DBMS Lab Manual

1. DDL COMMANDS

AIM:

To write a query with a functions of DDL commands such as create, alter, drop.

DDL COMMANDS:

1. CREATE A TABLE

Syntax: create table tablename(columnname1 datatype1, columnname2 datatype2,…)

Q1: Create a table student with following fields

Name	Data Type	
Rollno	number(6)	primary key
Name	varchar(15)	
Dept	varchar(4)	
City	varchar(15)	
DOB	date	NOT NULL
Gender	char(1)	

Query:

SQL> create table student (rollno number(6) primary key,name varchar(15), dept varchar(4),city varchar(15),dob date not null,gender char(1));

Output:

Table Created

SQL>desc student

```
Name                     Null?          Type
-------------------------------------- -------- ----------------------------
ROLLNO                   NOT NULL       NUMBER(6)
NAME                                    VARCHAR2(15)
DEPT                                    VARCHAR2(4)
CITY                                    VARCHAR2(15)
DOB                      NOT NULL       DATE
GENDER                                  CHAR(1)
```

Q2: Create a table studmarks with following attributes

Name	**datatype**
Rollno	number(6)
Regnumber	number(14)
Semester	number(1)
CGPA	number(2,4)

Query:

SQL> create table studmarks(rollno number(6),regnumber number(14), semester number(1),cgpa number(2,4));

Output:

Table created.

SQL> desc studmarks;

```
 Name                              Null?              Type
 --------------------------------------- -------- ----------------------------
 ROLLNO                                             NUMBER(6)
 REGNUMBER                                          NUMBER(14)
 SEMESTER                                           NUMBER(1)
 CGPA                                               NUMBER(2,4)
```

2. DESCRIBE THE SCHEMA OF THE TABLE

Syntax: desc tablename;

Q: Describe the table student

Query:

SQL> desc student;

Output:

```
 Name                              Null?              Type
 --------------------------------------- -------- ----------------------------
  ROLLNO                       NOT NULL          NUMBER(6)
 NAME                                               VARCHAR2(15)
 DEPT                                               VARCHAR2(4)
```

```
CITY                              VARCHAR2(15)
DOB                NOT NULL       DATE
GENDER                            CHAR(1)
```

3. ALTER THE TABLE

Syntax: alter table tablename add/drop/modify(columnname datatype);

Q1: Add the constraint UNIQUE for regnumber attribute from studmarks table

Query:

```
SQL> alter table studmarks add constraint s unique(regnumber);
```

Output:

```
Table altered.
```

Q2: Remove the constraint for the regnumber attribute

Query:

```
SQL> alter table studmarks drop constraint s;
```

Output:

```
Table altered.
```

Q3: Add foreign key constraint for the column rollno from studmarks that refers rollno from student table.

Query:

```
SQL> alter table studmarks add foreign key(rollno) references student(rollno);
```

Output:

```
Table altered.
```

Q4: Add one more column age in student table with NOT NULL constraint in student table

Query:

```
SQL> alter table student add(age number(2) not null);
```

Output:

```
Table altered.
```

```
SQL> desc student;
Name                  Null?          Type
```

```
 -------------------------------------- -------- ---------------------------
 ROLLNO                    NOT NULL           NUMBER(6)
 NAME                                         VARCHAR2(15)
 DEPT                                         VARCHAR2(4)
 CITY                                         VARCHAR2(15)
 DOB                       NOT NULL           DATE
 GENDER                                       CHAR(1)
 AGE                       NOT NULL           NUMBER(2)
```

Q5: Remove the column city from the student table

Query:

SQL> alter table student drop(city);

Output:

Table altered.

SQL> desc student;

```
Name                             Null?                Type
 -------------------------------------- -------- ---------------------------
 ROLLNO                    NOT NULL        NUMBER(6)
 NAME                                      VARCHAR2(15)
 DEPT                                      VARCHAR2(4)
 DOB                       NOT NULL        DATE
 GENDER                                    CHAR(1)
 AGE                       NOT NULL        NUMBER(2)
```

Q6: Modify the data type of regnumber to varchar(16)

Query:

SQL> alter table studmarks modify(regnumber varchar(16));

Output:

Table altered.

SQL> desc studmarks;

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

```
ROLLNO                                  NUMBER(6)
REGNUMBER                               VARCHAR2(16)
SEMESTER                                NUMBER(1)
CGPA                                    NUMBER(2,4)
```

4. RENAME THE TABLE

Syntax: rename oldtablename to newtablename;

Q1: Change the name of the table student to stud

Query:

SQL> rename student to stud;

Output:

```
Table renamed.
```

SQL> desc student;

```
ERROR:
ORA-04043: object student does not exist
```

SQL> desc stud;

```
 Name                           Null?        Type
 ----------------------------------------- -------- ----------------------------
 ROLLNO                         NOT NULL     NUMBER(6)
 NAME                                        VARCHAR2(15)
 DEPT                                        VARCHAR2(4)
 DOB                            NOT NULL     DATE
 GENDER                                      CHAR(1)
 AGE                            NOT NULL     NUMBER(2)
```

Q2: Change the name of the attribute dob to dateofbirth

Query:

SQL> alter table stud rename column dob to dateofbirth;

Output:

SQL> desc stud;

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

```
ROLLNO              NOT NULL     NUMBER(6)
NAME                             VARCHAR2(15)
DEPT                             VARCHAR2(4)
DATEOFBIRTH         NOT NULL     DATE
GENDER                           CHAR(1)
AGE                 NOT NULL     NUMBER(2)
```

5. DROP THE TABLE

Syntax: drop table tablename;

Q: Drop the table stud

Query:

SQL> drop table stud;

Output:

Table dropped.

SQL> desc stud;

ERROR:

ORA-04043: object stud does not exist

RESULT:

Thus the query using DDL commands was executed successfully.

2. DML COMMANDS

AIM:

To write a query with a functions of DML commands such as create, alter, drop.

DML COMMANDS:

1. CREATE A TABLE

Q1: Create a table books with attributes bookno, title, publication, author, price, quantity,edition

Query:

SQL> create table bokks(bookno number(3),title varchar(25),publication varchar(25),author varchar(25),price number(6,2),quantity number(3),edition number(2));

Output:

Table Created

SQL>desc books;

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BOOKNO                                             NUMBER(3)
 TITLE                                              VARCHAR2(25)
 PUBLICATION                                        VARCHAR2(25)
 AUTHOR                                             VARCHAR2(25)
 PRICE                                              NUMBER(6,2)
 QUANTITY                                           NUMBER(3)
 EDITION                                            NUMBER(2)
```

2. INSERTING A RECORD

Q1: Insert few records into the table employee.

Query:

SQL> insert into books values(&bookno,'&title','&publication','&author',&price,&quantity,&edition);

```
Enter value for bookno: 1
Enter value for title: database concepts
Enter value for publication: tata
Enter value for author: balagurusamy
Enter value for price: 850.00
Enter value for quantity: 10
Enter value for edition: 4
```

Output:

```
old   1: insert into books values(&bookno,'&title','&publication','&author',&price,&quantity,&edit
new   1: insert into books values(1,'database concepts','tata','balagurusamy',800.00,10,4)
1 row created.
```

Query:

SQL> /

```
Enter value for bookno: 2
Enter value for title: database system
Enter value for publication: tata
Enter value for author: silberschatz
Enter value for price: 700.00
Enter value for quantity: 17
Enter value for edition: 3
```

Output:

```
old   1: insert into books values(&bookno,'&title','&publication','&author',&price,&quantity,&edit
new   1: insert into books values(2,'database system','tata','silberschatz',700.00,17,3)
1 row created.
```

3. SELECT THE RECORDS

Q: Describe the table student

Query:

SQL>select * from books;

Output:

```
 BOOKNO     TITLE               PUBLICATION      AUTHOR           PRICE  QUANTITY   EDITION
------------------------ ---------- ---------- ------------------------------------------------------------------------
1        database concepts        tata                 balagurusamy        850            10      4
2        database system          tata                 silberschatz        700            17      3
3        database oracle          s.chand & co         s.s.khandars         500           12      2
4        mastering of database    bpb publications     ivan bayross          900          18      1
```

```
5         database system                 prentice hall           jeffrey d.ullman          550              20       2
```

Q2: Show the list of titles with their authors.

Query:

SQL> select title,author from books;

Output:

```
TITLE                  AUTHOR
------------------------ ------------------------
database concepts              balagurusamy
database system                silberschatz
database oracle                dr.s.s.khandare
mastering of database          ivan bayross
database system                jeffrey d.ullman
```

Q3: List various authors for the book title 'database system'

Query:

SQL> select * from books where title='database system';

Output:

```
BOOKNO        TITLE                PUBLICATION      AUTHOR           PRICE   QUANTITY   EDITION
------------------------ ---------- ---------- -----------------------------------------------------------------------
2         database system                 tata                    silberschatz             700              17       3
5         database system                 prentice hall           jeffrey d.ullman          550              20       2
```

Q4: Show the authors details for the table books

Query:

SQL> select author from books;

Output:

```
AUTHOR
------------------------
balagurusamy
silberschatz
dr.s.s.khandare
```

```
ivan bayross
jeffrey d.ullman
```

Q5: Select the list of book details whose price is greater than 800.

Query:

SQL> select * from books where price>800.00;

Output:

```
BOOKNO     TITLE              PUBLICATION     AUTHOR          PRICE  QUANTITY  EDITION
------------------------ ---------- ---------- --------------------------------------------------------------------------
1       database concepts       tata              balagurusamy        850          10     4
4       mastering of database   bpb publications  ivan bayross        900          18     1
```

Q6: List the details of books which have more than 15 copies in the order of price.

Query:

SQL> select * from books where quantity>15;

Output:

```
  BOOKNO   TITLE              PUBLICATION     AUTHOR          PRICE  QUANTITY  EDITION
------------------------ ---------- ---------- --------------------------------------------------------------------------
2       database system         tata              silberschatz         700         17     3
4       mastering of database   bpb publications  ivan bayross         900         18     1
5       database system         prentice hall     jeffrey d.ullman     550         20     2
```

Q7: Display the list of books published by the author ‘balagurusamy’ with the publication ‘TATA’

Query:

SQL> select * from books where author='balagurusamy' and publication='tata';

Output:

```
BOOKNO     TITLE              PUBLICATION     AUTHOR          PRICE  QUANTITY  EDITION
------------------------ ---------- ---------- --------------------------------------------------------------------------
1       database concepts       tata              balagurusamy        850          10     4
```

Q8: List the names of the books that consists of 'database concepts'

Query:

SQL> select title from books where title='database concepts'

Output:

```
Table renamed.
SQL> desc student;
ERROR:
ORA-04043: object student does not exist
SQL> desc stud;
 Name                              Null?              Type
 --------------------------------------- -------- ----------------------------
 ROLLNO                         NOT NULL         NUMBER(6)
 NAME                                            VARCHAR2(15)
 DEPT                                            VARCHAR2(4)
 DOB                            NOT NULL         DATE
 GENDER                                          CHAR(1)
 AGE                            NOT NULL         NUMBER(2)
```

Q2: Change the name of the attribute dob to dateofbirth

Query:

SQL> alter table stud rename column dob to dateofbirth;

Output:

```
SQL> desc stud;
 Name                              Null?                   Type
 --------------------------------------- -------- ----------------------------
 ROLLNO                         NOT NULL           NUMBER(6)
  NAME                                             VARCHAR2(15)
 DEPT                                              VARCHAR2(4)
 DATEOFBIRTH                    NOT NULL           DATE
 GENDER                                            CHAR(1)
 AGE                            NOT NULL           NUMBER(2)
```

5. DROP THE TABLE

Syntax: drop table tablename;

Q: Drop the table stud

Query:

SQL> drop table stud;

Output:

Table dropped.

SQL> desc stud;

ERROR:

ORA-04043: object stud does not exist

RESULT:

Thus the query using DDL commands was executed successfully.

3. NESTED QUERIES AND JOIN QUERIES

AIM:

To write and execute SQL queries for the nested and join queries.

NESTED QUERIES OR SUB QUERIES:

Syntax:

```
Select<column(s)>from table where<condn operator>
                  (select <column>from table);
```

Q1: Create a table employee with attributes ssn, name,bdate,salary,mgrssn,dno with appropriate data type. Insert few records in to the table employee.

Query:

```
select * from employee;
```

Output:

```
SSN    NAME       BDATE          SALARY       MGRSSN     DNO
---------- ------------- --------- ---------- ---------- -------------------------------
1111   sathya        17-DEC-88    50000        4323          3
1112   vinotha       02-AUG-90    32000        5462          1
1113   nandu         07-OCT-93    30000        6452          2
1114   rajesh        05-APR-85    40000        8264          2
1115   nive          06-OCT-92    45000        7241          1
```

Also create a table department with attributes dno, dname, loc. Insert few records into the department table.

Query:

```
select * from department;
```

Output:

```
     DNO       DNAME       LOC
  ----------------------------------------------
     1           admin       madurai
     2          research      chennai
     3          accounts        bangalore
```

Q2: Display the names of the employees working for Accounts department.

Query:

SQL> select name from employee where dno=(select dno from department where dname='accounts');

Output:

```
 NAME
---------------
 sathya
```

Q2: Display names of employees whose salary is greater than the employee SSN=1234

Query:

SQL> select name from employee where salary>(select salary from employee where ssn=1234);

Output:

```
NAME
------------
sathya
vinotha
rajesh
nive
```

Q3: Display all the employees drawing more than or equal to the average salary of department number 3.

Query:

SQL> select name from employee where salary>=(select avg(salary) from employee group by dno having dno=3);

Output:

```
NAME
--------------
sathya
```

Q4: Display the name of the highest paid employee.

Query:

```
SQL> select name from employee where salary=(select max(salary) from employee);
```

Output:

```
NAME
--------------
sathya
```

Q5: Find the Name and Salary of people who draw in the range Rs. 20,000 to Rs. 40,000.

Query:

```
SQL> select name, salary from employee where salary in(select salary from employee where salary
between 20000 and 40000);
```

Output:

```
NAME            SALARY
----------------------------------
vinotha          32000
nandu            30000
rajesh           40000
```

Q6: Update the salary by 0.25 times for all employees who works in research department.

Query:

```
SQL> update employee set salary=(salary*0.25)where dno=(select dno from department where
dname='research');
```

Output:

```
2 rows updated.
SQL> select * from employee;
SSN    NAME       BDATE         SALARY      MGRSSN     DNO
---------- ------------- --------- ---------- ---------- -------------------------------
1111   sathya       17-DEC-88    50000       4323         3
1112   vinotha      02-AUG-90   32000        5462         1
1113   nandu        07-OCT-93    37500       6452         2
1114   rajesh       05-APR-85    50000       8264         2
```

```
1115   nive              06-OCT-92    45000           7241             1
```

Q7: Delete all the employee details from admin department.

Query:

SQL> delete from employee where dno=(select dno from department where dname='admin');

Output:

```
2 rows deleted.
```

SQL> select * from employee;

```
      SSN NAME          BDATE        SALARY     MGRSSN       DNO
---------- ------------- --------- ---------- ---------- ----------
     1111 sathya        17-DEC-88      50000           4323             3
     1113 nandu         07-OCT-93     30000           6452             2
     1114 rajesh        05-APR-85     40000           8264             2
```

Q8: Display the department name in which employee that has highest salary.

Query:

SQL> select dname from department where dno=(select dno from employee where salary=(select max(salary)from employee);

Output:

```
     DNAME
   ---------------
    accounts
```

Q9: Display the employee details of all employees who earn more than that of ‘nandu’ and in the same department as ‘sathya’

Query:

SQL> select name from employee where salary>(select salary from employee where name= ‘nandu’)and dno=(select dno from employee where name= ‘sathya’);

Output:

```
  SSN    NAME          BDATE        SALARY     MGRSSN       DNO
---------- ------------- --------- ---------- ---------- ----------
```

```
1111  sathya     17-DEC-88   50000      4323      3
1112  vinotha    02-AUG-90   32000      5462      1
1113  nandu      07-OCT-93   37500      6452      2
1114  rajesh     05-APR-85   50000      8264      2
1115  nive       06-OCT-92   45000      7241      1
```

JOIN QUERIES

An SQL join clause combines records from two or more tables in a database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables by using values common to each. Types are:

- ✓ Cross join
- ✓ Inner Join
  - Equi Join
    - Natural Join
- ✓ Outer Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

(i). Cross Join

CROSS JOIN returns the Cartesian product of rows from tables in the join. In other words, it will produce rows which combine each row from the first table with each row from the second table

Syntax:

Select * from table1 cross join table2;

(or)

Select * from table1,table2;

Query:

SQL> select * from employee cross join department;

Output:

```
   SSN      NAME      BDATE      SALARY   MGRSSN    DNO     DNO   DNAME     LOC
--------------------------------------------------------------------------------------
```

```
     1111      sathya     17-DEC-88    50000    4323        3        1       admin      madurai
     1111      sathya     17-DEC-88    50000     4323         3      2       research    chennai
     1111      sathya     17-DEC-88    50000     4323         3      3       accounts   bangalore
     1112      vinotha    02-AUG-90   32000     5462         1      1       admin      madurai
     1112      vinotha    02-AUG-90   32000     5462         1      2       research    chennai
     1112      vinotha    02-AUG-90   32000     5462         1      3       accounts   bangalore
     1113      nandu      07-OCT-93    37500     6452         2      1       admin      madurai
     1113      nandu      07-OCT-93    37500     6452         2      2       research    Chennai
     1113      nandu      07-OCT-93    37500     6452         2      3       accounts   Bangalore
     1114      rajesh      05-APR-85   50000     8264         2      1       admin      madurai
     1114      rajesh      05-APR-85   50000     8264         2      2       research    chennai
     1114      rajesh      05-APR-85   50000     8264         2      3       accounts   bangalore
     1115      nive       06-OCT-92    45000     7241         1      1       admin      madurai
     1115      nive       06-OCT-92    45000     7241         1      2       research    chennai
     1115      nive       06-OCT-92    45000     7241         1      3       accounts   bangalore
```

ii). Inner Join

Inner join creates a new result table by combining column values of two tables (A and B) based upon the join-predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. The result of the join can be defined as the outcome of first taking the Cartesian product (or Cross join) of all records in the tables (combining every record in table A with every record in table B)—then return all records which satisfy the join predicate

Syntax:

Select * from table1 inner join table2 on table1.column=table2.column;

Query:

SQL> select * from employee inner join department on employee.dno=department.dno;

Output:

```
SSN     NAME        BDATE       SALARY    MGRSSN      DNO      DNO    DNAME      LOC
------- --------------- ------------------ --------------- ------------- ----------------- ------------------- ------------
 1115   nive         06-OCT-92     45000            7241          1         1         admin      madurai
 1112  vinotha     02-AUG-90    32000            5462        1        1        admin      madurai
 1114  rajesh      05-APR-85    50000            8264        2        2        research    chennai
 1113  nandu       07-OCT-93    37500            6452        2        2        research    chennai
 1111  sathya      17-DEC-88    50000            4323        3        3        accounts    bangalore
```

(iii). Equi Join

An **equi-join** is a specific type of comparator-based join, that uses only equality comparisons in the join-predicate. Using other comparison operators (such as <) disqualifies a join as an equi-join.

Syntax:

Select * from table1 join table2 on table1.column=table2.column;

Query:

SQL> select * from employee join department on employee.dno=department.dno;

Output:

```
  SSN      NAME     BDATE     SALARY    MGRSSN      DNO     DNO   DNAME     LOC
--------------------------------------------------------------------------------------------------------------
  1114      rajesh     05-APR-85     50000      8264           2          2      research    chennai
  1113      nandu      07-OCT-93    37500       6452           2         2      research    chennai
  1111      sathya     17-DEC-88     50000      4323           3          3    accounts    bangalore
  1111      sathya     17-DEC-88     50000      4323           3          3    accounts    bangalore
  1111      sathya     17-DEC-88     50000      4323           3          3    accounts    bangalore
```

(iv). Natural Join

A natural join is a type of equi-join where the join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables. The resulting joined table contains only one column for each pair of equally named columns.

Syntax:

Select * from table1 natural join table2 ;

Query:

Sql> select * from employee natural join department;

Output:

```
DNO      SSN  NAME        BDATE       SALARY    MGRSSN        DNAME      LOC
----------------------------------------------------------------------------------------------------------
```

```
1        1115   nive        06-OCT-92        45000         7241               admin      Madurai
1        1112   vinotha      02-AUG-90      32000         5462               admin      Madurai
2        1114   rajesh       05-APR-85      50000         8264                research    chennai
2        1113   nandu        07-OCT-93      37500         6452               research    chennai
3        1111   sathya       17-DEC-88       50000          4323               accounts    bangalore
```

(v). Outer Join

An **outer join** does not require each record in the two joined tables to have a matching record. The joined table retains each record—even if no other matching record exists. Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table's rows are retained (left, right, or both).

SQL> select * from employee;

```
     SSN     NAME      BDATE       SALARY    MGRSSN       DNO
   ---------------------------------------------------------------------------------------
     1115        nive        06-OCT-92     45000           7241            1
     1112        vinotha     02-AUG-90    32000          5462            1
     1114        rajesh       05-APR-85    50000          8264           2
     1113        nandu        07-OCT-93    37500          6452           2
     1111        sathya      17-DEC-88     50000         4323            3
     1116        palani       06-OCT-60     90000         6372           4
```

SQL> select * from department;

```
   DNO    DNAME      LOC
----------------------------------------
       1        admin       chennai
       2        research    banglore
       3       accounts    mumbai
       5       sales         trichy
```

Left Outer Join

The result of a *left outer join* (or simply **left join**) for table A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B).

Syntax:

Select * from table1 left outer join table2 on table1.column=table2.column;

Query:

Sql> select * from employee left outer join department on employee.dno=department.dno;

Output:

```
 SSN    NAME      BDATE      SALARY    MGRSSN    DNO    DNO    DNAME     LOC
---------------------------------------------------------------------------------------------------------------------
 1115    nive        06-OCT-92     45000      7241        1           1            admin      madurai
 1112    vinotha      02-AUG-90     32000      5462        1           1            admin      madurai
 1114    rajesh        05-APR-85     50000      8264        2           2          research    chennai
 1113    nandu        07-OCT-93     37500      6452        2           2          research    chennai
 1111     sathya       17-DEC-88    50000      4323        3           3         accounts   bangalore
 1116     palani       06-OCT-60     90000      6372        4
```

Right Outer Join

A right outer join returns all the values from the right table and matched values from the left table (NULL in case of no matching join predicate).

Syntax:

Select * from table1 right outer join table2 on table1.column=table2.column;

Query:

Sql> select * from employee right outer join department on employee.dno=department.dno;

Output:

```
 SSN    NAME       BDATE       SALARY    MGRSSN      DNO      DNO        DNAME      LOC
------------ ------------ ---------- ------------ ---------------- --------------- ------------------ ----------------
 1111   sathya       17-DEC-88     50000      4323          3          3          accounts    bangalore
 1112   vinotha       02-AUG-90    32000      5462          1         1           admin       madurai
 1113   nandu        07-OCT-93     37500      6452          2         2           research    chennai
 1114   rajesh       05-APR-85     50000      8264          2         2           research    chennai
 1115   nive          06-OCT-92     45000      7241          1          1           admin       madurai
                                                                      5          madical     delhi
```

Full Outer Join

Conceptually, a **full outer join** combines the effect of applying both left and right outer joins. Where records in the FULL OUTER JOINed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row. For those records that do match, a single row will be produced in the result set (containing fields populated from both tables).

Syntax:

Select * from table1 full outer join table2 on table1.column=table2.column;

Query:

Sql> select * from employee full outer join department on employee.dno=department.dno;

Output:

```
 SSN    NAME     BDATE       SALARY    MGRSSN     DNO    DNO     DNAME     LOC
------------------------------------------------------------------------------------------------------------------
 1115    nive        06-OCT-92     45000              7241        1        1              admin    madurai
 1112    vinotha     02-AUG-90    32000              5462        1        1              admin    madurai
 1114    rajesh       05-APR-85    50000              8264        2        2              research  chennai
 1113    nandu        07-OCT-93    37500              6452        2        2              research  chennai
 1111    sathya      17-DEC-88    50000              4323        3        3          accounts Bangalore
                                                                           5              madical   delhi
 1116    palani       06-OCT-60    90000              6372        4
```

<u>RESULT:</u>

Thus the Nested queries and join queries was executed successfully.

4. VIEWS

AIM:

To write an SQL query for views.

PROCEDURES:

Q1: Create a table employee with attributes empno, ename, job, mgr, hiredate, sal, commission, deptno with appropriate data type. Also create a table department with attributes deptno, dname, loc.

Query:

```
SQL> create table employee1(empno number(5),ename varchar(19),mgrssn number(5),salary number(7),deptno number(3));
```

Output:

```
Table Created
```

Query:

```
SQL> insert into employee1 values(&empno,'&ename',&mgrssn,&salary,&deptno,'&job');
Enter value for empno: 111
Enter value for ename: nirmala
Enter value for mgrssn: 3456
Enter value for salary: 40000
Enter value for deptno: 2
Enter value for job: lecturer
```

Output:

```
old   1: insert into employee1 values(&empno,'&ename',&mgrssn,&salary,&deptno,'&job')
new   1: insert into employee1 values(111,'nirmala',3456,40000,2,'lecturer')
1 row created.
```

```
SQL> select * from employee1;
    EMPNO     ENAME          MGRSSN     SALARY     DEPTNO        JOB
-----------------------------------------------------------------------------------------------------
      111         nirmala              3456         40000        2            lecturer
      112         nandhu                5678         80000        1            surgeon
      113          sathya                  2345         90000         3              marketing business
      114          vino                    6789        50000        2              doctor
```

Query:

SQL> create table department(deptno number(3),dname varchar(30),loc varchar(20));

Output:

```
Table Created
```

Query:

SQL> insert into department values(&deptno,'&dname','&loc');

```
Enter value for deptno: 1
Enter value for dname: medical
Enter value for loc: mumbai
```

Output:

```
old   1: insert into department values(&deptno,'&dname','&loc')
new   1: insert into department values(1,'sales','chennai')
1 row created.
```

SQL> select * from department;

Output:

```
    DEPTNO     DNAME        LOC
   ---------- ---------------------------- --------------------
         1  medical          mumbai
         2  english          chennai
         3  sales            madurai
```

Q2: Create a view V1, which contain employee names and their manager names working in sales department

Query:

SQL> create view v1 as select ename,mgrssn from employee1 where deptno=(select deptno from department where dname='sales');

Output:

```
View created.
```

SQL> select * from v1;

```
ENAME              MGRSSN
------------------ ----------
sathya                   2345
```

Q3: Create a view called V2 which contains column such as empno, name, job from employee table. If V2 already exists your view command has to delete existing view and recreated it with the current query.

Query:

SQL> create view v2 as select empno,ename,job from employee1;

Output:

View created.

SQL> select * from v2;

```
     EMPNO ENAME              JOB
---------- ------------------ ------------------
       111 nirmala            lecturer
       112 nandhu             surgeon
       113 sathya             marketing business
       114 vino               doctor
```

SQL> create or replace view v2 as select empno,job from employee1;

View created.

SQL> select * from v2;

```
     EMPNO      JOB
---------- ------------------
       111      lecturer
       112      surgeon
       113      marketing business
       114      doctor
```

Q3: Create a view V3 from employee table with check option on column sal>45000.

Query:

SQL> create view v3 as select empno,ename,salary,job from employee1 where salary>50000;

Output:

```
View created.
```

SQL> select * from v3;

```
    EMPNO ENAME                SALARY        JOB
---------- ------------------ ---------- ------------------
      112          nandhu               80000       surgeon
      113 s         athya                  90000       marketing business
```

Q4: Create a view called V4 for the table employee, which does not exists currently.

Query:

SQL> create view v4 as select ename,job from employee1;

Output:

```
View created.
```

SQL> select * from v4;

```
ENAME             JOB
------------------ ------------------
nirmala           lecturer
nandhu            surgeon
sathya            marketing business
vino              doctor
```

SQL> create view v4 as select ename,empno from employee1;

```
create view v4 as select ename,empno from employee1
            *
ERROR at line 1:
ORA-00955: name is already used by an existing object
```

Q5: Create a view V5 which performs read only operation

Query:

SQL> create view v5 as select deptno,dname from departmnt with read only;

Output:

```
View created.
SQL> select * from v5;
   DEPTNO DNAME
---------- ----------
       1  medical
       2  english
       3  sales
SQL> insert into v5 values('empno',’&name’,&sal);
Enter value for ename:4
Enter value for hiredate: nanz
Enter value for sal: 52000
old   1: insert into v5 values('empno',’&name’,&sal)
new   1: insert into v5 values(4,'07-SEP-93',52000)
insert into v5 values(4,'07-SEP-93',52000)
*
ERROR at line 1:
ORA-42399: cannot perform a DML operation on a read-only view
```

Q5: Remove all the views.

Query:

```
SQL> drop view v1;
View dropped.
SQL> drop view v2;
View dropped.
SQL> drop view v3;
View dropped.
SQL> drop view v4;
View dropped.
SQL> drop view v5;
View dropped.
SQL> select * from v1;
```

```
select * from v1
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

RESULT:

Thus the query for views was executed successfully.

5. TRIGGERS

AIM:

To create and execute low level and statement level triggers.

TRIGGERS:

Q1: Create a table employee with attributes ssn,ename,salary,deptno with appropriate data type. Insert few records in to the table employee.

Query:

SQL> select * from employee;

Output:

```
SSN      ENAME       SALARY    DEPTNO
---------- -------- ---------- -------------------------
111     vino              60000           1
222     adhavan           50000           2
333     nanz              70000           1
444     sath              80000           3
```

Also create a table department with attributes deptno, dname, loc. Insert few records into the department table.

Query:

SQL>select * from department;

Output:

```
 DEPTNO         DNAME        LOC
---------------------------------------------
   1                medical         mumbai
   2                english         chennai
   3                sales           madurai
   4                it              madurai
   4                ba              chennai
```

Also create a table student with attributes sid,sname,marks. Insert few records into the student table.

Query:

SQL> select * from student;

Output:

```
      SID         SNAME      MARKS
     -------- ------- --------------------
      111         vino           80
      222         nandu          70
      333         sath           90
```

Q2: Create a trigger that should put restriction on user, who trying to insert or delete or modify records on employee table (row level).

<u>Create a Trigger</u>

Query:

SQL> create trigger pp after update on employee for each row begin insert into department values(5,'civil','karur');

```
            2  end;
            3  /
```

Output:

Trigger updated.

<u>update a record in employee table</u>

SQL> update employee set salary=50000 where deptno=1;

2 rows updated

<u>View the record from employee and department table</u>

SQL> select * from employee;

```
            SSN      ENAME       SALARY    DEPTNO
            ---------- -------- ---------- -------------------------
            111     vino           50000          1
            222     adhavan        50000          2
            333     nanz           50000          1
            444     sath           80000          3
```

SQL>select * from department;

```
DEPTNO      DNAME          LOC
---------------------------------------------------------
1           medical        mumbai
2           english        chennai
3           sales          madurai
4           it             madurai
4           ba             chennai
5           civil          karur
5           civil          karur
```

Q3: Create a trigger that should put restriction on user, who trying to insert or delete or modify records on student table(statement level).

<u>Create a Trigger</u>

Query:

```
SQL>  create trigger t2 after deletion on student
    2 begin
    3 delete from department where deptno=4;
    4 end;
      3   /
```

Output:

Trigger updated

<u>Delete a record from student table</u>

```
SQL> delete from student where sid=333;
1 row deleted
```

<u>View the record from student and department table</u>

```
SQL>select * from student;
     SID      SNAME      MARKS
     -------- ------- -------------------
     111      vino         80
     222      nandu        70
```

SQL>select * from department;

```
DEPTNO      DNAME       LOC
------------------------------------------------
 1          medical             mumbai
 2          english             chennai
 3          sales               madurai
 5          civil               karur
 5          civil               karur
```

RESULT:

Thus the row level trigger and statement level triggers were executed successfully.

6. PROCEDURES AND FUNTIONS

AIM:

To write PL/SQL stored procedure to perform various operations and produce the formatted output.

PROCEDURE:

Q1: Create the following tables:

'em' with empid, name, and dept.

'salary' with empid, basic, hra

Query:

SQL> create table em(empid number(3),name varchar(25),dept varchar(20));

Output:

Table created.

Query:

SQL>create table salary(empid number(3),basic number(6),hra number(6));

Output:

Table created.

Q2: Insert few records into the table em and salary.

Query:

SQL> insert into em values(&empid,'&name','&dept');

Output:

```
Enter value for empid: 10
Enter value for name: priya
Enter value for dept: IT
old   1: insert into em values(&empid,'&name','&dept')
new   1: insert into em values(10,'priya','IT')
1 row created.
```

Query:

SQL> insert into salary values(&empid,&basic,&hra);

Output:

```
Enter value for empid: 10
Enter value for basic: 23000
Enter value for hra: 20000
old   1: insert into salary values(&empid,&basic,&hra)
new   1: insert into salary values(10,23000,20000)
1 row created.
```

Q3. Display the records from the table em and salary.

Query:

SQL> select * from em;

Output:

```
EMPID       NAME            DEPT
---------- ------------------------ --------------------
10          priya             IT
11          reena             ECE
12          meena             EEE
```

Query:

SQL> select * from salary;

Output:

```
EMPID    BASIC     HRA
---------- ---------- -------------
10     23000        20000
11     33000        30000
12     43000        40000
```

Q4. Write a PL/SQL Procedure to display all the records in employee table as "The Employer <empname> has a ID <empid> working in <Dept> Department".

Query:

```
SQL> CREATE OR REPLACE PROCEDURE disp
IS
CURSOR emp_cur is
Select EmpId,Name,Dept from em;
emp_rec emp_cur%rowtype;
BEGIN
```

```
FOR emp_rec in emp_cur
LOOP
dbms_output.put_line('The Employer ' || emp_rec.name || ' has id' || emp_rec.empid || '  Working in
the Department : ' || emp_rec.dept);
END LOOP;
END;
/
```

Output:

```
Procedure created.
```

Q5: Write a Query to call the above procedure to display the output.

Query:

SQL> Set serveroutput on;

SQL> exec disp;

Output:

```
The Employer priya has id10  Working in the Department : IT
The Employer reena has id11  Working in the Department : ECE
The Employer meena has id12  Working in the Department : EEE
PL/SQL procedure successfully completed.
```

Q6: Write a PL/SQL function to return the name of the employee for the employee id mention in the function.

Query:

```
SQL>CREATE OR REPLACE FUNCTION em_dtl_func
RETURN em.name%type
IS
emp_name em.name%type;
BEGIN
SELECT name INTO emp_name FROM em WHERE empID = 12;
RETURN emp_name;
END;
/
```

Output:

```
Function created.
```

Q7: Write a Query to display the Output for the above function

Query:

SQL> select em_dtl_func from dual;

Output:

```
EM_DTL_FUNC
--------------------------------------------------------------------------------
meena
```

Q8: Write PL/SQL Procedure to get the Employee Id from the input and store the Employer Name for the given ID to Out Parameter.

Query:

SQL>CREATE OR REPLACE PROCEDURE emp_name (id IN em.empid%type, ename OUT em.name%type)

```
IS
BEGIN
SELECT name INTO ename
FROM em WHERE empid = id;
END;
/
```

Output:

```
Procedure created.
```

Q9: Write a PL/SQL Block to call the above Procedure using the Cursor. The Cursor will contain the entire Employee id from the em table and give the ID to the above procedure. The PL/SQL Block code that retrieve the OUT Parameter from the above Procedure and display the Output.

Query:

SQL> DECLARE

```
ename em.name%type;
CURSOR id_cur is SELECT empid FROM em;
```

```
emp_rec id_cur%rowtype;
BEGIN
FOR emp_rec in id_cur
LOOP
emp_name(emp_rec.empid, ename);
dbms_output.put_line('The employee ' || ename || ' has id ' || emp_rec.empid);
END LOOP;
END;
/
```

Output:

```
The employee priya has id 10
The employee reena has id 11
The employee meena has id 12
PL/SQL procedure successfully completed.
```

Q10: Write a PL/SQL Procedure to get the Employee Id from the table salary as input and Basic as IN OUT Parameter and calculate the bonus based on their Basic as per the following condition.

If the Basic below 10000 then increase the Basic to 8%

If the Basic between 10000 and 20000 then increase the Basic to 12%

If the Basic between 20000 and 30000 then increase the Basic to 15%

If the Basic above 30000 then increase the Basic to 20%

Query:

```
SQL> CREATE OR REPLACE PROCEDURE emp_Bonus  ( id IN salary.empid%type , Bas IN OUT
Salary.Basic%type)
IS
tmp_sal salary.Basic%type;
BEGIN
tmp_sal:=Bas;
IF tmp_sal < 10000 THEN
Bas := tmp_sal +(tmp_sal * .08);
ELSIF tmp_sal between 10000 and 20000 THEN
```

```
Bas := tmp_sal +(tmp_sal * .12);
ELSIF tmp_sal between 20000 and 30000 THEN
Bas := tmp_sal +(tmp_sal * .15);
ELSIF tmp_sal > 30000 THEN
Bas := tmp_sal +(tmp_sal * .20);
END IF;
END;
/
```

Output:

```
Procedure created.
```

Q11: Write PL/SQL Block for the above procedure to display the output.

Query:

```
  SQL > DECLARE
CURSOR updated_sal is
SELECT empid, Basic FROM Salary;
pre_sal salary.Basic%type;
BEGIN
FOR emp_rec IN updated_sal
LOOP
pre_sal := emp_rec.Basic;
emp_Bonus(emp_rec.empID, emp_rec.Basic);
dbms_output.put_line('The Bonus of ' || emp_rec.empID || ' increased from '|| pre_sal || ' to
'||emp_rec.Basic);
END LOOP;
END;
/
```

Output:

```
The Bonus of 10 increased from 23000 to    26450
The Bonus of 11 increased from 33000 to    39600
The Bonus of 12 increased from 43000 to    51600
PL/SQL procedure successfully completed.
```

Q12: Write a PL/SQL Function to find the Net Salary for the given Employee

Query:

```
SQL>  CREATE OR REPLACE FUNCTION NETSAL(id IN salary.empid%type)
RETURN salary.basic%type
IS
netsal salary.basic%type;
BEGIN
SELECT sum(basic) + sum(hra) INTO netsal FROM salary WHERE empid = id;
RETURN (netsal);
END;
/
```

Output:

```
Function created.
```

Q13: Write PL/SQL Block to display the output for the above Function.

Query:

```
SQL> variable sal number
SQL> execute :sal := netsal(12)
```

Output:

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

Query:

```
SQL> print sal
```

Output:

```
SAL
----------
83000
```

<u>RESULT:</u>

Thus the PL/SQL stored procedures are successfully executed to perform various operations like calculation of Net Salary of the given employee through the parameter (IN and OUT) and using the cursor to display the output in formatted way.

7. FRONT END TOOLS

AIM:

To create an application using visual basic to perform the operations such as insert, delete & move the records in the oracle database.

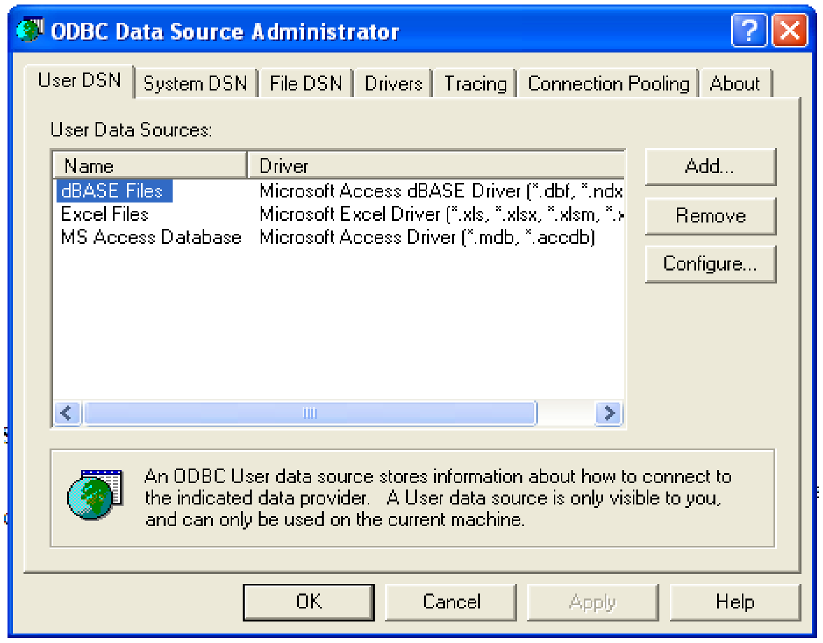
PROCEDURE:

Step 1: Create a table employee with attributes empno, ename, job(i.e., designation), mgr, hiredate, sal, deptno.

Step 2: Insert few records in to the table employee

Step 3: Save the employee table

Step 4: Open Control Panel-> Administrative Tools->ODBC



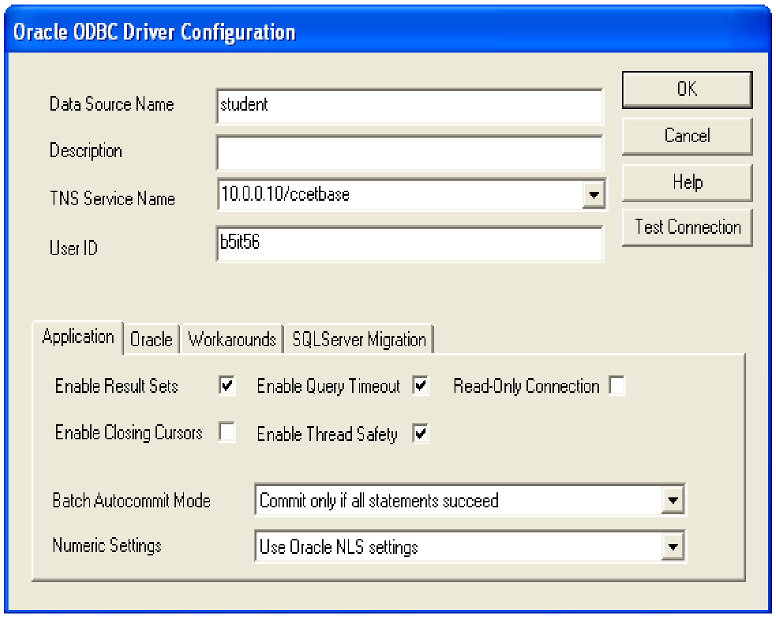
User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.ndx
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.x
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Step 5: Create a New Data Source Name for the project by click Add button, choose 'Oracle in OraDb10g_home1 and then click Finish button.

Step 6: Enter the Data Source Name, Service Name and User id.



Oracle ODBC Driver Configuration

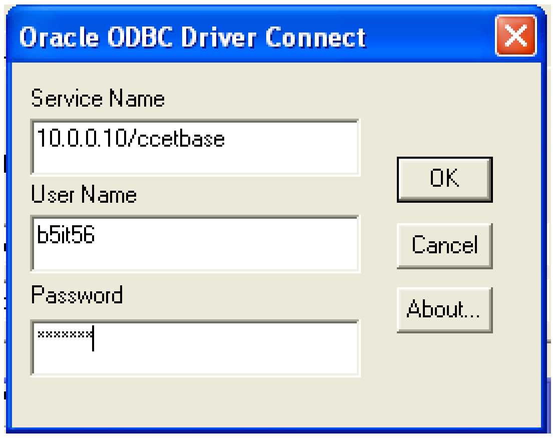
Data Source Name	student
Description	
TNS Service Name	10.0.0.10/ccetbase
User ID	b5it56

Enable Result Sets ✓ Enable Query Timeout ✓ Read-Only Connection ☐

Enable Closing Cursors ☐ Enable Thread Safety ✓

Batch Autocommit Mode	Commit only if all statements succeed
Numeric Settings	Use Oracle NLS settings

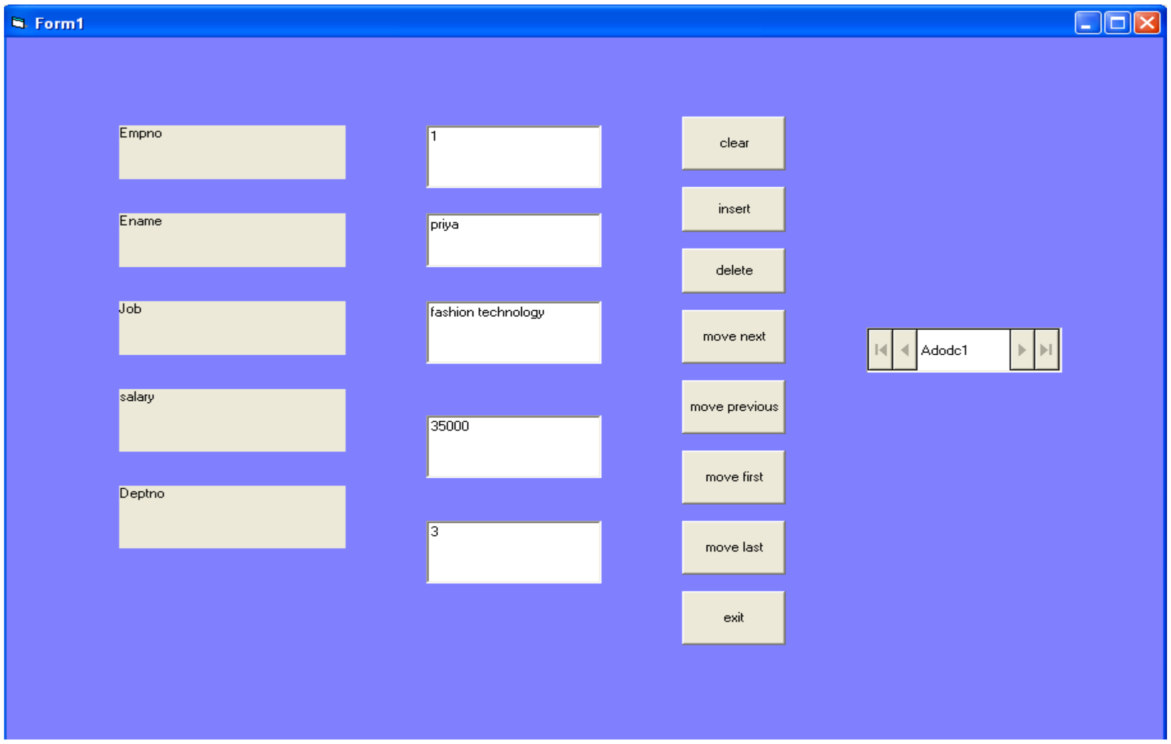
Step 7: Check the Test Connection by entering the correct password and then press OK.



Oracle ODBC Driver Connect

Service Name	10.0.0.10/ccetbase
User Name	b5it56
Password	*******

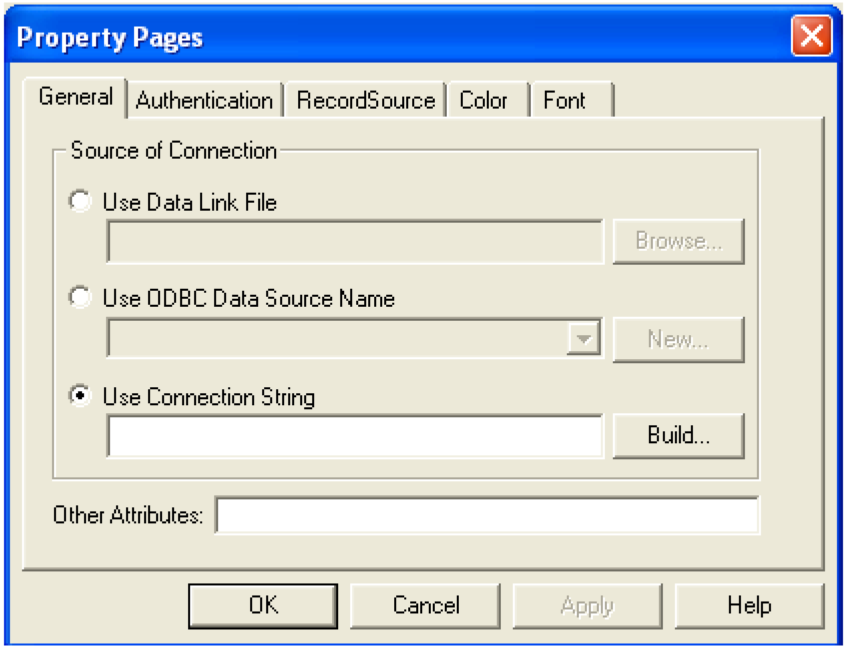
Step 8: Create a new Project in Visual Basic and design the forms as follow



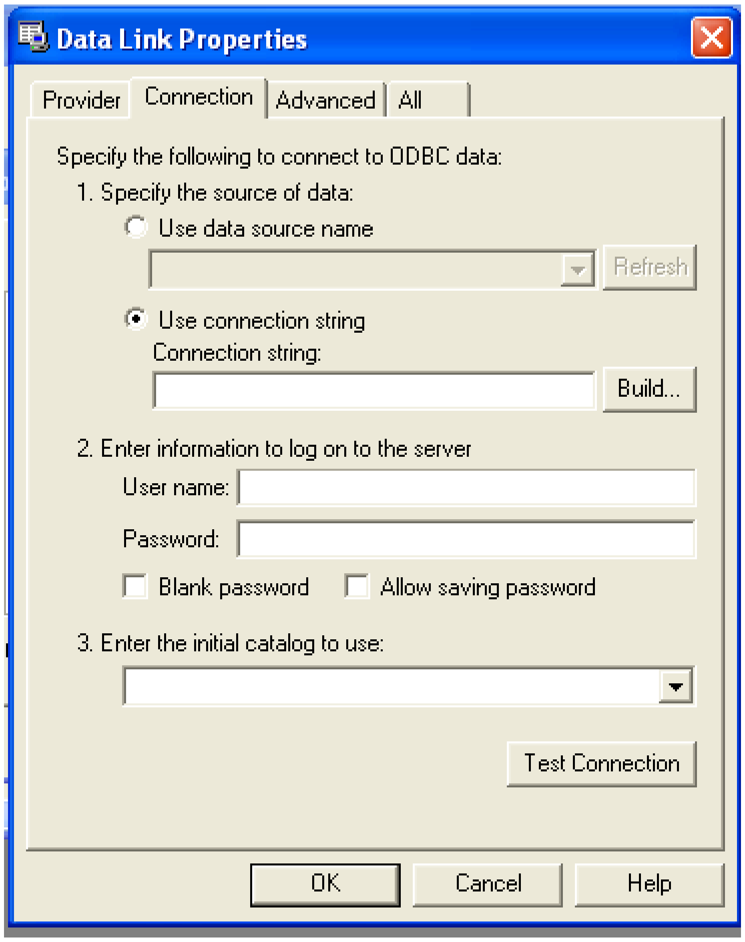
Empno	1
Ename	priya
Job	fashion technology
salary	35000
Deptno	3

Step 9: Add the component Microsoft ADO Control 6.0 and insert it into the project form.

Step 10: Identify the connection string by right click the ADO Control 6.o and select the property window.



Step 11: In Use Connection String, Select Build button.



Specify the following to connect to ODBC data:

1. Specify the source of data:
   - ( ) Use data source name
   - (●) Use connection string

   Connection string:

2. Enter information to log on to the server

   User name:

   Password:

   [ ] Blank password [ ] Allow saving password

3. Enter the initial catalog to use:

Step 12: Click Build button, Go to Machine Data Source Tab and select the data source name and press OK.

Step 13: Test the connection string by entering the password and choose the connection string.

Step 14: Add the code for the various operations like insert, delete, exit, clear, move next, move first, move last, move previous.

Step 15: Execute the project.

Program Code

```
Dim con As ADODB.Connection
Dim rs As ADODB.Recordset
```

Private Sub Command1_Click()

```
'code for move the next record
rs.MoveNext
```

```
If (rs.EOF) Then
rs.MoveFirst
MsgBox "You are in Last Record"
End If
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
Text6.Text = rs.Fields(5)
Text7.Text = rs.Fields(6)
End Sub

Private Sub Command2_Click()
'code for inserting a record
rs.Close
rs.Open "insert into employee values(" & Text1.Text & ",'" & Text2.Text & "','" & Text3.Text & "','" &
Text4.Text & "','" & Text5.Text & "')", con, adOpenDynamic
MsgBox ("Record Inserted")
rs.Open "select * from employee", con, adOpenDynamic
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)

Text6.Text = rs.Fields(5)
Text7.Text = rs.Fields(6)
End Sub

Private Sub Command3_Click()
'code for deleting the record
```

```
rs.Close
rs.Open "delete  from employee where empno=" & Text1.Text, con, adOpenDynamic
MsgBox ("Record Deleted")
rs.Open "select * from employee", con, adOpenDynamic
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
Text6.Text = rs.Fields(5)
Text7.Text = rs.Fields(6)
```
End Sub

Private Sub Command4_Click()
```
‘code for clear button
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = "”
Text6.Text = “”
Text7.Text = “”
```
End Sub

Private Sub Command5_Click()
```
‘code for move to the first record
rs.MoveFirst
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
```

```
Text6.Text = rs.Fields(5)
Text7.Text = rs.Fields(6)
End Sub

Private Sub Command6_Click()
‘code for move the previous record
rs.MovePrevious
If (rs.EOF) Then
rs.MoveLast
MsgBox ("You are in First Record")
End If
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
Text6.Text = rs.Fields(5)
Text7.Text = rs.Fields(6)
End Sub

Private Sub Command7_Click()
‘code for move the last record
rs.MoveLast
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
Text6.Text = rs.Fields(5)
Text7.Text = rs.Fields(6)
End Sub
```

Private Sub Command8_Click()

‘code for exit the project

End

End Sub

Private Sub Form_Load()

Set con = New ADODB.Connection

Set rs = New ADODB.Recordset

con.Open

"DSN=Aadharsha;UID=b5it18;PWD=student;DBQ=10.0.0.10/CCETBASE;DBA=W;APA=T;EXC=F;FEN=T;QTO=T;FRC=10;FDL=10;LOB=T;RST=T;BTD=F;BAM=IfAllSuccessful;NUM=NLS;DPM=F;MTS=T;MDI=F;CSR=F;FWC=F;FBS=64000;TLO=O;"

rs.Open "select * from employee", con, adOpenDynamic

Text1.Text = rs.Fields(0)

Text2.Text = rs.Fields(1)

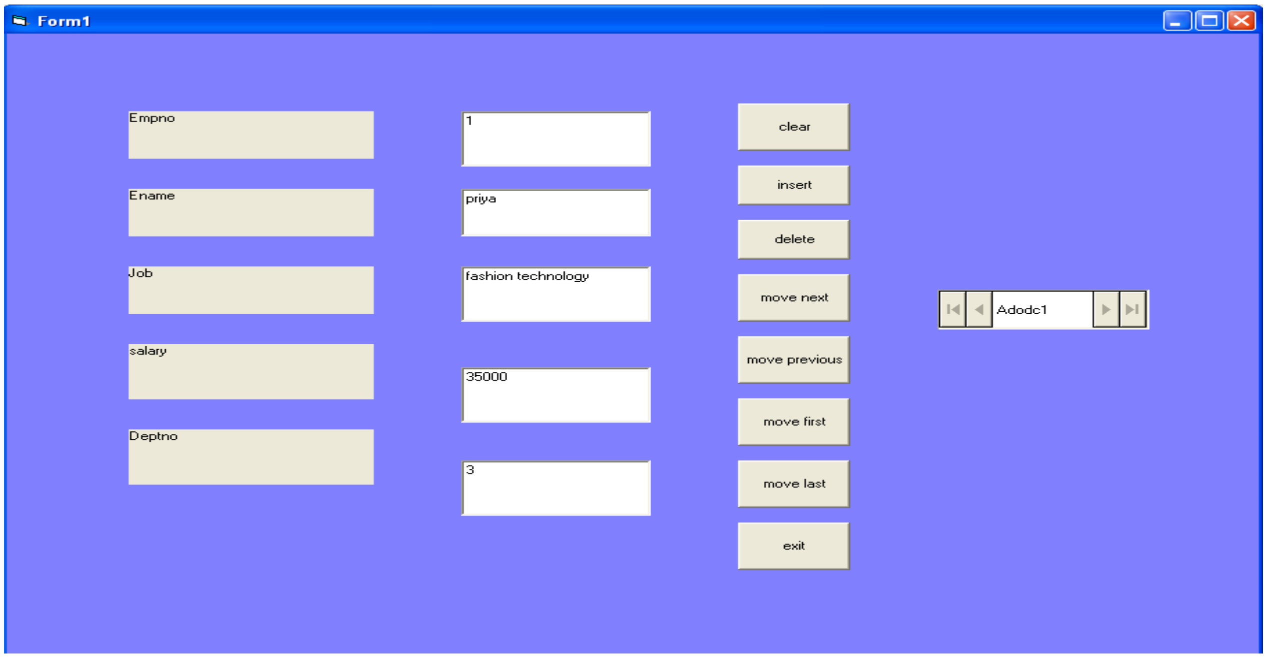
Text3.Text = rs.Fields(2)

Text4.Text = rs.Fields(3)

Text5.Text = rs.Fields(4)

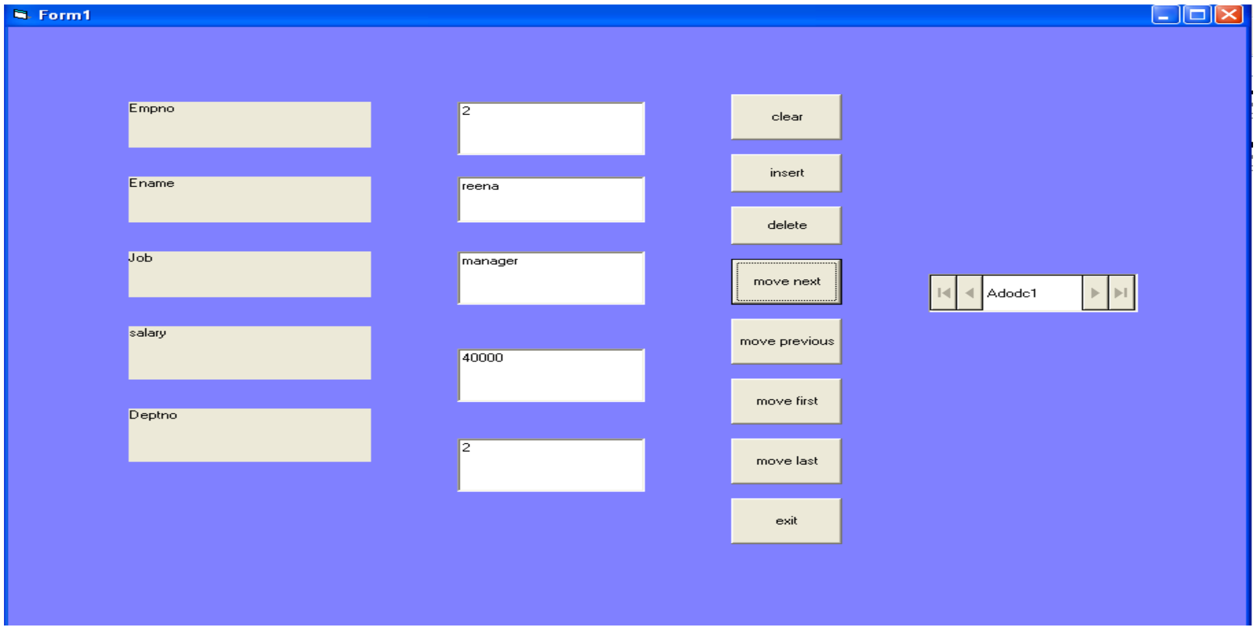
End Sub

SNAPSHOTS



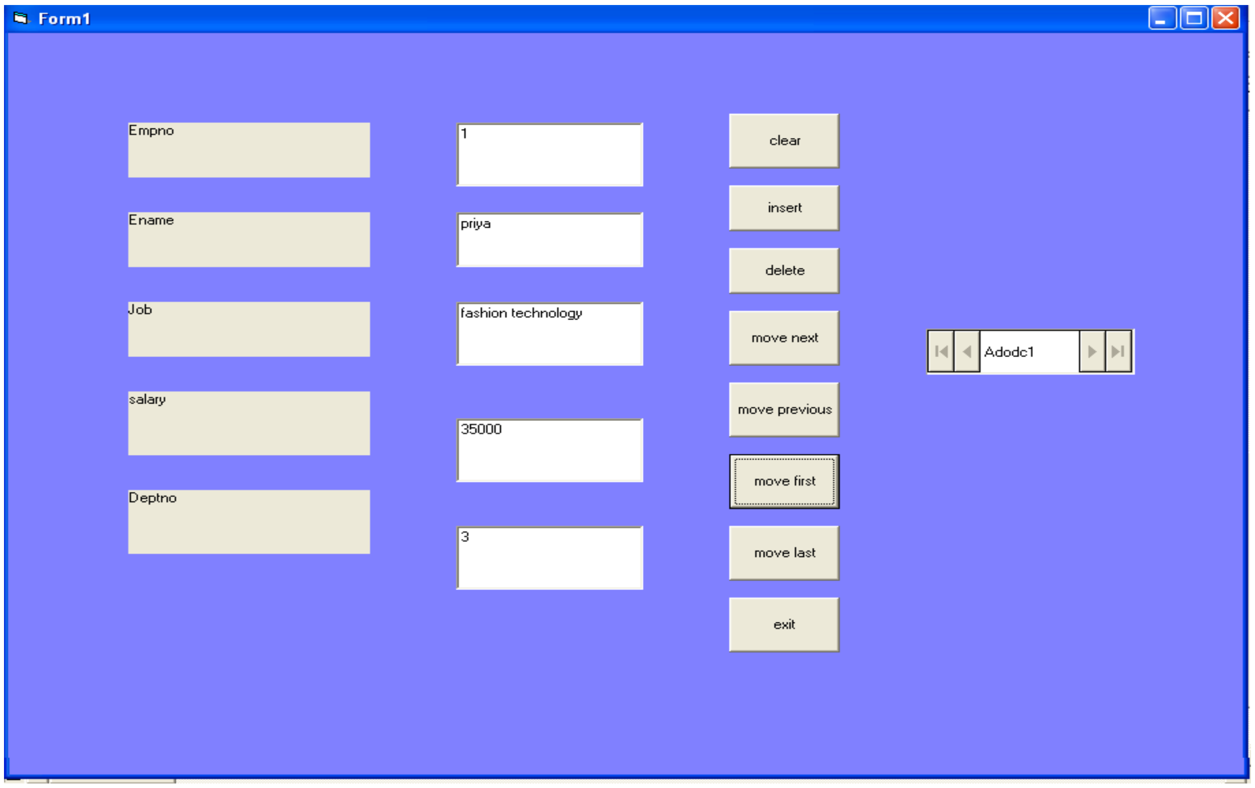
Empno	1
Ename	priya
Job	fashion technology
salary	35000
Deptno	3

MOVE NEXT



Empno	2
Ename	reena
Job	manager
salary	40000
Deptno	2

MOVE FIRST



Empno 1

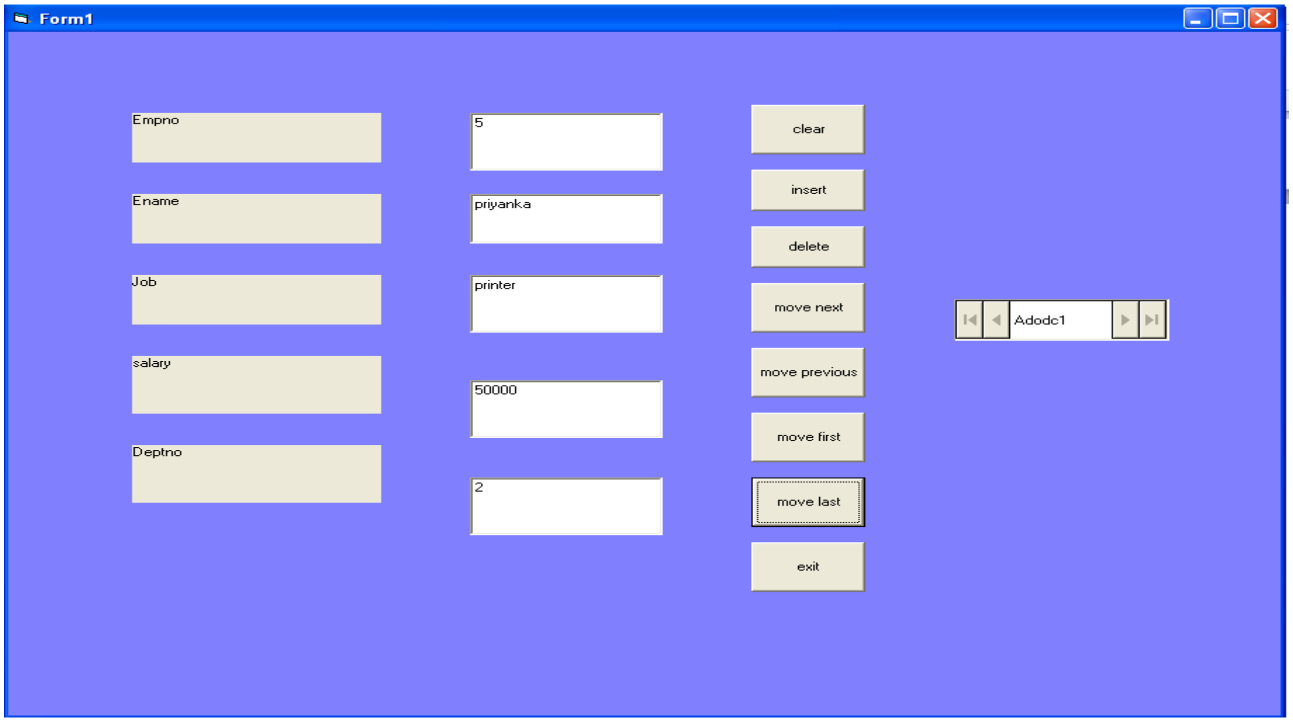
Ename priya

Job fashion technology

salary 35000

Deptno 3

MOVE LAST



Empno 5

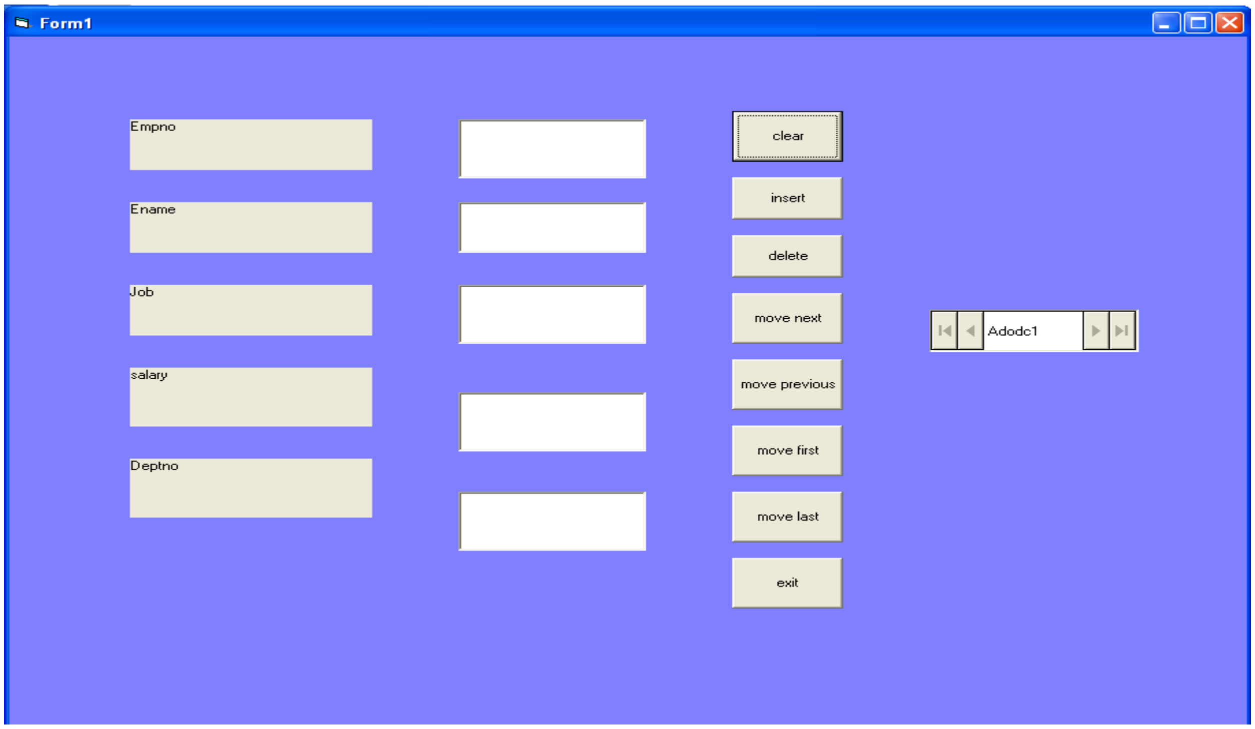
Ename priyanka

Job printer

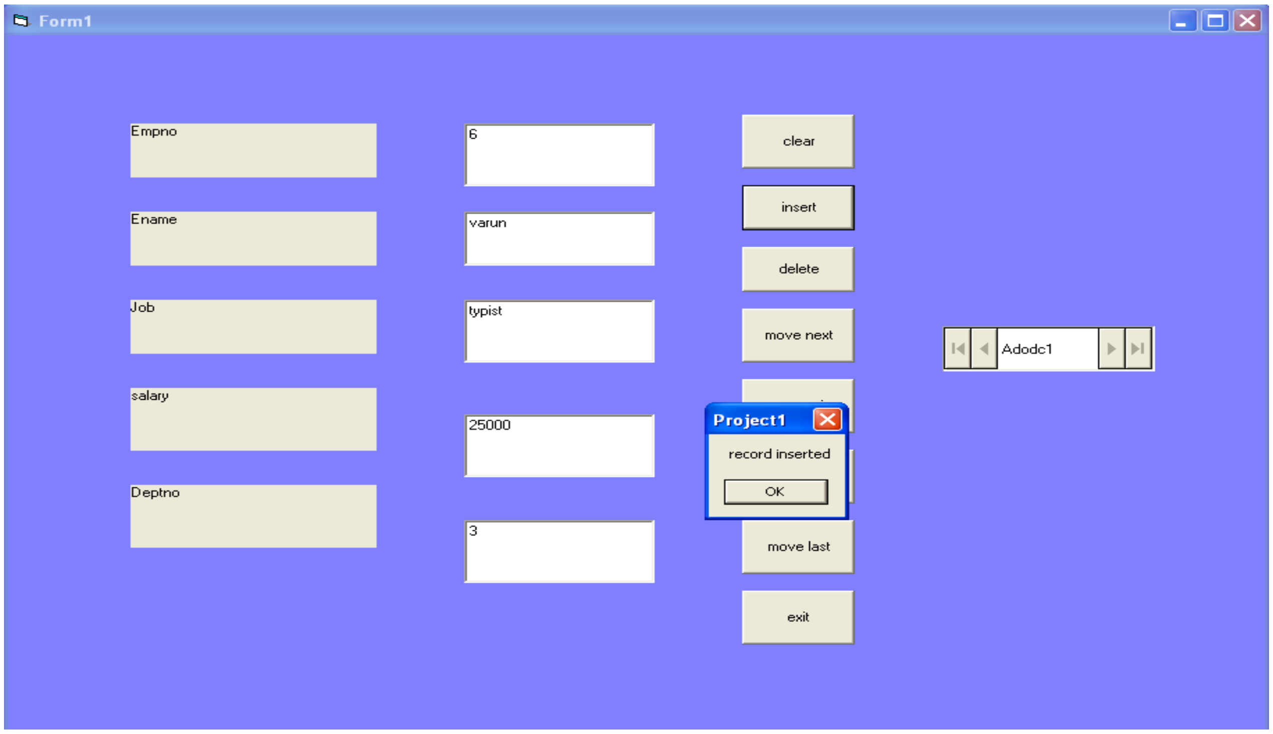
salary 50000

Deptno 2

CLEAR



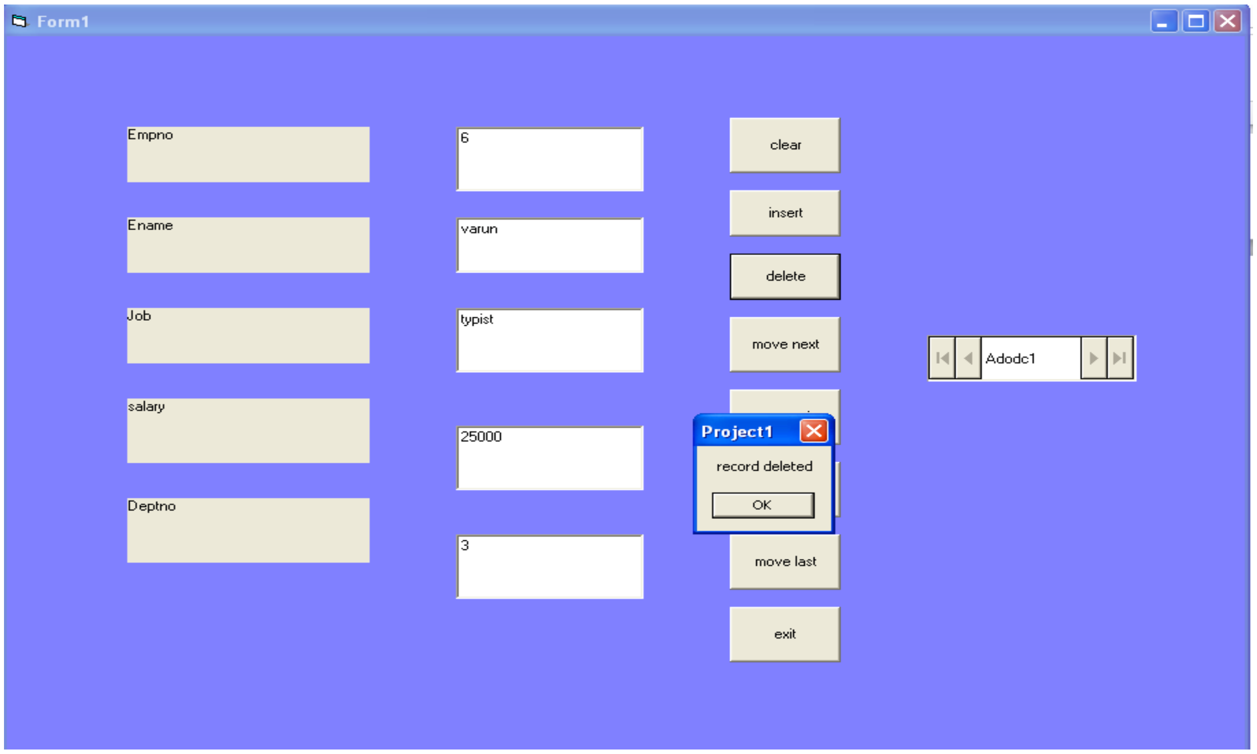
INSERT A RECORD



Empno	6
Ename	varun
Job	typist
salary	25000
Deptno	3

record inserted

DELETE A RECORD



Empno	6
Ename	varun
Job	typist
salary	25000
Deptno	3

record deleted

RESULT:

Thus the application using visual basic that perform the operations such as insert,delete & move the records in the oracle database are done successfully .

8. ORACLE FORMS

AIM:

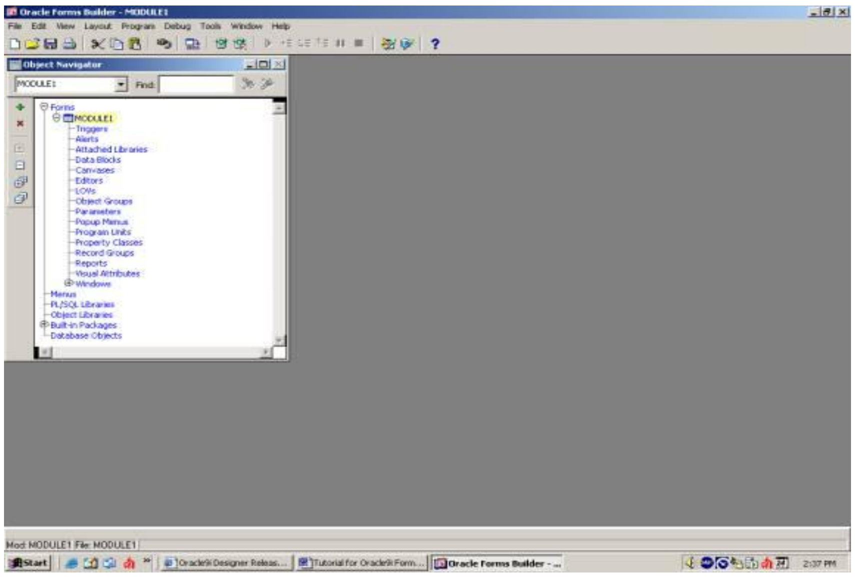
To create web-enabled forms based on the tables that you have created using Oracle10g Designer or SQL Plus commands.

PROCEDURES:

Step1: Create a Table named Student with ftpt_status,major,start_sem,start_year,studname with relevant data types and insert some records into the table.

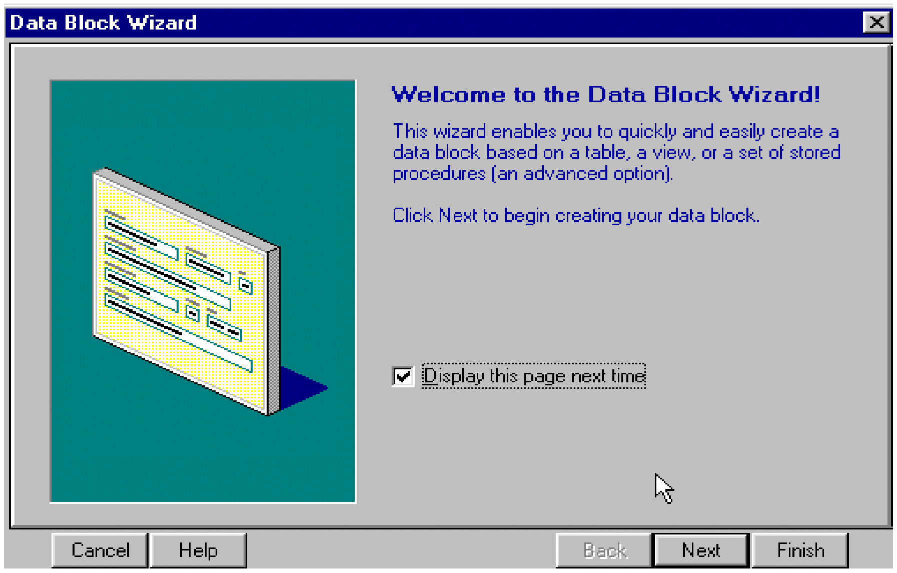
Stpe2: Create the Form using Oracle10g Developer Suite. **Logging On to** Oracle10g Forms. To log on to Oracle10g Forms/Reports, go to Start → Programs → Oracle10g Developer Suite → Forms Developer, click Start OC4J Instance (Very important: You must keep OC4J running while using Oracle10g Forms/Reports !!!), after OC4J initialized, then click Forms Builder.

Step 3: Immediately, you will see the window for Forms Builder:



Step 4: Double click Data Blocks, Select 'Use the Datablock Wizard' and click OK--This is the easiest method to design a new form.

Step 5: Will now see the Welcome to the Datablock Wizard Window. Click Next to proceed.



Data Block Wizard

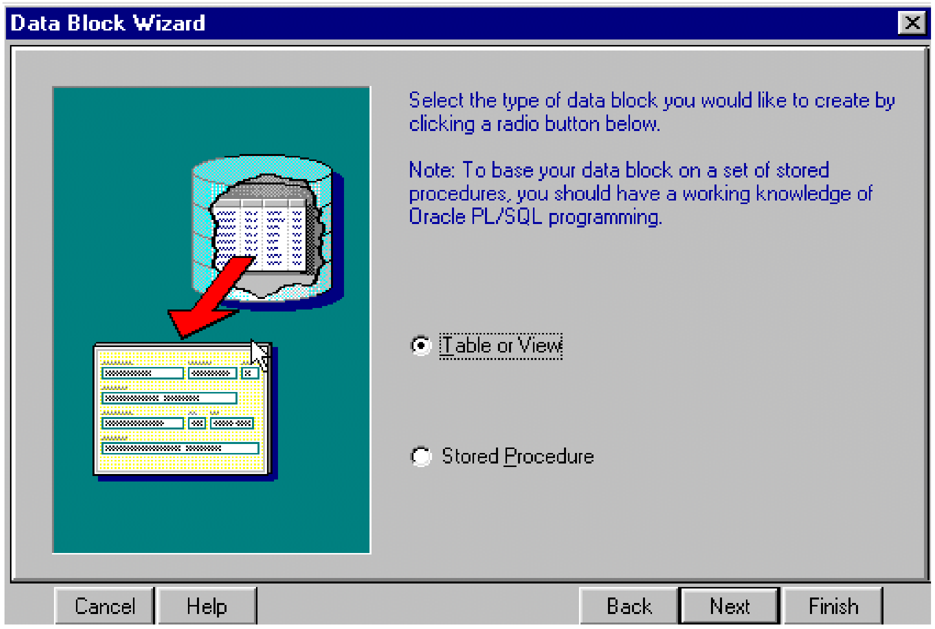
Welcome to the Data Block Wizard!

This wizard enables you to quickly and easily create a data block based on a table, a view, or a set of stored procedures (an advanced option).

Click Next to begin creating your data block.

[✓] Display this page next time

Step 6: The window for the Datablock Wizard. Select Table or View as in the figure and click Next.



Data Block Wizard

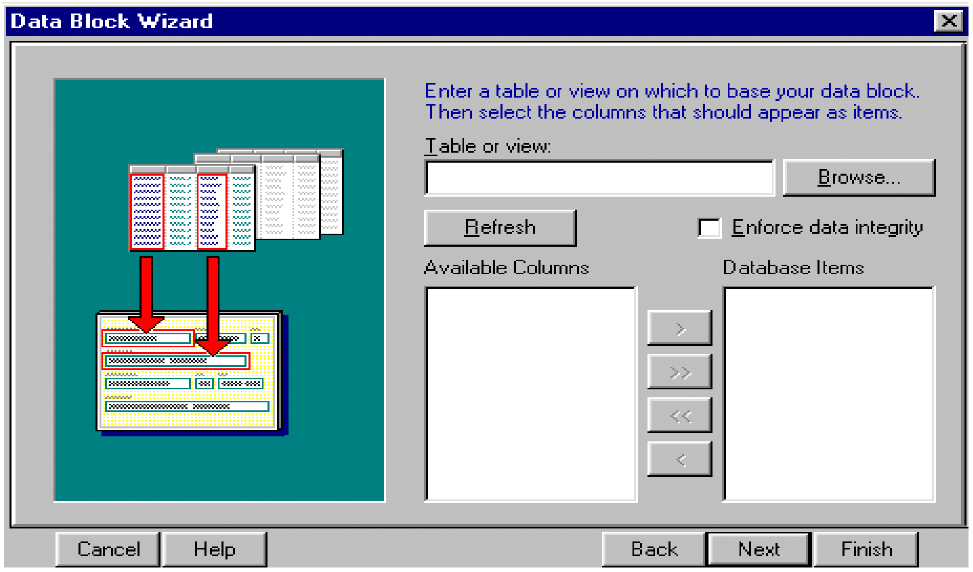
Select the type of data block you would like to create by clicking a radio button below.

Note: To base your data block on a set of stored procedures, you should have a working knowledge of Oracle PL/SQL programming.

[✓ Table or View]

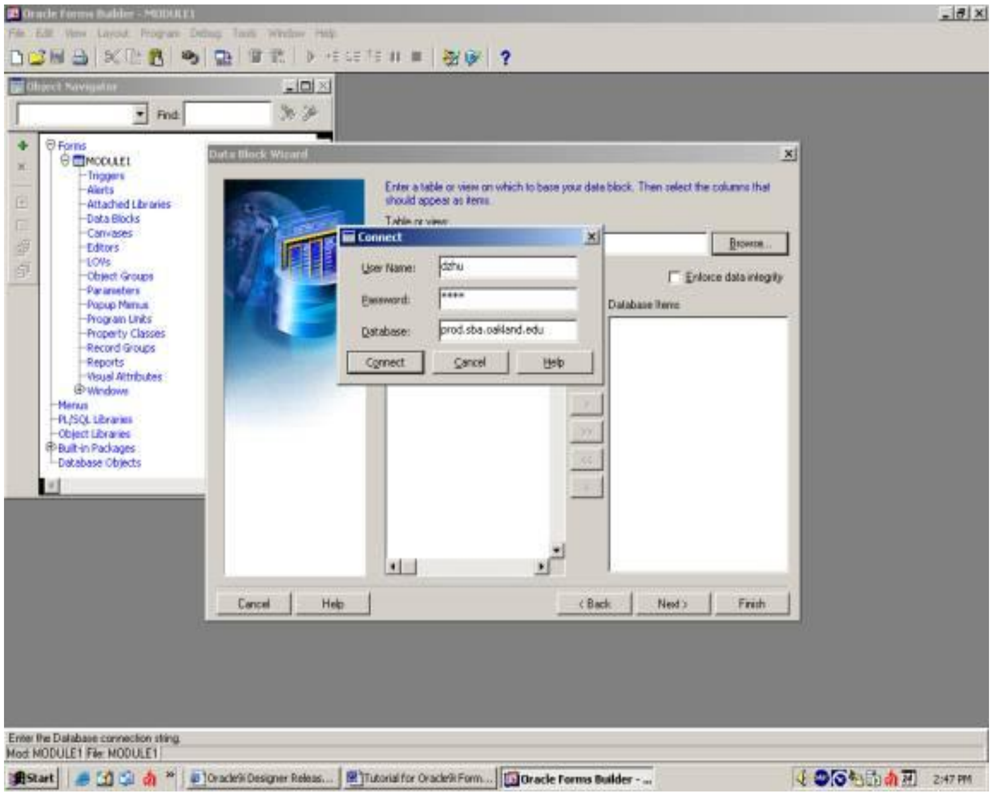
Stored Procedure

Step 7: Will now see the window that prompts you to select a table or a view--your form will be created based on this selection. Since no table or view is being shown, click on browse to look at the list of tables and views in your database.



Enter a table or view on which to base your data block. Then select the columns that should appear as items.

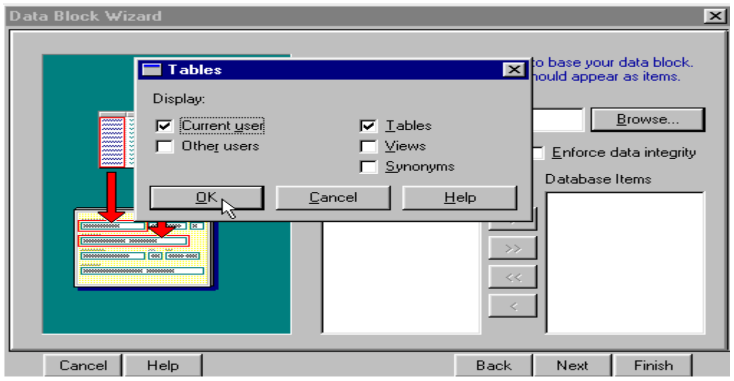
Step 8: Click browse, the connect window will appear. Type your username, password and database to connect to the database.



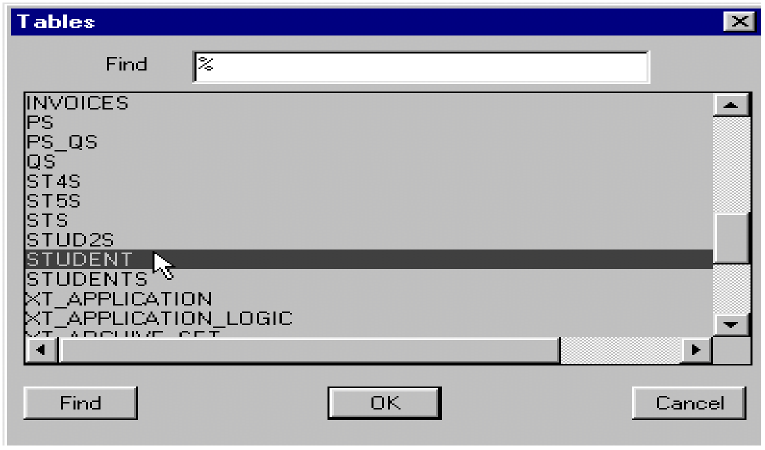
User Name: dzhu
Password: ****
Database: prod.sba.oakland.edu

Enter the Database connection string.

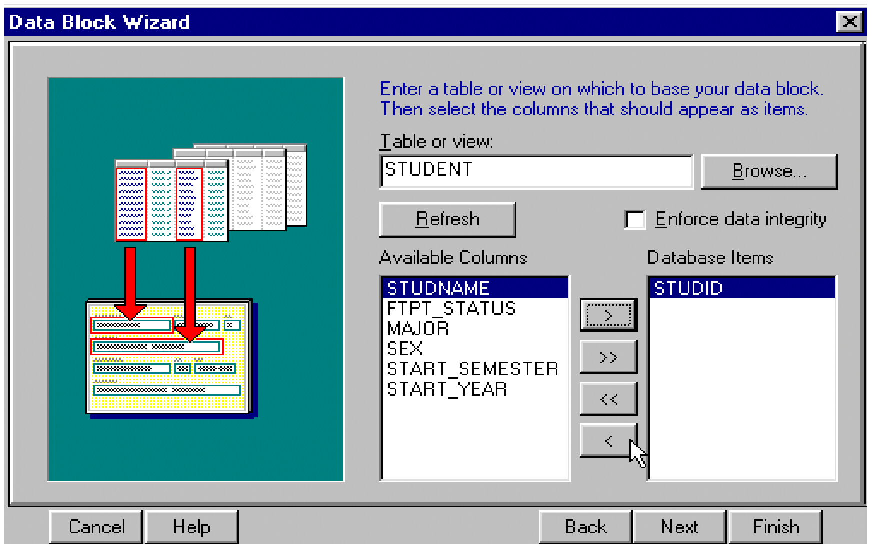
Step 9: The tables window will displayed. Select current users and tables and click OK.



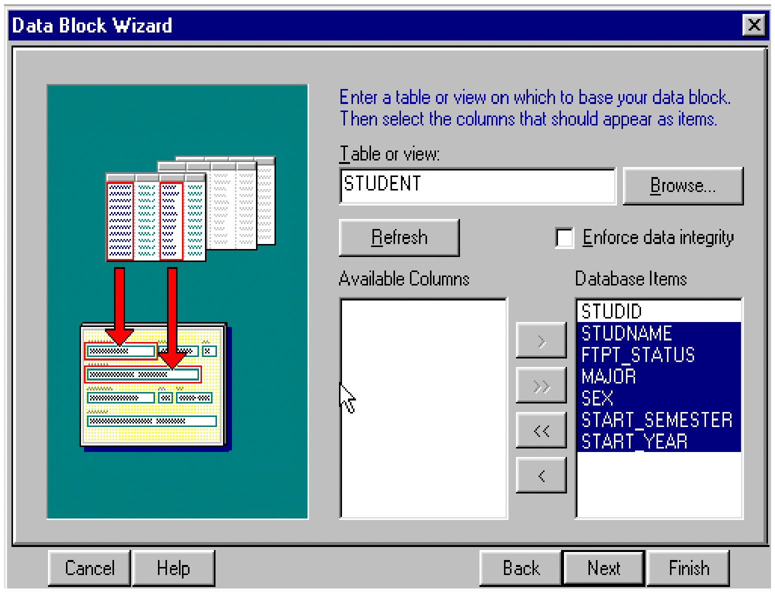
Step 10: The list of tables created in the database is displayed. Select Students and click OK.



Step 11: The selected table and its available columns on your screen. Click on the single right arrow to select the first column to be shown in your form; in this case the STUDID column. You will now see this column under the database items selected sub-window.



Step 12: To move the rest of the columns, simply click on the double right arrow and this will select all your columns in to the database items.



Enter a table or view on which to base your data block.
Then select the columns that should appear as items.

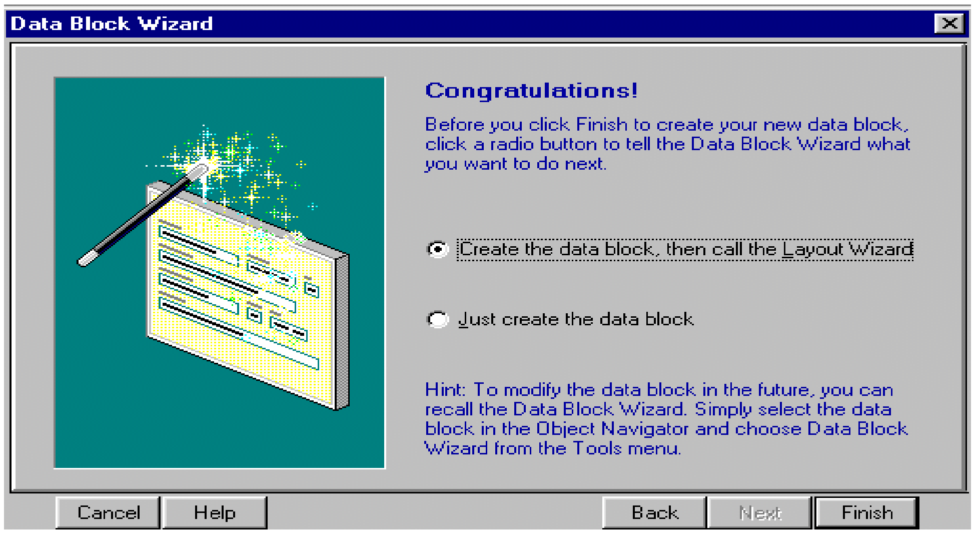
Table or view: STUDENT

☐ Enforce data integrity

Available Columns

Database Items
STUDID
STUDNAME
FTPT_STATUS
MAJOR
SEX
START_SEMESTER
START_YEAR

Step 13: The Congratulations window will displayed. Make sure that "Create the data block, then call the Layout Wizard" is selected and click on Finish.



Congratulations!

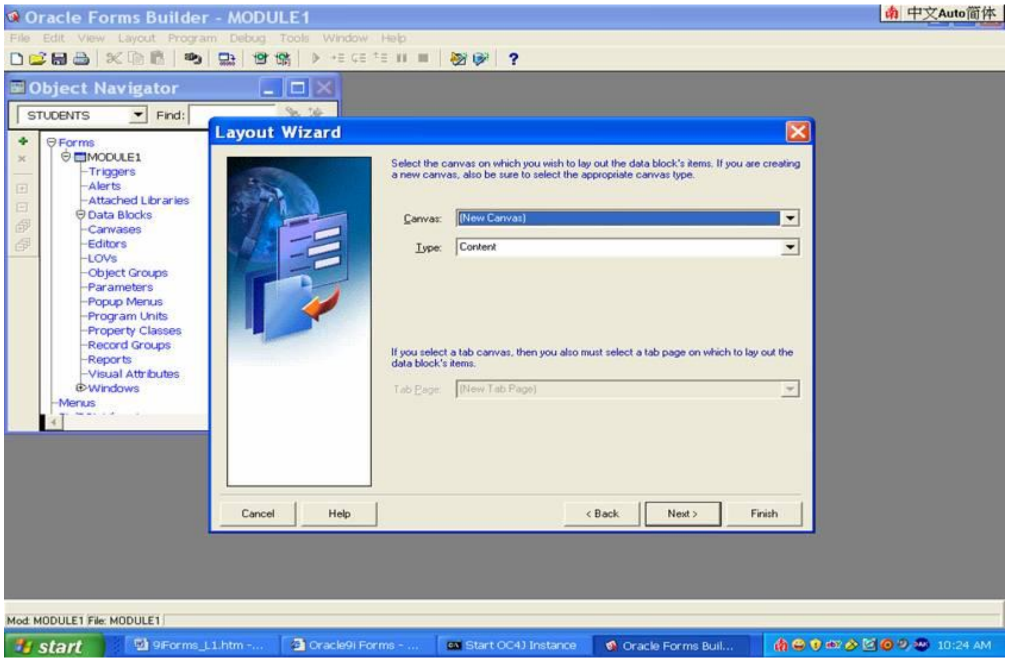
Before you click Finish to create your new data block, click a radio button to tell the Data Block Wizard what you want to do next.

● Create the data block, then call the Layout Wizard

○ Just create the data block

Hint: To modify the data block in the future, you can recall the Data Block Wizard. Simply select the data block in the Object Navigator and choose Data Block Wizard from the Tools menu.

Step 14: Will now see the Layout Wizard Welcome window, click next. You will see the following screen, click next.



Select the canvas on which you wish to lay out the data block's items. If you are creating a new canvas, also be sure to select the appropriate canvas type.

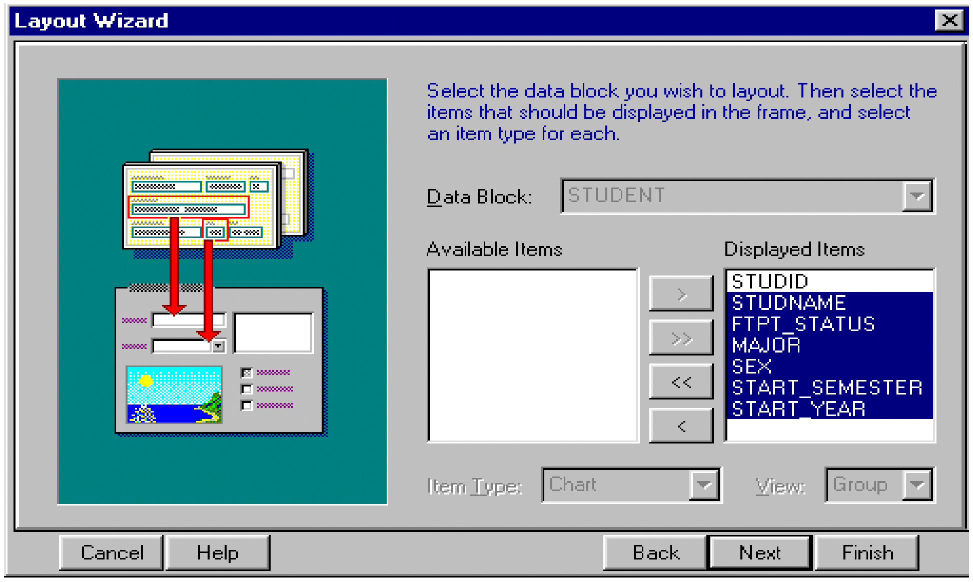
Canvas: (New Canvas)

Type: Content

If you select a tab canvas, then you also must select a tab page on which to lay out the data block's items.

Tab Page: (New Tab Page)

Step 15: select the items that you would like to show in the form. Make sure that the data block selected is Students and then click the double right arrow to move all the columns of the Student block from the available items to the displayed items. Click on Next to continue.



Select the data block you wish to layout. Then select the items that should be displayed in the frame, and select an item type for each.

Data Block: STUDENT

Available Items

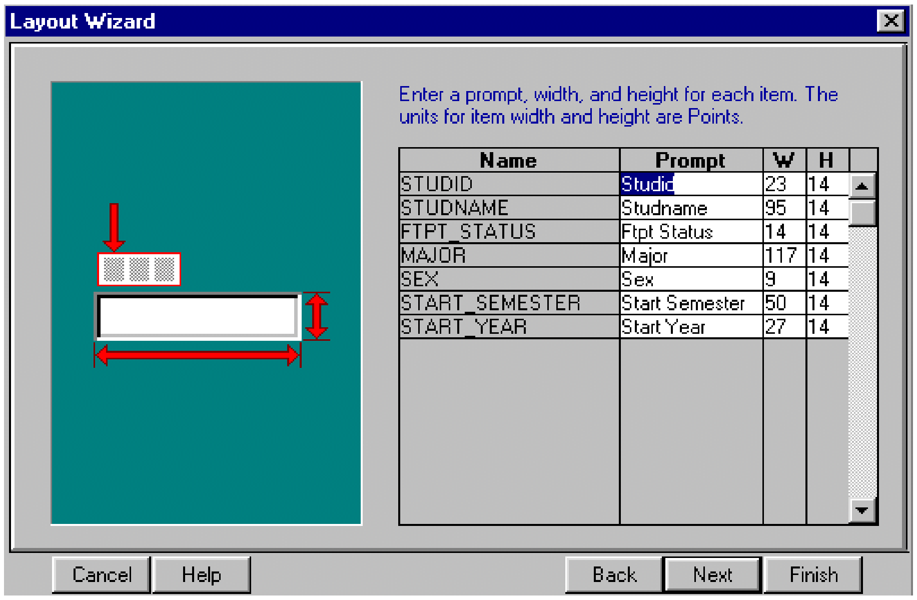
Displayed Items

STUDID
STUDNAME
FTPT_STATUS
MAJOR
SEX
START_SEMESTER
START_YEAR

Item Type: Chart

View: Group

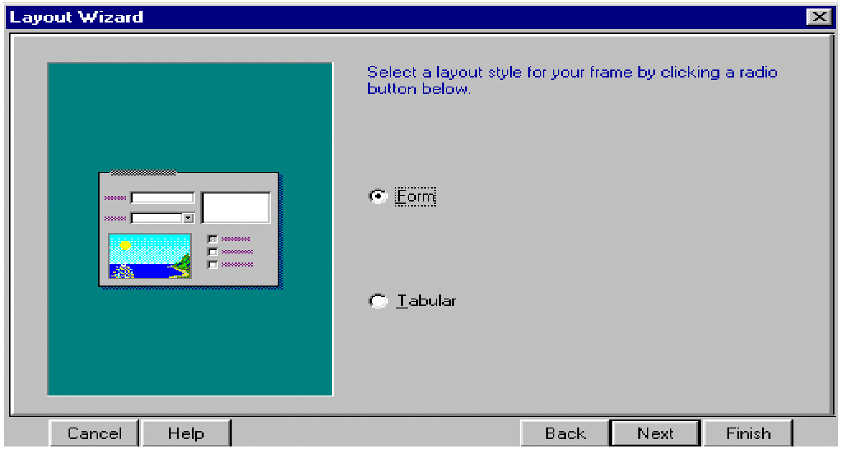
Step 16: The window with the prompt for the height and width of the items will appear. Click Next to accept the default values.



Enter a prompt, width, and height for each item. The units for item width and height are Points.

Name	Prompt	W	H
STUDID	Studid	23	14
STUDNAME	Studname	95	14
FTPT_STATUS	Ftpt Status	14	14
MAJOR	Major	117	14
SEX	Sex	9	14
START_SEMESTER	Start Semester	50	14
START_YEAR	Start Year	27	14

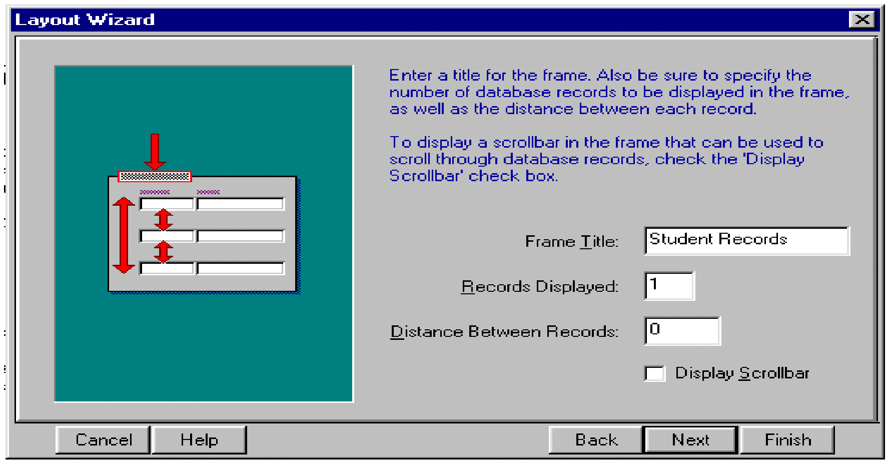
Step 17: The Layout Wizard will now prompt you to select the layout or view style of your block. Select Form and click Next.



Select a layout style for your frame by clicking a radio button below.

[✓ Form] Tabular

Step 18: The Layout Wizard will now prompt you to select a title for the form that you are creating. Type in Student Records. Click Next to continue.



Enter a title for the frame. Also be sure to specify the number of database records to be displayed in the frame, as well as the distance between each record.

To display a scrollbar in the frame that can be used to scroll through database records, check the 'Display Scrollbar' check box.

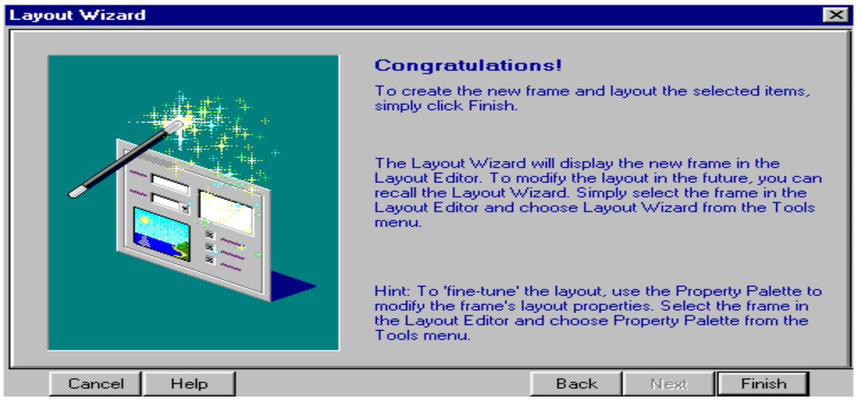
Frame Title: Student Records

Records Displayed: 1

Distance Between Records: 0

☐ Display Scrollbar

Step 19: Congratulations! You have now successfully created your first form. Click Finish to view your form.


Layout Wizard

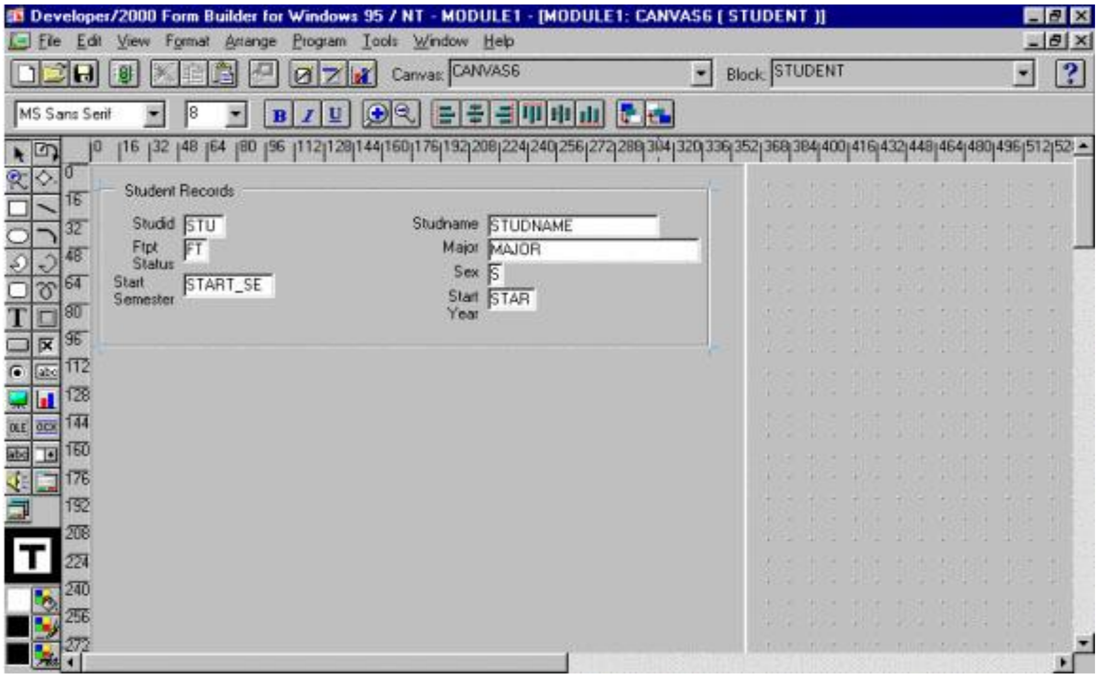
Congratulations!

To create the new frame and layout the selected items, simply click Finish.

The Layout Wizard will display the new frame in the Layout Editor. To modify the layout in the future, you can recall the Layout Wizard. Simply select the frame in the Layout Editor and choose Layout Wizard from the Tools menu.

Hint: To 'fine-tune' the layout, use the Property Palette to modify the frame's layout properties. Select the frame in the Layout Editor and choose Property Palette from the Tools menu.

Step 20: Will now see the canvas view of the form that you have created. You can now add various objects like push buttons, combo boxes and radio buttons to your form to make it more graphical and user friendly. We will do this in the next lesson.

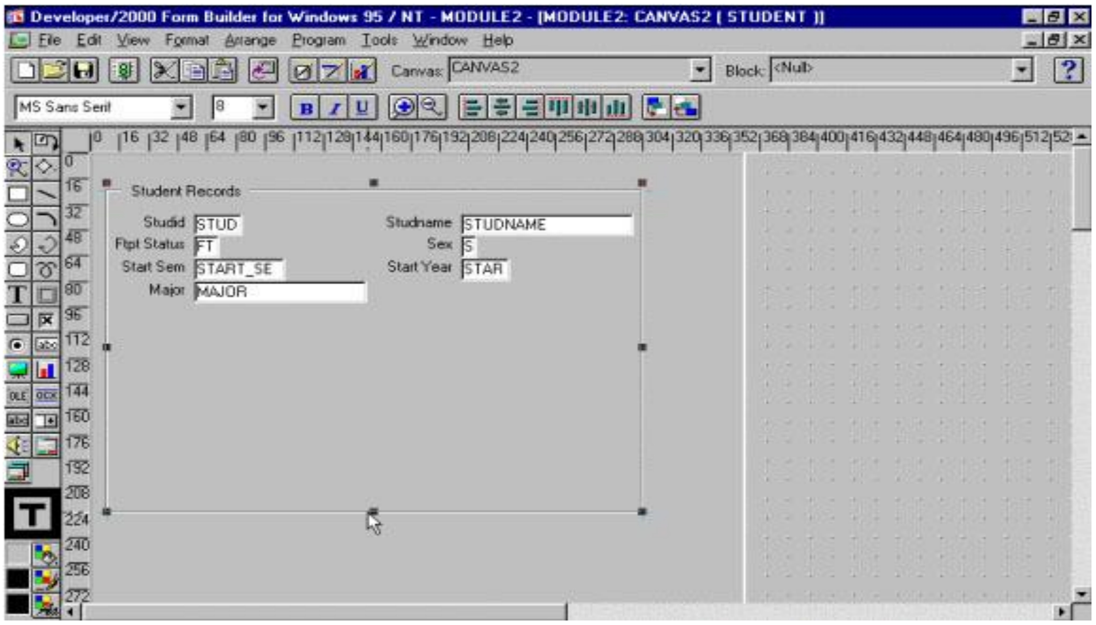

Developer/2000 Form Builder for Windows 95 / NT - MODULE1 - [MODULE1: CANVAS6 (STUDENT)]

Canvas: CANVAS6 Block: STUDENT

Student Records

Studid STU
Ftpt Status FT
Start Semester START_SE
Studname STUDNAME
Major MAJOR
Sex S
Start Year STAR

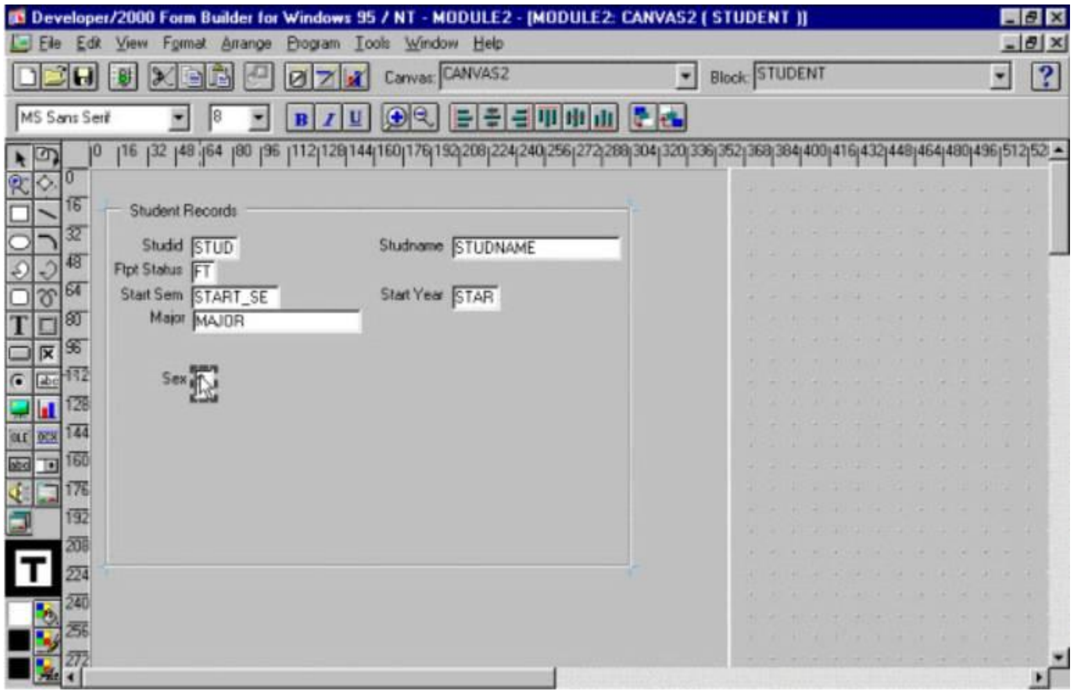
Step 21: Now format the form manually. Click on the frame to select it. Then drag the frame to make it bigger.


Developer/2000 Form Builder for Windows 95 / NT - MODULE2 - [MODULE2: CANVAS2 (STUDENT)]
File Edit View Format Arrange Program Tools Window Help
Canvas: CANVAS2
Block: <Null>
MS Sans Serif
8

Student Records

Studid STUD
Studname STUDNAME
Flpt Status FT
Sex S
Start Sem START_SE
Start Year STAR
Major MAJOR

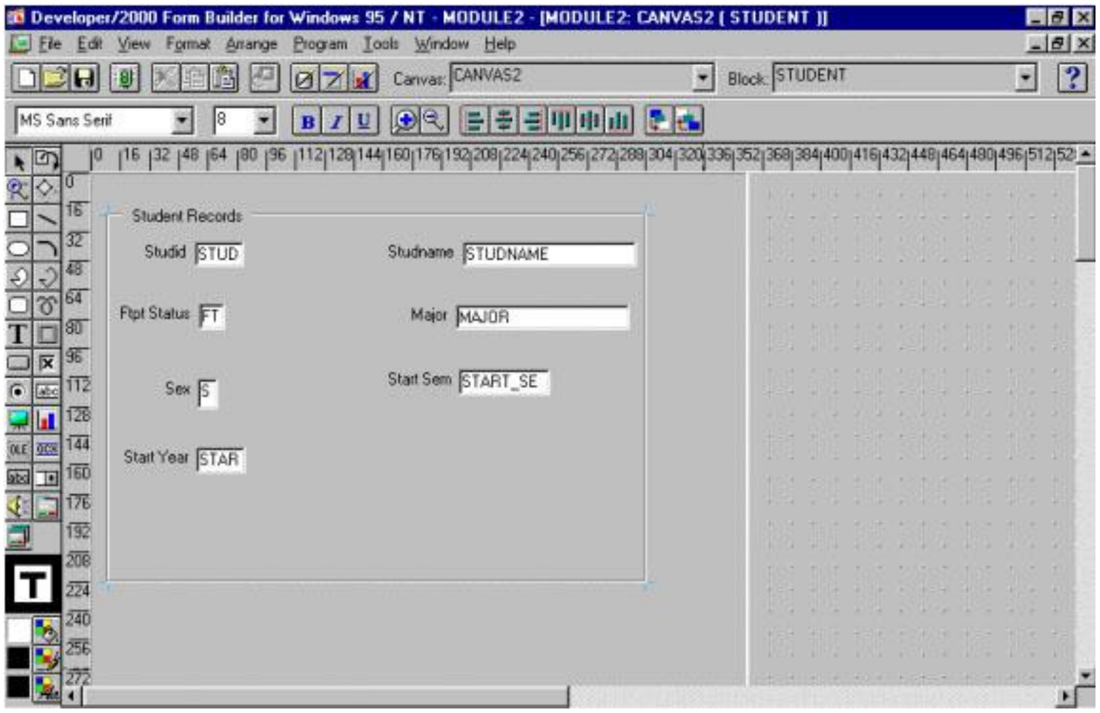
Step 22: Now space out the data fields to make your form more visually appealing. You can do this by simply selecting the data field and dragging it to your desired area.


Developer/2000 Form Builder for Windows 95 / NT - MODULE2 - [MODULE2: CANVAS2 (STUDENT)]
File Edit View Format Arrange Program Tools Window Help
Canvas: CANVAS2
Block: STUDENT
MS Sans Serif
8

Student Records

Studid STUD
Studname STUDNAME
Flpt Status FT
Start Sem START_SE
Start Year STAR
Major MAJOR
Sex S

Step 23: After formatted all the data fields, the form should look like the following:



Developer/2000 Form Builder for Windows 95 / NT - MODULE2 - [MODULE2: CANVAS2 (STUDENT)]

Canvas: CANVAS2 Block: STUDENT

Student Records

Studid	STUD	Studname	STUDNAME
Ftpt Status	FT	Major	MAJOR
Sex	S	Start Sem	START_SE
Start Year	STAR		

Step 24: Run the Form.

NOTE: We must run the form in Internet explorer only so we must change the browser from Mozilla Firefox.

Edit -→ Preferences -→ Runtime -→ Web Browser location -→ Select the path for Internet Explorer.

<u>RESULT:</u>

Thus the web enabled forms based on tables are created using Oracle 10g designer or SQL plus commands.

9. ORACLE REPORTS

AIM:

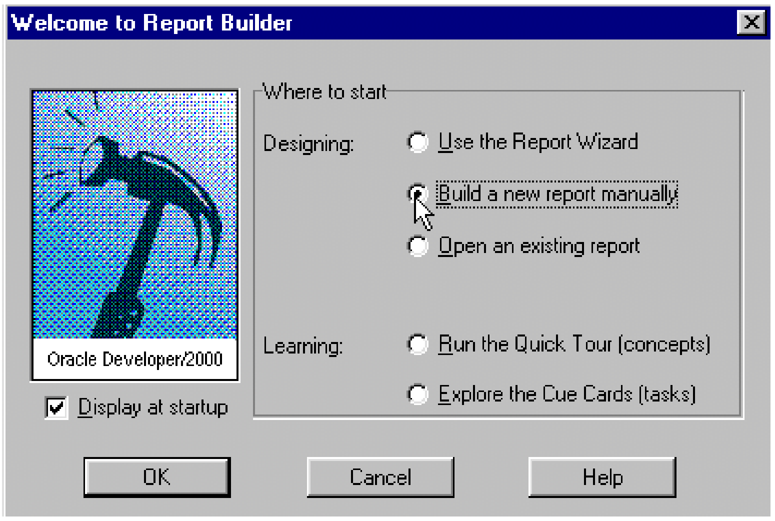
Our objective will be to create a simple report that will list students along with some student attributes. Students in this report will be categorized by major.

PROCEDURES:

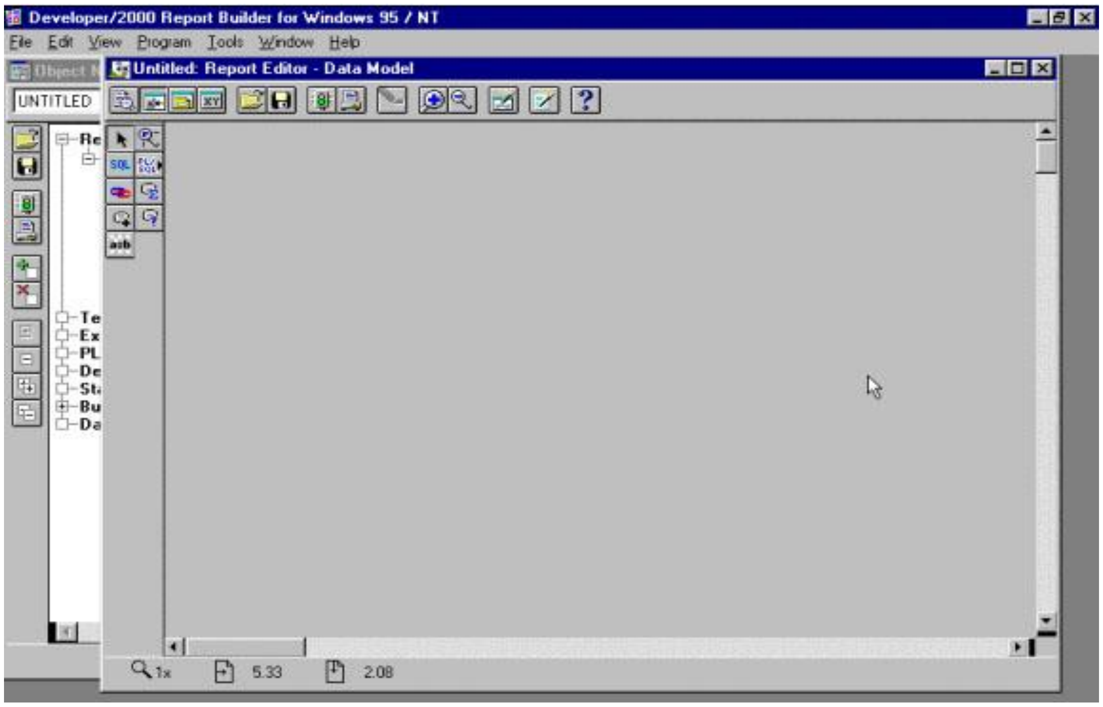
Step1: Create a Table named Student with ftpt_status,major,start_sem,start_year,studname with relevant data types and insert some records into the table.

Step 2: Create the Report using Oracle10g Developer Suite. In order to create reports, Go to the Reports Builder in Oracle10g Forms/Reports. To do this, go to the Start button and select Programs →Oracle10g Developer → Reports Developer →Report Builder

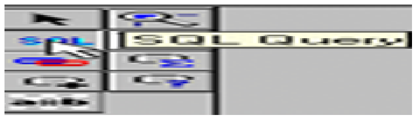
Step 3: Immediately, will see the Welcome to Report Builder Window. Select the radio button for the *Build a new report manually* and click OK.

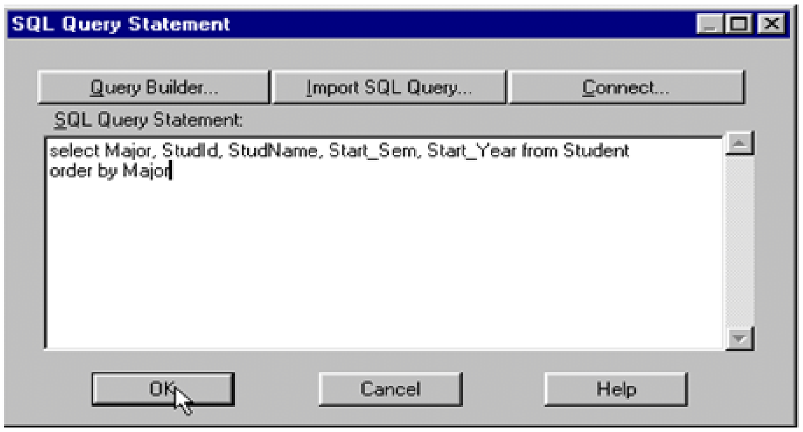


Step 4: click OK, you will see the Report Editor-Data Model window, with a default name for the Data Model.


Developer/2000 Report Builder for Windows 95 / NT
File Edit View Program Tools Window Help
Untitled: Report Editor - Data Model
UNTITLED

Step 5: The Report Editor is the tool that you will use to create your data model for the report. Click on the SQL icon (See the following) in the toolbar located on the left hand side, and drag and drop it on the palette. Immediately, the SQL Query Statement window will appear. Type in the displayed SQL query to view student information in order of major.

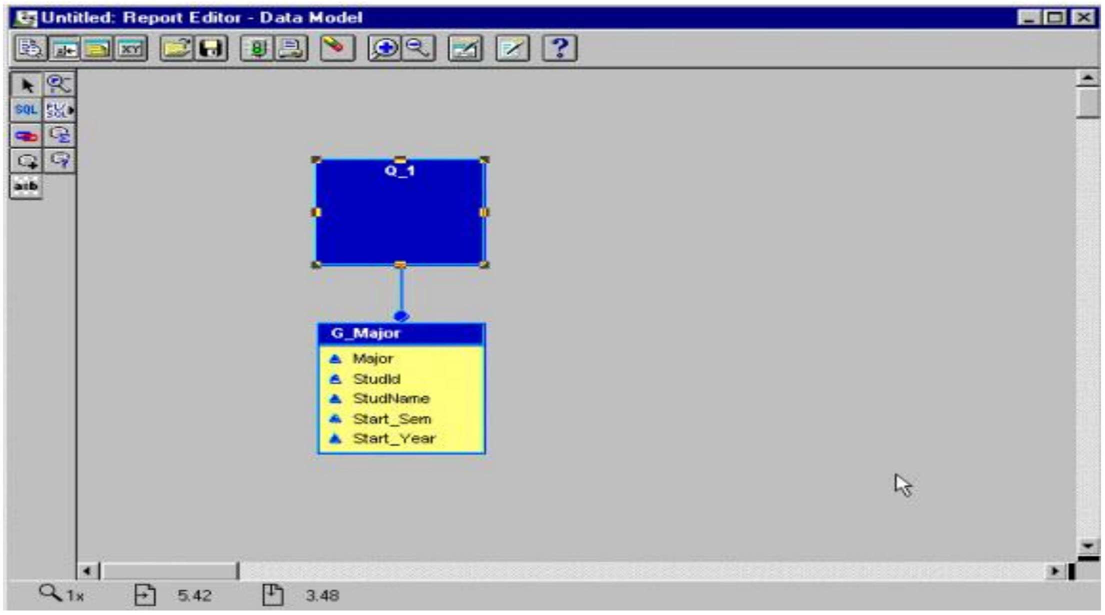




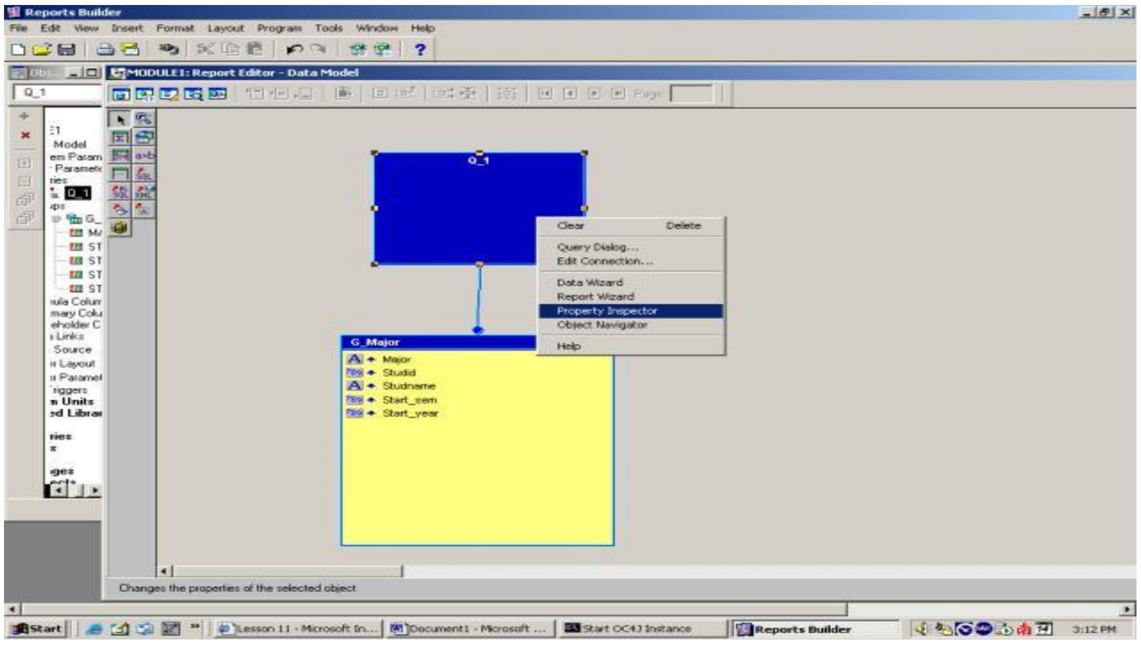
```
select Major, StudId, StudName, Start_Sem, Start_Year from Student
order by Major
```

Step 6: Once you click OK, Oracle10g Forms will prompt you to connect to the database. Type your User Name, Password and Database.

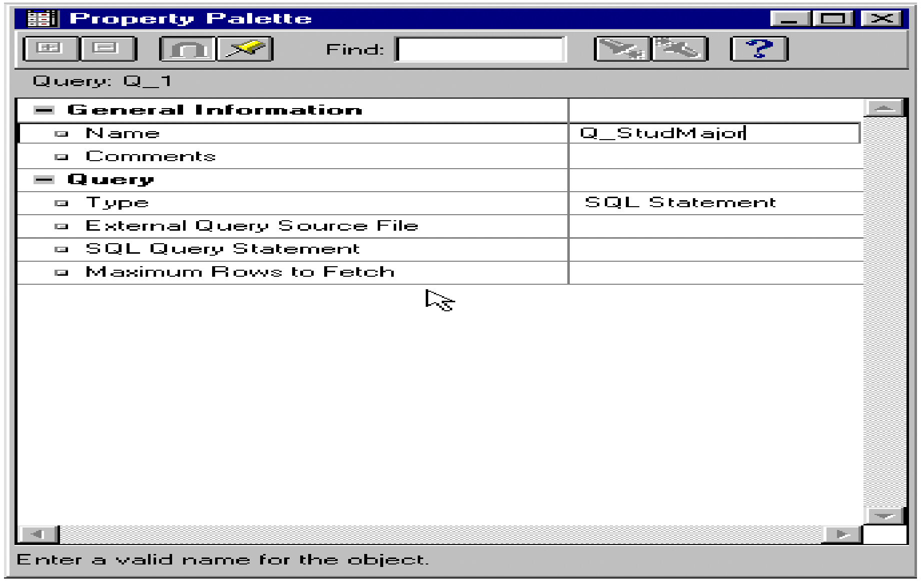
Step 7: See the data model, where Q_1 stands for the name of the query and G_Major, stands for its associated record group which contains the list of fields that you will be able to view in your report.



Step 8: To change the name of your query, right click on it and select the Property Inspector:

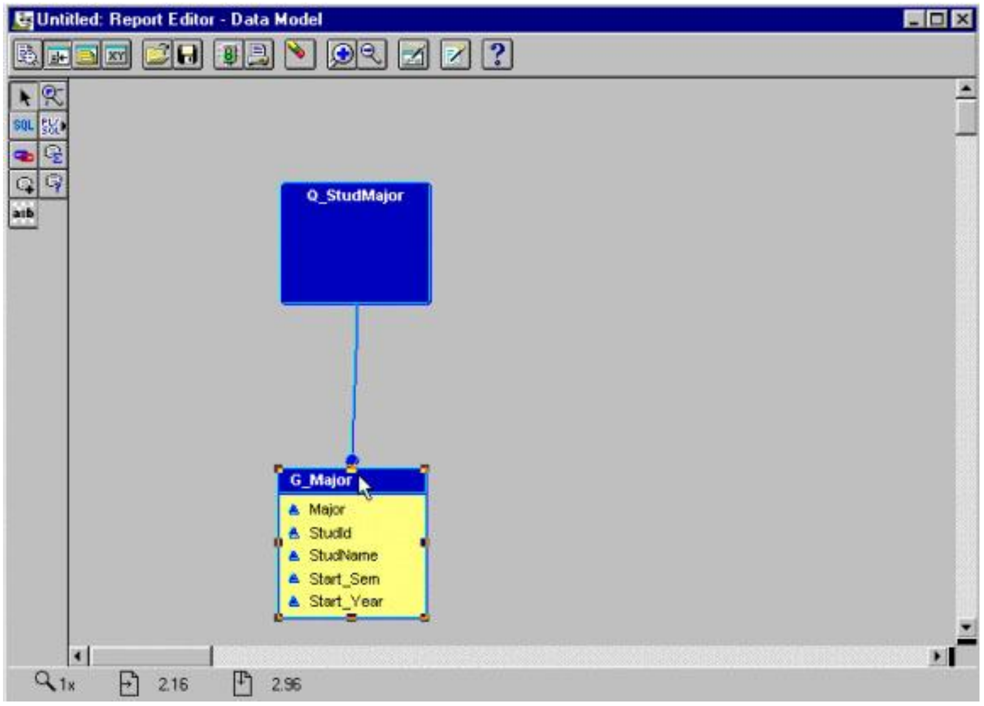


Step 9: Immediately, you will see the window for the Property Inspector. Change the name by typing in the name (Q_StudMajor) beside the 'Name' tab, and press enter. You can also change or edit your SQL query by clicking on the space beside the SQL Query Statement tab.



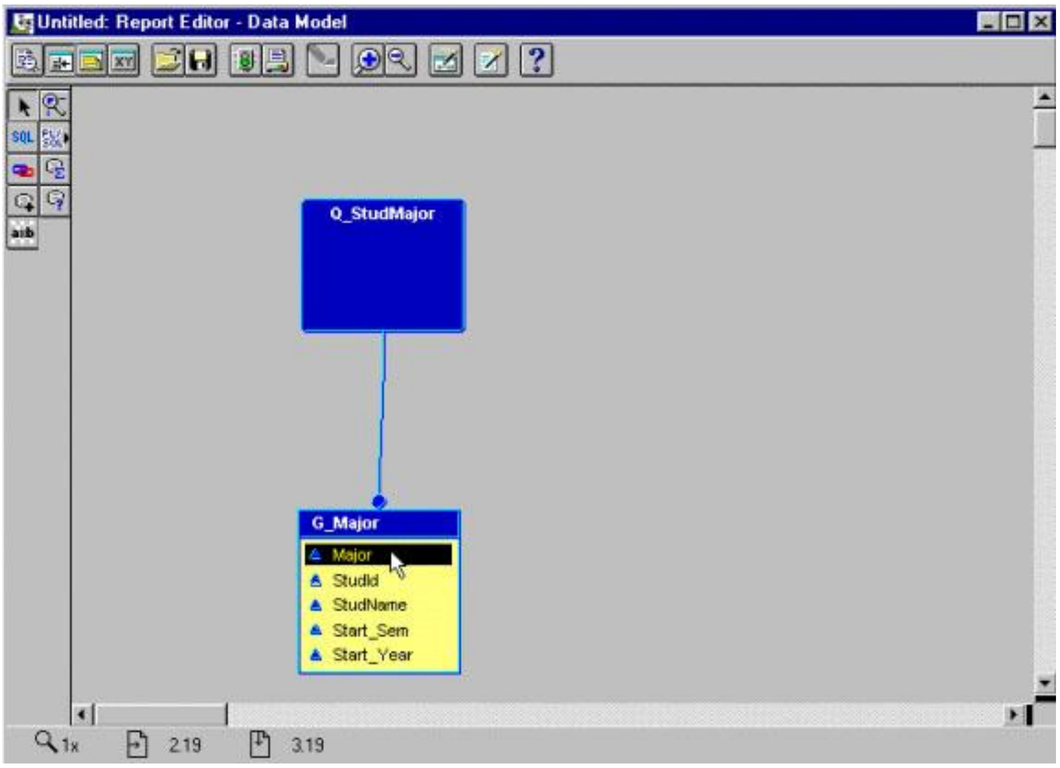
General Information	
Name	Q_StudMajor
Comments	
Query	
Type	SQL Statement
External Query Source File	
SQL Query Statement	
Maximum Rows to Fetch	

Step 10: The Data Model should now look like the following:

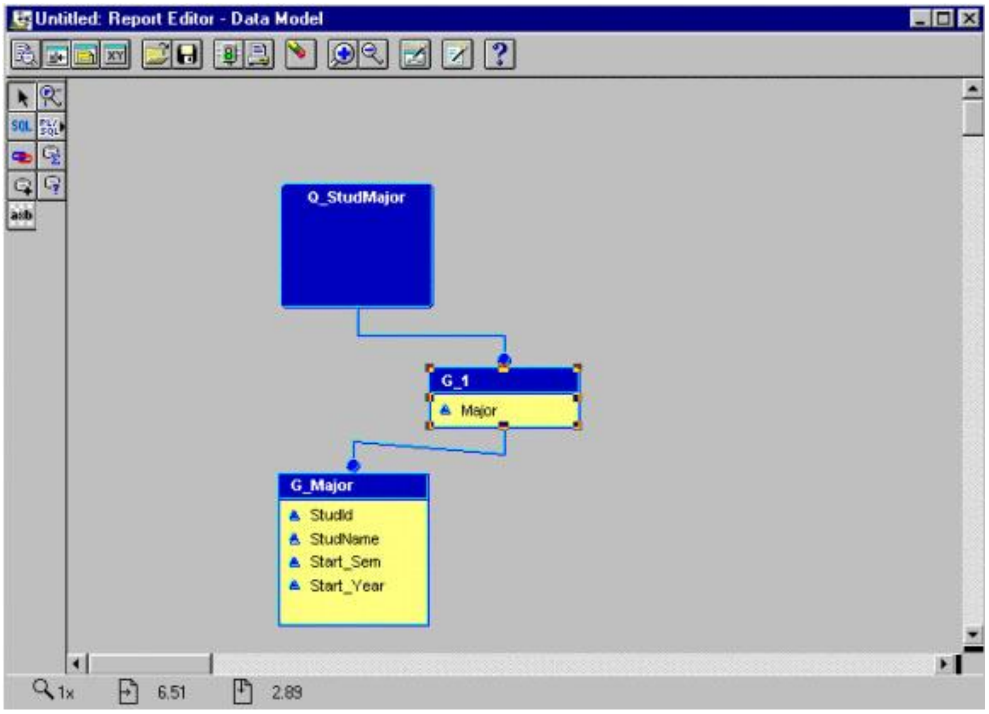


Step 11: Recall *that we have been asked to create a report that will display a list of students and their related information organized by Majors*. To do this, move the Major records into a separate record group. In Oracle10g Reports terms, it is called to 'break out'. First, click on the G_Major, and drag and

lower it to create more space between the record group and the query. Then select Major, and drag and drop it on the line connecting Q_StudMajor and G_Major.

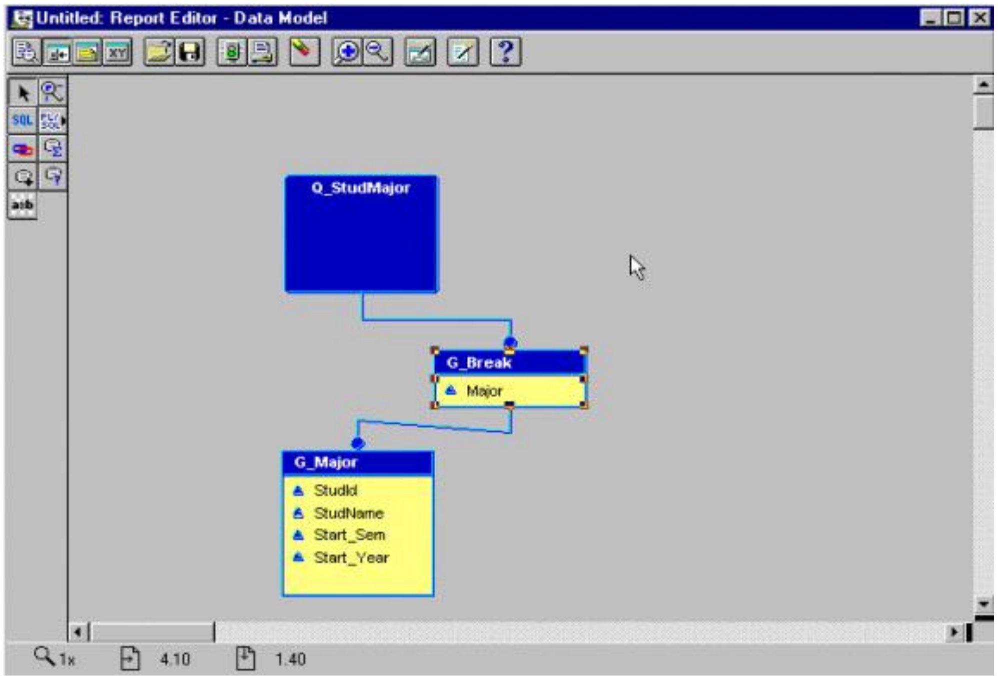


Step 12: The Data Model should now look like the one in Figure 11.13 with a new group for Major.



Step 13: Right click on the G_1 to go to its Property Palette. Change its name to G_Break.

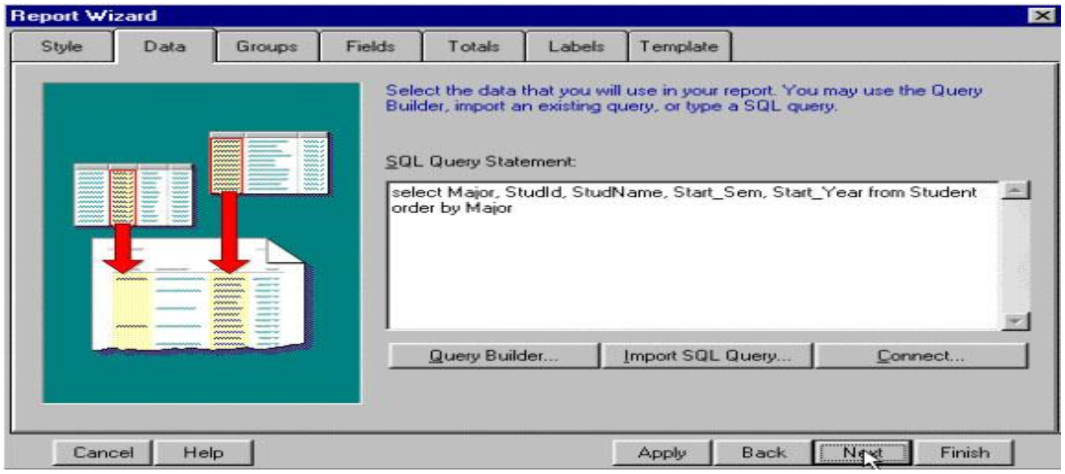
Step 14: The Data Model should now look like the following:



Step 15: Now select Report Wizard from the Tools Menu to generate the report.

Step 16: Will now see the first screen of the Report Wizard. Select "Create both Web & Paper Layout", click next, and Type in "List of Students by Major" in the Title box. Next, select the radio button for *Group Above* in order to create breaks after record groups for each Major. Now, click Next.

Step 17: Will now see the SQL statement again. You can edit your statement here if you choose to. At this time we will use the query that we had entered earlier. Click Next.

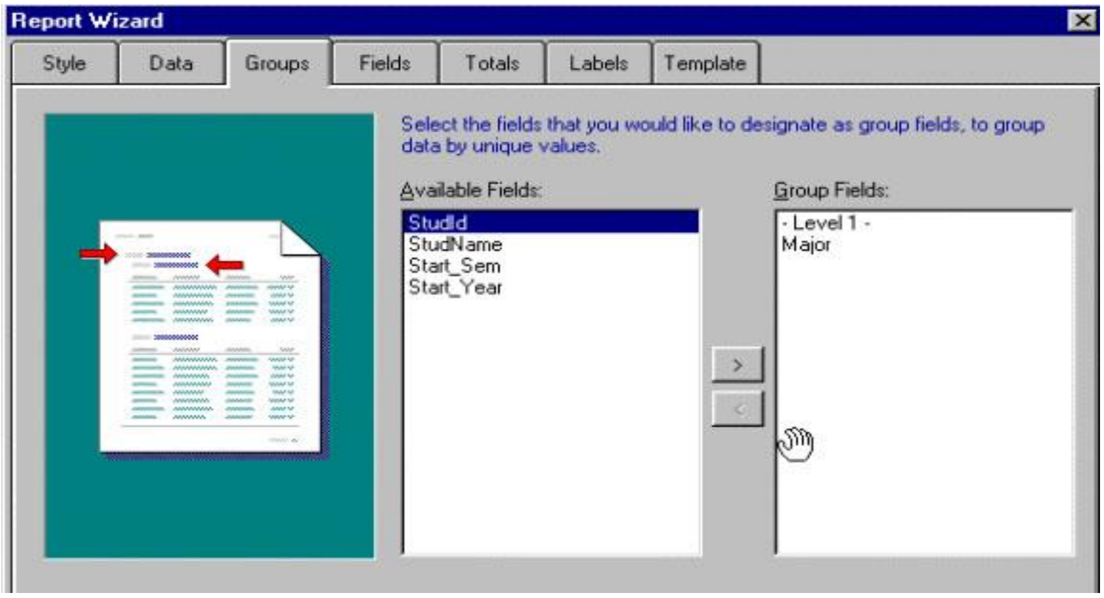


Select the data that you will use in your report. You may use the Query Builder, import an existing query, or type a SQL query.

SQL Query Statement:

```
select Major, StudId, StudName, Start_Sem, Start_Year from Student
order by Major
```

Step 18: Will now be prompted to select the fields that you would like to designate as group fields. Selected Major into Group Fields window (see the following). Now, select the next tab, Fields.

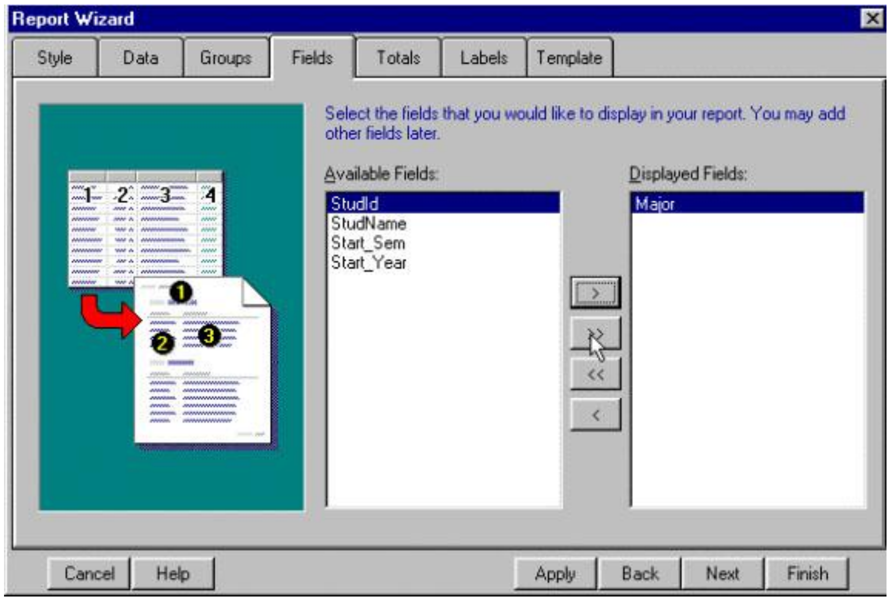


Select the fields that you would like to designate as group fields, to group data by unique values.

Available Fields:
StudId
StudName
Start_Sem
Start_Year

Group Fields:
- Level 1 -
Major

Step 19: Will now be asked to select the fields that you will display in your report. We would like to see all the fields, so click on the double right facing arrows to select all the fields and click next.



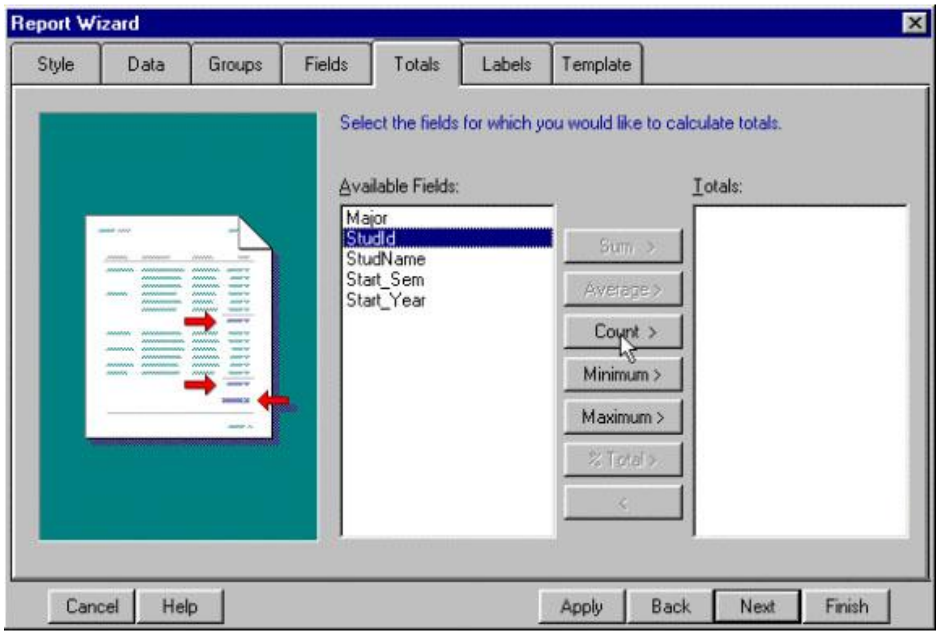
Select the fields that you would like to display in your report. You may add other fields later.

Available Fields:
StudId
StudName
Start_Sem
Start_Year

Displayed Fields:
Major

Selecting the Fields that are to be displayed in the Report

Step 20: Will now be prompted to select fields for which you would like to calculate totals. Let us assume that we have been asked to provide the total number of students in each major and also the Grand total of the number of students. To do this, select StudID, and click on Count.

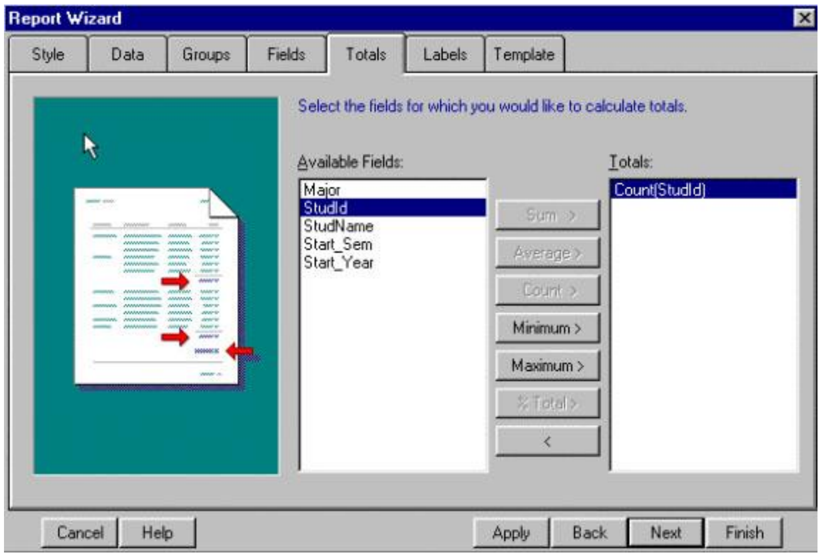


Select the fields for which you would like to calculate totals.

Available Fields:
Major
StudId
StudName
Start_Sem
Start_Year

Totals:

Step 21: The Screen should look like the following with Count (StudId) in the Totals column). Click Next.

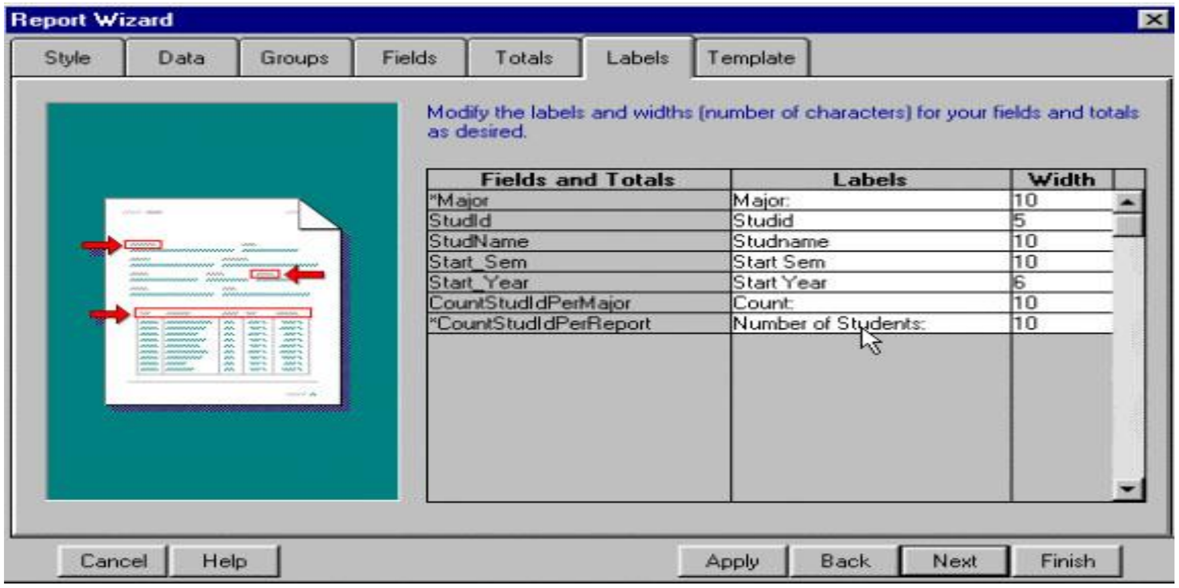


Select the fields for which you would like to calculate totals.

Available Fields:
Major
StudId
StudName
Start_Sem
Start_Year

Totals:
Count(StudId)

Step 22: Now modify the labels and their width. In this case we have put a colon and a space after Major and have changed the label for CountStudIdPerReport to "Number of Students: " and click Next.

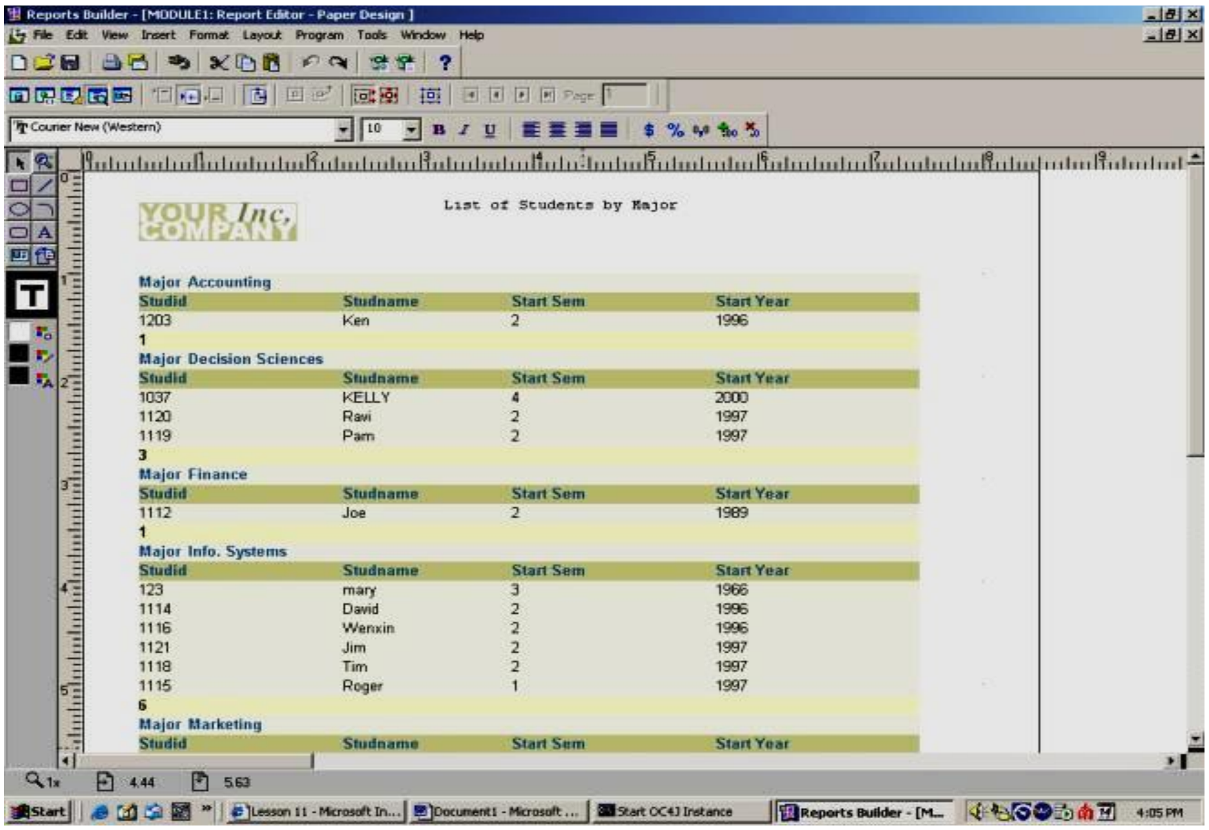


Modify the labels and widths (number of characters) for your fields and totals as desired.

Fields and Totals	Labels	Width
*Major	Major:	10
StudId	Studid	5
StudName	Studname	10
Start_Sem	Start Sem	10
Start_Year	Start Year	6
CountStudIdPerMajor	Count:	10
*CountStudIdPerReport	Number of Students:	10

Step 23: The final modification involves selecting an appropriate template for the report. In this case, we will select Beige from the list provided. You are free to select any template of your choice. Click Finish.

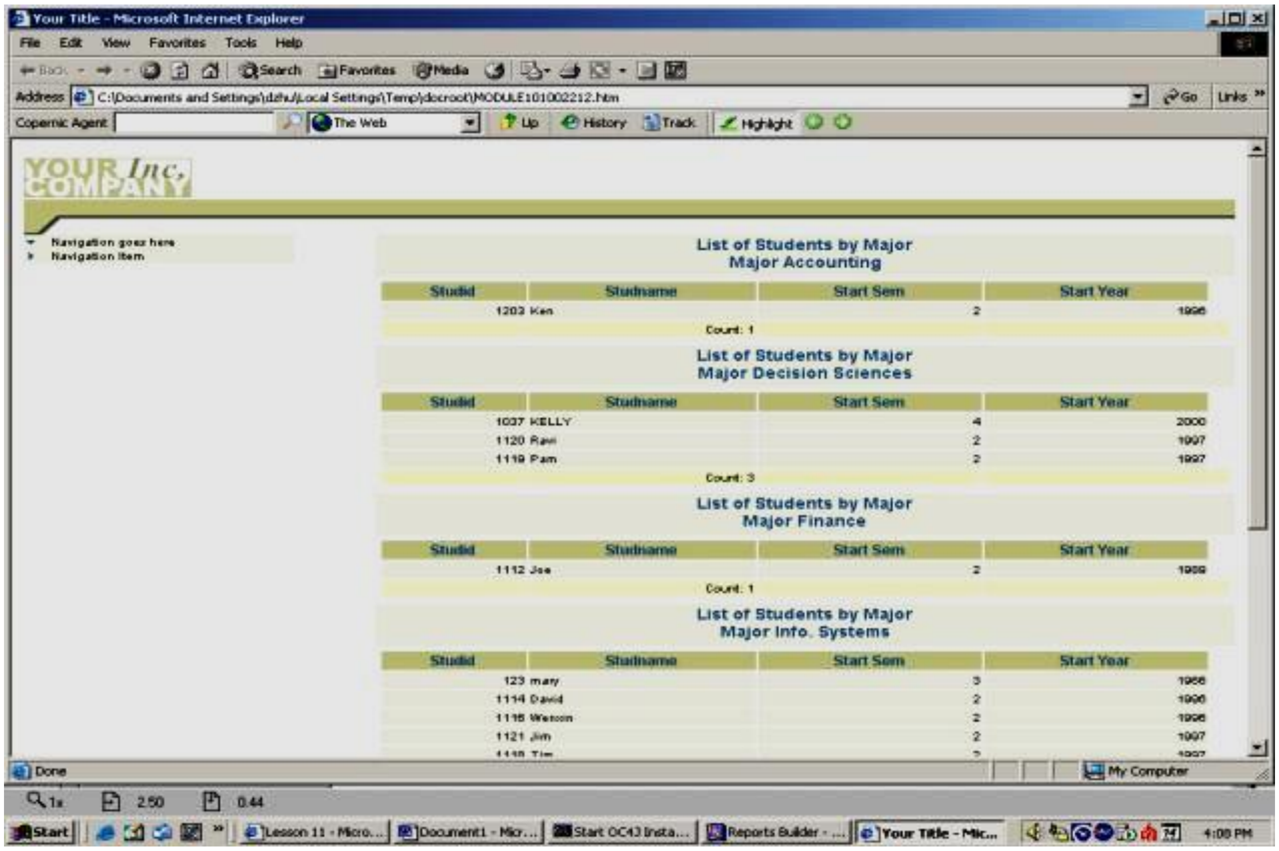
Step 24: Run the Report. Your report should now look like the following:


Reports Builder - [MODULE1: Report Editor - Paper Design]
File Edit View Insert Format Layout Program Tools Window Help
Courier New (Western)
10
YOUR Inc. COMPANY

```
List of Students by Major

Major Accounting
Studid      Studname      Start Sem      Start Year
1203        Ken           2              1996
1
Major Decision Sciences
Studid      Studname      Start Sem      Start Year
1037        KELLY         4              2000
1120        Ravi          2              1997
1119        Pam           2              1997
3
Major Finance
Studid      Studname      Start Sem      Start Year
1112        Joe           2              1989
1
Major Info. Systems
Studid      Studname      Start Sem      Start Year
123         mary          3              1966
1114        David         2              1996
1116        Wenxin        2              1996
1121        Jim           2              1997
1118        Tim           2              1997
1115        Roger         1              1997
6
Major Marketing
Studid      Studname      Start Sem      Start Year
```

Step 25: Web-enable Reports: To web enable the report, click Run button on the top of the window, Oracle10g Reports will generate a report in HTML version. You can deploy this file on a web server to publish it on-line.



List of Students by Major
Major Accounting

Studid	Studname	Start Sem	Start Year
1203	Ken	2	1996

Count: 1

List of Students by Major
Major Decision Sciences

Studid	Studname	Start Sem	Start Year
1037	KELLY	4	2000
1120	Ravi	2	1997
1119	Pam	2	1997

Count: 3

List of Students by Major
Major Finance

Studid	Studname	Start Sem	Start Year
1112	Joe	2	1999

Count: 1

List of Students by Major
Major Info. Systems

Studid	Studname	Start Sem	Start Year
123	mary	3	1966
1114	David	2	1996
1116	Watson	2	1996
1121	Jim	2	1997

<u>RESULT:</u>

Thus the Oracle reports for the list of students categorized by major is generated successfully.

10. BANKING MANAGEMENT SYSTEM

AIM:

To develop a mini project named “Banking Management System” that performs deposit, withdrawal and mini statement operations in Visual Basic.

ALGORITHM:

Step1:Create a table custlist with the following attributes

NAME	CHAR
ACCNO	NUMBER
CITY	VARCHAR2
CONTACT	NUMBER
DOB	DATE
AMOUNT	NUMBER

Step2:Insert few records in custlist table

Step3:Create a table ministate with the following attributes

ACCNO	NUMBER
TRANSDATE	CHAR
TYPE	CHAR
AMOUNT	NUMBER

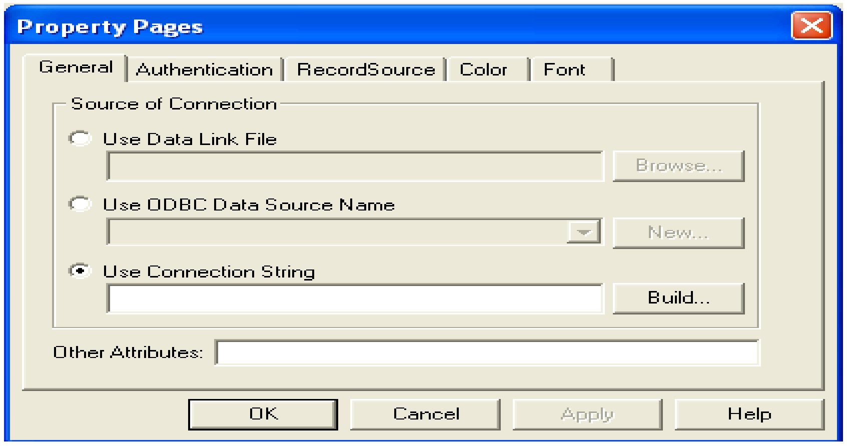
Step4:Save the table

Step5:Create a data source name using Control Panel

Step6:Create a new Project in Visual Basic by choosing Start-> All Programs->Microsoft Visual Studio 6.0->Microsoft Visual Basic 6.0

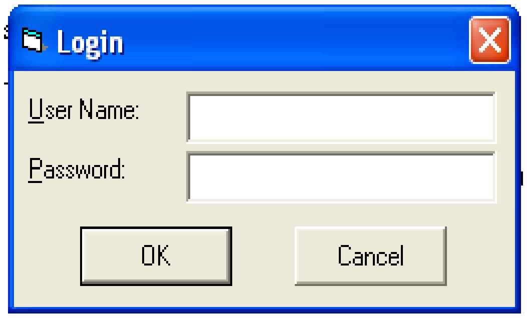
Step7:Add the component Microsoft ADO Control 6.0 and insert it into the project form

Step8:Identify the connection string by right click the ADO Control 6.o and select the property window

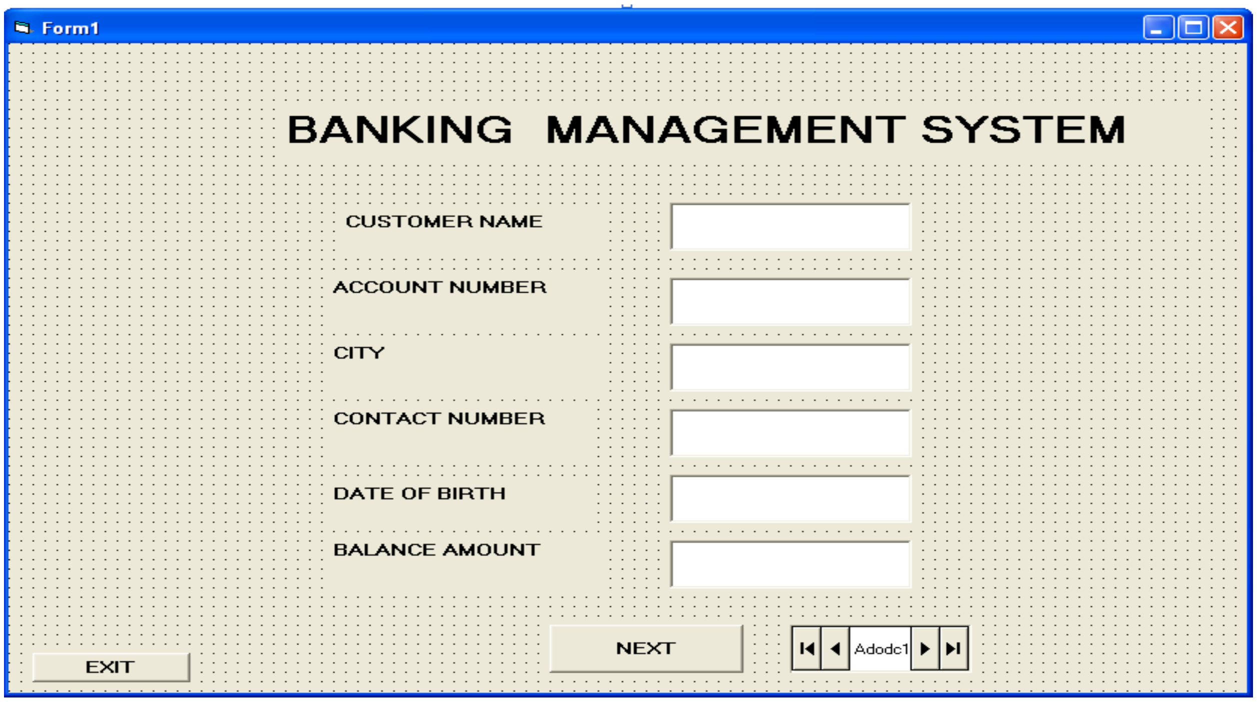


Step9:Design the following form

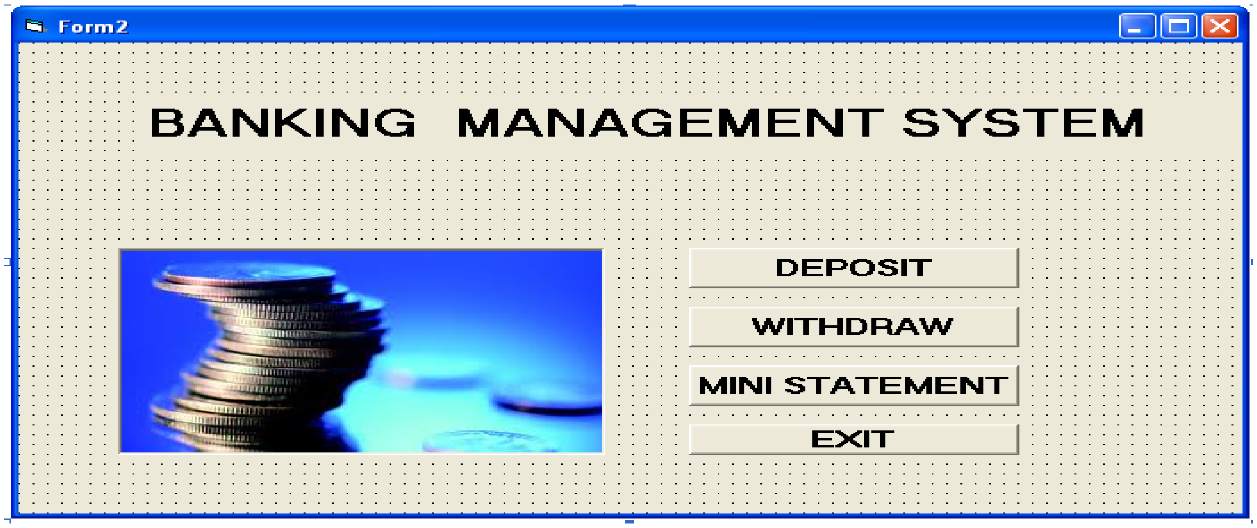
<u>Login Form (FrmLogin.frm)</u>



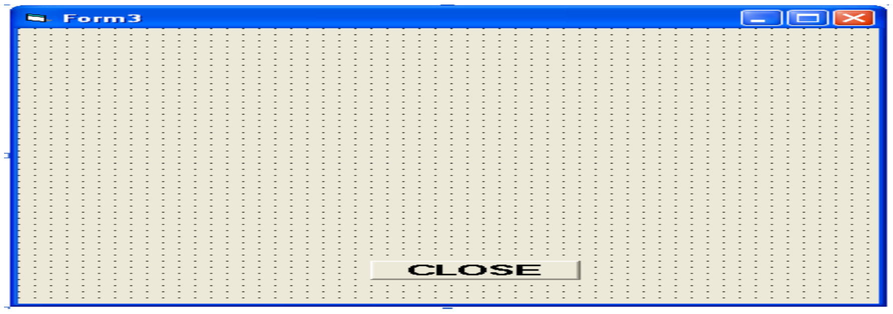
<u>Administrator Module (Form1.frm)</u>



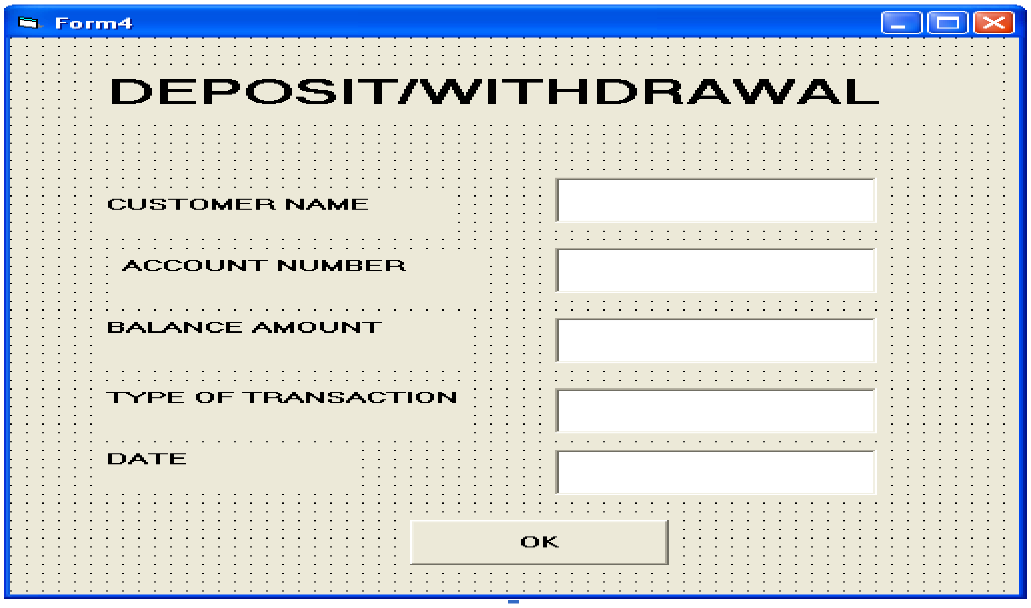
User Module (Form2.frm



Mini statement Module (Form3.frm)



Deposit/Withdrawal Processing Module (Form4.frm)



Step10:Add the code for the various operations such as withdrawal, deposit and ministatement.

Step11:Execute the project

Source Code

<u>Frmlogin.frm</u>

```
Option Explicit
Public LoginSucceeded As Boolean
----------------------------------------------------------------------------------------------------------------
Private Sub cmdCancel_Click()
   'set the global var to false
   'to denote a failed login
   LoginSucceeded = False
   Me.Hide
End Sub
----------------------------------------------------------------------------------------------------------------
Private Sub cmdOK_Click()
   'check for correct password
   If (txtUserName = "11998765" Or txtUserName = "11665432" Or txtUserName = "12887656" Or
txtUserName = "11212222") And txtPassword = "password" Then
      'place code to here to pass the
      'success to the calling sub
      'setting a global var is the easiest
      LoginSucceeded = True
      Form2.Show
      frmLogin.Hide
   ElseIf txtUserName = "admin" And txtPassword = "pass" Then
   LoginSucceeded = True
   Form1.Show
   Else
      MsgBox "Invalid Password, try again!", , "Login"
      txtPassword.SetFocus
      SendKeys "{Home}+{End}"
   End If
End Sub
```

Form1.frm

```
Dim con As ADODB.Connection
Dim rs As ADODB.Recordset
-------------------------------------------------------------------------------------------------------------------Private
Sub Command1_Click()
'code for move the next record
rs.MoveNext
If (rs.EOF) Then
rs.MoveFirst
MsgBox "You are in Last Record"
End If
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
Text6.Text = rs.Fields(5)
End Sub
```

```
Private Sub Command2_Click()
End
End Sub
-------------------------------------------------------------------------------------------------------------------
Private Sub Form_Load()
Set con = New ADODB.Connection
Set rs = New ADODB.Recordset
con.Open
"DSN=bank;UID=b5it56;PWD=student;DBQ=10.0.0.10/CCETBASE;DBA=W;APA=T;EXC=F;FEN=T;QT
O=T;FRC=10;FDL=10;LOB=T;RST=T;BTD=F;BAM=IfAllSuccessful;NUM=NLS;DPM=F;MTS=T;MDI=F;C
SR=F;FWC=F;FBS=64000;TLO=O;"
rs.Open "select * from custlist", con, adOpenDynamic
```

```
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
Text3.Text = rs.Fields(2)
Text4.Text = rs.Fields(3)
Text5.Text = rs.Fields(4)
Text6.Text = rs.Fields(5)
End Sub
-----------------------------------------------------------------------------------------------------------
```

Form2.frm

```
Private Sub Command1_Click()
Form4.Show
Form4.Text3.Text = "DEPOSIT"
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Command2_Click()
Form4.Show
Form4.Text3.Text = "WITHDRAW"
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Command3_Click()
End
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Command4_Click()
Form3.Show
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Form_Load()
Form1.Hide
frmLogin.Hide
End Sub
```

Form4.frm

```
Dim con As ADODB.Connection
Dim rs As ADODB.Recordset
Dim amt As Integer
Dim dtmTest As Date
-----------------------------------------------------------------------------------------------------------------
Private Sub Command1_Click()
dtmTest = DateValue(Now)
Text4.Text = dtmTest
amt = Val(Text6.Text)
If Text3.Text = "DEPOSIT" Then
rs.Close
query = "update custlist set amount = amount + " & amt & " where accno = " & frmLogin.txtUserName
rs.Open query, con, adOpenDynamic
MsgBox " Your Amount Deposited"
Else
rs.Close
query2 = "update custlist set amount = amount - " & amt & " where accno = " &
frmLogin.txtUserName
rs.Open query2, con, adOpenDynamic
MsgBox " Please Collect Your Amount "
End If
rs.Open "insert into ministate values(" & Text2.Text & ", '" & Text4.Text & "','" & Text3.Text & "'," &
Text6.Text & ")", con, adOpenDynamic
Form2.Show
Form4.Hide
End Sub
-----------------------------------------------------------------------------------------------------------------
Private Sub Form_Load()
Label5.Visible = False
Text4.Visible = False
Set con = New ADODB.Connection
Set rs = New ADODB.Recordset
con.Open
"DSN=bank;UID=b5it56;PWD=student;DBQ=10.0.0.10/CCETBASE;DBA=W;APA=T;EXC=F;FEN=T;QT
O=T;FRC=10;FDL=10;LOB=T;RST=T;BTD=F;BAM=IfAllSuccessful;NUM=NLS;DPM=F;MTS=T;MDI=F;C
SR=F;FWC=F;FBS=64000;TLO=O;"
rs.Open "select * from custlist where accno=" & frmLogin.txtUserName, con, adOpenDynamic
Text1.Text = rs.Fields(0)
Text2.Text = rs.Fields(1)
End Sub
-----------------------------------------------------------------------------------------------------------------
```

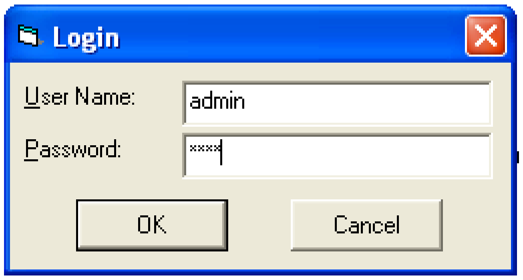
Form3.frm

```
Dim con As ADODB.Connection
Dim rs As ADODB.Recordset
-----------------------------------------------------------------------------------------------------------------
Private Sub Command1_Click()
End
End Sub
-----------------------------------------------------------------------------------------------------------------
Private Sub Form_Load()
Set con = New ADODB.Connection
Set rs = New ADODB.Recordset
con.Open
"DSN=bank;UID=b5it56;PWD=student;DBQ=10.0.0.10/CCETBASE;DBA=W;APA=T;EXC=F;FEN=T;QT
O=T;FRC=10;FDL=10;LOB=T;RST=T;BTD=F;BAM=IfAllSuccessful;NUM=NLS;DPM=F;MTS=T;MDI=F;C
SR=F;FWC=F;FBS=64000;TLO=O;"
rs.Open "select * from ministate where accno = " & frmLogin.txtUserName, con, adOpenDynamic
Print "-----------------------------------------------------------------------------"
Print "AccountNo    Date of Transaction          Mode of Transaction     Amount Transffered"
Print "_____________________________________________________________________________"
While Not rs.EOF
Print rs(0) & "                " & rs(1) & "          " & rs(2) & "              " & rs(3)
```

```
rs.MoveNext
Wend
End Sub
```

Snapshots

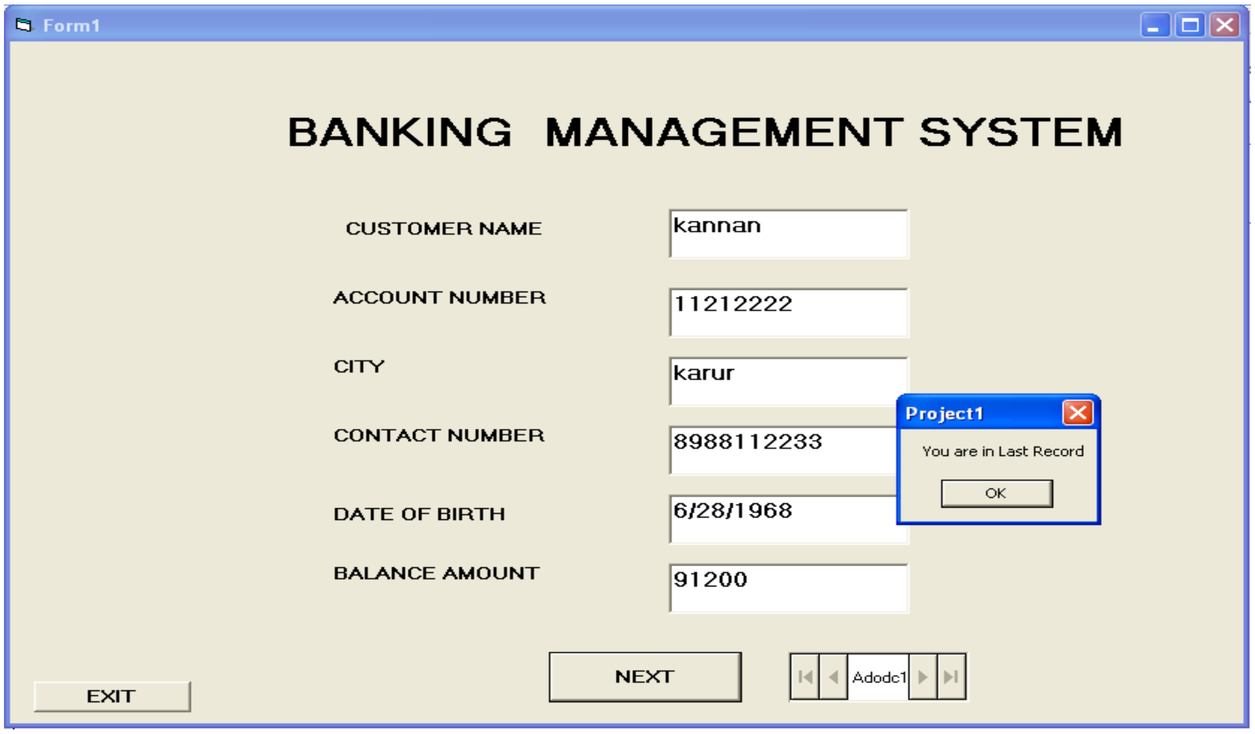
Login Page



User Name: admin

Password: ****

Administrator Module



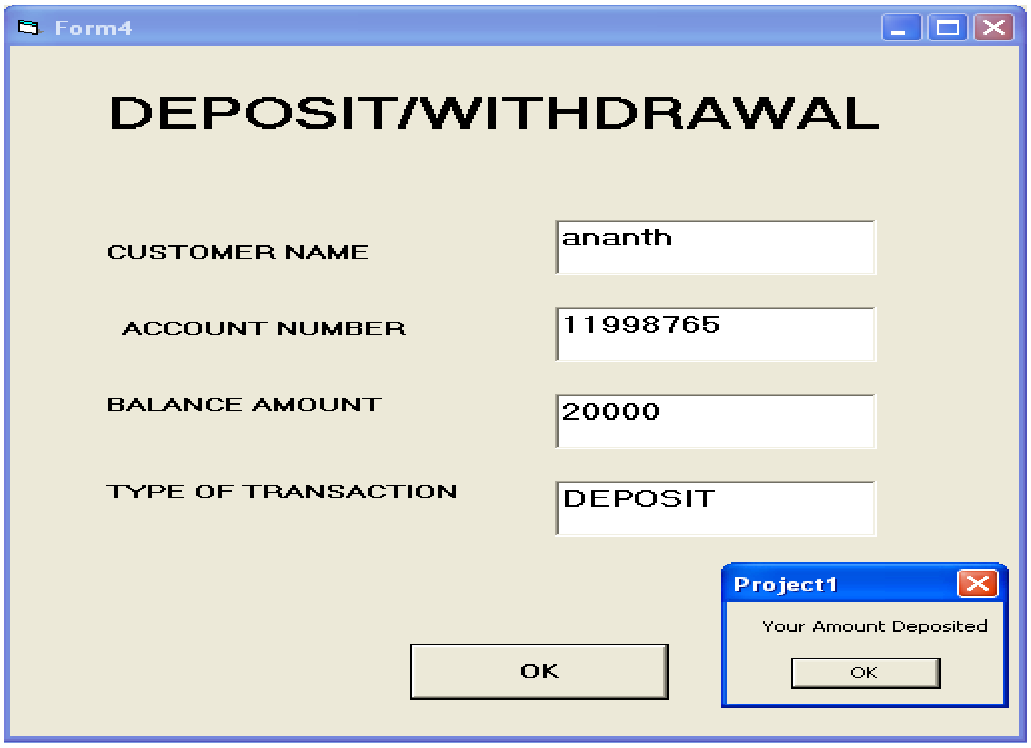
BANKING MANAGEMENT SYSTEM

CUSTOMER NAME	kannan
ACCOUNT NUMBER	11212222
CITY	karur
CONTACT NUMBER	8988112233
DATE OF BIRTH	6/28/1968
BALANCE AMOUNT	91200

Project1

You are in Last Record

Deposit Module



Form4

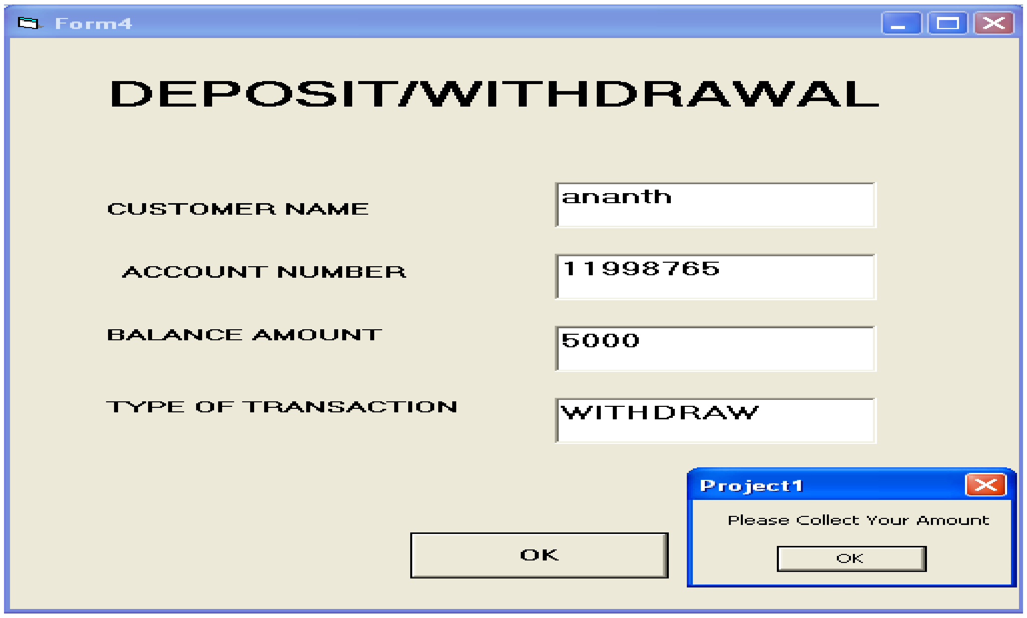
DEPOSIT/WITHDRAWAL

CUSTOMER NAME	ananth
ACCOUNT NUMBER	11998765
BALANCE AMOUNT	20000
TYPE OF TRANSACTION	DEPOSIT

Project1

Your Amount Deposited

Withdrawal Module



Form4

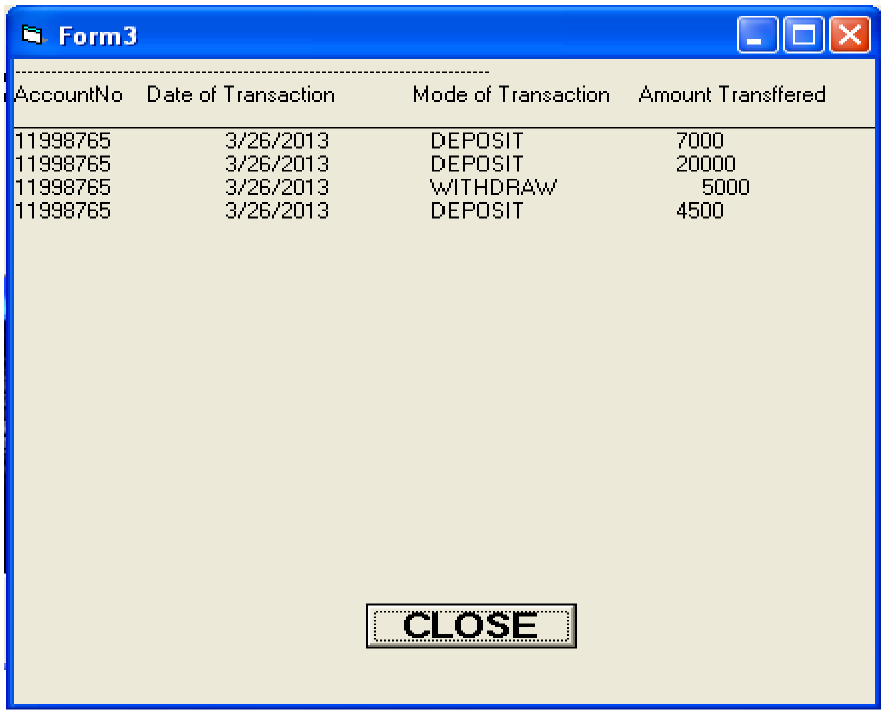
DEPOSIT/WITHDRAWAL

CUSTOMER NAME	ananth
ACCOUNT NUMBER	11998765
BALANCE AMOUNT	5000
TYPE OF TRANSACTION	WITHDRAW

Project1

Please Collect Your Amount

Mini statement module



Form3

```
------------------------------------------------------------------------
AccountNo   Date of Transaction      Mode of Transaction    Amount Transffered

11998765            3/26/2013         DEPOSIT              7000
11998765            3/26/2013         DEPOSIT              20000
11998765            3/26/2013         WITHDRAW               5000
11998765            3/26/2013         DEPOSIT              4500
```

RESULT:

Thus the project named "Banking Management System" has been developed and various operations such as deposit, withdrawal and mini statement operations have been performed using Visual Basic.