<u>DATA DEVELOPMENT LIFE CYCLE</u>

EX NO: 1, (A)

1. <u>CREATE</u>

Creating a Database

Syntax : CREATE DATABASE <DB_NAME>;
Example : CREATE DATABASE Test;

Creating a Table

Command : CREATE

Purpose : To create objects in the database

Syntax : CREATE TABLE table_name (column_name1 data_type,column_name2 data_type, column_name3 data_type,...);

Example : create table emp11(emp_namevarchar(20),emp_phone number(10),emp_addr varchar(30));

Sample output: Table created.

2. <u>ALTER</u>

Command : ALTER

Purpose : Alters the structure of the database

Syntax :

i) To add a column in a table

```
ALTER TABLE table_name ADD column_name datatype;
```

Example1: alter table emp11 add emp_id varchar(20);
Sample output: Table altered.
Example2: alter table emp11 add emp_salary number(10);
Sample output: Table altered.

ii) To delete a column in a table

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Example: alter table emp11 drop column emp_salary;
Sample output: Table altered.

iii) To change the data type of a column in a table

```
ALTER TABLE table_nameMODIFY column_name datatype;
```

Example: alter table emp11 modify emp_id number(10);

Sample output: Table altered.

3. RENAME

Command	: RENAME
Purpose	: Rename an object
Syntax	: RENAME old_table_name to new_table_name;
Example	: rename emp11 to emp123;
Sample output	: Table renamed.

4. DESCRIBE

Command	: DESCRIBE
Purpose	: Describes the structure of the table
Syntax	: Desc table_name;
Example	: desc emp11;

Sample output:

```
Name                                      Null?      Type
----------------------------------------- -------- ------------------
EMP_NAME                                             VARCHAR2(20)
EMP_PHONE                                            NUMBER(10)
EMP_ADDR                                             VARCHAR2(30)
EMP_ID                                               NUMBER(10)
EMP_SALARY                                           NUMBER(10)
```

5. DROP

Command	: DROP
Purpose	: Delete objects from the database
Syntax	: DROP TABLE table_name;
Example	: drop table emp123;

Sample output : Table dropped.

6. TRUNCATE

Command : TRUNCATE

Purpose : Remove all records from a table, including all spaces allocated for the records are removed

Syntax : TRUNCATE TABLE table_name;

Example : truncate table emp123;

Sample output : Table truncated.

EX NO: 1(B)

1. INSERT

Command : INSERT

Purpose : Insert data into a table.

Syntax :

i) To insert one row at at time

INSERT INTO table_name VALUES (value1, value2, value3,...);

Example : insert into emp123values('Ram',9787563641,'India');

Sample output: 1 row created.

ii) To insert many rows at a time

INSERT INTO table_name values (&column1,&column2,….);

Example : insert into emp123values('&emp_name','&emp_phone','&emp_addr');

```
Enter value for emp_name: Joseph

Enter value for emp_phone: 9787654123

Enter value for emp_addr: Tamilnadu

old 1: insert into emp123 values('&emp_name','&emp_phone','&emp_addr')
```

new 1: insert into emp123 values('Joseph','9787654123','Tamilnadu')

Sample output: 1 row created.

iii) Inserting Data's in specified columns:

INSERT INTO table_name(col1,col2,…….,coln) VALUES(val1,val2,……,valn);

2. UPDATE

Command : UPDATE

Purpose : Updates existing data within a table.

Syntax : UPDATE table_name SET column1=value, column2=value2,...
WHERE (condition);

Example : update emp123 set emp_id=40 where emp_addr='Tamilnadu';

Sample output: 1 row updated.

3. DELETE

Command : DELETE

Purpose : Deletes all records from a table, the space for the records remain.

Syntax : DELETE FROM table_name WHERE (condition);

Example : delete from emp123 where emp_id=30;

Sample output : 1 row deleted.

4. SELECT

Command : SELECT

Purpose : fetch the data from a table which returns data in the table.

Syntax : SELECT column1, column2, columnN FROM table_name;

Example1 : Select * from customers;

Sample output:

```
+----+---------+-----+----------+----------+
| ID | NAME        | AGE | ADDRESS  | SALARY          |
+----+---------+-----+----------+----------+
|  1 | Ramesh      |  32 |Ahmedabad |  2000.00 |
|     2 | Khilan          |     25 | Delhi                |         1500.00 |
|     3 | kaushik         |     23 | Kota                 |         2000.00 |
|     4 | Chaitali|              25 | Mumbai               |         6500.00 |
+----+---------+-----+----------+----------+
```

Example2 : Select id, name, salary from customers;

Sample output:

```
+----+---------+----------+
| ID | NAME    | SALARY   |
+----+---------+----------+
|     1 | Ramesh         |         2000.00 |
|     2 | Khilan   |         1500.00 |
|     3 | kaushik |         2000.00 |
|     4 | Chaitali|         6500.00 |
+----+---------+----------+
```

5. <u>Group by</u>

Command : GROUP BY

Purpose : Group the collection of values based on an object.

Syntax : select obj.name/col. Name sum () / count ()…. from table_name group

by (obj.name/col.name);

Example : select rollno, sum (hscmarks) from student group by (rollnumber);

Sample output:

```
+--------+---------------+
| rollno | SUM(HSCMARKS) |
+--------+---------------+

|        45  |    34                 |

|        56  |    30                 |

|        89  |    56                 |

+--------+---------------+
```

6. <u>Having clause</u>

Command : HAVING

Purpose : Filters the data on the group row but not on the individual row.

Syntax :

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

Example : select rollnumber,sum((hscmarks*.30)+hscmarks) from student group by(rollnumber) having sum(hscmarks)<50;

Sample output:

```
ROLLNUMBER SUM((HSCMARKS*.30)+HSCMARKS)

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

     1                     15.6

    45                     44.2

    56                     39
```

DESIGN AN ER DIAGRAM

EX.NO: 2

AIM: Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

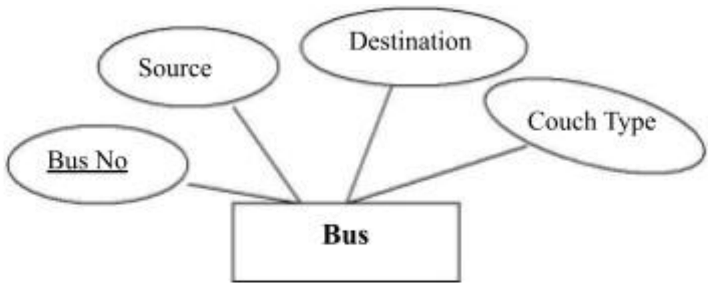
The Following are the entities:

1 .Bus

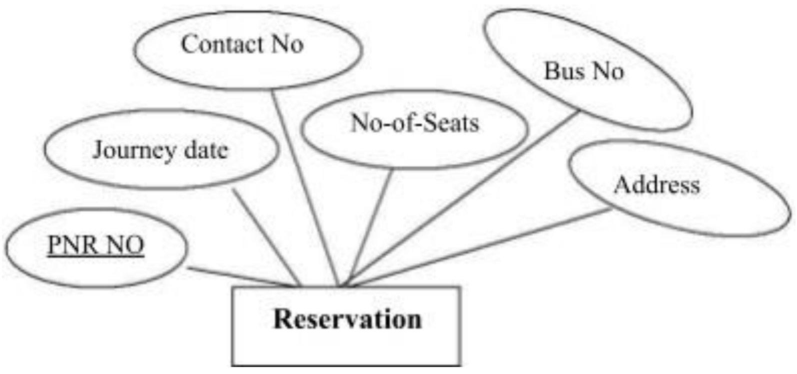
2. Reservation
3. Ticket
4. Passenger
5. Cancellation

The attributes in the Entities:

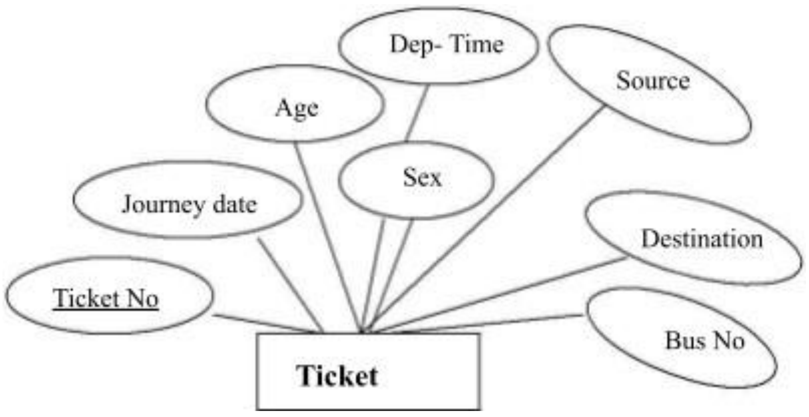
Bus:(Entity)



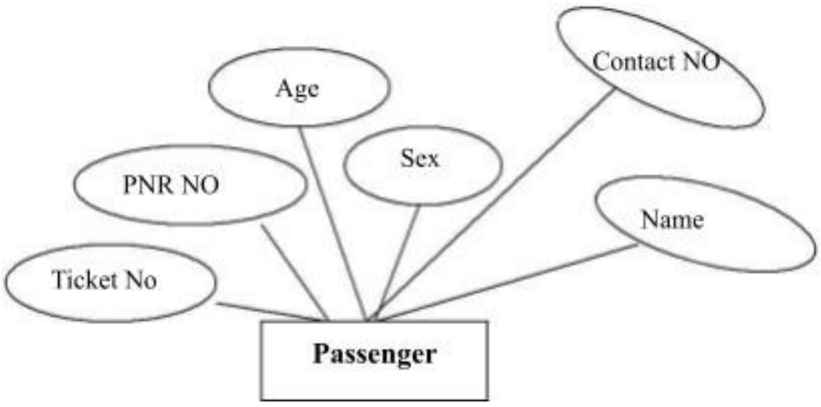
Reservation (Entity)



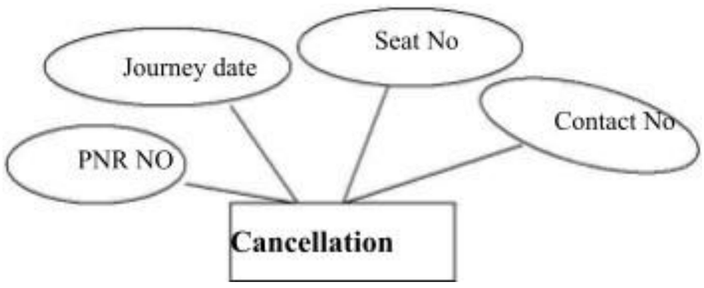
Ticket :(Entity)



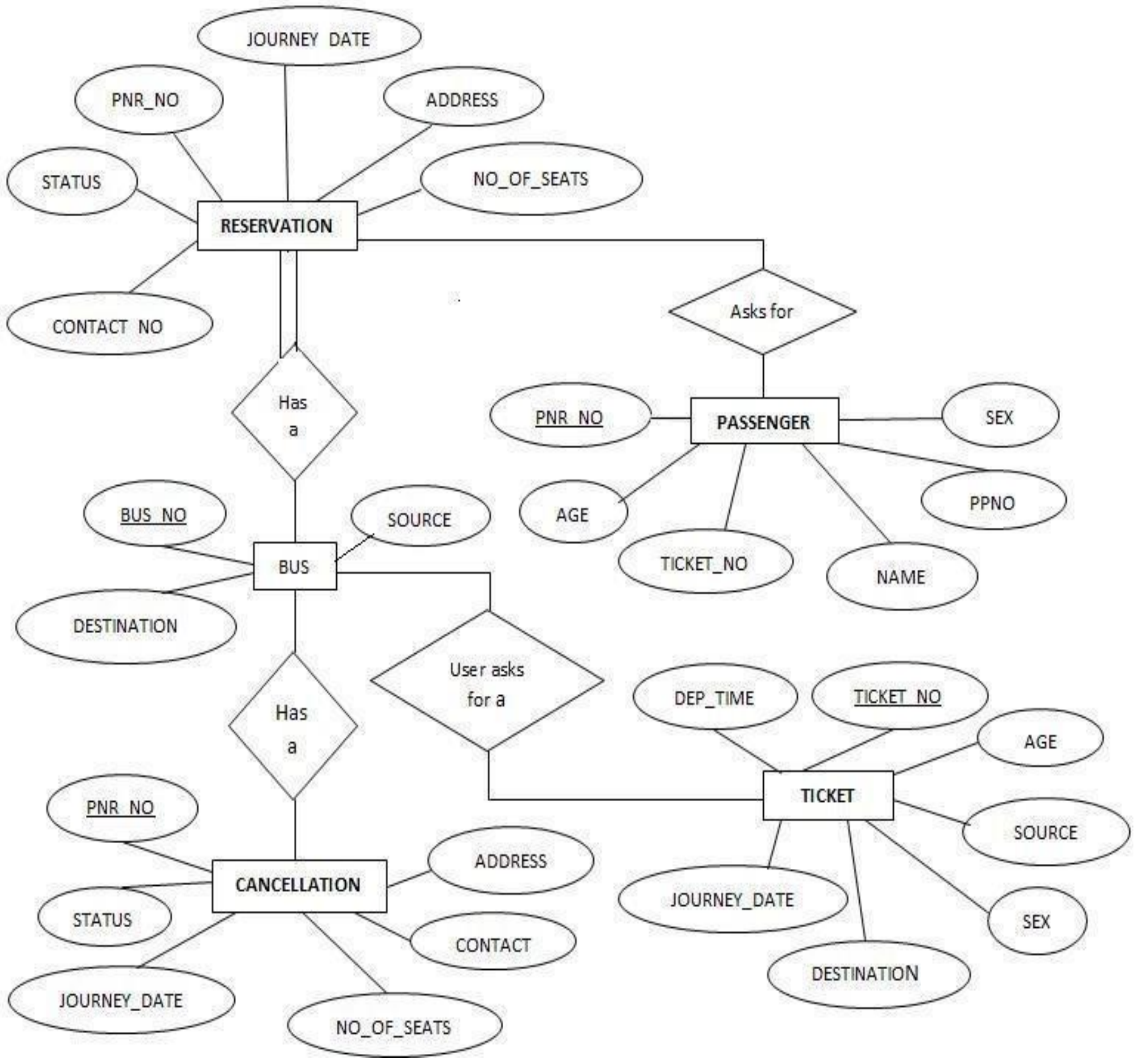
Passenger:



Cancellation (Entity)



Concept design with E-R Model:



RESULT: Thus the program to design a database using ER diagram has been done successfully.

EX.NO: 3 **VIEWS, SEQUENCES, SYNONYMS**

VIEWS

1. **Create Views**

   Syntax:

   CREATE VIEW view_name AS SELECT column1, column2... FROM table_name WHERE [condition];

   Example:

   Consider the CUSTOMERS table

```
+----+----------+-----+-----------+----------+

| ID | NAME       | AGE | ADDRESS    | SALARY   |

+----+----------+-----+-----------+----------+
|   1 | Ramesh        |    32  |  Ahmedabad    |    2000.00 |
|   2 | Khilan        |    25  |  Delhi        |    1500.00 |
|   3 | kaushik       |    23  |  Kota         |    2000.00 |
|   4 | Chaitali      |    25 | Mumbai        |       6500.00  |
|   5 | Hardik        |    27 | Bhopal        |       8500.00  |
|   6 | Komal         |    22 | MP            |       4500.00  |
|   7 | Muffy         |    24 | Indore        |      10000.00  |
+----+----------+-----+-----------+----------+
```

Mysql > create view customers_view as select name, age from customers;

Sample Output: View created.

2. **<u>Select Views</u>**

<u>Select ALL columns from View</u>

Mysql > select * from customers_view;

Sample Output:

```
+----------+-----+

| name       | age |

+----------+-----+

| Ramesh     |    32 |

| Khilan     |    25 |

| kaushik    |    23 |
| Chaitali |    25 |
| Hardik   |    27 |
| Komal    |    22 |
| Muffy    |    24 |
+----------+-----+
```

SELECT SPECIFIED COLUMNS FROM VIEW

Mysql > select name from customers_view where age=32;

```
+----------+-----+
| name     | age |
+----------+-----+
| Ramesh   |    32 |
+----------+-----+
```

3. **DROP VIEW:**

   Purpose: To delete the table

   Syntax: Drop view View_name ;

   Example: MYSQL> drop view customers_view ;

   Sample output: View dropped.

SEQUENCES

1. **Creating Sequences**

   Syntax:

   ```
   CREATE SEQUENCE sequence_name START WITH initial_value
   INCREMENT BY increment_value MINVALUE minimum value
   MAXVALUE maximum value CYCLE|NOCYCLE ;
   ```

   Where,

   sequence_name: Name of the sequence.

   initial_value : Starting value from where the sequence starts.

   Initial_value >= minimum value and <= maximum value.

increment_value: Value by which sequence will increment itself.

minimum_value: Minimum value of the sequence.

maximum_value: Maximum value of the sequence.

Cycle **:** When sequence reaches its set_limit it starts from beginning.

Nocycle **:** An exception will be thrown if sequence exceeds its max_value.

Example:

```
CREATE SEQUENCE sequence_1

start with 1

increment by 1

minvalue 0

maxvalue 100

cycle;
```

Example to use sequence :

```
CREATE TABLE students( ID number(10),NAME char(20));

INSERT into students VALUES(sequence_1.nextval,'Ramesh');

INSERT into students VALUES(sequence_1.nextval,'Suresh');
```

Output:

```
| ID    |          NAME       |

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

|    1   |        Ramesh       |

|    2   |        Suresh
```

SYNONYM

1. **Creating Sequences**

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema .] synonym_name
      FOR [schema .] object_name [@ dblink];
```

Example:

CREATE OR REPLACE PUBLIC SYNONYM suppliers FOR app.suppliers;

2. **Drop synonym**

   DROP [PUBLIC] SYNONYM [schema .] synonym_name [force];

   Example:

   Drop public synonym suppliers;

IMPLICIT CURSORS

EX.NO: 4

Write a PL/ SQL code for calculating total mark, percentage, grade for all the students in a student management system using implicit cursors.

SQL> create table student(id number, name varchar2(10), dept varchar2(10), percent number,m1 number,m2 number, m3 number, tot number, g varchar2(1));

Table created.

*SQL> select * from student;*

```
ID    NAME          DEP          PERCENT M1   M2     M3     TOT  G

1     Anu           it      0             90   89     80     0

2     Beena         cse     0             98   91     95     0

3     Bindhu        it      0             87   67     86     0

4     Varun         it      0             67   46     50     0

5     Rahul         cse     0             81   82     83     0
```

SQL> declare

```
2     cursor c is select * from student;
3     ctot number;
4     cgra varchar2(1);
5     cper number;
6     begin
7     for I in c
8     loop
9     ctot= i.m1+i.m2+i.m3;
10    cper :=ctot/3;
11    update student set tot = ctot where id =i.id;
12    update student set percent = cper where id =i.id;
13    if(cper between 91 and 100)then
14    cgra:= ‘S’
15    elsif(cper between 81 and 90)then
16    cgra:= ‘A’
17    elsif(cper between 71 and 80)then
```

```
18      cgra:= ‘B’
19      elsif(cper between 61 and 70)then
20      cgra:= ‘C’
21      elsif(cper between 56 and 60)then
22      cgra:= ‘D’
23      elsif(cper between 50 and 55)then
24      cgra:= ‘E’
25      else
26      cgra:= ‘F’
27      end if;
28      update student set g = cgra where id =i.id;
29      end loop;
30      end;
31      /

PL/ SQL procedure successfully completed.
```

*SQL> select * from student;*

```
ID    NAME      DEP   PERCENT        M1    M2    M3    TOT   G

1     Anu       it    86.3333333     90    89    80    259   A

2     Beena     cse   94.6666667     98    91    95    284   S

3     Bindhu    it    80             87    67    86    240   B

4     Varun     it    54.3333333     67    46    50    163   E

5     Rahul     cse   82             81    82    83    246   A
```

Select * from customers;

```
+----+----------+-----+-----------+----------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |

+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

EXPLICIT CURSORS

EX.NO: 5

SYNTAX:

cursor cursor_name is select * from table name;

To open the cursor:

open cursor_name;

To close the cursor:

close cursor_name;

Exercise:

Write PL/ SQL code for calculating hra , da, netsalary for all the employees in the Payroll Processing using Explicit cursor(uses employee table).

```
SQL> select * from employee;

EMPNO     NAME      HRADA          PF      NETSAL     BASICPAY

101       AAA       0         0      0          0        15000

102       BBB       0         0     0           0        18000

103       CCC       0         0     0           0        20000

104       DDD       0         0     0           0        10000

105       EEE       0         0     0           0        25000
```

```
SQL> declare

2     cursor c is select * from employee;
3     i employee% rowtype;
4     hrasal number;
```

```
dasal number;
pfsal number;
netsalary number;
begin
open c;
loop;
fetch c into i;
if c% notfound ten exit;
endif;
hrasal:=i.basicpay*0.1;
dasal:=i.basicpay*0.08;
pfsal:=i.basicpay*0.12;
netsalaray:= i.basicpay + hrasal + dasal + pfsal;
update employee set hra = hrasal, da= dasal, pf= pfsal, netsal= netsalaray where empno=i.empno;
end loop;
close c;
end;
/
PL/ SQL procedure successfully completed.
```

```
SQL> select * from employee;

    EMPNO      NAME        HRA    DA      PF      NETSAL     BASICPAY

    101        AAA         1500   1200    1800    15900      15000

    102        BBB         1800   1440    2160    19080      18000

    103        CCC         2000   1600    2400    21200      20000

    104        DDD         1000    800    1200    10600      10000

    105        EEE         2500   2000    3000    26500      25000
```

PROCEDURES AND FUNCTIONS

EX.NO: 6

PROGRAM:

```
declare
e name
varchar2(15);basic
number;
d a number; h r a number;
pfnumber;
net salary number;
yearsalary number;
begin
e name:='&e name';
basic:=&basic; da:=basic *
(30/100); hra:=basic *
(10/100); if (basic < 8000)
then pf:=basic * (8/100);
elsif (basic >= 8000 and basic <= 16000) then
pf:=basic * (10/100);

end if;
netsalary:=basic + da + hra - pf; yearsalary := netsalary*12;

dbms_output.put_line('Employee name : ' || ename); dbms_output.put_line('Providend Fund : ' || pf);
dbms_output.put_line('Net salary : ' || netsalary); dbms_output.put_line('Year salary : '|| yearsalary); end;
/
```

1) Create a function to find the factorial of a given number and hence findNCR.

```
SQL> create or replace function fact(n number) return number isa number:=n; f
number:=1; i number;
begin
for i in 1..n
loopf:=f*a;
a:=a-1;
end loop;
return f;
end;
/
```

```
SQL> create or replace function ncr(n number ,r number) return number isn1
number:=fact(n);
r1 number:=fact(r);
nr1number:=fact(n-
r); resultnumber;
begin result:=(n1)/(r1*nr1); return result; end;
/
```

1) ***Print Fibonacci series using localfunctions.***

```
sql>create or replace function fib (n positive) return integer is begin if (n
= 1) or (n = 2) then -- terminating condition return
1;else
return fib(n - 1) + fib(n - 2); -- recursive call end if;
endfib;
        /
```

-- Test Fibonacci Series:

```
SQL>SELECT fib(1), fib(2), fib(3), fib(4), fib(5) FROM dual;
```

2) ***write a pl/sql function accept date of birth as "dd-mm-yyyy" and sum all digits tillyou get single digit number to show as he luckynumber.***

```
SQL> set serverout on SQL> declare
l_input varchar2(20) := '31/01/1978';
l_output int;
begin loop
dbms_output.put_line('
');
dbms_output.put_line('l_input='||l_input); l_output := 0;
for iin 1 .. length(l_input) loop
if substr(l_input,i,1) between '0' and '9' then
l_output := l_output + to_number(substr(l_input,i,1));
endif;
end loop; dbms_output.put_line('l_output='||l_output); exit when
l_output< 10; l_input := to_char(l_output);
 end loop;
dbms_output.put_line('');
dbms_output.put_line('Lucky='||l_output); end;
/
```

```
___________-
l_input=31/01/1978l_output=30
```

```
___________-
l_input=30l_output=3
___________-
Lucky=3

PL/SQL procedure successfully completed
```

RESULT: Thus the program to practice Pl/SQL commands in database has been donesuccessfully.

EX.NO:7 **TRIGGERS**

1. Create a row level trigger for the customers table that would fire forINSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

CUSTOMERS table:

ID	NAME	AGE	ADDRESS	SALARY
1	Alive	24	Khammam	2000
2	Bob	27	Kadappa	3000
3	Catri	25	Guntur	4000
4	Dena	28	Hyderabad	5000
5	Eeshwar	27	Kurnool	6000
6	Farooq	28	Nellur	7000

```
CREATE OR REPLACE TRIGGER display_salary_changes BEFORE DELETE OR INSERT
ORUPDATE ON customers FOR EACH ROW

WHEN (NEW.ID > 0) DECLARE
sal_diff number; BEGIN
sal_diff := :NEW .salary - :OLD .salary; dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary); dbms_output.put_line('Salary difference: '
|| sal_diff);
    END;
    /
```

Trigger created.

Here following two points are important and should be noted carefully:

OLD and NEW references are not available for table level triggers, rather you can use them for record leveltriggers.

If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.

Above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using DELETE operation on the table.

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table:

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7,
'Kriti', 22,'HP', 7500.00 );
```

When a record is created in CUSTOMERS table, above create trigger display_salary_changeswill be fired and it will display the following result:

```
Old salary:
New salary: 7500 Salary difference:
```

2) Convert employee name into uppercase whenever an employee record is inserted or updated. Trigger to fire before the insert orupdate.

```
SQL> create table Employee(
1   ID              VARCHAR2(4 BYTE) NOTNULL,
2   First_Name      VARCHAR2(10BYTE),
3   Last_Name       VARCHAR2(10BYTE),
4   Start_Date      DATE,
5   End_Date        DATE,
6   Salary          NUMBER(8,2),
7   City            VARCHAR2(10BYTE),
8   Description
    VARCHAR2(15
BYTE) 10)

11 /

Table created.
```

```
SQL> CREATE OR REPLACE TRIGGER employee_ insert_ update
BEFORE INSERT OR UPDATE ON employee FOR
EACH ROW 4DECLARE

5       dup_ flag INTEGER; 6BEGIN
--Force all employee names toupper case.

:NEW. first_ name :=UPPER(:NEW. first_name); 9END;

10 /
```

```
Trigger created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)
2          values('01','Jason',       'Martin',
to_date('19960725','YYYYMMDD'),
to_date('20060725','YYYYMMDD'), 1234.56, 'Toronto', 'Programmer')
3 /

1 row created



SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)
1 values('02','Alison', 'Mathews', to_date('19760321','YYYYMMDD'),
   to_date('19860221','YYYYMMDD'), 6661.78, 'Vancouver','Tester')
3 /

1 row created.



SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
2          values('03','James',      'Smith',
to_date('19781212','YYYYMMDD'),
to_date('19900315','YYYYMMDD'), 6544.78, 'Vancouver','Tester')
3 /

1 row created.



SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
2          values('04','Celia',       'Rice',
to_date('19821024','YYYYMMDD'),
to_date('19990421','YYYYMMDD'), 2344.78,'Vancouver','Manager')
3 /
1 row created.
```

```
SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City,
Description)
      2            values('05','Robert',
                   'Black',to_date('19840115','YYYYMMDD'),
     to_date('19980808','YYYYMMDD'),
                   2334.78,'Vancouver','Tester')
      3 /
     1 row created.


     SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
     City, Description)
      2            values('06','Linda', 'Green',
     to_date('19870730','YYYYMMDD'),
     to_date('19960104','YYYYMMDD'),
                   4322.78,'NewYork','Tester')
      3 /
     1 row created.


     SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
     City, Description)
      2            values('07','David', 'Larry',
     to_date('19901231','YYYYMMDD'),
     to_date('19980212','YYYYMMDD'),
                   7897.78,'NewYork','Manager')3 /
```

```
1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary,
City, Description)
  2         values('08','James', 'Cat',
            to_date('19960917','YYYYMMD
            D'),
to_date('20020415','YYYYMMDD'),
1232.78,'Vancouver','Tester')
3 /1 row created.

SQL> Select * from
Employee 2 /
    ID FIRST_NAME LAST_NAMESTART_DATEND_DATE   SALARYCITY
                                               DESCRIPTION
    --------------------------------------------------------------------------------------------------------------------------------------------------

___-
```

01	JASON	Martin	25-JUL-96	25-JUL-06 1234.56	Toronto Programm e r
02	ALISON	Mathews	21-MAR-76	21-FEB-86 6661.78	Vancouver Tester
03	JAMES	Smith	12-DEC-78	15-MAR-90 6544.78	Vancouver Tester
04	CELIA	Rice	24-OCT-82	21-APR-99 2344.78	Vancouver Manager
05	ROBER T	Black	15-JAN-84	08-AUG-98 2334.78	Vancouver Tester
06	LINDA	Green	30-JUL-87	04-JAN-96 4322.78	New YorkTester
07	DAVID	Larry	31-DEC-90	12-FEB-98 7897.78	New York Manager
08	JAMES	Cat	17-SEP-96	15-APR-02 1232.78	Vancouver Tester

```
8 rows selected.

SQL> drop table Employee 2 /
Table dropped.
```

3) Trigger before deleting a record from emp table. Trigger will insert the row to be deleted into another table and also record the user who has deleted therecord.

```
SQL>   CREATE OR REPLACE TRIGGERemployee_before_delete
  2    BEFOREDELETE
  3       ON employee
  4       FOR EACHROW
  5    DECLARE
  6       v_usernamevarchar2(10);
  7    BEGIN
 8      -- Find username of person performing the DELETE on thetable
 9       SELECT user INTOv_username
 10        FROMdual;
 11     -- Insert record into audit table
 12       INSERT INTO employee_audit (id, salary, delete_date,deleted_by)
 13                    VALUES (:old.id,:old.salary, sysdate, v_username);
 14    END;
 15    /

Trigger created.

SQL> delete from employee; 8
rows deleted.
SQL> select * from
employee_audit;

    ID          SALARY DELETE  _DADELETED_BY
     ---- ---------- --------- ---------------
01    234.56 09-SEP-
      06JAVA2S
02    6661.78 09-SEP-
       06JAVA2S
     03    6544.78 09-SEP-
           06JAVA2S
     04    2344.78 09-SEP-
           06JAVA2S
     05    2334.78 09-SEP-
           06JAVA2S
     06    4322.78 09-SEP-
           06JAVA2S
```

```
07     7897.78 09-SEP-
       06JAVA2S
08     1232.78 09-SEP-
       06JAVA2S

     8 rows selected.


SQL>drop table
employee_audit;Table
dropped
```

RESULT: Thus the program to practice triggers in Sql commands in database has been done successfully.

EXP NO: 08

INTRODUCTION TO IDE

AIM: Installation of MySQL

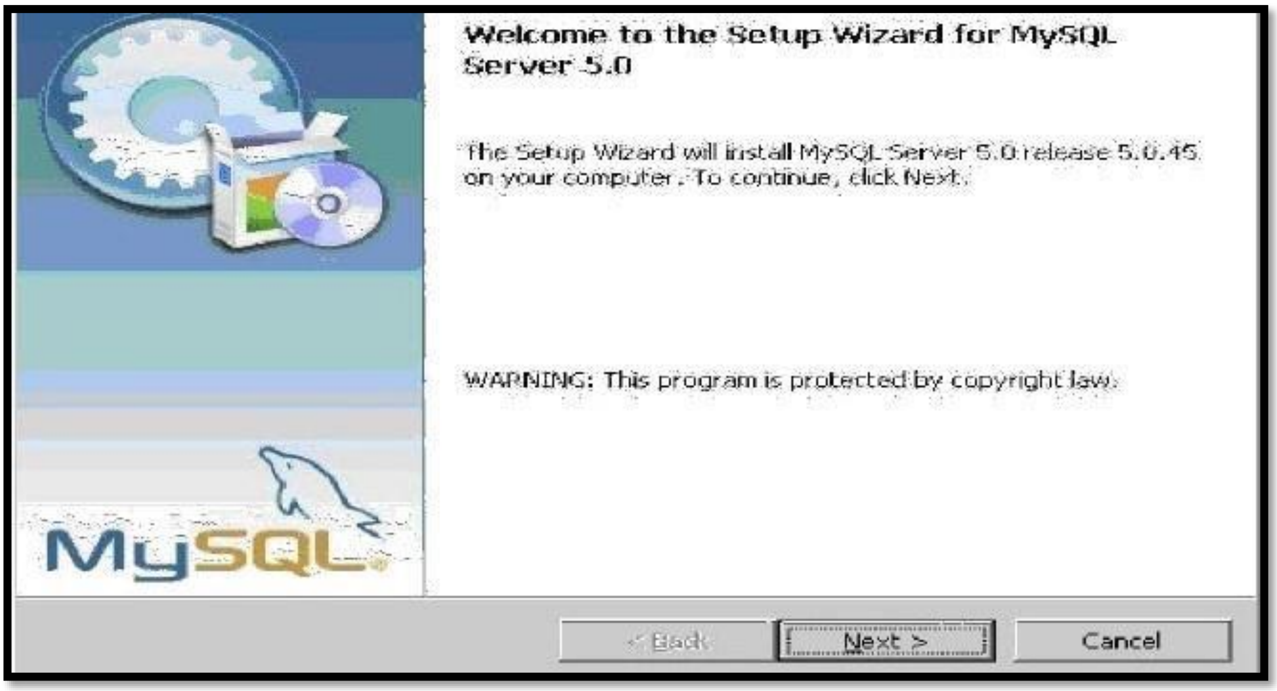
Steps for installing MySQL

Step1:-

Make sure you already downloaded the **MySQL essential 5.0.45 win32.msi file**. Double click on the .msi file.

Step2:-

This is MySQL Server 5.0 setup wizard. The setup wizard will install MySQL Server 5.0 release 5.0.45 on your computer. To continue, click **next.**

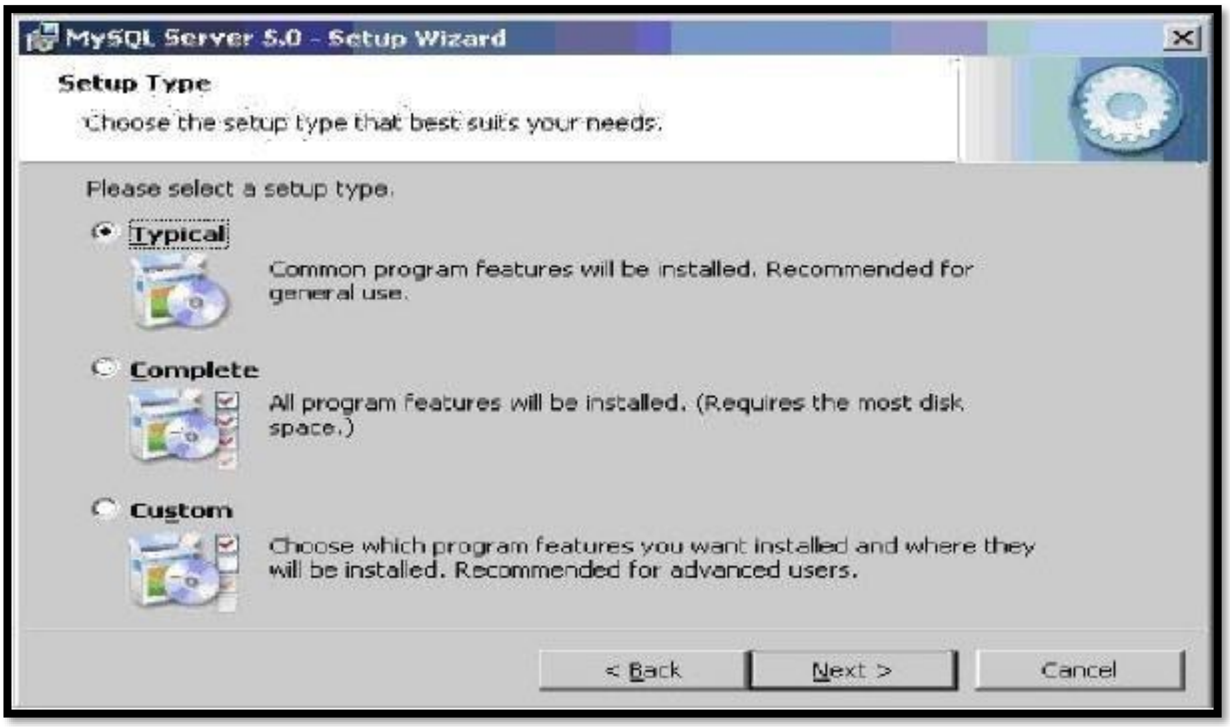

Welcome to the Setup Wizard for MySQL Server 5.0

The Setup Wizard will install MySQL Server 5.0 release 5.0.45 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law.

Step3:-

Choose the setup type that best suits your needs. For common program features select ***Typical*** and it’s recommended for general use. To continue, click **next**.


MySQL Server 5.0 - Setup Wizard

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

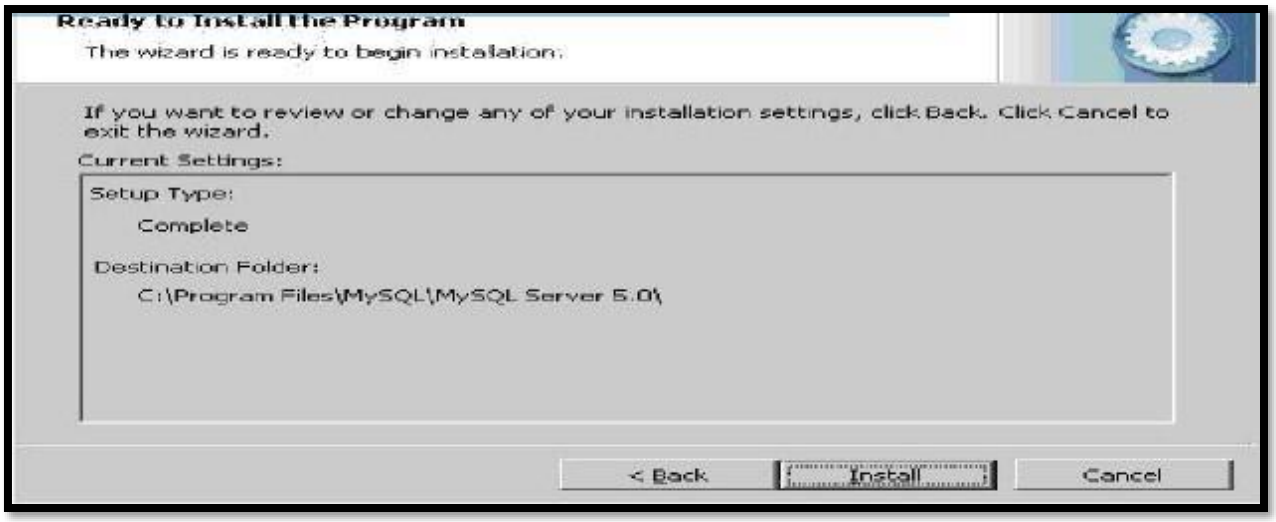
[✓ Typical]
Common program features will be installed. Recommended for general use.

Complete
All program features will be installed. (Requires the most disk space.)

Custom
Choose which program features you want installed and where they will be installed. Recommended for advanced users.

Step4:-

This wizard is ready to begin installation. Destination folder will be in **C:\Program Files\MySQL\MySQL Server 5.0**. To continue, click **next**.



Ready to Install the Program

The wizard is ready to begin installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

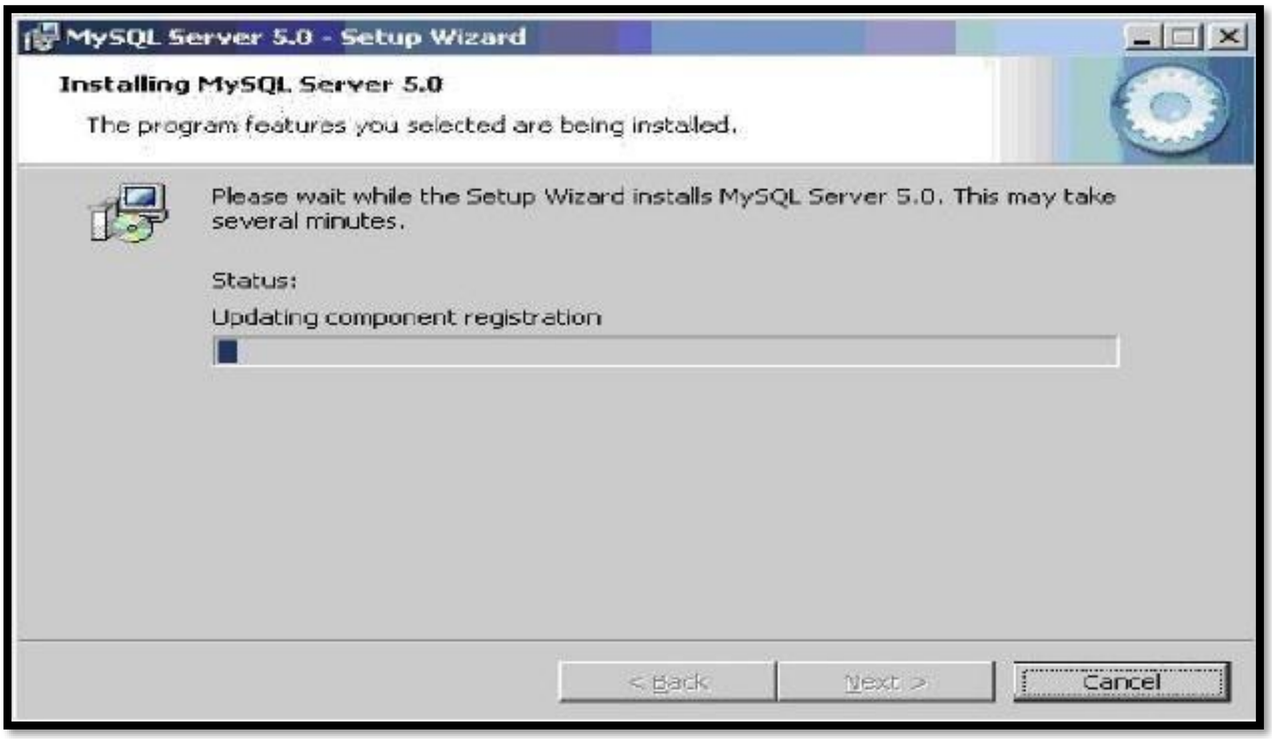
Current Settings:

Setup Type:
Complete

Destination Folder:
C:\Program Files\MySQL\MySQL Server 5.0\

Step5:-

The program features you selected are being installed. Please wait while the setup wizard installs MySQL 5.0.This may take several minutes.



MySQL Server 5.0 - Setup Wizard

Installing MySQL Server 5.0

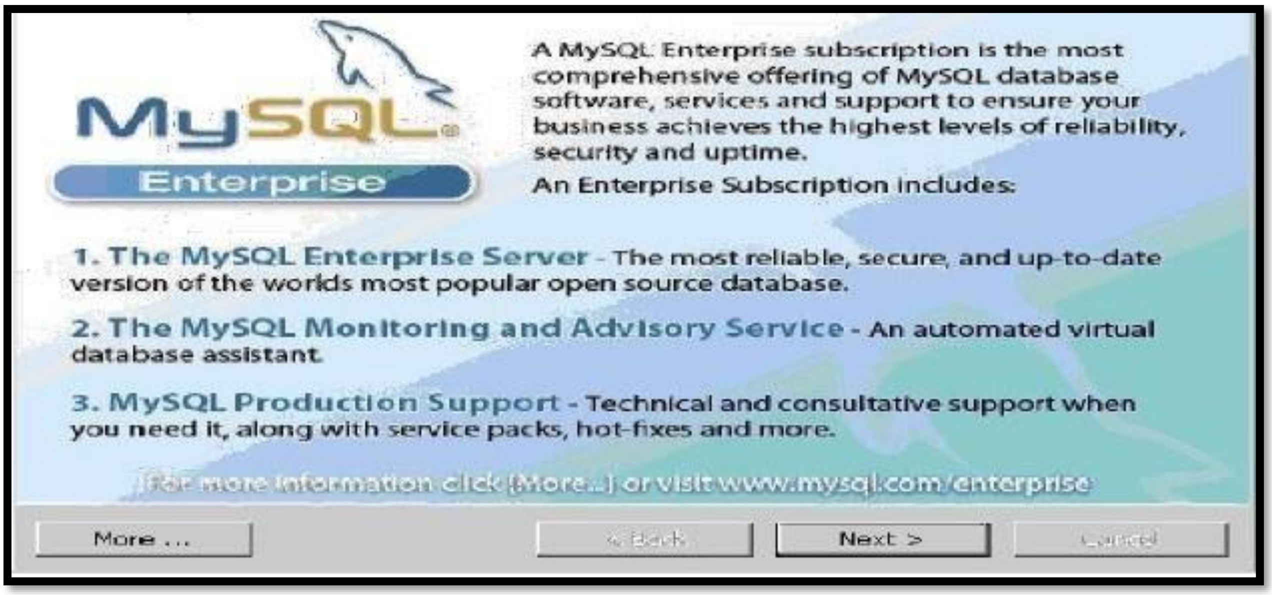
The program features you selected are being installed.

Please wait while the Setup Wizard installs MySQL Server 5.0. This may take several minutes.

Status:
Updating component registration

Step6:-

To continue, click **next**.


A MySQL Enterprise subscription is the most comprehensive offering of MySQL database software, services and support to ensure your business achieves the highest levels of reliability, security and uptime.

An Enterprise Subscription includes:

1. The MySQL Enterprise Server - The most reliable, secure, and up-to-date version of the worlds most popular open source database.

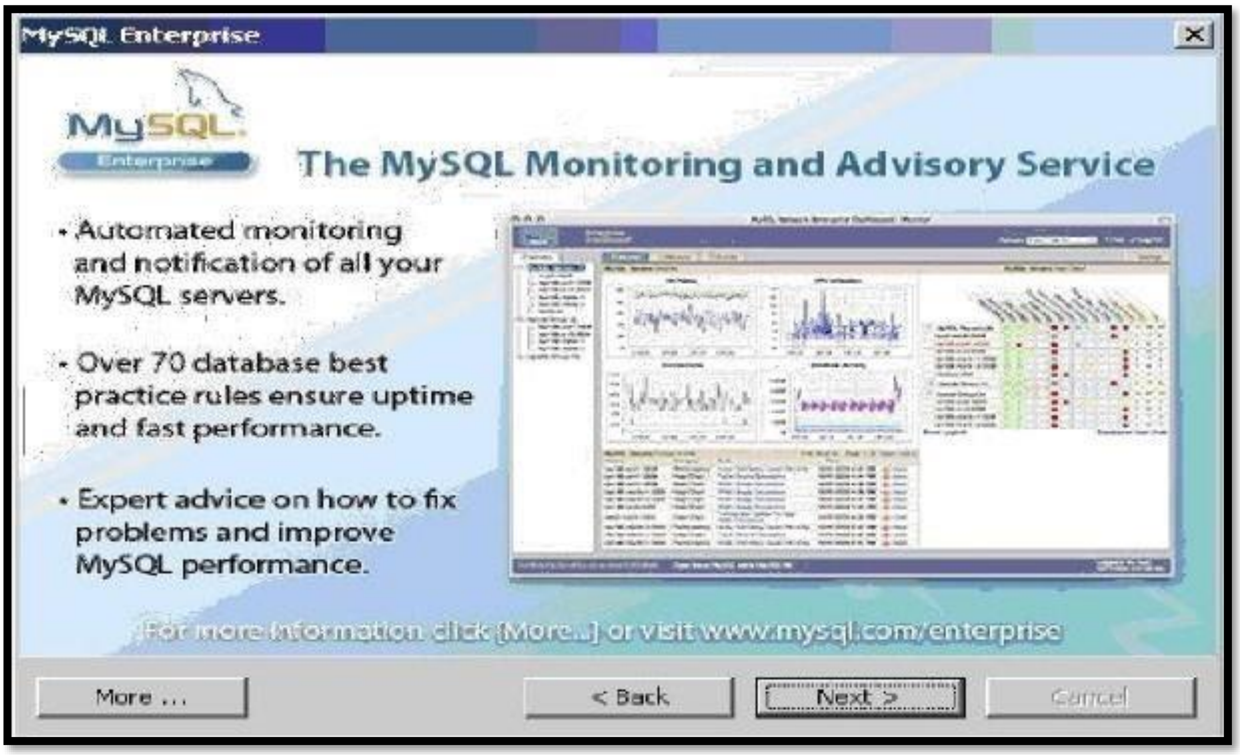
2. The MySQL Monitoring and Advisory Service - An automated virtual database assistant.

3. MySQL Production Support - Technical and consultative support when you need it, along with service packs, hot-fixes and more.

For more information click [More...] or visit www.mysql.com/enterprise

Step7:-

To continue, click **next**.


MySQL Enterprise

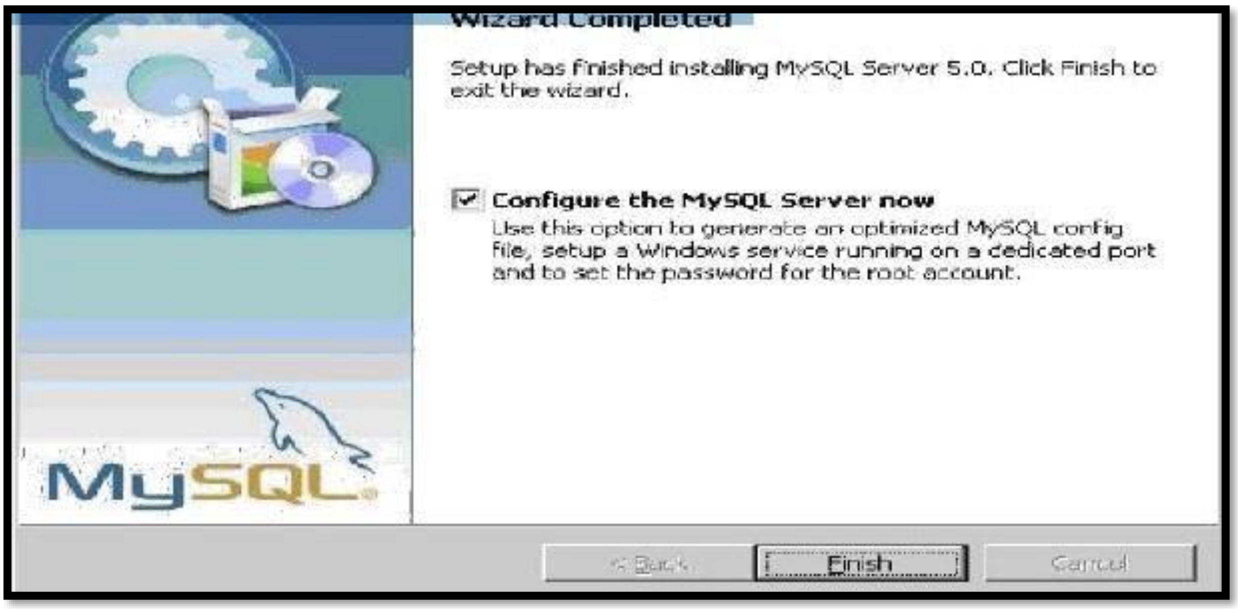
The MySQL Monitoring and Advisory Service

- Automated monitoring and notification of all your MySQL servers.
- Over 70 database best practice rules ensure uptime and fast performance.
- Expert advice on how to fix problems and improve MySQL performance.

For more information click [More...] or visit www.mysql.com/enterprise

Step8:-

Wizard Completed. Setup has finished installing MySQL 5.0. **Check** the configure the MySQL server now to continue. Click **Finish** to exit the wizard

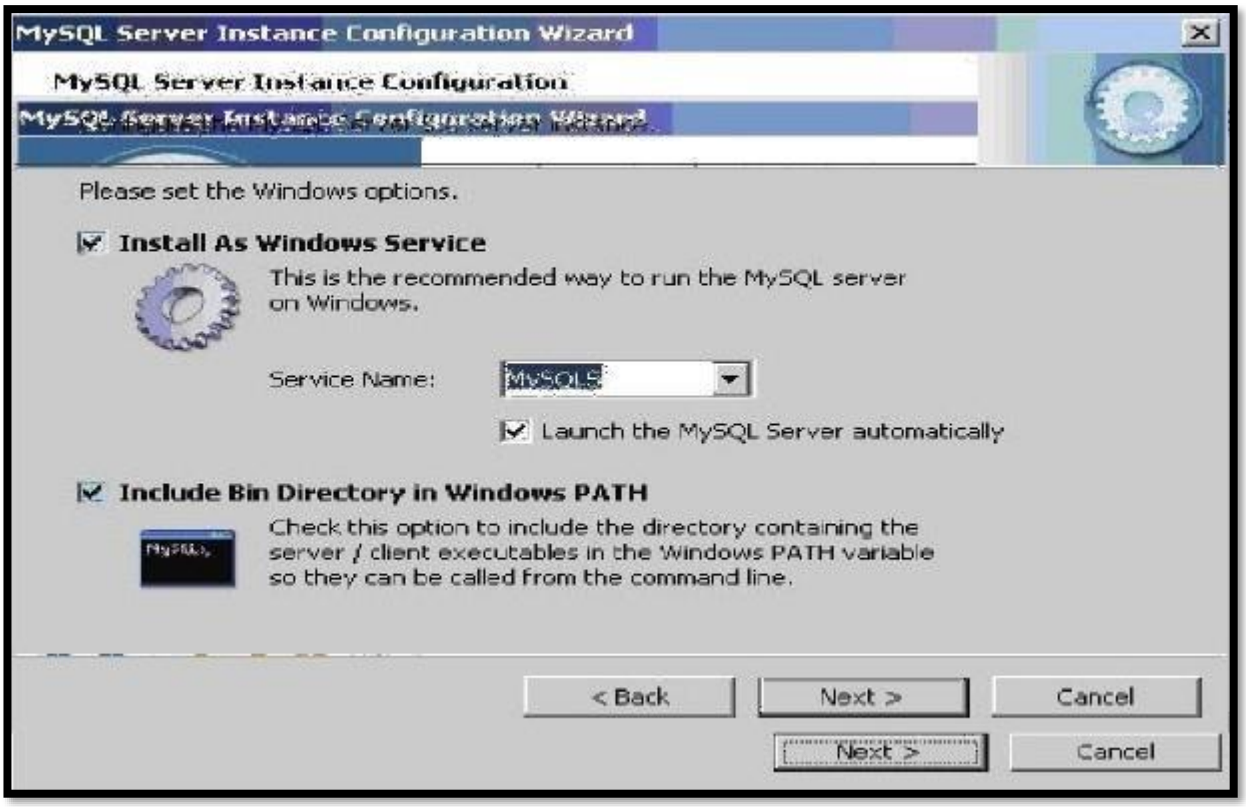

Wizard Completed

Setup has finished installing MySQL Server 5.0. Click Finish to exit the wizard.

☑ **Configure the MySQL Server now**
Use this option to generate an optimized MySQL config file, setup a Windows service running on a dedicated port and to set the password for the root account.

Step9:-

The configuration wizard will allow you to configure the MySQL Server 5.0 server instance. To continue, click **next**.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Please set the Windows options.

☑ Install As Windows Service

This is the recommended way to run the MySQL server on Windows.

Service Name: MySQL5

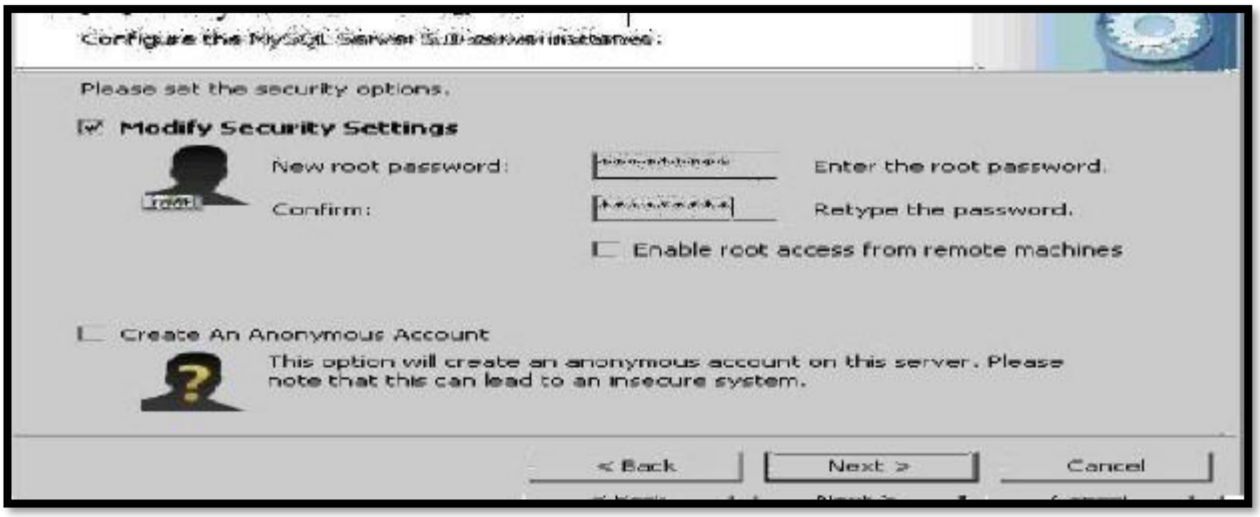
☑ Launch the MySQL Server automatically

☑ Include Bin Directory in Windows PATH

Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.

Step10:-

Select a **standard configuration** and this will use a general purpose configuration for the server that can be tuned manually. To continue, click **next**.



Configure the MySQL Server 5.0 server instance.

Please set the security options.

☑ Modify Security Settings

New root password: ********* Enter the root password.

Confirm: ********* Retype the password.

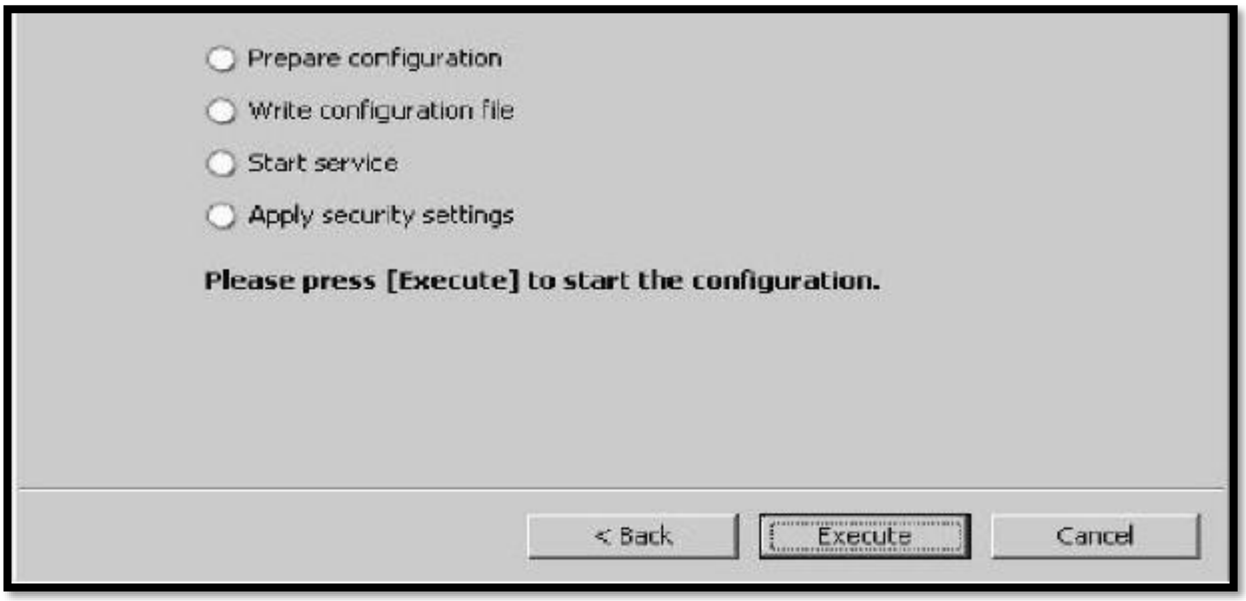
☐ Enable root access from remote machines

☐ Create An Anonymous Account

This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

Step11:-

Check on the **install as windows service** and **include bin directory in windows path**. To continue, click **next**


Prepare configuration

Write configuration file

Start service

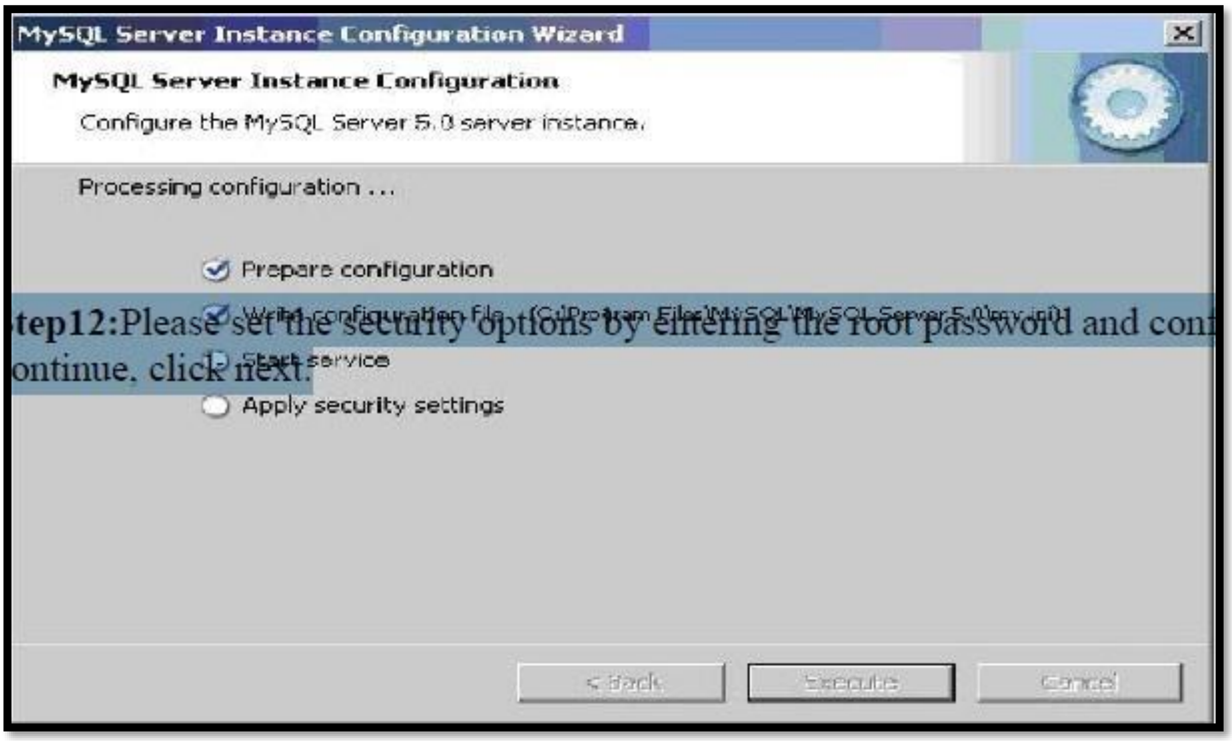
Apply security settings

Please press [Execute] to start the configuration.

Step12:-

Please set the security options by entering the root password and confirm retype the password. continue, click next.

tep12:Please set the security options by entering the root password and con
ontinue, click next.



MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Processing configuration ...

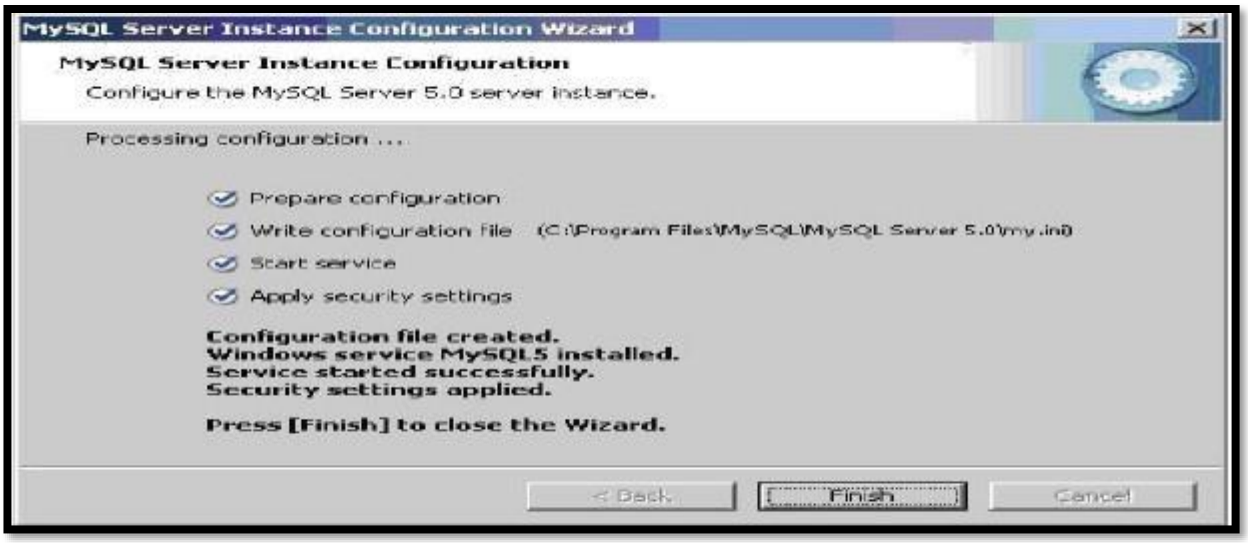
- ✓ Prepare configuration
- ✓ Write configuration file (C:\Program Files\MySQL\MySQL Server 5.0\my.ini)
- Start service
- Apply security settings

Step13:-

Ready to execute? Clicks **execute** to continue.

Step14:-

Processing configuration in progress.



MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Processing configuration ...

- ✓ Prepare configuration
- ✓ Write configuration file (C:\Program Files\MySQL\MySQL Server 5.0\my.ini)
- ✓ Start service
- ✓ Apply security settings

Configuration file created.
Windows service MySQL5 installed.
Service started successfully.
Security settings applied.

Press [Finish] to close the Wizard.

Step15:-

Configuration file created. Windows service MySQL5 installed. Press **finish** to close the wizard.

Result:-

Thus the installation of MySql is done successfully.

EX.NO:9 ***DATABASE CONNECTIVITY WITH FRONT END TOOLS***

Coding:-

```
Private Sub ab_Click()
RichTextBox1.SelFontName = "Arial Black"
End Sub
Private Sub bold_Click()
RichTextBox1.SelBold = True
 End Sub
Private Sub cb_Click()
 RichTextBox1.SelColor = vbblue
End Sub
Private Sub cl_Click()
RichTextBox1.SelColor = vbred
End Sub
Private Sub copy_Click()
'Clipboard.SetText "richtextbox1.seltext", 1 'MsgBox Clipboard.GetText
Clipboard.SetText RichTextBox1.SelText, 1 RichTextBox1.SelText =
Clipboard.GetText MsgBox Clipboard.GetText End
Sub

Private Sub eighteen_Click()
RichTextBox1.SelFontSize = 18
End Sub
Private Sub exit_Click()
End
End Sub

Private Sub fcg_Click()
RichTextBox1.SelColor = vbgreen
End Sub
Private Sub fourteen_Click()
RichTextBox1.SelFontSize = 14
```

```
End Sub
Private Sub helpp_Click()
ans = MsgBox("visual basic sample notepad !", vbYes + vbinforamtion, "Help")
If ans = vbYes Then Unload Me
End If
End Sub


Private Sub italic_Click()
RichTextBox1.SelItalic = True
End Sub


Private Sub MC_Click() RichTextBox1.SelFontName = "Monotype Corsiva" End Sub
Private Sub new_Click() RichTextBox1 = "" End Sub

Private Sub open_Click()
RichTextBox1.LoadFile ("C:\Notepad\Document.rtf")
End Sub
Private Sub paste_Click()
RichTextBox1.SelText = Clipboard.GetText
End Sub
Private Sub save_Click()
RichTextBox1.SaveFile ("C:\Notepad\Document.rtf")
End Sub
Private Sub sixteen_Click()
RichTextBox1.SelFontSize = 16 End Sub
Private Sub Th_Click()
RichTextBox1.SelFontName = "Tahoma"
End Sub
Private Sub tn_Click()
RichTextBox1.SelFontName = "Times New Roman"
End Sub
Private Sub twele_Click()
RichTextBox1.SelFontSize = 12  End
Sub
Private Sub underline_Click()
RichTextBox1.SelUnderline = True
End Sub
Private Sub vbblue_Click()
RichTextBox1.SelColor = vbblue
End Sub
Private Sub vbgreen_Click()
RichTextBox1.SelColor = vbgreen
End Sub
```

```
Private Sub vbred_Click()
RichTextBox1.SelColor = vbred
End Sub
```

EX.NO:10 ***INVENTORY CONTROL SYSTEM***

TABLE NAME:SUPPLIER

```
SQL> create table supplier(supno number(10),supname
varchar2(20),supdate date, price number(20),quantity number(10),ITEM_NAME VARCHAR2(20));

Table created.
SQL> insert into supplier values(1,'pit','12-jan-2014',8000,2,'MONITOR');  1 row created.
SQL> insert into supplier values(2,'PEC','6-MAR2014',4000,1,'KEYBOARD');  1
row created.
```

TABLE NAME:ITEM

```
SQL> CREATE TABLE ITEM(ITEMNO NUMBER(10),ITEM_NAME
VARCHAR2(10),PRICE NUMBER(10),QUAT_AVAILABLE
NUMBER(10));
Table created.
SQL> INSERT INTO ITEM VALUES(101,'MONITOR',80000,3);
1 row created.
SQL> insert into ITEM VALUES(102,'MOUSE',70000,10);
1 row created.
SQL>
COMMIT;
Commit complete.
```

CODING FORM1:

```
Private Sub Item_Click()
Form3.Show
End Sub


Private Sub
Supplier_Click()
Form2.Show
End Sub
```

FORM2:

```
Private Sub Command1_Click()
Adodc1.Recordset.AddNew
Adodc1.Recordset.Fields("supno") = Text1.Text
Adodc1.Recordset.Fields("supname") = Text2.Text
```

```
Adodc1.Recordset.Fields("supdate") = Text3.Text
Adodc1.Recordset.Fields("price") = Text4.Text
Adodc1.Recordset.Fields("quantity") = Text5.Text
Adodc1.Recordset.Fields("item_name") =
Text6Text Adodc1.Recordset.Update
MsgBox "Data Added"  End Sub

Private Sub
Command10_Click()
Form3.Show
End Sub

Private Sub
Command3_Click()
Adodc1.Recordset.Delete
Adodc1.Recordset.Update
End Sub

Private Sub
Command4_Click() Unload
Me
End Sub

Private Sub Command5_Click()
If Adodc1.Recordset.EOF =
False Then
Adodc1.Recordset.MoveNext
Else
MsgBox "END OF FILE!", vbOKOnly, "Warning"
End If
End Sub

Private Sub
Command6_Click()
Adodc1.Recordset.MoveFirst
End Sub

Private Sub
Command7_Click()
Adodc1.Recordset.MoveLast  End Sub
```

```
Private Sub Command8_Click()
If Adodc1.Recordset.BOF = False
Then Adodc1.Recordset.MovePrevious
Else
MsgBox "BEGIN OF FILE!!", vbOKOnly, "Warning"
End If
End Sub

Private Sub
Command9_Click() Form1.Show

End Sub
```

FORM3:

```
Private Sub Command1_Click()

Adodc1.Recordset.AddNew
Adodc1.Recordset.Fields("itemno") = Text1.Text
Adodc1.Recordset.Fields("item_name") = Text2.Text
Adodc1.Recordset.Fields("price") = Text4.Text
Adodc1.Recordset.Fields("quat_available") =
Text5.Text Adodc1.Recordset.Update
MsgBox "Data Added"
End Sub
Private Sub Command10_Click()
Form2.Show End Sub

Private Sub
Command3_Click()
Adodc1.Recordset.Delete
Adodc1.Recordset.Update
End Sub

Private Sub
Command4_Click() Unload
Me
End Sub


Private Sub Command5_Click()
If Adodc1.Recordset.EOF =
False Then
Adodc1.Recordset.MoveNext
Else
MsgBox "END OF FILE!", vbOKOnly, "Warning"
End If
End Sub
```

```
Private Sub
Command6_Click()
Adodc1.Recordset.MoveFirst
End Sub

Private Sub
Command7_Click()
Adodc1.Recordset.MoveLast

End Sub



Private Sub Command8_Click()
If Adodc1.Recordset.BOF =
False Then
Adodc1.Recordset.MovePrevio us Else
MsgBox "BEGIN OF FILE!!", vbOKOnly, "Warning"
End If
End Sub

Private Sub Command9_Click()
Form1.Show
End Sub
```