Lee</author>
        <published>1960</published>
    </book>')
);

-- Step 3: Create an XML schema
CREATE TABLE book_schema (
    schema_id NUMBER,
    schema_xml XMLTYPE
);

-- Step 4: Insert the XML schema into the book schema table
INSERT INTO book_schema VALUES (
    1,
    XMLTYPE('<?xml version="1.0"?>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="books">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="book" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="title" type="xs:string"/>
                  <xs:element name="author" type="xs:string"/>
                  <xs:element name="published" type="xs:integer"/>
                </xs:sequence>
                <xs:attribute name="id" type="xs:integer" use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>')
);
```

```
-- Step 5: Validate the XML data against the XML schema
SELECT
  book_id,
  book_xml
FROM books
WHERE XMLVALIDATE(book_xml, XMLSCHEMA('book_schema.schema_xml')) = 1;
```

OUTPUT:

BOOK_ID	BOOK_XML
1	<book id="1"><title>The Catcher in the Rye</t...
2	<book id="2"><title>To Kill a Mockingbird</t...
*	

RESULT

Thus the XML database and validate it using an XML schema using SQL has been implemented and output was verified.

EX.NO:12 IMPLEMENTATION OF NOSQL

AIM:

Create a MongoDB database and a collection to store documents containing information about books.

ALGORITHM:

1. Connect to the MongoDB server.
2. Create a database named "library".
3. Switch to the "library" database.
4. Create a collection named "books".
5. Insert some sample book documents into the "books" collection.

PROGRAM:

```
// Step 1: Connect to the MongoDB server
const MongoClient = require('mongodb').MongoClient;
const uri = "mongodb://localhost:27017/";
const client = new MongoClient(uri, { useNewUrlParser: true });
client.connect(err => {
  if (err) throw err;
  console.log("Connected to MongoDB server");

  // Step 2: Create a database named "library"
  const db = client.db("library");

  // Step 3: Switch to the "library" database
  db.collection("books", function(err, collection) {
    if (err) throw err;

    // Step 4: Create a collection named "books"
    console.log("Created collection 'books'");

    // Step 5: Insert some sample book documents into the "books" collection
    const books = [
      { title: "The Catcher in the Rye", author: "J.D. Salinger", published: 1951 },
      { title: "To Kill a Mockingbird", author: "Harper Lee", published: 1960 },
      { title: "1984", author: "George Orwell", published: 1949 }
    ];
    collection.insertMany(books, function(err, result) {
      if (err) throw err;
      console.log("Inserted " + result.insertedCount + " book documents");
      client.close();
    });
  });
});
```

OUTPUT:

```
Connected to MongoDB server
Created collection 'books'
Inserted 3 book documents
```

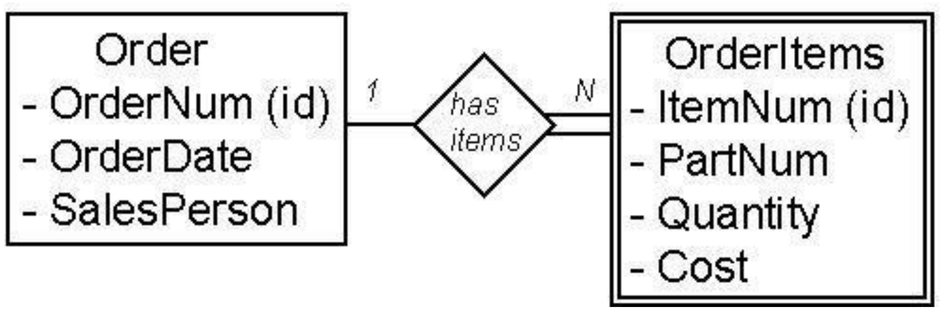
RESULT:

Thus the MongoDB database and a collection to store documents containing information about books was implementedsuccessfully.

EX.NO:13 DATABASE DESIGN USING E-R MODEL AND NORMALIZATION

ER diagram:

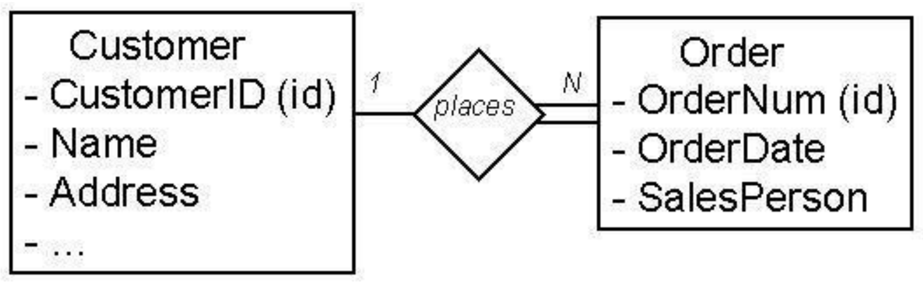
Chen Notation



- **ORDER** (<u>OrderNum</u> (key), OrderDate, SalesPerson)

  ORDERITEMS (<u>OrderNum</u> (key)(fk) , <u>ItemNum</u> (key), PartNum, Quantity, Cost)
  - In the above example, in the ORDERITEMS Relation: OrderNum is the *ForeignKey* and OrderNum plus ItemNum is the *Composite Key*.

Chen Notation



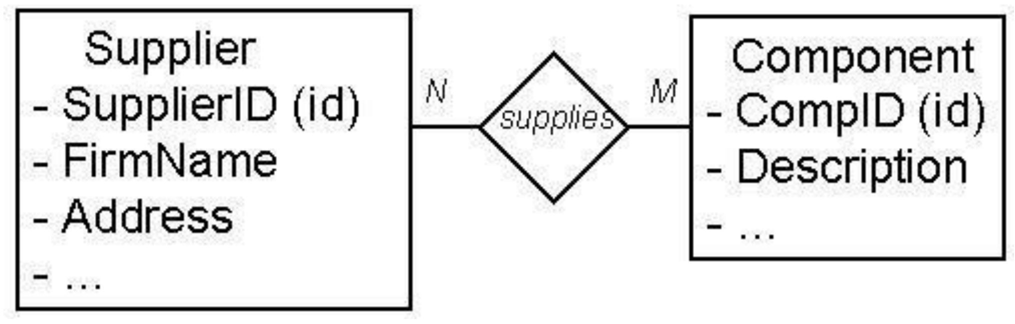
In the ORDER Relation: OrderNum is the *Key*.

Representing Relationships

- **1:1** Relationships. The key of one relation is stored in the second relation. Lookat example queries to determine which key is queried most often.

- **1:N** Relationships.
  Parent - Relation on the "1" side. **Child**
  - Relation on the "Many" side.
- Represent each Entity as a relation.
  Copy the key of the parent into the child relation.
- CUSTOMER (CustomerID (key), Name, Address, ...)
  ORDER (OrderNum (key), OrderDate, SalesPerson, CustomerID (fk))
- **M:N** Relationships. Many to Many relationships can not be directly implemented inrelations.
- Solution: Introduce a third *Intersection relation* and copy keys from original tworelations.

Chen Notation



- SUPPLIER (SupplierID (key), FirmName, Address, ...)
  COMPONENT (CompID (key), Description, ...)
  SUPPLIER_COMPONENT (SupplierID (key), CompID (key))
- Note that this can also be shown in the ER diagram. Also, look for potential added attributes in the intersection relation.

RESULT:

Thus the ER Database design using E-R model and Normalization was implementedsuccessfully.

EX.NO: 14 EMPLOYEE INFORMATION- DATABASE CONNECTIVITY

AIM:

To create the following Form using Database Grid tool in Visual Basic.

DESCRIPTION:

1. The connection of database with the visual basic form window is made possible usingDatabase Grid.
2. The database Table to be connected is specified in the record source field in thedbgrid properties window.
3. Text boxes or labels associated with the data fields are connected to the data grid using the "Data source" and the filed in the data table is connected using "Data Field" from the properties window of the respective textboxes or labels

The following commands are used to perform the data grid operations

1) Data_grid_name.recordset.addnew Adds new record
2) Data_grid_name.recordset.delete Deletes a record
3) Data_grid_name.recordset.movenext Moves to the next record
4) Data_grid_name.recordset.moveprevious Moves to the previous record
5) Data_grid_name.recordset. movefirst Moves to the first record
6) Data_grid_name.recordset.movel ast Moves to the last record

1) Data_grid_name.recordset.edit

2) Data_grid_name.recordset.update

CODING:

Private Sub Command1_Click()Data1.Recordset.MoveFirst End Sub

Private Sub Command2_Click()Data1.Recordset.MoveLast

End Sub

Private Sub Command3_Click()Data1.Recordset.MovePreviousEnd Sub

Prepares a row of a Recordset for editing

 Cancels any pending Update statements.

Private Sub

Command4_Click()

Data1.Recordset.MoveNext

End Sub

Private Sub

Command5_Click()

Data1.Recordset.MoveLast

Data1.Recordset.AddNew

End Sub

Private Sub

Command6_Click()

Data1.Recordset.Delete

Data1.Recordset.MoveLast

End Sub

Private Sub

Command7_Click()

Data1.Recordset.Edit

Data1.Recordset.Update

```
End Sub
Private                     Sub

Command8_Click()End

End Sub
```

RESULT:

Thus the employee information was created using DBGrid tool in Visual Basic

EX.NO: 15 IMPLEMENTATION OF PAYROLL PROCESSING

AIM:

To design and implement the pay roll processing System.

STEPS:

1. Create a database for payroll processing which request the using SQL

2. Establish ODBC connection

3. In the administrator tools open data source ODBC

4. Click add button and select oracle in ORA home 90, click finish

5. A window will appear given the data source home as oracle and select TNS source name as lion andgive the used id as SWTT

6. ADODC CONTROL FOR SALARY FORM:-

7. The above procedure must be follow except the table , A select the table as salary

8. Write appropriate Program in form each from created in VB from each from created in VB formproject.

```
SQL>create table emp(eno number primary key,enamr varchar(20),age number,addr
varchar(20),DOB date,phno number(10));
Table created.

SQL>create    table    salary(eno    number,edesig    varchar(10),basic    number,da
number,hra number,pf number,mc number,met number,foreign key(eno) references
emp); Table created.TRIGGER to calculate DA,HRA,PF,MC
SQL>  create  or  replace  trigger
 employ2 after insert on salary
3 declare
4 cursor  cur  is  select  eno,basic  from
salary;5 begin
6  for  cur1  in  cur
loop     7     update
salary set
8  hra=basic*0.1,da=basic*0.07,pf=basic*0.05,mc=basic*0.03  where  hra=0;  9
end loop;10 end;
11 / Trigger created.
```

PROGRAM FOR FORM 1

```
Private    Sub    emp_Click()
Form2.Show End
Sub Private
Sub
exit_Click()
Unload Me
End  Sub  Private
Sub
salary_Click()
Form3.Show
End Sub
```

PROGRAM FOR FORM 2

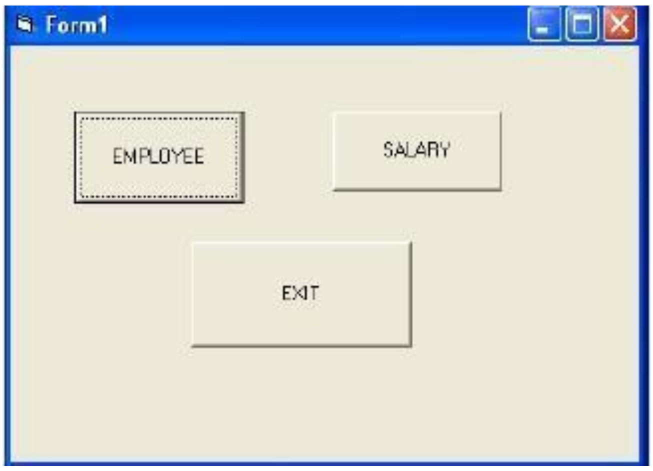
```
Private Sub add_Click()
Adodc1.Recordset.AddNew MsgBox "Record added"

End Sub Private
Sub
clear_Click()
Text1.Text  =  ""
Text2.Text  =  ""
Text3.Text  =  ""
Text4.Text  =  ""
Text5.Text  =  ""
Text6.Text = ""
End      Sub      Private      Sub      delte      _Click()
Adodc1.Recordset.Delete       MsgBox       "Record
Deleted" If Adodc1.Recordset.EOF  =             True
                                      Then
Adodc1.Recordset.MovePrevious
End
IfEnd
Sub        Private        Sub
exit_Click()Unload Me
End Sub
Private                   Sub
main_Click()
Form1.Show
End Sub
Private                      Sub
modify_Click()
Adodc1.Recordset.Updat
e End Sub
```

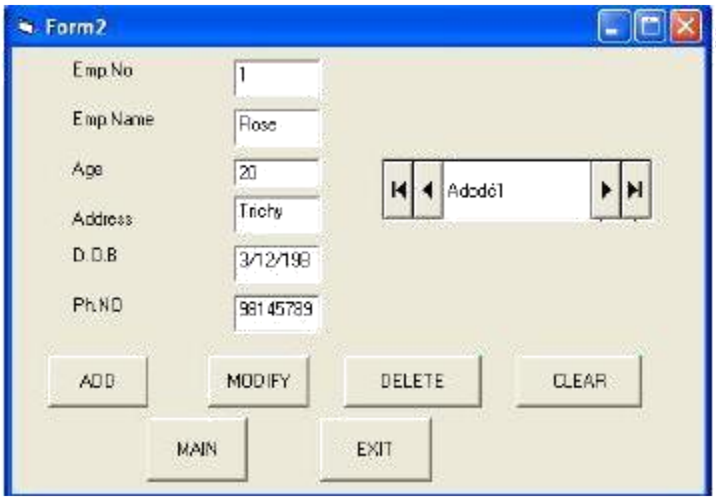
PROGRAM FOR FORM 3

```
Private Sub add_Click()
Adodc1.Recordset.AddNew    MsgBox    "Record
added"End Sub
Private     Sub
```

```
clear_Click()
Text1.Text   =
""
Text2.Text   =
""  Text3.Text
=               ""
Text4.Text   =
""  Text5.Text
=               ""
Text6.Text   =
""End Sub
Private Sub delte_Click()
Adodc1.Recordset.Delete     MsgBox     "Record
Deleted"If Adodc1.Recordset.EOF = True
Then Adodc1.Recordset.MovePrevious
End   If
End
Sub
Private                 Sub
exit_Click()    Unload
Me
End Sub
Private                  Sub
main_Click()
Form1.Show
End          Sub
Private     Sub
modify_Click
()
Adodc1.Recordset.Update
End Sub
```

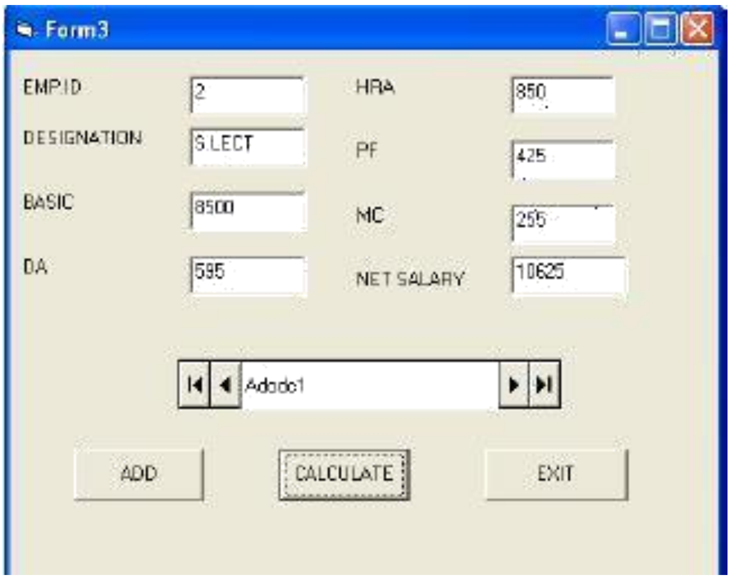
Output:





Form2

Emp.No	1
Emp.Name	Rose
Age	20
Address	Trichy
D.O.B	3/12/198
Ph.NO	98145789



Form3

EMP.ID	2	HRA	850
DESIGNATION	S.LECT	PF	425
BASIC	8500	MC	255
DA	595	NET SALARY	10625

RESULT:

Thus payroll system was designed and implemented successfully.

EX.NO:16 DESIGN AND IMPLEMENTATION OF BANKING SYSTEM

AIM :

To design and implement the pay roll processing System.

STEPS:

1.Create the DB for banking system source request the using SQL.

2.Establishing ODBC connection.

3. ISUAL BASIC APPLICATION:-

Create standard exe project in to and design ms from in request format

To add ADODC project select component and check ms ADO data control click ok

Nowthe control is added in the tool book

Create standard exe project in to and design ms from in request format

4.ADODC CONTEOL FOR ACCOUNT FROM:- Click customs and property window and windowwill appear and select ODBC data source name as oracle and click apply as the some window.

CREATE A TABLE IN ORACLE

```
SQL>create table account(cname varchar(20),accno number(10),balance number);
TableCreated
SQL> insert into account values('&cname',&accno,&balance);
Enter value for cname: Mathi
Enter value for accno: 1234
Enter value for balance:
10000
old 1: insert into account
values('&cname',&accno,&balance) new 1: insert into emp
values('Mathi',1234,10000) 1 row created.
```

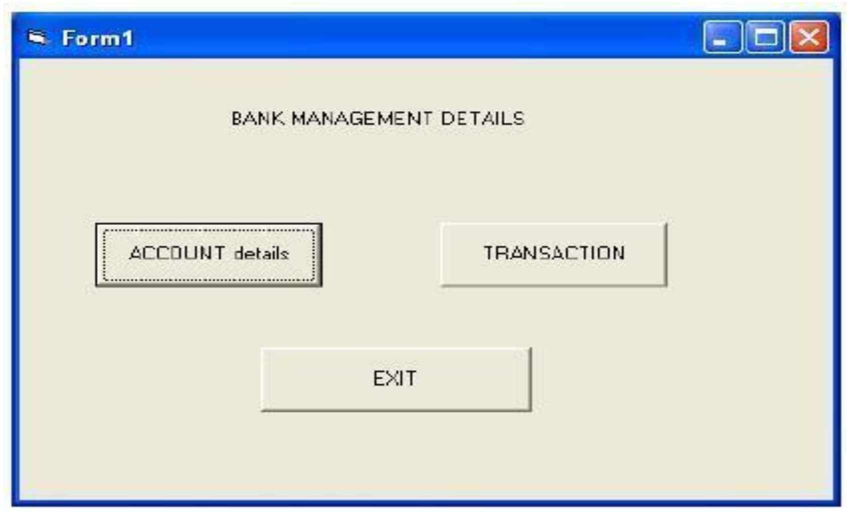
SOURCE CODE FOR FORM1

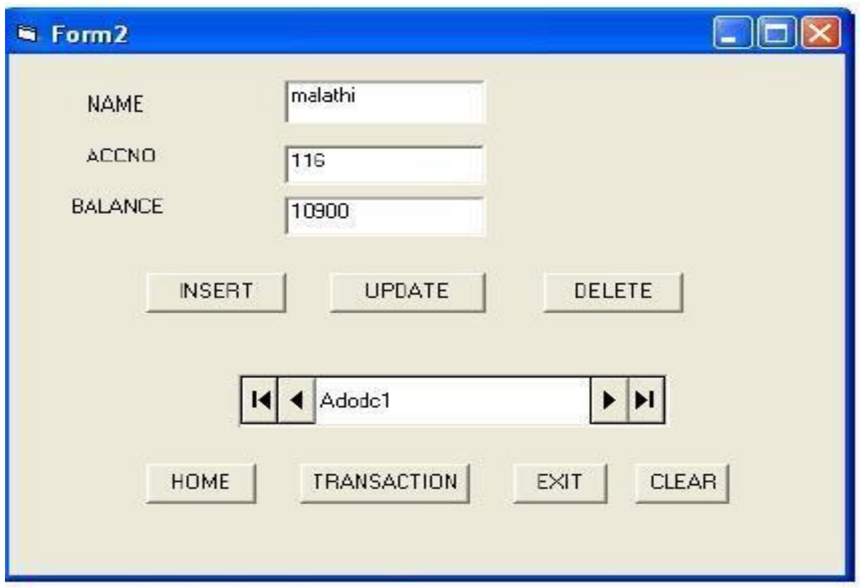
```
Private Sub
ACCOUNT_Click()
Form2.Show
End Sub
Private Sub
EXIT_Click
() Unload
Me End Sub
Private Sub
TRANSACTION_Click()
Form3.Show

End Sub
```

SOURCE CODE FOR FORM 2

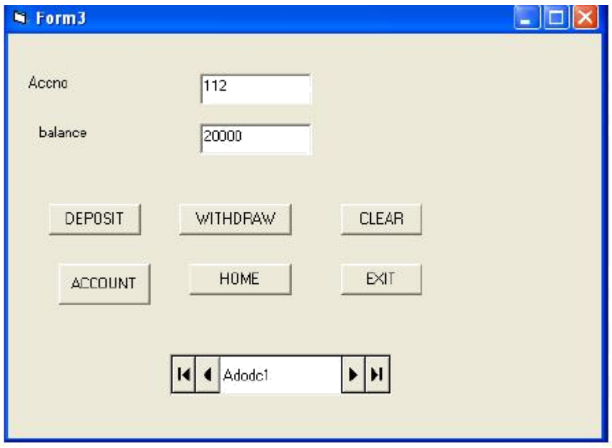
```
Private                    Sub
CLEAR_Click()
Text1.Text = ""
 Text2.Text   =
""   Text3.Text
= "" End Sub
Private Sub
DELETE_Click()
Adodc1.Recordset.DELETE      MsgBox      "record      deleted"
Adodc1.Recordset.MoveNext If Adodc1.Recordset.EOF = True
ThenAdodc1.Recordset.MovePrevious
End   If
End
Sub
Private   Sub   EXIT_Click()
Unload Me
End          Sub
Private      Sub
HOME_Click()
Form1.Show
End          Sub
Private Sub
INSERT_Click()
Adodc1.Recordset.AddNewEnd Sub
Private                        Sub
TRANSACTION_Click()
Form3.Sho
wEnd Sub
Private   Sub   UPDATE_Click()   Adodc1.Recordset.UPDATE   MsgBox   "record
updatedsuccessfully"
End Sub
SOURCE CODE FOR FORM 3
Private                       Sub
ACCOUNT_Click()
Form2.Show
End Sub
Private                    Sub
CLEAR_Click()
Text1.Text = ""
Text2.Text   =
""   End    Sub
Private Sub
DEPOSIT_Click()
Dim s As String s = InputBox("enter the amount to be deposited")
Text2.Text = Val(Text2.Text) + Val(s) A = Text2.Text MsgBox "CURRENT BALANCE IS
Rs" +Str(A) Adodc1.Recordset.Save Adodc1.Recordset.UPDATE
```

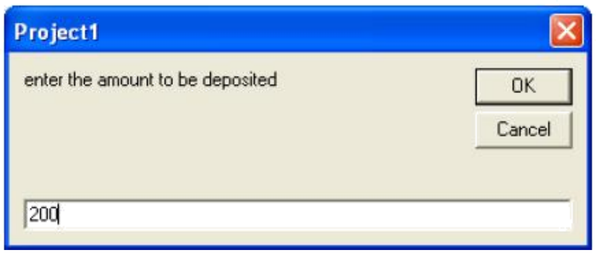
```
End          Sub
Private      Sub
EXIT_Click()
Unload       Me
End          Sub
Private      Sub
HOME_Click
()
Form1.Show
End Sub  Private
Sub
WITHDRAW_Click()
Dim s As String s = InputBox("enter the amount to be deleted")
Text2.Text = Val(Text2.Text) - Val(s) A = Text2.Text MsgBox "current balance
is Rs" +Str(A)
Adodc1.Recordset.Save
Adodc1.Recordset.UPDA
TEEnd Sub
```





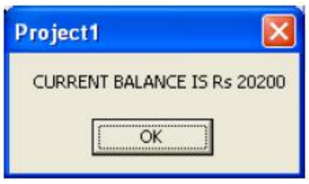
NAME	malathi
ACCNO	116
BALANCE	10900





enter the amount to be deposited

200



CURRENT BALANCE IS Rs 20200

RESULT:

Thus the banking system was designed and implemented successfully.

EX.NO:17 DESIGN AND IMPLEMENTATION OF LIBRARY MANAGEMENT SYSTEM

AIM :

To design and implement the library management System.

STEPS:

1. Create a database for library which request the using SQL
2. Establish ODBC connection
3. In the administrator tools open data source ODBC
4. Click add button and select oracle in ORA home 90, click finish
5. A window will appear given the data source home as oracle and select TNS source name as lionand give the used id as SWTT
6. ADODC CONTROL FOR library FORM:-
7. The above procedure must be follow except the table , A select the table as library
8. Write appropriate Program in form each from created in VB from each from created in VB formproject.

Relational Database Schema							
Status	code	Description					
Media	media_id	Code					
Book	ISBN	Title	author	year	dewey	price	
BookMedia	media_id	ISBN					
Customer	ID	Name	addr	DOB	phone	username	Password
Card	num	Fines	ID				
Checkout	media_id	Num	since	until			
Location	name	Addr	phone				
Hold	media_id	Num	name	until	queue		

Stored_In	media_id	Name	

Librarian	Eid	ID	Pay	name	since	
Video	title	Year	director	rating	price	
VideoMedia	media_id	Title	year			

CREATE TABLE Status (code INTEGER, description CHAR(30), PRIMARY KEY (code)); CREATE TABLE Media(media_id INTEGER, code INTEGER, PRIMARY KEY (media_id),FOREIGN KEY (code) REFERENCES Status);

CREATE TABLE Book(ISBNCHAR(14), title CHAR(128), author CHAR(64), year INTEGER, dewey INTEGER, price REAL, PRIMARY KEY (ISBN));

CREATE TABLE BookMedia(media_id INTEGER, ISBN CHAR(14), PRIMARY KEY (media_id),

FOREIGN KEY (media_id) REFERENCES Media,

FOREIGNKEY (ISBN) REFERENCES Book);

CREATE TABLE Customer(ID INTEGER, name CHAR(64), addr CHAR(256), DOBCHAR(10),

phone CHAR(30), username CHAR(16), password CHAR(32), PRIMARY KEY (ID),UNIQUE (username));

CREATE TABLE Card(num INTEGER, fines REAL, ID INTEGER, PRIMARY KEY (num),FOREIGN KEY (ID) REFERENCES Customer);

CREATE TABLE Checkout(media_id INTEGER, num INTEGER, since CHAR(10), until CHAR(10),PRIMARY KEY (media_id),

FOREIGN KEY (media_id) REFERENCES Media,

FOREIGNKEY (num) REFERENCES Card);

CREATE TABLE Location(name CHAR(64), addr CHAR(256), phone CHAR(30), PRIMARY

KEY (name));

CREATE TABLE Hold(media_id INTEGER, num INTEGER, name CHAR(64), untilCHAR(10),

queue INTEGER, PRIMARY KEY (media_id, num),

```
FOREIGNKEY (name) REFERENCES Location,
FOREIGN KEY (num) REFERENCES Card, FOREIGN KEY
(media_id) REFERENCES Media );
CREATE TABLE Stored_In( media_id INTEGER, name char(64), PRIMARY KEY
(media_id), FOREIGN KEY (media_id) REFERENCES Media ON DELETE
CASCADE, FOREIGN
KEY (name) REFERENCES Location );
CREATE TABLE Librarian( eid INTEGER, ID INTEGER NOT NULL, Pay REAL,
Loc_nameCHAR(64) NOT NULL, PRIMARY KEY (eid),
FOREIGN KEY (ID) REFERENCES Customer ON DELETE CASCADE, FOREIGN
KEY (Loc_name) REFERENCES Location(name) );
CREATE TABLE Video( title CHAR(128), year INTEGER, director CHAR(64), rating
REAL,price REAL, PRIMARY KEY (title, year) );
CREATE TABLE VideoMedia( media_id INTEGER, title CHAR(128), year INTEGER,
PRIMARY KEY (media_id), FOREIGN KEY (media_id) REFERENCES Media,
FOREIGNKEY (title, year) REFERENCES Video );
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (60201, 'Jason L. Gray', '2087 Timberbrook Lane, Gypsum, CO
81637', '09/09/1958', '970-273-9237', 'jlgray', 'password1');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (89682, 'Mary L. Prieto', '1465 Marion Drive, Tampa, FL 33602',
'11/20/1961',
'813-487-4873', 'mlprieto', 'password2');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES(64937, 'Roger Hurst', '974 Bingamon Branch Rd, Bensenville, IL 60106',
'08/22/1973',
'847-221-4986', 'rhurst', 'password3');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES(31430, 'Warren V. Woodson', '3022 Lords Way, Parsons, TN 38363',
'03/07/1945', '731-845-0077', 'wvwoodson', 'password4');
```

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (79916, 'Steven Jensen', '93 Sunny Glen Ln, Garfield Heights, OH
44125', '12/14/1968','216-789-6442', 'sjensen', 'password5');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(93265, 'David Bain', '4356 Pooh Bear Lane, Travelers Rest, SC 29690',
'08/10/1947','864-610-9558', 'dbain', 'password6');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (58359, 'Ruth P. Alber', '3842 Willow Oaks Lane, Lafayette, LA
70507', '02/18/1976', '337-316-3161', 'rpalber', 'password7');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (88564, 'Sally J. Schilling', '1894 Wines Lane, Houston, TX 77002',
'07/02/1954',
'832-366-9035', 'sjschilling', 'password8');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (57054, 'John M. Byler', '279 Raver Croft Drive, La Follette, TN
37766', '11/27/1954', '423-592-8630', 'jmbyler', 'password9');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password)
VALUES (49312, 'Kevin Spruell', '1124 Broadcast Drive, Beltsville, VA
20705', '03/04/1984', '703-953-1216', 'kspruell', 'password10');
INSERT INTO Card(num, fines, ID) VALUES ( 5767052, 0.0, 60201); INSERT
INTOCard(num, fines, ID) VALUES ( 5532681, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 2197620, 10.0, 89682);
 INSERT INTOCard(num, fines, ID) VALUES ( 9780749, 0.0, 64937); INSERT
INTO Card(num, fines, ID) VALUES ( 1521412, 0.0, 31430); INSERT INTO
Card(num, fines, ID) VALUES ( 3920486, 0.0, 79916); INSERT INTO Card(num,
fines, ID) VALUES ( 2323953, 0.0,93265); INSERT INTO Card(num, fines, ID)
VALUES ( 4387969, 0.0, 58359); INSERTINTO Card(num, fines, ID) VALUES
( 4444172, 0.0, 88564); INSERT INTO
Card(num, fines, ID) VALUES ( 2645634, 0.0, 57054); INSERT INTO
Card(num, fines, ID) VALUES ( 3688632, 0.0, 49312); INSERT INTO
Location(name, addr, phone) VALUES ('Texas Branch', '4832 Deercove Drive,
Dallas, TX 75208', '214-948-7102'); INSERT INTO Location(name, addr,
phone) VALUES ('Illinois Branch', '2888 Oak Avenue, Des Plaines, IL 60016',
'847-953-8130');
INSERT INTO Location(name, addr, phone) VALUES ('Louisiana Branch', '2063
Washburn Street, Baton Rouge, LA 70802', '225-346-0068'); INSERT INTO
Status(code, description) VALUES (1, 'Available'); INSERT INTO Status(code,
description) VALUES (2, 'In Transit'); INSERT INTO Status(code, description)
VALUES (3, 'Checked Out'); INSERT INTO Status(code, description) VALUES
(4, 'On Hold'); INSERT INTO Media( media_id, code) VALUES (8733, 1);
INSERT INTO Media( media_id, code) VALUES (9982, 1);
INSERT INTO Media( media_id, code) VALUES (3725, 1);
INSERT INTO Media( media_id, code) VALUES (2150, 1);
INSERT INTO Media( media_id, code) VALUES (4188, 1);
INSERT INTO Media( media_id, code) VALUES (5271, 2);
INSERT INTO Media( media_id, code) VALUES (2220, 3);
INSERT INTO Media( media_id, code) VALUES (7757, 1);
INSERT INTO Media( media_id, code) VALUES (4589, 1);
INSERT INTO Media( media_id, code) VALUES (5748, 1);
INSERT INTO Media( media_id, code) VALUES (1734, 1);
INSERT INTO Media( media_id, code) VALUES (5725, 1);
INSERT INTO Media( media_id, code) VALUES (1716, 4);
INSERT INTO Media( media_id, code) VALUES (8388, 1);
INSERT INTO Media( media_id, code) VALUES (8714, 1);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES ('978-0743289412',
'Lisey''s Story', 'Stephen King',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
```

```
('978-1596912366', 'Restless: A Novel', 'William Boyd', 2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0312351588', 'Beachglass', 'Wendy Blackburn', 2006, 813, 10.0);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978- 0156031561', 'The Places In Between', 'Rory Stewart', 2006, 910,
10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0060583002', 'The Last Season', 'Eric Blehm', 2006, 902, 10.0);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978- 0316740401', 'Case Histories: A Novel', 'Kate Atkinson', 2006, 813,
10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978- 0316013949', 'Step on a Crack', 'James Patterson, et al.',2007, 813,
10.0);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES ('978-
0374105235', 'Long Way Gone: Memoirs of a Boy Soldier', 'Ishmael Beah',
2007, 916,10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0385340229', 'Sisters', 'Danielle Steel', 2006, 813, 10.0);
INSERT INTO BookMedia(media_id, ISBN) VALUES (8733, '978-
0743289412'); INSERT INTO BookMedia(media_id, ISBN) VALUES
(9982, '978-1596912366'); INSERT INTO BookMedia(media_id, ISBN)
VALUES (3725, '978-1596912366'); INSERT INTO BookMedia(media_id,
ISBN) VALUES (2150, '978-0312351588'); INSERT INTO
BookMedia(media_id, ISBN) VALUES (4188, '978-0156031561'); INSERT
INTO BookMedia(media_id, ISBN) VALUES (5271, '978-0060583002');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2220, '978-
0316740401'); INSERT INTO BookMedia(media_id, ISBN) VALUES
(7757, '978-0316013949'); INSERT INTO BookMedia(media_id, ISBN)
VALUES (4589, '978-0374105235'); INSERT INTO BookMedia(media_id,
ISBN) VALUES (5748, '978-0385340229');
INSERT INTO Checkout(media_id, num, since, until) VALUES (2220, 9780749, '02/15/2007',
'03/15/2007');
INSERT INTO Video(title, year, director, rating, price) VALUES
('Terminator2: Judgment Day', 1991, 'James Cameron', 8.3, 20.0);
 INSERT INTO Video(title, year, director, rating, price) VALUES
('Raiders ofthe Lost Ark', 1981, 'Steven Spielberg', 8.7, 20.0);
 INSERT INTO Video(title, year, director, rating, price) VALUES
('Aliens',1986, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES ('Die
Hard',1988, 'John McTiernan', 8.0, 20.0);
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 1734, 'Terminator 2: Judgment Day', 1991);
INSERT INTO VideoMedia(media_id, title, year) VALUES ( 5725,
'Raiders of the Lost Ark', 1981);
INSERT INTO VideoMedia(media_id, title, year) VALUES ( 1716,
'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES ( 8388,
'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES ( 8714,
'DieHard', 1988);
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(1716,4444172, 'Texas Branch', '02/20/2008', 1);
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(2591051,88564, 30000.00, 'Texas Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name)
```

```
Values(6190164, 64937, 30000.00, 'Illinois Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name)
Values (1810386, 58359, 30000.00, 'Louisiana
Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8733, 'Texas
Branch'); INSERT INTO Stored_In(media_id, name) VALUES(9982,
'Texas Branch'); INSERT INTO Stored_In(media_id, name)
VALUES(1716, 'Texas Branch'); INSERT INTO Stored_In(media_id,
name) VALUES(1734, 'Texas Branch'); INSERT INTO
Stored_In(media_id, name) VALUES(4589, 'Texas Branch'); INSERT
INTO Stored_In(media_id, name) VALUES(4188, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5271, 'Illinois
Branch'); INSERT INTO Stored_In(media_id, name) VALUES(3725,
'Illinois Branch'); INSERT INTO Stored_In(media_id, name)
VALUES(8388, 'Illinois Branch'); INSERT INTO Stored_In(media_id,
name) VALUES(5748, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(2150, 'Louisiana
Branch'); INSERT INTO Stored_In(media_id, name) VALUES(8714,
'Louisiana Branch'); INSERT INTO Stored_In(media_id, name)
VALUES(7757, 'Louisiana Branch'); INSERT INTO Stored_In(media_id,
name) VALUES(5725, 'Louisiana Branch');
SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,nvl((SELECT
'Librarian' FROM Librarian L WHERE L.ID = C.ID), 'Customer') AS role
FROM Customer C WHERE C.username = <user input> AND C.password =
<user input>;
/* Book search for customers */
SELECT B.ISBN, B.title, B.author, B.year,(SELECT COUNT(*) FROM
BookMedia BM WHERE BM.ISBN = B.ISBN AND BM.code = 1) AS
num_available FROM
```

```
Book B WHERE B.title LIKE '%<user input>%' AND B.author LIKE
'%<user input>%' AND B.year <= <user input> AND B.year >=
<userinput>;
/* Find all copies of a book (used for placing holds or viewing
detailedinformation). */
SELECT BM.media_id, S.description, nvl((SELECT SI.name FROM
Stored_In SIWHERE SI.media_id = BM.media_id), 'none') AS name
FROM BookMedia BM, Media M, Status S
WHERE BM.ISBN = <user input> AND M.media_id = BM.media_id AND S.code = M.code;
/* Video search for customers */
SELECT V.title, V.year, V.director, V.rating (SELECT COUNT(*) FROM
VideoMedia VMWHERE VM.ID = V.ID AND VM.code = 1) AS num_available
FROM Video V WHERE V.title LIKE '%<user input>%' AND V.year
<= <user input> AND V.year <= <user input>
AND V.director LIKE '%<user input>%' AND V.rating >= <user input>;
/* Find all copies of a video (used for placing holds or viewing detailed
information). */ SELECT VM.media_id, S.description, nvl((SELECT
SI.name FROM Stored_In SI WHERE SI.media_id = VM.media_id),
'none') AS name FROM VideoMedia VM,Media M, Status S
WHERE VM.title = <user input> AND VM.year = <user input> AND
M.media_id = VM.media_id AND S.code = M.code;
/* Find the status of a given media item */
SELECT S.description FROM Status S, Media M WHERE S.code = M.code AND
M.media_id = <userinput>;
 /* Create a new Hold */
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(<user input>, <user input>, <user input>, <user input>,
nvl((SELECT MAX(H.queue) FROM Hold H WHERE
H.media_id = <user input>), 0)
+ 1 );
```

```
/* Cancel Hold, Step 1: Remove the entry from hold
*/DELETE FROM Hold
WHERE media_id = <user input> AND num = <user input>
/* Cancel Hold, Step 2: Update queue for this item */
UPDATE HoldSET queue = queue-1

WHERE media_id = <user input> AND queue > <user
input>; /* Functions needed to view information about a
customer */ /* View the customer's card(s) */ SELECT
CR.num, CR.fines
FROM Card CR
WHERE CR.ID = <user input>;
/* View media checked out on a given card */
SELECT B.title, B.author, B.year, BM.media_id, CO.since,
CO.untilFROM Checkout CO, BookMedia BM, Book B
WHERE CO.num = <user input> AND CO.media_id = BM.media_id AND B.ISBN
=BM.ISBN UNION
SELECT V.title, V.director, V.year, VM.media_id, CO.since,
CO.untilFROM Checkout CO, VideoMedia VM, Book B
WHERE CO.num = <user input> AND CO.media_id = VM.media_id
ANDVM.title = V.title AND VM.year = V.year;
/* View media currently on hold for a given card */
SELECT B.title, B.author, B.year, BM.media_id, H.until, H.queue, SI.name
FROMHold H, BookMedia BM, Book B, Stored_In SI
WHERE H.num = <user input> AND H.media_id = BM.media_id AND B.ISBN =
BM.ISBN
AND SI.media_id =
H.media_idUNION
SELECT V.title, V.director, V.year, VM.media_id, H.until, H.queue, SI.name
FROMHold H, VideoMedia VM, Book B, Stored_In SI
```

```
WHERE H.num = <user input> AND H.media_id = VM.media_id AND
VM.title =V.title AND VM.year = V.year AND SI.media_id = H.media_id;
/* View the total amount of fines the customer has to pay */ SELECT
SUM(CR.fines)
FROM Card CR
WHERE CR.ID = <user input>;
/* *\
Functions reserved for librarians
\* */
/* Add new customer */
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(<userinput>, <user input>, <user input>, <user input>, <user input>, <user input>, <user
input>, );
/* Find a customer */
SELECT   C.ID,   C.name,   C.addr,   C.DOB,   C.phone,
C.username,nvl((SELECT 'Librarian'
FROM Librarian L
WHERE L.ID = C.ID), 'Customer') AS
roleFROM Customer C
WHERE C.username = <user input> AND C.name LIKE '%<user input>%';
/* Addnew card and assign it to a customer */
INSERT INTO Card(num, fines, ID) VALUES ( <user input>, 0, <user input>);
/*Create an entry in Checkout */
INSERT INTO Checkout(media_id, num, since, until) VALUES
(<user input>, <user input>, <user input>, <user input>); /*
Remove the entryfor Stored_In */
DELETE    FROM    Stored_In
WHERE    media_id    =    <user
input>;
```

```
/* Change the status code of the
media */UPDATE Media
SET code = <user input>
WHERE media_id = <user input>;
/* Remove the entry from
Checkout */ DELETE FROM
Checkout
WHERE media_id = <user input>;
/* Create the entry in Stored_In */
INSERT INTO Stored_In(media_id, name) VALUES (<user input>, <user input>);
/* Findthe next Hold entry for a given media */ SELECT H.num, H.name, H.until

FROM Hold H
WHERE H.queue = 1 AND H.media_id = <user input>;
/* Change the Stored_In entry to the target library
branch */UPDATE Stored_In
SET name = <user input>
WHERE media_id = <user
input>;
/* Find the customer that should be notified about book arrival */
SELECT C.name, C.phone, CR.num FROM Customer C, Card
CR, Hold H
WHERE H.queue = 1 AND H.name = <user input> AND H.media_id = <user
input> ANDCR.num = H.num AND C.ID = CR.ID;
```

/* Add a new entry into the Book table */

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES

(<user input>, <user input>, <user input>, <user input>, <user input>,

<user input>);

/* Add a new entry into the Video table */

INSERT INTO Video(title, year, director, rating, price) VALUES

(<user input>, <user input>, <user input>, <user input>, <user input>);

/* Add a newMedia object */

INSERT INTO Media(media_id, code) VALUES (<user input>, 1);

/* Adda new BookMedia object */

INSERT INTO BookMedia(media_id, ISBN) VALUES (<user input>, <user

input>); /* Adda new VideoMedia object */

INSERT INTO VideoMedia(media_id, title, year)

VALUES(<user input>, <user input>, <user input>);

/* Remove an entry from the BookMedia

table */DELETE FROM BookMedia WHERE

media_id =

<user input>;

/* Remove an entry from the VideoMedia

table */DELETE FROM VideoMedia WHERE

media_id =

<user input>;

/* Remove an entry from the Media

table */DELETE FROM Media

WHERE media_id = <user input>;

/* Remove an entry from the Book

table */ DELETE FROM Book

WHERE ISBN = <user input>;

/* Remove an entry from the Video

table */DELETE FROM Video

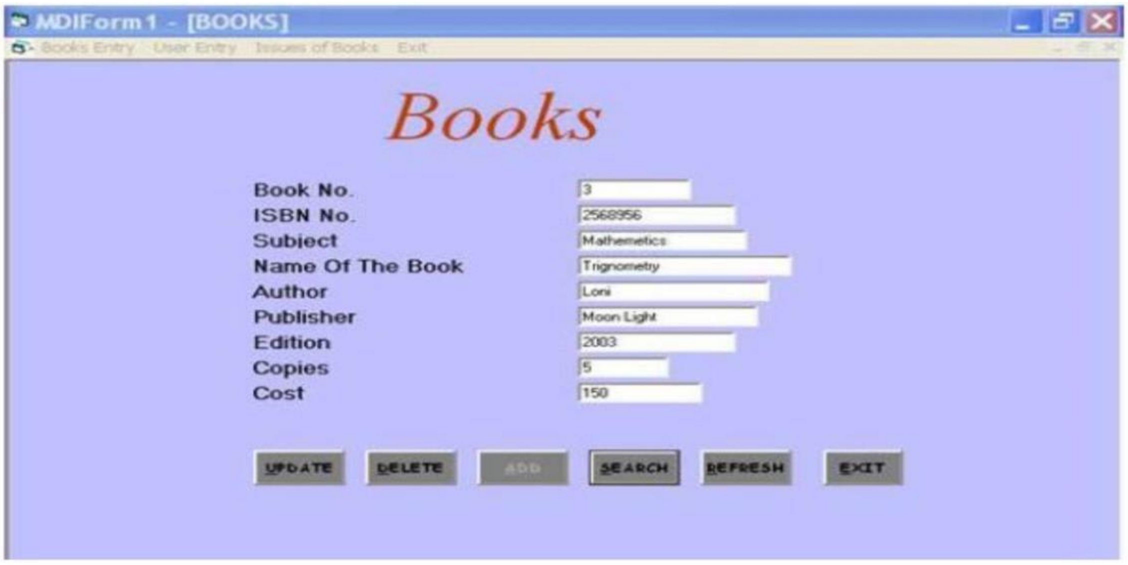
WHERE title = <user input> AND year = <user

input>; /*Update the customer's fines */ UPDATE Card

SET fines = <user input>

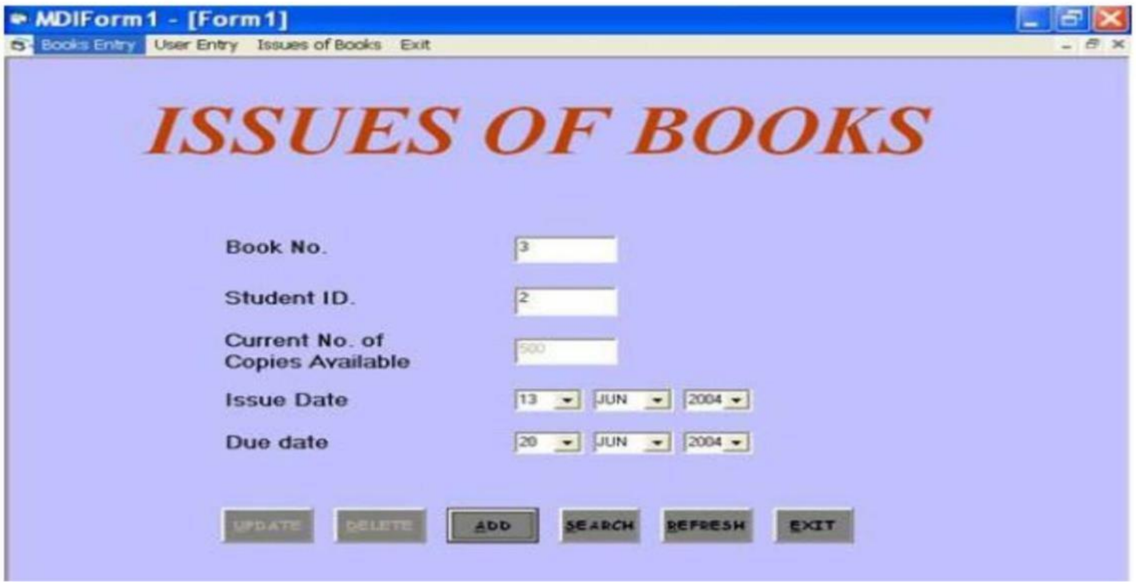
WHERE num = <user

input>



Books

Book No.	3
ISBN No.	2568956
Subject	Mathemetics
Name Of The Book	Trignometry
Author	Loni
Publisher	Moon Light
Edition	2003
Copies	5
Cost	150



ISSUES OF BOOKS

Book No.	3
Student ID.	2
Current No. of Copies Available	500
Issue Date	13 JUN 2004
Due date	20 JUN 2004

RESULT:

Thus the library management System by using the front end tools was executed successfully.

EX.NO:18 SIMPLE CALCULATOR

AIM :

To implement a simple calculator by using Visual Basic front end tools.

PROCEDURE:

Step1: create a new project in visual basic using the option file---> new project.

Step2: In the form use the front end tools in the toolbox like textbox, label,command button andcreate a front end Design for the simple calculator.

Step3: Open the properties window for the tool sand select properties. Now the propertieswindow is opened.

Step4: Set properties for each tool in the form like caption, name, etc.

Step5: Double click each and every tool to open the project code

window.Step6: write the code for the events of the tools.

Step7: write the code for the simple operations in the calculator like Addition, subtraction,

multiplicationand division.

Step7: The code is Automatically compiled at the end of each line while pressing the

Enter key.Step7: now execute the code by click the F5 button in the keyboard or select

Run--->start.

Step8: after successfully executing the project create the executable file

bySelect the option file---> make file.exe.

CODING:

```
Dim a, b, c, d As Integer

Private                    Sub

button0_Click()

display.Text      =      display.Text      +

button0.CaptionEnd Sub
```

```
Private Sub button1_Click()
display.Text        =        display.Text        +

button1.CaptionEnd Sub

Private Sub button2_Click()

display.Text        =        display.Text        +

button2.CaptionEnd Sub

Private Sub button3_Click()

display.Text        =        display.Text        +

button3.CaptionEnd Sub
Private Sub button4_Click()

display.Text        =        display.Text        +

button4.CaptionEnd Sub
Private Sub button5_Click()

display.Text        =        display.Text        +

button5.CaptionEnd Sub

Private Sub button6_Click()

display.Text        =        display.Text        +

button6.CaptionEnd Sub

Private Sub button7_Click()

display.Text        =        display.Text        +

button7.CaptionEnd Sub

Private Sub button8_Click()

display.Text        =        display.Text        +

button8.CaptionEnd Sub

Private Sub button9_Click()

display.Text        =        display.Text        +

button9.CaptionEnd Sub
```

```
Private                  Sub
add_Click()      a       =
Val(display.Text)
display.Text = ""d = 1
End Sub
Private                  Sub
sub_Click()      a       =
Val(display.Text)
display.Text = ""
d   =   2
End
Sub
Private                  Sub
mul_Click()      a       =
Val(display.Text)
display.Text = ""
d   =   3
End
Sub
Private Sub div_Click()
a                        =
Val(display.Text)
display.Text = ""
d   =   4
End
Sub
```

```
Private Sub equalto_Click()

b = Val(display.Text)

If   d   =   1

Thenc = a +

b

display.Text = c

ElseIf   d   =   2

Thenc = a - b

display.Text = c

ElseIf   d   =   3

Thenc = a * b

display.Text = c

ElseIf   d   =   4

Then c  =  a  /  b

display.Text = c

End If

End Sub

Private                    Sub

clear_Click()a = 0

b = 0

c = 0 display.Text = ""

End Sub
Private Sub off_Click()

MSG = MsgBox("THANKS FOR USING FX990ES FROM NASA COPY RIGHTS
RESERVED", vbOKOnly, "BYE")

End

End
```

```
Sub

Private Sub decimalpoint_Click()

display.Text          =          display.Text          +

decimalpoint.CaptionEnd Sub
```

RESULT:

Thus the simple calculator created by using the front end tools was executed successfully.