Laboratory Assignment File

For

Database Management System Lab

(CSPC-222)

B.Tech (Computer Science and Engineering)

IV Semester Sec-B

Prepared during (Jan-June 2024) By

Name: Sunandhit Gupta

Roll No: 22103148

Department of Computer Science and Engineering DR. B.R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY JALANDHAR, PUNJAB-144027

INDEX

S. No.	Name	Page no.
1	To create a library management system	3-10
2	To implement DDL and DML queries	11-18
3	To implement data integrity constraints	19-25
4	To implement aggregate and scalar function	26-37
5	To implement ER diagram	38-47
6	To implement JOIN	48-60
7	To implement VIEW, INDEX, SEQUENCE	61-71
8	To implement join (exam question)	71-74
9	To implement user role and privilege	75-78
10	To implement pl/sql	79-83
11	To implement trigger	84-89

Lab Assignment 1

Program: Using C++, WAP of a Library Management Systems with the concepts of File Handling. It must Allow the Library to Manage the records of Issues and Deposits of Books and Add, Retrieve, and update Details of any Student or Book.

Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

class Student {
private:
public:
   static int nextID;
   int studentID;
   string studentName;
   string className;
   int booksIssued;

   Student() : studentID(++nextID) {}

   void addStudent() {
   cout << "Enter student name: ";
   cin.ignore();
   getline(cin, studentName);
   cout << "Enter number of books issued: ";
   cin >> booksIssued;
   cout << "Enter class name: ";
   cin.ignore();
   getline(cin, className);
```

```
    // Increment nextID before assigning it to studentID
    studentID = ++nextID;

    ofstream outFile("students_data.txt", ios::app);
    outFile << "\n" << studentID;
    outFile << "\n" << studentName;
    outFile << "\n" << className;
    outFile << "\n" << booksIssued;
    outFile.close();

    // Update ID in studentID.txt
    ofstream idFile("studentID.txt", ios::app);
    idFile << "\n" << nextID;
    idFile.close();

    cout << "Record Added Successfully" << endl;
}

    void printStudents() {
        ifstream inFile("students_data.txt");
        Student student;

        while (inFile >> student.studentID >> ws && getline(inFile, student.studentName) &&
               getline(inFile, student.className) && inFile >> student.booksIssued) {
            cout << "\nStudent ID: " << student.studentID;
            cout << "\nName: " << student.studentName;
            cout << "\nClass name: " << student.className;
            cout << "\nIssued Book Number: " << student.booksIssued << endl
                 << endl
                 << endl;
        }
        inFile.close();
```

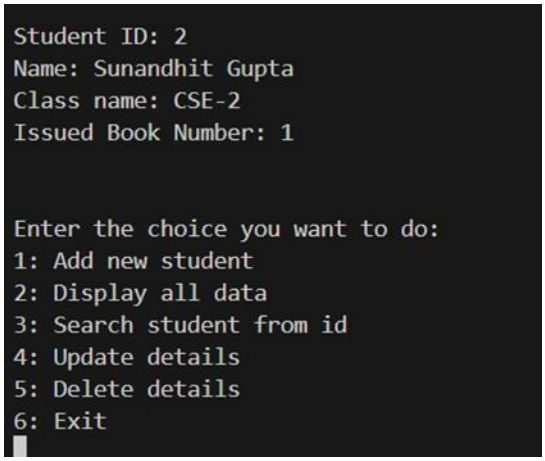
```
}

void updateStudent(int studentID) {
    deleteStudent(studentID);

    cout << "Enter student name: ";
    cin.ignore();
    getline(cin, studentName);
    cout << "Enter number of books issued: ";
    cin >> booksIssued;
    cout << "Enter class name: ";
    cin.ignore();
    getline(cin, className);

    ofstream outFile("students_data.txt", ios::app);
    outFile << "\n" << studentID;
    outFile << "\n" << studentName;
    outFile << "\n" << className;
    outFile << "\n" << booksIssued;
    outFile.close();

    cout << "Record Updated Successfully" << endl;
}

void deleteStudent(int studentID) {
    ifstream inFile("students_data.txt");
    ofstream outFile("temp.txt");

    Student student;
    while (inFile >> student.studentID >> ws && getline(inFile, student.studentName) &&
           getline(inFile, student.className) && inFile >> student.booksIssued) {
        if (studentID != student.studentID) {
            outFile << "\n" << student.studentID;
```

```
            outFile << "\n" << student.studentName;
            outFile << "\n" << student.className;
            outFile << "\n" << student.booksIssued;
        }
    }
    inFile.close();
    outFile.close();
    remove("students_data.txt");
    rename("temp.txt", "students_data.txt");

    cout << "Record Deleted Successfully" << endl;
}

void searchStudent(int studentID) {
    ifstream inFile("students_data.txt");
    Student student;

    while (inFile >> student.studentID >> ws && getline(inFile, student.studentName) &&
           getline(inFile, student.className) && inFile >> student.booksIssued) {
        if (studentID == student.studentID) {
            cout << "\nStudent ID: " << student.studentID;
            cout << "\nName: " << student.studentName;
            cout << "\nClass name: " << student.className;
            cout << "\nIssued Book Number: " << student.booksIssued << endl
                 << endl
                 << endl;
            break;
        }
    }
    inFile.close();
}
};
```

```
// Initialize static member
int Student::nextID = 0;

int main() {
    int choice;
    Student student;
    int studentID;

    ifstream idFile("studentID.txt");

    if (!idFile) {
        cout << "File Not Found" << endl;
    } else {
        while (idFile >> Student::nextID) {
        }
    }
    idFile.close();

    while (true) {
        cout << "Enter the choice you want to do:" << endl;
        cout << "1: Add new student" << endl;
        cout << "2: Display all data" << endl;
        cout << "3: Search student from id" << endl;
        cout << "4: Update details" << endl;
        cout << "5: Delete details" << endl;
        cout << "6: Exit" << endl;
        cin >> choice;
        switch (choice) {
            case 1:
                student.addStudent();
                break;
            case 2:
                student.printStudents();
```

```
            break;
        case 3:
            cout << "Enter student ID: ";
            cin >> studentID;
            student.searchStudent(studentID);
            break;
        case 4:
            cout << "Enter student ID: ";
            cin >> studentID;
            student.updateStudent(studentID);
            break;
        case 5:
            cout << "Enter student ID: ";
            cin >> studentID;
            student.deleteStudent(studentID);
            break;
        case 6:
            return 0;
        default:
            cout << "Invalid choice. Please try again." << endl;
      }
   }
   return 0;
}
```

Output:


Student ID: 2
Name: Sunandhit Gupta
Class name: CSE-2
Issued Book Number: 1
Enter the choice you want to do:
1: Add new student
2: Display all data
3: Search student from id
4: Update details
5: Delete details
6: Exit

Lab-Assignment 2

1. Create a database named as 'Inventory' and Create the tables described below:
2. Insert the following data into their respective tables:
3. a. Find out all the tables present in the current database and other databases.

   b. Find out the names of all the clients.

   c. Retrieve the entire contents of the CLIENT_MASTER table.

   d. Retrieve the list of names, city, and state of all the clients.

   e. List the various products available from the PRODUCT_MASTER table.

   f. List all the clients who are located in Mumbai.

   g. Find the names of the salesman who have a salary equal to Rs. 3000.

4. Exercise on updating records in a table

a. Change the city of ClientNo 'C00005' to Bangalore.

b. Change the BalDue of ClientNo 'C00001' to Rs. 1000.

c. Change the CostPrice of 'Trousers' to Rs. 950.00.

d. Change the city of the salesman to Pune.

5. Exercise on deleting records in a table

a. Delete all salesmen from the SALESMAN_MASTER whose salaries are equal to Rs. 3500.

b. Delete all products from PRODUCT_MASTER where the quantity on hand is equal to 100.

c. Delete from CLIENT_MASTER where the column state holds the value 'Tamil Nadu'.

6. Exercise on altering the table structure

a. Add a column called 'Telephone' of data type 'Number' and size ='10' to the CLIENT_MASTER

table.

b. Change the size of SellPrice column in PRODUCT_MASTER TO 10,2.

7. Exercise on deleting the table structure along with the data

a. Destroy the table CLIENT_MASTER along with its data.

8. Exercise on renaming the table

a. Change the name of the SALESMAN_MASTER table to SMAN_MAST.

SOURCE CODE:

```
create database inventory;
use inventory;
create table client_master(
clientno varchar(6),
name varchar(20),
address1 varchar(30),
address2 varchar(30),
city varchar(15),
pincode int(8),
state varchar(15),
```

```
baldue int(10));
create table product_master(
productno varchar(6),
description varchar(15),
profitpercent int(4),
unitmeasure varchar(10),
qtyonhand int(8),
reorderlvl int(8),
sellprice int(8),
costprice int(8));
create table sales_master(
salesmanno varchar(6),
salesmanname varchar(20),
address1 varchar(30),
address2 varchar(30),
city varchar(20),
pincode int(8),
state varchar(20),
salamt int(8),
tgttoget int(6),
ytdsales int(6),
remarks varchar(60));
insert into client_master (clientno,name,city,pincode,state,baldue)
values ('c00001','ivan bayross','mumbai','400054','maharastra',15000),
('c00002','mamta muzumdar','madras','780001','tamil naidu',0),
('c00003','chhaya bankar','mumbai','400057','maharastra',5000),
('c00004','ashvini joshi','banglore','560001','karnataka',0),
('c00005','hancel colaco','mumbai','40060','maharastra',2000),
('c00006','deepak sharma','mangalore','560050','karnataka',0);
```

```
insert into
product_master(productno,description,profitpercent,unitmeasure,qtyonhand,reorderlvl,sell
price,costprice)

values('p00001','t-shirts',5,'piece',200,50,350,250),

('p0345','shirts',6,'piece',150,50,500,350),

('p06734','cotton jeans',5,'piece',100,20,600,450),

('p07865','jeans',5,'piece',100,20,750,500),

('p07868','trousers',2,'piece',150,50,850,550),

('p07885','pull overs',2.5,'piece',80,30,700,450),

('p07965','denim shirts',4,'piece',100,40,350,250),

('p07975','lycra tops',5,'piece',70,30,300,175),

('p08865','skirts',5,'piece',75,30,450,300);

insert into
sales_master(salesmanno,salesmanname,address1,address2,city,pincode,state,salamt,tgttog
et,ytdsales,remarks)

values('s00001','aman','a/14','worli','mumbai',400002,'maharastra',3000,100,50,'good'),

('s00002','omkar','65','nariman','mumbai',400001,'maharastra',3000,200,100,'good'),

('s00003','raj','p-7','bandra','mumbai',400032,'maharastra',3000,200,100,'good'),

('s00004','ashish','a/5','juhu','mumbai',400044,'maharastra',3500,200,150,'good');

show tables;

select name from client_master;

select * from client_master;

select name,city,state from client_master;

select description from product_master;

select name from client_master where city='mumbai';

select salesmanname from sales_master where salamt=3000;

/*exercise 4 on update*/

SET SQL_SAFE_UPDATES = 0;

update client_master

set city='bangalore'

WHERE clientno='c00005';
```

```
update client_master
set baldue='100'
where clientno='c00001';
update product_master
set costprice=950
where description='toursers';
update sales_master
set city='pune';
delete from sales_master
where salamt=3500;
delete from product_master
where qtyonhand=100;
delete from client_master
where state='tamil naidu';
/*exercise on altering the table*/
ALTER TABLE client_master
ADD Telephone INT(10);
ALTER TABLE product_master
MODIFY sellprice INT(10);
DROP TABLE client_master;
Rename TABLE sales_master to sman_mast;
select * from client_master;
select * from sales_master;
select * from sman_mast;
select * from product_master;
```

OUTPUT:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_xyz
dient_master
employee
product_master
sales_master
student

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name
ivan bayross
mamta muzumdar
chhaya bankar
ashvini joshi
hancel colaco
deepak sharma

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

clientno	name	address1	address2	city	pincode	state	baldue
c00001	ivan bayross	NULL	NULL	mumbai	400054	maharastra	15000
c00002	mamta muzumdar	NULL	NULL	madras	780001	tamil naidu	0
c00003	chhaya bankar	NULL	NULL	mumbai	400057	maharastra	5000
c00004	ashvini joshi	NULL	NULL	banglore	560001	karnataka	0
c00005	hancel colaco	NULL	NULL	mumbai	40060	maharastra	2000
c00006	deepak sharma	NULL	NULL	mangalore	560050	karnataka	0

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	city	state
ivan bayross	mumbai	maharastra
mamta muzumdar	madras	tamil naidu
chhaya bankar	mumbai	maharastra
ashvini joshi	banglore	karnataka
hancel colaco	mumbai	maharastra
deepak sharma	mangalore	karnataka

Result Grid | Filter Rows: | Export: | Wrap C

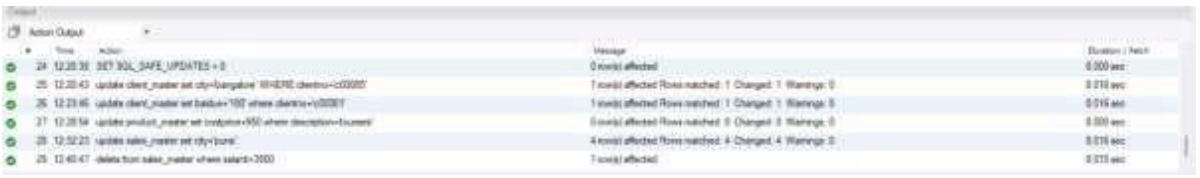
description
t-shirts
shirts
cotton jeans
jeans
trousers
pull overs
denim shirts
lycra tops
skirts

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name
ivan bayross
chhaya bankar
hancel colaco

Result Grid | Filter Rows: | Export: | Wrap Cell Co

salesmanname
aman
omkar
raj



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

salesmanno	salesmanname	address1	address2	city	pincode	state	salamt	tgttoget	ytdsales	remarks
s00001	aman	a/14	worli	pune	400002	maharastra	3000	100	50	good
s00002	omkar	65	nariman	pune	400001	maharastra	3000	200	100	good
s00003	raj	p-7	bandra	pune	400032	maharastra	3000	200	100	good

sman_mast 1

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

productno	description	profitpercent	unitmeasure	qtyonhand	reorderlvl	sellprice	costprice
p00001	t-shirts	5	piece	200	50	350	250
p0345	shirts	6	piece	150	50	500	350
p07868	trousers	2	piece	150	50	850	550
p07885	pull overs	3	piece	80	30	700	450
p07975	lycra tops	5	piece	70	30	300	175
p08865	skirts	5	piece	75	30	450	300

product_master 2

Lab Assignment 3

Program 1: Create the tables described below in the same database created in assignment 2 ‘inventory’

SALES_ORDER

SALES_ORDER_DETAILES

Source Code:

```
CREATE TABLE SALES_ORDER (
   ORDERNO VARCHAR(6) PRIMARY KEY CHECK (ORDERNO LIKE 'O%'),
   CLIENTNO VARCHAR(6),
   ORDERDATE DATE,
   SALESMANNO VARCHAR(6),
   DELYTYPE CHAR(1) CHECK (DELYTYPE IN ('P', 'F')),
   BILLYN CHAR(1),
   DELYDATE DATE, CHECK (DELYDATE >= ORDERDATE),
   ORDERSTATUS VARCHAR(15) CHECK (ORDERSTATUS IN ('In Process', 'Fulfilled',
'BackOrder', 'Cancelled')),
   CONSTRAINT fk_clientno FOREIGN KEY (CLIENTNO) REFERENCES
CLIENT_MASTER(CLIENTNO),
   CONSTRAINT fk_salesmanno FOREIGN KEY (SALESMANNO) REFERENCES
SALESMAN_MASTER(SALESMANNO)
);


CREATE TABLE SALES_ORDER_DETAILS (
   ORDERNO VARCHAR(6),
```

```
    PRODUCTNO VARCHAR(6),
    QTYORDERED INT(8),
    QTYDISP INT(8),
    PRODUCTRATE FLOAT(10,2),
    CONSTRAINT fk_orderno FOREIGN KEY (ORDERNO) REFERENCES
SALES_ORDER(ORDERNO),
    CONSTRAINT fk_productno FOREIGN KEY (PRODUCTNO) REFERENCES
PRODUCT_MASTER(PRODUCTNO)
);
```

Output:

	Field	Type	Null	Key	Default	Extra
▶	ORDERNO	varchar(6)	NO	PRI	NULL	
	CLIENTNO	varchar(6)	YES	MUL	NULL	
	ORDERDATE	date	YES		NULL	
	SALESMANNO	varchar(6)	YES	MUL	NULL	
	DELYTYPE	char(1)	YES		NULL	
	BILLYN	char(1)	YES		NULL	
	DELYDATE	date	YES		NULL	
	ORDERSTATUS	varchar(15)	YES		NULL	

	Field	Type	Null	Key	Default	Extra
▶	ORDERNO	varchar(6)	YES	MUL	NULL	
	PRODUCTNO	varchar(6)	YES	MUL	NULL	
	QTYORDERED	int	YES		NULL	
	QTYDISP	int	YES		NULL	
	PRODUCTRATE	float(10,2)	YES		NULL	

Program 2: Insert the data into their respective table

Source Code:

```
INSERT INTO SALES_ORDER (ORDERNO, CLIENTNO, ORDERDATE,  SALESMANNO,
DELYTYPE, BILLYN, DELYDATE, ORDERSTATUS)

VALUES

('O19001', 'C00001', '2004-06-12', 'S00001', 'F', 'N', '2004-07-20', 'In Progress'),

('O19002', 'C00002', '2004-06-25', 'S00002', 'P', 'N', '2004-06-27', 'Cancelled'),

('O46865', 'C00003', '2004-02-18', 'S00003', 'F', 'Y', '2004-02-20', 'Fulfilled'),

('O19003', 'C00001', '2004-04-03', 'S00001', 'F', 'Y', '2004-04-07', 'Fulfilled'),

('O46866', 'C00004', '2004-05-20', 'S00002', 'P', 'N', '2004-05-22', 'Cancelled'),

('O19008', 'C00005', '2004-05-24','S00004', 'F', 'N', '2004-07-26', 'In Progress');


INSERT INTO SALES_ORDER_DETAILS (ORDERNO, PRODUCTNO, QTYORDERED,
QTYDISP, PRODUCTRATE)

VALUES

('O19001', 'P00001', 4, 4, 525),

('O19001', 'P07965', 2, 1, 8400),

('O19001', 'P07885', 2, 1, 5250),

('O19002', 'P00001', 10, 0, 525),

('O46865', 'P07868', 3, 3, 3150),

('O46865', 'P07885', 3, 1, 5250),

('O46865', 'P00001', 10, 10, 525),

('O46865', 'P0345', 4, 4, 1050),
```

('O19003', 'P03453', 2, 2, 1050),

('O19003', 'P06734', 1, 1, 12000),

('O46866', 'P07965', 1, 0, 8400),

('O19008', 'P07975', 1, 0, 1050),

('O19008', 'P00001', 10, 5, 525),

('O19008', 'P07975', 5, 3, 1050);

Output:

ORDERNO	CLIENTNO	ORDERDATE	SALESMANNO	DELYTYPE	BILLYN	DELYDATE	ORDERSTATUS
O19001	C00001	2004-06-12	S00001	F	N	2004-07-20	In Progress
O19002	C00002	2004-06-25	S00002	P	N	2004-06-27	Cancelled
O19003	C00001	2004-04-03	S00001	F	Y	2004-04-07	Fulfilled
O19008	C00005	2004-05-24	S00004	F	N	2004-07-26	In Progress
O46865	C00003	2004-02-18	S00003	F	Y	2004-02-20	Fulfilled
O46866	C00004	2004-05-20	S00002	P	N	2004-05-22	Cancelled
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Program 3: Modify the tables created in the Assignment 2 with following constraint

Source Code:

```
ALTER TABLE CLIENT_MASTER

MODIFY CLIENTNO VARCHAR(6) CHECK (CLIENTNO LIKE 'C%') PRIMARY KEY,

MODIFY NAME VARCHAR(20) NOT NULL,

MODIFY CITY VARCHAR(15),

MODIFY PINCODE INT(8),

MODIFY STATE VARCHAR(15),

MODIFY BALDUE FLOAT(10,2);


ALTER TABLE PRODUCT_MASTER

MODIFY PRODUCTNO VARCHAR(6) CHECK (PRODUCTNO LIKE 'P%') PRIMARY
KEY,

MODIFY DESCRIPTION VARCHAR(15) NOT NULL,

MODIFY PROFITPERCENT FLOAT(4,2) NOT NULL,

MODIFY UNITMEASURE VARCHAR(10) NOT NULL,

MODIFY QTYONHAND INT(8) NOT NULL,

MODIFY REORDERLVL INT(8) NOT NULL,

MODIFY SELLPRICE FLOAT(8,2) NOT NULL CHECK (SELLPRICE <> 0),

MODIFY COSTPRICE FLOAT(8,2) NOT NULL CHECK (COSTPRICE <> 0);
```

```
ALTER TABLE SALESMAN_MASTER

MODIFY SALESMANNO VARCHAR(6) CHECK (SALESMANNO LIKE 'S%') PRIMARY
KEY,

MODIFY SALESMANNAME VARCHAR(20) NOT NULL,

MODIFY ADDRESS1 VARCHAR(30) NOT NULL,

MODIFY ADDRESS2 VARCHAR(30),

MODIFY CITY VARCHAR(20),

MODIFY PINCODE INT(8),

MODIFY STATE VARCHAR(20),

MODIFY SALAMT FLOAT(8,2) NOT NULL CHECK (SALAMT <> 0),

MODIFY TGTTOGET FLOAT(6,2) NOT NULL CHECK (TGTTOGET <> 0),

MODIFY YTDSALES FLOAT(6,2) NOT NULL,

MODIFY REMARKS VARCHAR(60);
```

Output:

Field	Type	Null	Key	Default	Extra
CLIENTNO	varchar(6)	NO	PRI	NULL	
NAME	varchar(20)	NO		NULL	
CITY	varchar(15)	YES		NULL	
PINCODE	int	YES		NULL	
STATE	varchar(15)	YES		NULL	
BALDUE	float(10,2)	YES		NULL	

	Field	Type	Null	Key	Default	Extra
▶	PRODUCTNO	varchar(6)	NO	PRI	NULL	
	DESCRIPTION	varchar(15)	NO		NULL	
	PROFITPERCENT	float(4,2)	NO		NULL	
	UNITMEASURE	varchar(10)	NO		NULL	
	QTYONHAND	int	NO		NULL	
	REORDERLVL	int	NO		NULL	
	SELLPRICE	float(8,2)	NO		NULL	
	COSTPRICE	float(8,2)	NO		NULL	

	Field	Type	Null	Key	Default	Extra
▶	SALESMANNO	varchar(6)	NO	PRI	NULL	
	SALESMANNAME	varchar(20)	NO		NULL	
	ADDRESS1	varchar(30)	NO		NULL	
	ADDRESS2	varchar(30)	YES		NULL	
	CITY	varchar(20)	YES		NULL	
	PINCODE	int	YES		NULL	
	STATE	varchar(20)	YES		NULL	
	SALAMT	float(8,2)	NO		NULL	
	TGTTOGET	float(6,2)	NO		NULL	
	YTDSALES	float(6,2)	NO		NULL	
	REMARKS	varchar(60)	YES		NULL	

Lab Assignment 4

Program 1: **Change the client_master table by converting each value of the field in uppercase.**

Source Code:

```
SELECT UCASE(NAME) FROM CLIENT_MASTER;
```

Output:

	UCASE(NAME)
▶	IVAN BAYROSS
	MAMTA MUZUMDAR
	CHHAYA BANKAR
	ASHVINI JOSHI
	HANCEL COLACO
	DEEPAK SHARMA

Program 2: **Display the length of the value in the description field from PRODUCT_MASTER table.**

Source Code:

```
SELECT LENGTH (DESCRIPTION) FROM PRODUCT_MASTER;
```

Output:

	LENGTH(DESCRIPTION)
▶	8
	6
	12
	5
	8
	10
	12
	10
	6

Program 3: **Change the Salesman Name and Product Description to All Capital letters.**

Source Code:

```
SELECT UCASE(DESCRIPTION) FROM PRODUCT_MASTER;

SELECT UCASE(SALESMANNAME) FROM SALESMAN_MASTER;
```

Output:

	UCASE(SALESMANNAME)
▶	AMAN
	OMKAR
	RAJ
	ASHISH

	UCASE(DESCRIPTION)
▶	T-SHIRTS
	SHIRTS
	COTTON JEANS
	JEANS
	TROUSERS
	PULL OVERS
	DENIM SHIRTS
	LYCRA TOPS
	SKIRTS

Program 4: **Display the system date and time.**

Source Code:

```
SELECT NOW() FROM CLIENT_MASTER;
```

Output:

	NOW()
▶	2024-02-17 21:52:08
	2024-02-17 21:52:08
	2024-02-17 21:52:08
	2024-02-17 21:52:08

Program 5: **List the order numbers delivered in the month of July.**

Source Code:

```
SELECT COUNT(ORDERDATE) FROM SALES_ORDER WHERE MONTH(ORDERDATE) =7;
```

Output:

	COUNT(ORDERDATE)
▶	0

Program 6: **Display the number of days in between the delivery date and order date for each order.**

Source Code:

```
SELECTDATEDIFF( DELYDATE, ORDERDATE) AS DAYS_BETWEEN FROM SALES_ORDER ;
```

Output:

	DAYS_BETWEEN
▶	38
	2
	4
	63
	2
	2

Program 7: **Display the number of orders delivered in between 20-may-04 and 25-june-04.**

Source Code:

```
SELECT COUNT(ORDERSTATUS) FROM SALES_ORDER WHERE ORDERSTATUS="FULFILLED" AND DELYDATE BETWEEN'2004-06-12' AND '2004-06-12';
```

Output:

	COUNT(ORDERSTATUS)
▶	0

Program 8: **Retrieve the order no and name of the weekday for all the order_date.**

Source Code:

```
SELECT ORDERNO,DAYNAME(ORDERDATE) FROM SALES_ORDER;
```

Output:

	ORDERNO	DAYNAME(ORDERDATE)
▶	O19001	Saturday
	O19002	Friday
	O19003	Saturday
	O19008	Monday
	O46865	Wednesday
	O46866	Thursday

Program 9: **Find the order_no who have cancelled or in processed orders in the month of May.**

Source Code:

```
SELECT ORDERNO  FROM SALES_ORDER WHERE ORDERSTATUS="IN PROGRESS" OR ORDERSTATUS="CANCELLED" AND MONTH(ORDERDATE)=5;
```

Output:

	ORDERNO
▶	O19001
	O19008
	O46866
*	NULL

Program 10: **Retrieve minimum qtyordered.**

Source Code:

```
SELECT MIN(QTYORDERED)FROM SALES_ORDER_DETAILS;
```

Output:

	MIN(QTYORDERED)
▶	1

Program 11: **Display the order no and the product no who got maximum rate of the product.**

Source Code:

```
SELECT ORDERNO,PRODUCTNO,PRODUCTRATE FROM SALES_ORDER_DETAILS WHERE PRODUCTRATE=(SELECT  MAX(PRODUCTRATE) FROM SALES_ORDER_DETAILS);
```

Output:

	ORDERNO	PRODUCTNO	PRODUCTRATE
▶	O19003	P06734	12000.00

Program 12: **Retrieve the product no. and description of the product who got maximum profit.**

Source Code:

```
SELECT DESCRIPTION,PRODUCTNO,PROFITPERCENT FROM PRODUCT_MASTER WHERE PROFITPERCENT=(SELECT  MAX(PROFITPERCENT) FROM PRODUCT_MASTER);
```

Output:

	DESCRIPTION	PRODUCTNO	PROFITPERCENT
▶	Shirts	P0345	6.00
*	NULL	NULL	NULL

Program 13: **Retrieve the minimum rate of the product for each order.**

Source Code:

```
SELECT MIN(PRODUCTRATE) PRODUCTNO,ORDERNO FROM SALES_ORDER_DETAILS GROUP BY ORDERNO ;
```

Output:

	PRODUCTNO	ORDERNO
	525.00	O19001
	525.00	O19002
	1050.00	O19003
	525.00	O19008
	525.00	O46865
	8400.00	O46866

Program 14: **Find the description and sell price of the product who get at least 3% profit.**

Source Code:

```
SELECT DESCRIPTION,SELLPRICE,PROFITPERCENT FROM PRODUCT_MASTER WHERE PROFITPERCENT>3 OR PROFITPERCENT=3;\
```

Output:

	DESCRIPTION	SELLPRICE	PROFITPERCENT
▶	T-Shirts	350.00	5.00
	Shirts	500.00	6.00
	Cotton Jeans	600.00	5.00
	Jeans	750.00	5.00
	Denim Shirts	350.00	4.00
	Lycra Tops	300.00	5.00
	Skirts	450.00	5.00

Program 15: **Retrieve the order number that delivered maximum and minimum orders.**

Source Code:

```
SELECT QTYORDERED,ORDERNO FROM SALES_ORDER_DETAILS WHERE
QTYORDERED=(SELECT MAX(QTYORDERED) FROM SALES_ORDER_DETAILS);
```

Output:

	QTYORDERED	ORDERNO
▶	10	O19002
	10	O46865
	10	O19008

Program 16: **Display the total quality ordered for each order_no.**

Source Code:

```
SELECT QTYORDERED,ORDERNO FROM SALES_ORDER_DETAILS ;
```

Output:

QTYORDERED	ORDERNO
10	O19002
3	O46865
3	O46865
10	O46865
4	O46865
2	O19003
1	O19003
1	O46866
1	O19008
10	O19008
5	O19008

Program 17: **Increase the selling price by 13% of all products with cost price less than 310.**

Source Code:

```
UPDATE PRODUCT_MASTER  SET SELLPRICE=SELLPRICE * 1.13 WHERE COSTP
RICE
<310;
```

```
SELECT SELLPRICE , COSTPRICE FROM PRODUCT_MASTER ;
```

Output:

	SELLPRICE	COSTPRICE
▸	395.50	250.00
	500.00	350.00
	600.00	450.00
	750.00	500.00
	850.00	550.00
	700.00	450.00
	395.50	250.00
	339.00	175.00
	508.50	300.00

Program 18: **Count the number of products where profit is more than 4%.**

Source Code:

```
SELECT COUNT(PROFITPERCENT)FROM PRODUCT_MASTER W
HERE PROFITPERCENT>4;
```

Output:

	COUNT(PROFITPERCENT)
	6

Program 19: **List all the items of Sales_Order_Details table in decreasing order of Product_rate.**

Source Code:

```
SELECT * FROM SALES_ORDER_DETAILS ORDER BY PRODUCTRATE DESC;
```

Output:

ORDERNO	PRODUCTNO	QTYORDERED	QTYDISP	PRODUCTRATE
O19001	P07885	2	1	5250.00
O46865	P07885	3	1	5250.00
O46865	P07868	3	3	3150.00
O46865	P0345	4	4	1050.00
O19003	P0345	2	2	1050.00
O19008	P07975	1	0	1050.00
O19008	P07975	5	3	1050.00
O19001	P00001	4	4	525.00
O19002	P00001	10	0	525.00
O46865	P00001	10	10	525.00
O19008	P00001	10	5	525.00

Program 20: **Display the Product details in Ascending order with selling price a bove 400**

grouped based on profit percent.

Source Code:

```
SELECT * FROM PRODUCT_MASTER WHERE SELLPRICE > 400 ORD
ER BY PROFITPERCENT;
```

Output:

PRODUCTNO	DESCRIPTION	PROFITPERCENT	UNITMEASURE	QTYONHAND	REORDERLVL	SELLPRICE
P07868	Trousers	2.00	Piece	150	50	850.00
P07885	Pull Overs	2.50	Piece	80	30	700.00
P06734	Cotton Jeans	5.00	Piece	100	20	600.00
P07865	Jeans	5.00	Piece	100	20	750.00
P08865	Skirts	5.00	Piece	75	30	508.50
P0345	Shirts	6.00	Piece	150	50	500.00

Program 21: **Display the product no and description of product for which sell p rice is more than or equal to 500 in descending order of the Selling Price**

Source Code:

```
SELECT DESCRIPTION, SELLPRICE  PRODUCTNO FROM PRODUCT_MASTER WHERE SELLPRICE > 500 ORDER BY SELLPRICE DESC ;
```

Output:

DESCRIPTION	PRODUCTNO
Trousers	850.00
Jeans	750.00
Pull Overs	700.00
Cotton Jeans	600.00
Skirts	508.50
NULL	NULL

Program 22: **Count the client no in which product is ordered after 20-June-02.**

Source Code:

```
SELECT COUNT(CLIENTNO) FROM SALES_ORDER WHERE ORDERDATE > "2002-06-20";
```

Output:

COUNT(CLIENTNO)
6

Program 23: **Count the order no grouped by Order Status and Delivery Type.**

Source Code:

```
SELECT COUNT(ORDERNO) ORDERNO,DELYTYPE FROM SALES_ORDER AS ORDER_COUNT GROUP BY ORDERSTATUS , DELYTYPE;
```

Output:

ORDERNO	DELYTYPE
2	F
2	P
2	F

Program 24: **Find the total number of orders for each product number that have maximum 1200 product rate.**

Source Code:

```
SELECTCOUNT(ORDERNO) AS
PRODUCT_COUNT_HAVING_PRICE_GREATER_THAN_1200 FROM
SALES_ORDER_DETAILS WHERE PRODUCTRATE=(SELECT MAX(PRODUCTRA
TE) FROMSALES_ORDER_DETAILS);
```

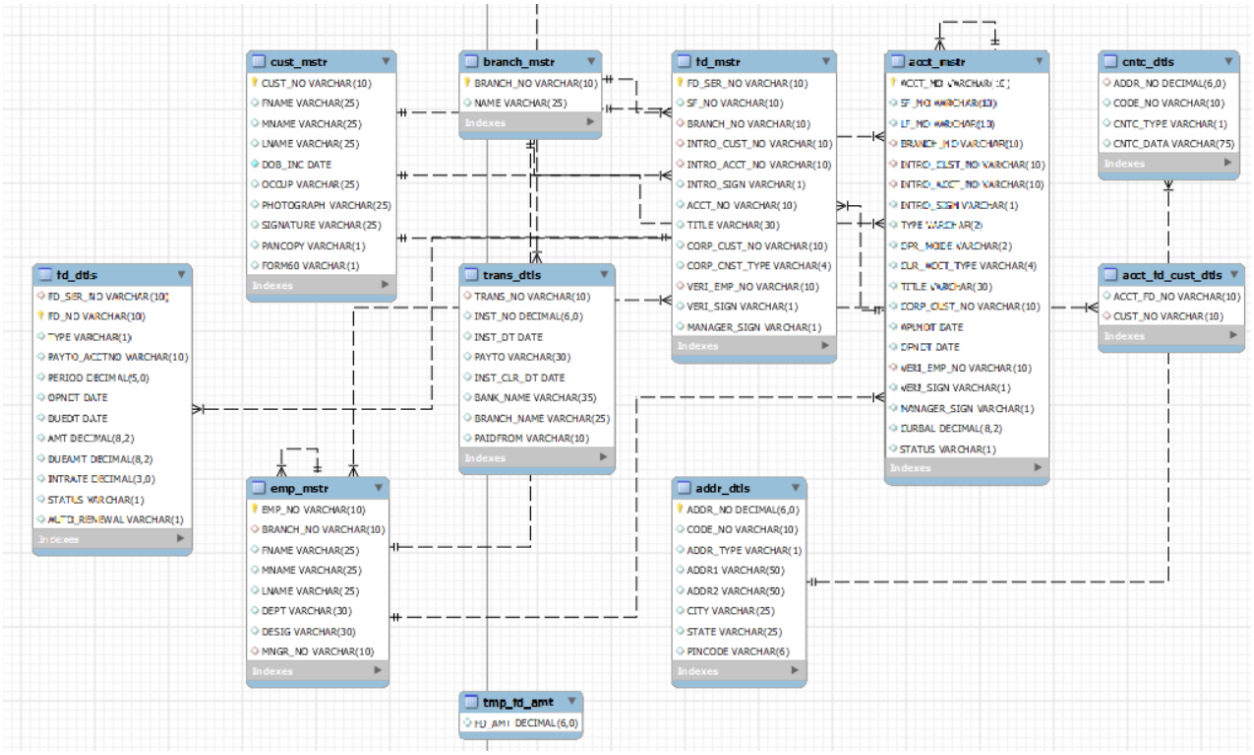
Output:

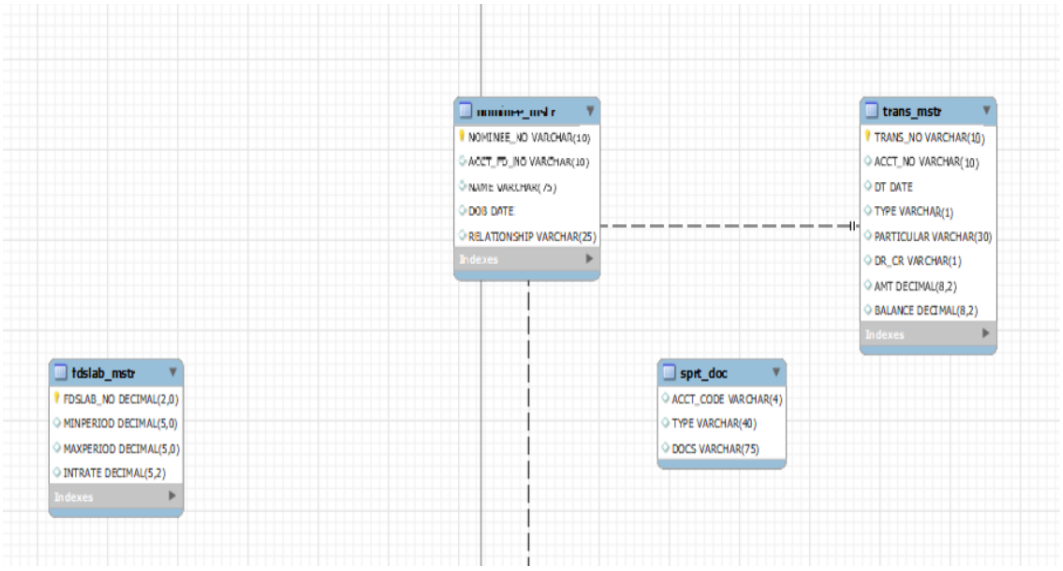
	PRODUCT_COUNT_HAVING_PRICE_GREATER_THAN_1200
1	1

Lab -5

Program 1: Draw ER diagram of the given bank database

Output:





Program 2: Create this database named 'Bank', create all tables as given in attachment.

Source Code:

```
CREATE DATABASE BANK;
```

Output:

	Database
▶	bank

Program 3: Insert all the data as given in the attachment.

Source Code:

```
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT,
DESIG, MNGR_NO)

VALUES('E8', 'B3', 'Seema', 'P.', 'Apte', 'Client Servicing', 'Clerk', 'E3');
```

Output:

✓	319	11:55:29	INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, B...	1 row(s) affected
✓	320	11:55:30	INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, B...	1 row(s) affected
✓	321	11:55:30	INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, B...	1 row(s) affected
✓	322	11:55:30	INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, B...	1 row(s) affected
✓	323	11:55:30	INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, B...	1 row(s) affected

Program 4: List accounts details of those accounts which are neither singly nor joint accounts.

Source Code:

```
SELECT * FROM ACCT_MSTR WHERE OPR_MODE NOT IN ('SI', 'JO');
```

Output:

ACCT_NO	SF_NO	LF_NO	BRANCH_NO	INTRO_CUST_NO	INTRO_ACCT_NO	INTRO_SIGN	TYPE	OPR_MODE
CA10	SF-0010	FEB04-19	B6	C10	SB9	Y	CA	AS
CA14	SF-0014	APR04-05	B5	C4	SB3	Y	CA	AS
CA4	SF-0004	DEC03-05	B5	C4	SB3	Y	CA	AS
CA7	SF-0007	JAN04-14	B1	C8	CA7	Y	CA	AS

Program 5: List the transactions performed in the months of January to March.

Source Code:

```
SELECT * FROM TRANS_DTLS WHERE MONTH(INST_DT) BETWEEN 1 AND 3;
```

Output:

TRANS_NO	INST_NO	INST_DT	PAYTO	INST_CLR_DT	BANK_NAME	BRANCH_NAME	PAIDFROM
T9	434560	2004-01-14	Self	2004-01-14	ICICI Bank	Bandra (West)	4882
T12	204907	2004-02-14	Self	2004-02-15	Memon Co-operative Bank	Jogeshwari (West)	1767
T14	100907	2004-02-19	Self	2004-02-19	Memon Co-operative Bank	Jogeshwari (West)	2001

Program 6: List all the accounts, which have not been accessed in the fourth quarter of the financial year.

Source Code:

```
SELECT * FROM ACCT_MSTR WHERE ACCT_NO NOT IN ( SELECT DISTINCT
ACCT_NO FROM TRANS_DTLS WHERE YEAR(INST_DT) = 2023 AND
QUARTER(INST_DT) = 4);
```

Output:

ACCT_NO	SF_NO	LF_NO	BRANCH_NO	INTRO_CUST_NO	INTRO_ACCT_NO	INTRO_SIGN	TYPE	OPR_MODE
CA10	SF-0010	FEB04-19	B6	C10	SB9	Y	CA	AS
CA12	SF-0012	MAR04-10	B2	C1	SB5	Y	CA	JO
CA14	SF-0014	APR04-05	B5	C4	SB3	Y	CA	AS
CA2	SF-0002	NOV03-10	B2	C1	SB1	Y	CA	JO

Program 7: List the customers whose name have the second character as a or s.

Source Code:

```
SELECT * FROM CUST_MSTR WHERE FNAME LIKE '_A%' OR '_S%';
```

Output:

CUST_NO	FNAME
C10	Namita
C3	Mamta
C6	Hansel

Program 8: List the customers whose names begin with the letters Iv and it is a four letter word.

Source Code:

```
SELECT * FROM CUST_MSTR WHERE FNAME LIKE 'Iv__';
```

Output:

CUST_NO	FNAME	MNAME	LNAME	DOB_INC	OCCUP
C1	Ivan	Nelson	Bayross	1952-06-25	Self Employed

Program 9: List the customer details of the customers named Hansel, Mamta, Namita and Aruna.

Source Code:

```
SELECT * FROM CUST_MSTR WHERE FNAME IN ('Hansel', 'Mamta', 'Namita', 'Aruna') OR
LNAME IN ('Hansel', 'Mamta', 'Namita', 'Aruna');
```

Output:

CUST_NO	FNAME	MNAME	LNAME	DOB_INC	OCCUP	PHOTOGRAPH	SIGNATURE	PANCOPY	FORM60
C10	Namita	S.	Kanade	1978-06-10	Self Employed	D:/ClntPht/C10.gif	D:/ClntSgnt/C10.gif	Y	Y
C3	Mamta	Arvind	Muzumdar	1975-08-28	Service	D:/ClntPht/C3.gif	D:/ClntSgnt/C3.gif	Y	Y
C6	Hansel	I.	Colaco	1982-01-01	Service	D:/ClntPht/C6.gif	D:/ClntSgnt/C6.gif	N	Y

Program 10: List the customer details of the customers other than Hansel, Mamta, Namita and Aruna.

Source Code:

```
SELECT * FROM CUST_MSTR WHERE FNAME NOT IN ('Hansel', 'Mamta', 'Namita',
'Aruna');
```

Output:

CUST_NO	FNAME	MNAME	LNAME	DOB_INC	OCCUP	PHOTOGRAPH	SIGNATURE
C1	Ivan	Nelson	Bayross	1952-06-25	Self Employed	D:/ClntPht/C1.gif	D:/ClntSgnt/C1.gif
C2	Chriselle	Ivan	Bayross	1982-10-29	Service	D:/ClntPht/C2.gif	D:/ClntSgnt/C2.gif
C4	Chhaya	Sudhakar	Bankar	1976-10-06	Service	D:/ClntPht/C4.gif	D:/ClntSgnt/C4.gif
C5	Ashwini	Dilip	Joshi	1978-11-20	Business	D:/ClntPht/C5.gif	D:/ClntSgnt/C5.gif
C7	Anil	Arun	Dhone	1983-10-12	Self Employed	D:/ClntPht/C7.gif	D:/ClntSgnt/C7.gif

Program 11: Select average, minimum and maximum balance from ACCT_MSTR.

Source Code:

SELECT AVG(CURBAL) AS AverageBalance, MIN(CURBAL) AS MinimumBalance, MAX(CURBAL) AS MaximumBalance FROM ACCT_MSTR;

Output:

AverageBalance	MinimumBalance	MaximumBalance
5900.000000	500.00	32000.00

Program 12: Find out how many employees are there in each branch?

Source Code:

SELECT BRANCH_NO,COUNT(EMP_NO) AS NumOfEmployees FROM EMP_MSTR GROUP BY BRANCH_NO;

Output:

BRANCH_NO	NumOfEmployees
B1	2
B2	2
B3	2
B4	2
B6	2

Program 14: Find out the total number of accounts segregated on the basis of account per branch.

Source Code:

SELECT BRANCH_NO, COUNT(ACCT_NO) AS TotalAccounts FROM ACCT_MSTR GROUP BY BRANCH_NO;

Output:

BRANCH_NO	TotalAccounts
B1	3
B2	3
B3	2
B4	2
B5	2

Program 15: Find out the customers with their number of accounts having more than one account in the bank.

Source Code:

```
SELECT INTRO_CUST_NO, COUNT(ACCT_NO) AS NumOfAccounts FROM
ACCT_MSTR GROUP BY INTRO_CUST_NO HAVING COUNT(ACCT_NO) > 1;
```

Output:

INTRO_CUST_NO	NumOfAccounts
C1	6
C10	2
C4	4

Program 16: Find out the number of accounts opened at a branch after 3 rd January 2003, only if the number of accounts after 3 rd January 2003 exceeds q.

Source Code:

```
SELECT BRANCH_NO, COUNT(ACCT_NO) AS NewAccountsCount FROM ACCT_MSTR
WHERE OPNDT > '2003-01-03' GROUP BY BRANCH_NO;
```

Output:

BRANCH_NO	NewAccountsCount
B1	3
B2	3
B3	2
B4	2
B5	2
B6	3

Program 17: List customer numbers, which are associated with only one account (or fixed deposit) in the bank. (Unique Entries Only)

Source Code:

```
SELECT INTRO_CUST_NO, COUNT(ACCT_NO) AS TOTAL_COUNT FROM
ACCT_MSTR GROUP BY INTRO_CUST_NO HAVING COUNT(ACCT_NO) = 1;
```

Output:

INTRO_CUST_NO	TOTAL_COUNT
C5	1
C8	1
C9	1

Program 18: List the customer numbers associated with more than one account.(Or Fixed Deposits)(Non Unique Entries).

Source Code:

```
SELECT INTRO_CUST_NO, COUNT(ACCT_NO) AS TOTAL_COUNT FROM
ACCT_MSTR GROUP BY INTRO_CUST_NO HAVING COUNT(ACCT_NO) > 1;
```

Output:

INTRO_CUST_NO	TOTAL_COUNT
C1	6
C10	2
C4	4

Program 19: Create a report on the fixed deposit accounts available in the bank, providing the amount and the due amount per fixed deposit(per FD_NO in the FD_DTLS table) and per slot(per FD_SER_NO in the FD_MSTR table) of fixed deposit held by the customer.

Source Code:

Output:

Program 20: Retrieve the address of a customer named 'Ivan Bayross'.

Source Code:

```
SELECT ADDR1 FROM ADDR_DTLS WHERE CODE_NO =(SELECT CUST_NO FROM
CUST_MSTR WHERE FNAME = 'Ivan' AND LNAME = 'Bayross');
```

Output:

ADDR1	CODE_NO
F-12, Diamond Palace, West Avenue,	C1

Program 22: List Customers holding fixed deposits in the bank of amount more than 5000.

Source Code:

```
SELECT CUST_NO, FNAME, LNAME FROM CUST_MSTR WHERE CUST_NO IN
(SELECT CUST_NO FROM FD_DTLS WHERE AMT > 5000);
```

Output:

CUST_NO	FNAME	LNAME
C9	Ashwini	Apte
O11	NULL	NULL
O12	NULL	NULL

CUST_NO	FNAME	LNAME
C1	Ivan	Bayross
C10	Namita	Kanade
C2	Chriselle	Bayross
C3	Mamta	Muzumdar
C4	Chhaya	Bankar
C5	Ashwini	Joshi
C6	Hansel	Colaco
C7	Anil	Dhone
C8	Alex	Fernandes
C9	Ashwini	Apte

Program 23: Find out all the customers having same name as the employees.

Source Code:

SELECT CUST_NO, FNAME, LNAME FROM CUST_MSTR WHERE CONCAT(FNAME, ' ', LNAME) IN (SELECT CONCAT(FNAME, ' ', LNAME) FROM EMP_MSTR);

Output:

CUST_NO	FNAME	LNAME
C1	Ivan	Bayross
NULL	NULL	NULL

Program 24: Show details of the branch according to branch's name.

Source Code:

SELECT * FROM EMP_MSTR WHERE BRANCH_NO = (SELECT BRANCH_NO FROM BRANCH_MSTR WHERE NAME ='ANDHERI');

Output:

EMP_NO	BRANCH_NO	FNAME	MNAME	LNAME	DEPT	DESIG	MNGR_NO
E2	B2	Amit	NULL	Desai	Loans And Financing	Finance Manager	NULL
E9	B2	Vikram	Vilas	Randive	Marketing	Sales Asst.	E7

Program 25: Display all the customers whose last name is Bayross and are less than 25 yrs old or all those customers who are more than 25 but less than 50 yrs old.

Source Code:

```
SELECT * FROM  CUST_MSTR

WHERE LNAME = 'Bayross' AND (

FLOOR(DATEDIFF('2024-02-24', DOB_INC) / 365.25) < 25

OR (FLOOR(DATEDIFF( '2024-02-24', DOB_INC) / 365.25) > 25 AND
FLOOR(DATEDIFF( '2024-02-24', DOB_INC) / 365.25) < 50)

);
```

Output:

CUST_NO	FNAME	MNAME	LNAME	DOB_INC	OCCUP	PHOTOGRAPH	SIGNATURE	PANCOPY	FORM60
C2	Chriselle	Ivan	Bayross	1982-10-29	Service	D:/ClntPht/C2.gif	D:/ClntSgnt/C2.gif	N	Y

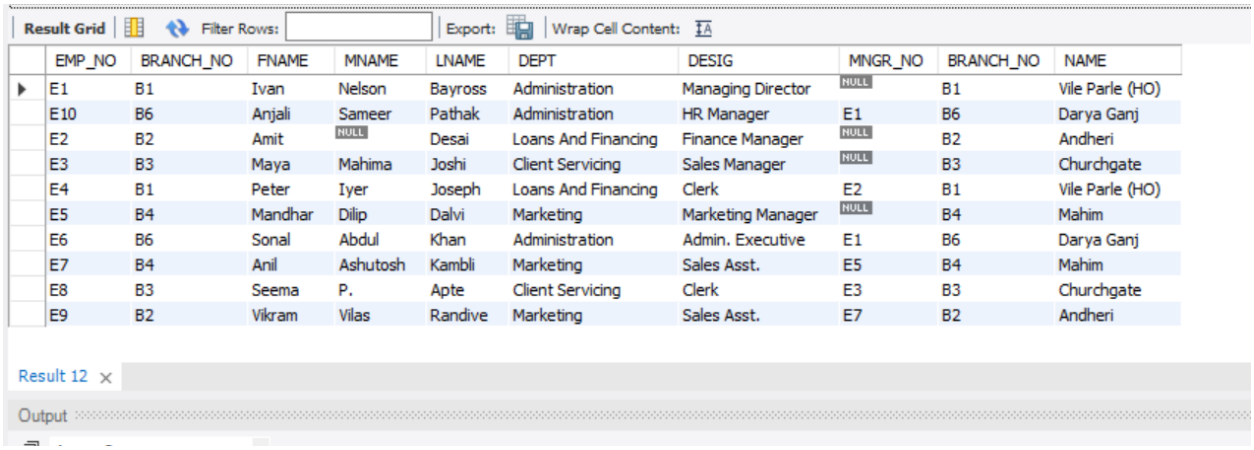
Lab Assignment 6

Program 1: List the employee details along with branch names to which they belong.

Source Code:

```
SELECT * FROM EMP_MSTR INNER JOIN BRANCH_MSTR ON
EMP_MSTR.BRANCH_NO = BRANCH_MSTR.BRANCH_NO;
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMP_NO	BRANCH_NO	FNAME	MNAME	LNAME	DEPT	DESIG	MNGR_NO	BRANCH_NO	NAME
E1	B1	Ivan	Nelson	Bayross	Administration	Managing Director	NULL	B1	Vile Parle (HO)
E10	B6	Anjali	Sameer	Pathak	Administration	HR Manager	E1	B6	Darya Ganj
E2	B2	Amit	NULL	Desai	Loans And Financing	Finance Manager	NULL	B2	Andheri
E3	B3	Maya	Mahima	Joshi	Client Servicing	Sales Manager	NULL	B3	Churchgate
E4	B1	Peter	Iyer	Joseph	Loans And Financing	Clerk	E2	B1	Vile Parle (HO)
E5	B4	Mandhar	Dilip	Dalvi	Marketing	Marketing Manager	NULL	B4	Mahim
E6	B6	Sonal	Abdul	Khan	Administration	Admin. Executive	E1	B6	Darya Ganj
E7	B4	Anil	Ashutosh	Kambli	Marketing	Sales Asst.	E5	B4	Mahim
E8	B3	Seema	P.	Apte	Client Servicing	Clerk	E3	B3	Churchgate
E9	B2	Vikram	Vilas	Randive	Marketing	Sales Asst.	E7	B2	Andheri

Result 12 ×

Output

Program 2: List the customers along with their multiple address details.

Source Code:

```
SELECT c.cust_no, c.fname, c.mname, c.lname, a.addr_no, a.addr1, a.ADDR2, a.CITY,
a.STATE, a.PINCODE FROM CUST_MSTR as c INNER JOIN ADDR_DTLS as a ON
c.CUST_NO = a.CODE_NO;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

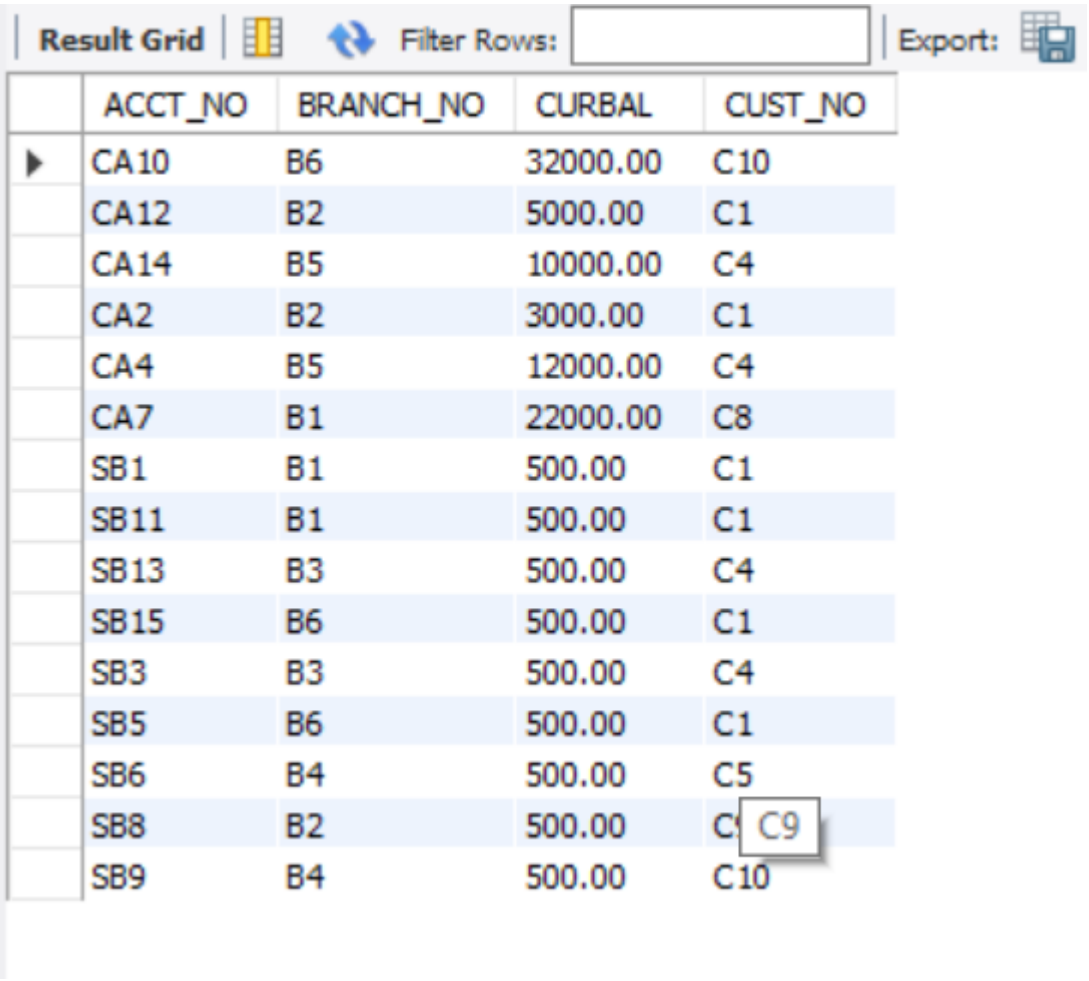
cust_no	fname	mname	lname	addr_no	addr1	ADDR2	CITY	STATE	PINCODE
C1	Ivan	Nelson	Bayross	17	F-12, Diamond Palace, West Avenue,	North Avenue, Santacruz (West),	Mumbai	Maharashtra	400056
C2	Chriselle	Ivan	Bayross	18	F-12, Silver Stream,	Santacruz (East),	Mumbai	Maharashtra	400056
C3	Mamta	Arvind	Muzumdar	19	Magesh Prasad,	Saraswati Baug, Jogeshwari(E),	Mumbai	Maharashtra	400060
C4	Chhaya	Sudhakar	Bankar	20	4, Sampada,	Kataria Road, Mahim,	Mumbai	Maharashtra	400016
C5	Ashwini	Dilip	Joshi	21	104, Vikram Apts. Bhagat Lane,	Shivaji Park, Mahim,	Mumbai	Maharashtra	400016
C6	Hansel	I.	Colaco	22	12, Radha Kunj, N.C Kelkar Road,	Dadar,	Mumbai	Maharashtra	400028
C7	Anil	Arun	Dhone	23	A/14, Shanti Society, Mogal Lane,	Mahim,	Mumbai	Maharashtra	400016
C8	Alex	Austin	Fernandes	24	5, Vagdevi, Senapati Bapat Rd.,	Dadar,	Mumbai	Maharashtra	400016
C9	Ashwini	Shankar	Apte	25	A-10 Nutan Vaishali,	Shivaji Park, Mahim,	Mumbai	Maharashtra	400016
C10	Namita	S.	Kanade	26	B-10, Makarand Society,	Cadal Road, Mahim,	Mumbai	Maharashtra	400016
O11	NULL	NULL	NULL	34	Shop No. 4, Simon Streams,	V. P. Road, Andheri (West),	Mumbai	Maharashtra	400058
O12	NULL	NULL	NULL	35	230-E, Patel Chambers,	Service Road, Vile Parle (East),	Mumbai	Maharashtra	400057
O13	NULL	NULL	NULL	36	G-2, Puru Hsg. Society,	Senapati Bapat Rd., Dadar,	Mumbai	Maharashtra	400016
O14	NULL	NULL	NULL	37	B-10, Makarand Society,	Cadal Road, Mahim,	Mumbai	Maharashtra	400016
C6	Hansel	I.	Colaco	45	203/A, Prachi Apmt.,	Andheri (East),	Mumbai	Maharashtra	400058
O15	NULL	NULL	NULL	46	Shop No. 4, Sai Compound,	Service Road, Vile Parle (East),	Mumbai	Maharashtra	400057
O15	NULL	NULL	NULL	47	G-4, Sagar Chambers,	G. P. Road, Andheri (West),	Mumbai	Maharashtra	400058

Program 3: List the Customers along with the account details associated with them.

Source code:

```
SELECT A.ACCT_NO,A.BRANCH_NO,A.CURBAL,C.CUST_NO FROM ACCT_MSTR
AS A INNER JOIN  CUST_MSTR AS C ON C.CUST_NO =A.INTRO_CUST_NO;
```

Output:



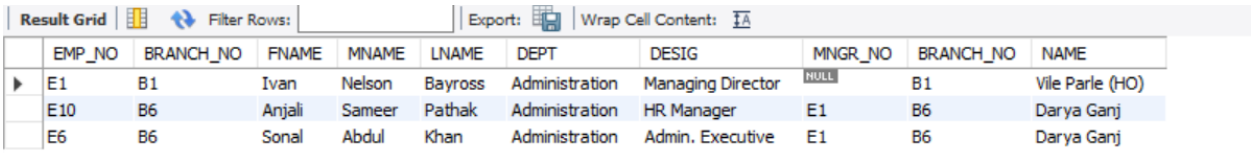
ACCT_NO	BRANCH_NO	CURBAL	CUST_NO
CA10	B6	32000.00	C10
CA12	B2	5000.00	C1
CA14	B5	10000.00	C4
CA2	B2	3000.00	C1
CA4	B5	12000.00	C4
CA7	B1	22000.00	C8
SB1	B1	500.00	C1
SB11	B1	500.00	C1
SB13	B3	500.00	C4
SB15	B6	500.00	C1
SB3	B3	500.00	C4
SB5	B6	500.00	C1
SB6	B4	500.00	C5
SB8	B2	500.00	C9
SB9	B4	500.00	C10

Program 4: List the employee details of only those employees who belong to the Administration department along branch names.

Source Code:

```
SELECT * FROM EMP_MSTR INNER JOIN BRANCH_MSTR ON
EMP_MSTR.BRANCH_NO = BRANCH_MSTR.BRANCH_NO WHERE
EMP_MSTR.DEPT='Administration';
```

Output:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMP_NO	BRANCH_NO	FNAME	MNAME	LNAME	DEPT	DESIG	MNGR_NO	BRANCH_NO	NAME
E1	B1	Ivan	Nelson	Bayross	Administration	Managing Director	NULL	B1	Vile Parle (HO)
E10	B6	Anjali	Sameer	Pathak	Administration	HR Manager	E1	B6	Darya Ganj
E6	B6	Sonal	Abdul	Khan	Administration	Admin. Executive	E1	B6	Darya Ganj

Program 5: List the employee details along with the contact details (if any) Using Left Outer Join.

Source code:

```
SELECT E.EMP_No,E.FNAME,E.LNAME,E.DEPT,C.CNTC_DATA FROM EMP_MSTR AS
E LEFT JOIN CNTC_DTLS AS C ON E.BRANCH_NO = C.CODE_NO;
```

Output:

EMP_No	FNAME	LNAME	DEPT	CNTC_DATA
E1	Ivan	Bayross	Administration	admin_vileparle@bom2.vsnl.in
E1	Ivan	Bayross	Administration	26124533
E1	Ivan	Bayross	Administration	26124571
E10	Anjali	Pathak	Administration	admin_matunga@bom2.vsnl.in
E10	Anjali	Pathak	Administration	24304545
E2	Amit	Desai	Loans And Financing	admin_andheri@bom2.vsnl.in
E2	Amit	Desai	Loans And Financing	26790014
E3	Maya	Joshi	Client Servicing	admin_churchgate@bom2.vsnl.in
E3	Maya	Joshi	Client Servicing	23457855
E4	Peter	Joseph	Loans And Financing	admin_vileparle@bom2.vsnl.in
E4	Peter	Joseph	Loans And Financing	26124533
E4	Peter	Joseph	Loans And Financing	26124571
E5	Mandhar	Dalvi	Marketing	admin_sion@bom2.vsnl.in
E5	Mandhar	Dalvi	Marketing	25545455
E6	Sonal	Khan	Administration	admin_matunga@bom2.vsnl.in
E6	Sonal	Khan	Administration	24304545
E7	Anil	Kambli	Marketing	admin_sion@bom2.vsnl.in
E7	Anil	Kambli	Marketing	25545455

Program 6: List the employee details along with the contact details (if any) Using right Outer Join

Source Code:

```
SELECT E.EMP_No,E.FNAME,E.LNAME,E.DEPT,C.CNTC_DATA FROM EMP_MSTR AS
E RIGHT JOIN CNTC_DTLS AS C ON E.BRANCH_NO = C.CODE_NO;
```

Output:

EMP_No	FNAME	LNAME	DEPT	CNTC_DATA
E1	Ivan	Bayross	Administration	26124571
E4	Peter	Joseph	Loans And Financing	26124571
E1	Ivan	Bayross	Administration	26124533
E4	Peter	Joseph	Loans And Financing	26124533
E1	Ivan	Bayross	Administration	admin_vileparle@bom2.vsnl.in
E4	Peter	Joseph	Loans And Financing	admin_vileparle@bom2.vsnl.in
E2	Amit	Desai	Loans And Financing	26790014
E9	Vikram	Randive	Marketing	26790014
E2	Amit	Desai	Loans And Financing	admin_andheri@bom2.vsnl.in
E9	Vikram	Randive	Marketing	admin_andheri@bom2.vsnl.in
E3	Maya	Joshi	Client Servicing	23457855
E8	Seema	Apte	Client Servicing	23457855
E3	Maya	Joshi	Client Servicing	admin_churchgate@bom2.vs...
E8	Seema	Apte	Client Servicing	admin_churchgate@bom2.vs...
E5	Mandhar	Dalvi	Marketing	25545455
E7	Anil	Kambli	Marketing	25545455
E5	Mandhar	Dalvi	Marketing	admin_sion@bom2.vsnl.in
E7	Anil	Kambli	Marketing	admin_sion@bom2.vsnl.in
NULL	NULL	NULL	NULL	

Dalvi

Program 8 Retrieve the names of the employees and the names of their respective managers from the employee table

Source Code:

```
SELECT E.FNAME AS employee_name,

M.emp_no  AS MNGR_NO, M.MNAME AS manager

FROM EMP_MSTR AS E

JOIN EMP_MSTR AS M ON  E.MNGR_NO = M.emp_no;
```

Output:

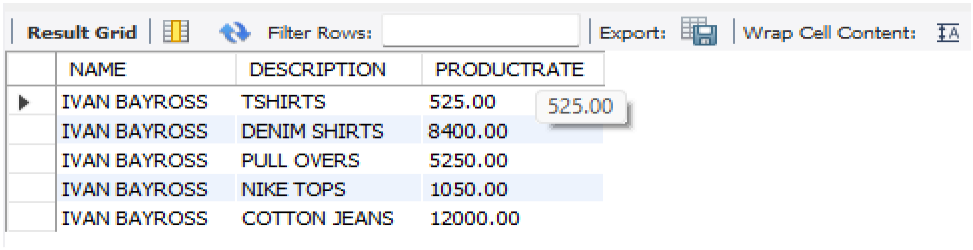
	employee_name	MNGR_NO	manager
▶	Anjali	E1	Nelson
	Peter	E2	NULL
	Sonal	E1	Nelson
	Anil	E5	Dilip
	Seema	E3	Mahima
	Vikram	E7	Ashutosh

Program 9 Retrieve the names of the employees and the names of their respective managers from the employee table

Source Code:

```
SELECT client_master.NAME,PRODUCT_MASTER.DESCRIPTION
,SALES_ORDER_DETAILS.PRODUCTRATE

FROM client_master

JOIN sales_order ON client_master.ClientNo = sales_order.ClientNo

JOIN SALES_ORDER_DETAILS ON sales_order.orderno =
SALES_ORDER_DETAILS.orderno

JOIN PRODUCT_MASTER ON SALES_ORDER_DETAILS.PRODUCTNO
=PRODUCT_MASTER.PRODUCTNO

WHERE client_master.NAME = "IVAN BAYROSS";
```

Output:



NAME	DESCRIPTION	PRODUCTRATE
IVAN BAYROSS	TSHIRTS	525.00
IVAN BAYROSS	DENIM SHIRTS	8400.00
IVAN BAYROSS	PULL OVERS	5250.00
IVAN BAYROSS	NIKE TOPS	1050.00
IVAN BAYROSS	COTTON JEANS	12000.00

Program 10 Find out the products and their quantities that will have to be delivered in the current month.

Source Code:

```
SELECT PRODUCT_MASTER.DESCRIPTION
,SALES_ORDER_DETAILS.QTYORDERED

FROM sales_order

JOIN SALES_ORDER_DETAILS ON sales_order.orderno =
SALES_ORDER_DETAILS.orderno

JOIN PRODUCT_MASTER ON SALES_ORDER_DETAILS.PRODUCTNO
=PRODUCT_MASTER.PRODUCTNO

WHERE MONTH(sales_order.orderdate) = MONTH(CURRENT_DATE);
```

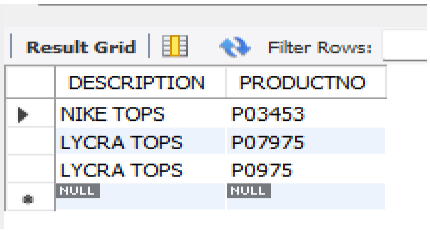
Output:

Program 11 List the ProductNo and description of constantly sold (i.e. rapidly moving) products.

Source Code:

```
SELECT PRODUCT_MASTER.DESCRIPTION,PRODUCT_MASTER.PRODUCTNO
FROM PRODUCT_MASTER
WHERE REORDERLVL = (SELECT MAX(REORDERLVL) FROM PRODUCT_MASTER);
```

Output:

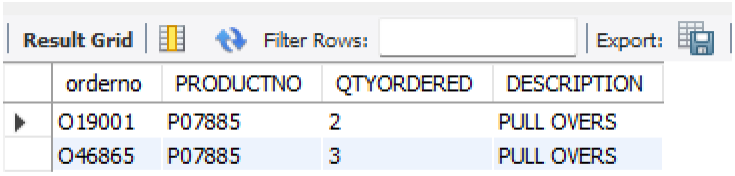


Program 12 List the products and orders from customers who have ordered less than 5 units of Pull Overs.

Source Code:

```
SELECT
SALES_ORDER_DETAILS.orderno,SALES_ORDER_DETAILS.PRODUCTNO,SALES_ORDER_DETAILS.QTYORDERED, PRODUCT_MASTER.DESCRIPTION

FROM SALES_ORDER_DETAILS

JOIN sales_order ON SALES_ORDER_DETAILS.orderno = sales_order.orderno

JOIN PRODUCT_MASTER ON SALES_ORDER_DETAILS.PRODUCTNO
=PRODUCT_MASTER.PRODUCTNO

WHERE PRODUCT_MASTER.DESCRIPTION = 'Pull Overs'

   AND SALES_ORDER_DETAILS.QTYORDERED < 5;
```

Output:



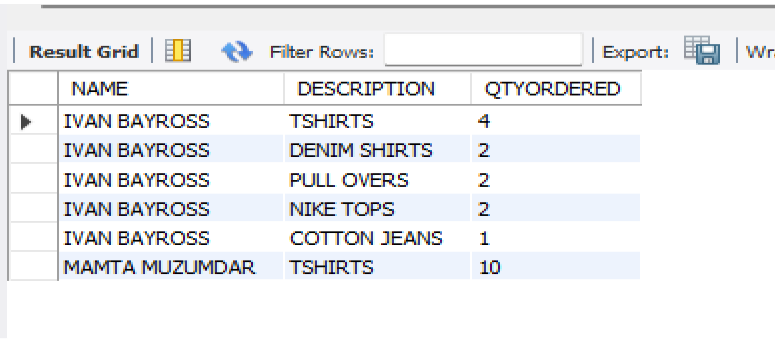
orderno	PRODUCTNO	QTYORDERED	DESCRIPTION
O19001	P07885	2	PULL OVERS
O46865	P07885	3	PULL OVERS

Program 13 Find the products and their quantities for the orders placed by lvan Bayross' and 'Mamta Muzumdar.

Source Code:

```
SELECT client_master.NAME,PRODUCT_MASTER.DESCRIPTION
,SALES_ORDER_DETAILS.QTYORDERED

FROM client_master

JOIN sales_order ON client_master.ClientNo = sales_order.ClientNo

JOIN SALES_ORDER_DETAILS ON sales_order.orderno =
SALES_ORDER_DETAILS.orderno

JOIN PRODUCT_MASTER ON SALES_ORDER_DETAILS.PRODUCTNO
=PRODUCT_MASTER.PRODUCTNO

WHERE client_master.NAME  IN ("MAMTA MUZUMDAR","IVAN BAYROSS")  ;
```

Output:



Result Grid | Filter Rows: | Export: | Wr

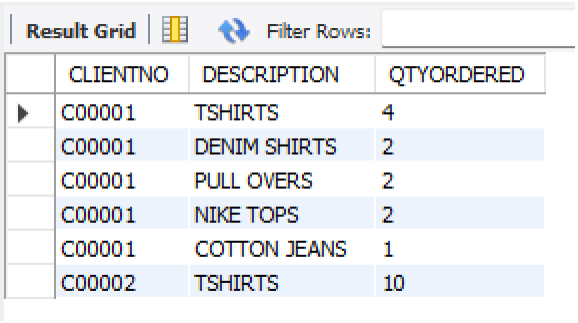
NAME	DESCRIPTION	QTYORDERED
IVAN BAYROSS	TSHIRTS	4
IVAN BAYROSS	DENIM SHIRTS	2
IVAN BAYROSS	PULL OVERS	2
IVAN BAYROSS	NIKE TOPS	2
IVAN BAYROSS	COTTON JEANS	1
MAMTA MUZUMDAR	TSHIRTS	10

Program 14 Find the products and their quantities for the orders placed by ClientNo 'C00001' and 'C00002'.

Source Code:

```
SELECT client_master.CLIENTNO,PRODUCT_MASTER.DESCRIPTION
,SALES_ORDER_DETAILS.QTYORDERED

FROM client_master

JOIN sales_order ON client_master.ClientNo = sales_order.ClientNo

JOIN SALES_ORDER_DETAILS ON sales_order.orderno =
SALES_ORDER_DETAILS.orderno

JOIN PRODUCT_MASTER ON SALES_ORDER_DETAILS.PRODUCTNO
=PRODUCT_MASTER.PRODUCTNO

WHERE client_master.CLIENTNO  IN ('C00001' , 'C00002');
```

Output:



Result Grid | Filter Rows:

CLIENTNO	DESCRIPTION	QTYORDERED
C00001	TSHIRTS	4
C00001	DENIM SHIRTS	2
C00001	PULL OVERS	2
C00001	NIKE TOPS	2
C00001	COTTON JEANS	1
C00002	TSHIRTS	10

Lab Assignment 7

Program 1. Create a view (Columns: CustNo, Full Name, and Occupation) called Customers on the CUST_MSTR.

Source Code:

```
CREATE VIEW CUSTOMERS AS SELECT Cust_No, FName , Occup FROM CUST_MSTR;

SELECT *FROM CUSTOMERS;
```

OUTPUT:

Result Grid | Filter Rows:

Cust_No	FName	Occup
C1	Ivan	Self Employed
C10	Namita	Self Employed
C2	Chriselle	Service
C3	Mamta	Service
C4	Chhaya	Service
C5	Ashwini	Business
C6	Hansel	Service
C7	Anil	Self Employed
C8	Alex	Executive
C9	Ashwini	Service
O11	NULL	Retail Business
O12	NULL	Information ...
O13	NULL	Community ...
O14	NULL	Retail Business
O15	NULL	Retail Business
O16	NULL	Marketing

-- list the names of the self-employed customers.

```
SELECT * FROM CUSTOMERS WHERE OCCUP = "Self employed";
```

OUTPUT:

Result Grid | Filter Rows:

Cust_No	FName	Occup
C1	Ivan	Self Employed
C10	Namita	Self Employed
C7	Anil	Self Employed

-- Change the Form60 to 'N' where CustNo is 'c10'. and see it is removed from the existing table Cust_mstr.

UPDATE CUSTOMERS SET FORM60 ='N' WHERE Cust_No='C10';

OUTPUT:

-- Insert a new row in the table and see if there is a change in the existing table.

INSERT INTO CUSTOMERS (Cust_No, FName , Occup)

VALUES('C10', 'AnilI','Self Employed');

OUTPUT:

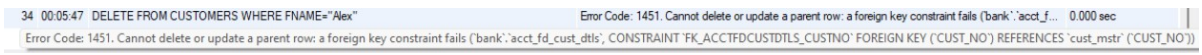
-- Remove the records of the customer whose Fname =' Alex' and see it is removed from the

-- existing table Cust_mstr.

DELETE FROM CUSTOMERS

WHERE FNAME="Alex";

OUTPUT:



Program 2. Create a view called Employees on the EMP_MSTR table. And perform the following queries:

Source Code:

-- List the structure as similar as the existing table.

```
CREATE VIEW Employees AS SELECT EMP_NO, BRANCH_NO, FNAME, MNAME,
LNAME, DEPT, DESIG, MNGR_NO FROM EMP_MSTR;
```

OUTPUT:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMP_NO	BRANCH_NO	FNAME	MNAME	LNAME	DEPT	DESIG	MNGR_NO
E1	B1	Ivan	Nelson	Bayross	Administration	Managing Director	NULL
E10	B6	Anjali	Sameer	Pathak	Administration	HR Manager	E1
E2	B2	Amit	NULL	Desai	Loans And Financing	Finance Manager	NULL
E3	B3	Maya	Mahima	Joshi	Client Servicing	Sales Manager	NULL
E4	B1	Peter	Iyer	Joseph	Loans And Financing	Manager	E2
E5	B4	Mandhar	Dilip	Dalvi	Marketing	Marketing Manager	NULL
E6	B6	Sonal	Abdul	Khan	Administration	Admin. Executive	E1
E7	B4	Anil	Ashutosh	Kambli	Marketing	Sales Asst.	E5
E8	B3	Seema	P.	Apte	Client Servicing	Manager	E3
E9	B2	Vikram	Vilas	Randive	Marketing	Sales Asst.	E7

-- List Name and Desig where DeptNo =' Administration'.

```
SELECT FNAME,DESIG FROM Employees

WHERE DEPT='Administration';
```

OUTPUT:

Result Grid | Filter Rows:

FNAME	DESIG
Ivan	Managing Director
Anjali	HR Manager
Sonal	Admin. Executive

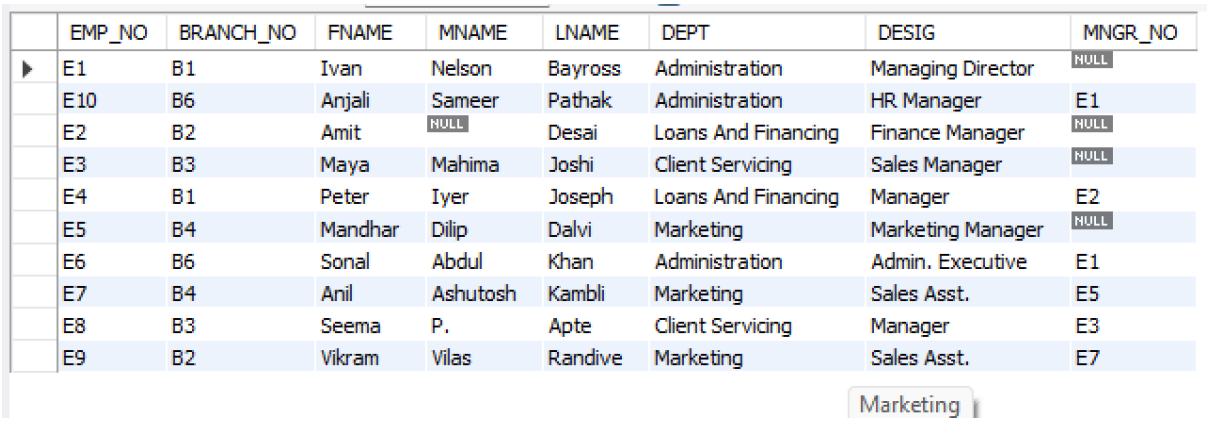
-- Change to 'Manager' where Desig =' Clerk' and see how it reflects the

-- original table.

```
UPDATE Employees SET DESIG ='Manager' WHERE Desig ='Clerk';

SELECT *FROM Employees;

SELECT *FROM Emp_mstr;
```

-- yes , changes also appeares in base table

OUTPUT:



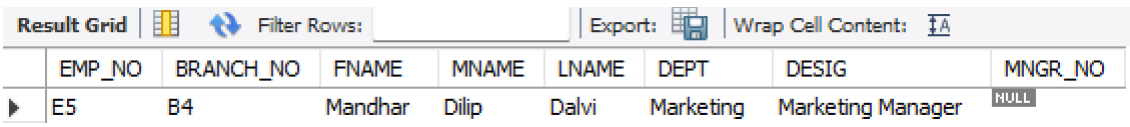
EMP_NO	BRANCH_NO	FNAME	MNAME	LNAME	DEPT	DESIG	MNGR_NO
E1	B1	Ivan	Nelson	Bayross	Administration	Managing Director	NULL
E10	B6	Anjali	Sameer	Pathak	Administration	HR Manager	E1
E2	B2	Amit	NULL	Desai	Loans And Financing	Finance Manager	NULL
E3	B3	Maya	Mahima	Joshi	Client Servicing	Sales Manager	NULL
E4	B1	Peter	Iyer	Joseph	Loans And Financing	Manager	E2
E5	B4	Mandhar	Dilip	Dalvi	Marketing	Marketing Manager	NULL
E6	B6	Sonal	Abdul	Khan	Administration	Admin. Executive	E1
E7	B4	Anil	Ashutosh	Kambli	Marketing	Sales Asst.	E5
E8	B3	Seema	P.	Apte	Client Servicing	Manager	E3
E9	B2	Vikram	Vilas	Randive	Marketing	Sales Asst.	E7

Marketing

-- show the employees whose name starts from 'M' and present in 'Marketing.

```
SELECT * FROM employees

WHERE FNAME LIKE 'M%' AND depT = 'Marketing';
```

OUTPUT:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMP_NO	BRANCH_NO	FNAME	MNAME	LNAME	DEPT	DESIG	MNGR_NO
E5	B4	Mandhar	Dilip	Dalvi	Marketing	Marketing Manager	NULL

Program 3. Print the Name, Relationship of the Nominees from the view created, where DOB is greater than the year 1980.

Source Code:

```
CREATE VIEW Nominees AS SELECT NOMINEE_NO, ACCT_FD_NO, NAME, DOB,
RELATIONSHIP FROM NOMINEE_MSTR;

SELECT *FROM Nominees

WHERE EXTRACT(YEAR FROM DOB) > 1980;
```

OUTPUT:

NOMINEE_NO	ACCT_FD_NO	NAME	DOB	RELATIONSHIP
N1	CA2	Joseph Martin Dias	1984-09-17	Colleague
N10	FS2	Nilesh Sawant	1987-08-25	Colleague
N14	FS5	Pramila P. Pius	1985-10-10	Niece
N2	CA2	Nilesh Sawant	-08-25	Colleague
N6	SB8	Rohit Rajan Sahakarkar	1985-05-30	Relative
N8	FS1	Rohit Rajan Sahakarkar	1985-05-30	Relative
N9	FS2	Joseph Martin Dias	1984-09-17	Colleague

Nilesh Sawant

Program 4. View on Multiple tables:

- Create a Master/Detail View from the tables given below. And also perform the following queries:
- Show BranchNo, Name, ADDR_NO, CITY, and PINCODE as a view where City is Like ‘M%.

Source Code:

```
CREATE VIEW Master AS

SELECT Branch_NO, Name, ADDR_NO, CITY, PINCODE

FROM BRANCH_MSTR

JOIN ADDR_DTLS ON BRANCH_MSTR.Branch_No = ADDR_DTLS.CODE_NO

WHERE ADDR_DTLS.CITY  LIKE 'M%';

SELECT *FROM MASTER;
```

OUTPUT:

Result Grid | Filter Rows: | Export:

Branch_NO	Name	ADDR_NO	CITY	PINCODE
B1	Vile Parle (HO)	1	Mumbai	400057
B2	Andheri	2	Mumbai	400058
B3	Churchgate	3	Mumbai	400004
B4	Mahim	4	Mumbai	400045
B5	Borivali	5	Mumbai	400078

- Delete the record from the view where BRANCH_NO = ‘B5’ from the view created on BRANCH_MSTR and see if the particular row is deleted from the original table or not.

Source Code:

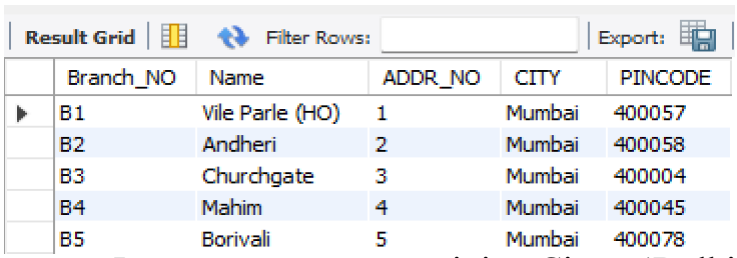
```
DELETE FROM Master WHERE Branch_NO = 'B5';

--Cannot be Deleted—
```

- List the Branch No, Name, and Complete Address where CITY =’ Mumbai’.

Source Code:

```
SELECT * FROM MASTER WHERE CITY = 'Mumbai';
```

OUTPUT:



Branch_NO	Name	ADDR_NO	CITY	PINCODE
B1	Vile Parle (HO)	1	Mumbai	400057
B2	Andheri	2	Mumbai	400058
B3	Churchgate	3	Mumbai	400004
B4	Mahim	4	Mumbai	400045
B5	Borivali	5	Mumbai	400078

- Insert a new row containing City = ‘Delhi’ in the view and compare the results with the existing table.

Source Code:

```
INSERT INTO MASTER (Branch_NO, Name, ADDR_NO, CITY, PINCODE)

VALUES ('B1','RAMAMANDI',4,'DELHI',400054);
```

Program 5. Destroy a View created on CUST_MSTR from the database.

Source Code:

```
DROP VIEW CUSTOMERS;
```

--cannot directly insert rows into a view gives ERROR—

Program 5. Destroy a View created on CUST_MSTR from the database.

Source Code:

```
DROP VIEW CUSTOMERS;
```

Program 6. Using Indexes:

1. Create a simple index on the VERI_EMP_NO column of the ACCT_MSTR table.
2. Create a composite index on the TRANS_MSTR table on columns TRANS_NO and ACCT_NO.
3. Create a unique index on the CUST_NO column of the CUST_MSTR table.
4. Create a reverse index on the CUST_NO column of the CUST_MSTR table.
5. Create a bitmap index on the TRANS_NO column of the TRANS_MSTR table.
6. Remove index idX_CustNo created for the table CUST_MSTR.
7. Retrieve the first three rows from the BRANCH_MSTR table using ROWNUM.

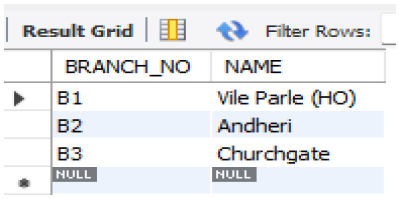
Source Code:

```
--1
CREATE INDEX idx_veri_emp_no ON ACCT_MSTR (VERI_EMP_NO);

--2
CREATE INDEX idx_trans_acct ON TRANS_MSTR (TRANS_NO, ACCT_NO);

--3
CREATE UNIQUE INDEX idx_cust_no ON CUST_MSTR (CUST_NO);

--4
CREATE INDEX idx_cust_no_reverse ON CUST_MSTR (CUST_NO DESC);

--5
CREATE BITMAP INDEX idx_trans_no ON TRANS_MSTR (TRANS_NO);

--6
DROP INDEX idx_cust_no ON CUST_MSTR;
```

--7

```
SELECT *
FROM BRANCH_MSTR
LIMIT 3;
```

OUTPUT:



Result Grid | Filter Rows:

	BRANCH_NO	NAME
▶	B1	Vile Parle (HO)
	B2	Andheri
	B3	Churchgate
*	NULL	NULL

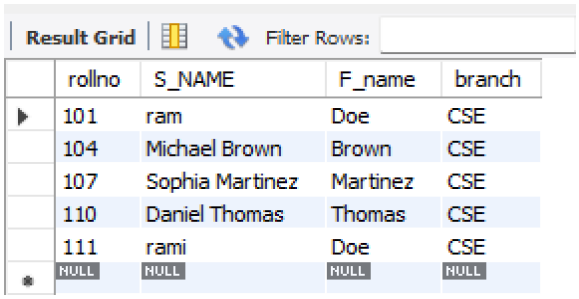
Lab Assignment 8

Program 1. List the roll no and name of all the students of branch cse.

Source Code:

```
select *from student

where  branch='cse';
```

OUTPUT:



Result Grid | Filter Rows:

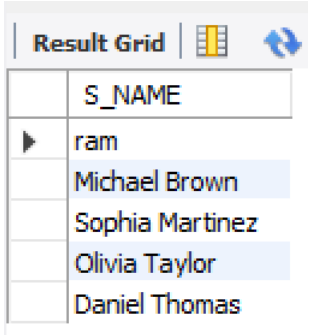
rollno	S_NAME	F_name	branch
101	ram	Doe	CSE
104	Michael Brown	Brown	CSE
107	Sophia Martinez	Martinez	CSE
110	Daniel Thomas	Thomas	CSE
111	rami	Doe	CSE
NULL	NULL	NULL	NULL

Program 2. Find the name of student who issued a book published by publisher abc.

Source Code:

```
SELECT student.S_NAME

FROM  student

INNER JOIN issue ON student.rollno = issue.rollno

INNER JOIN book ON issue.ISBN = book.ISBN

WHERE book.publisher='ABC';
```

OUTPUT:



Result Grid

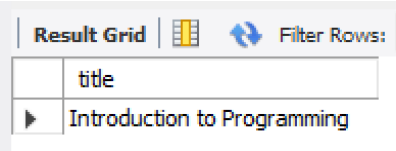
S_NAME
ram
Michael Brown
Sophia Martinez
Olivia Taylor
Daniel Thomas

Program 3. List the tittle and author of books issued to student ‘RAM’.

Source Code:

```
SELECT book.title
FROM book
INNER JOIN issue ON issue.ISBN = book.ISBN
INNER JOIN student ON issue.rollno =student.rollno
WHERE student.S_NAME='ram';
```

OUTPUT:

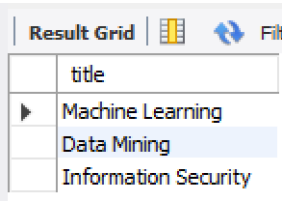


Program 4. List the tittle of books issued on or before December 1,2020 .

Source Code:

```
SELECT book.title
FROM book
INNER JOIN issue ON issue.ISBN = book.ISBN
WHERE issue.date_of_issue <='2020-12-01';
```

OUTPUT:

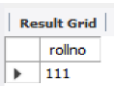


Program 5. List the name of students who had do not issued any book from Library.

Source Code:

```
SELECT student.rollno
FROM student
left JOIN issue ON issue.rollno= student.rollno
WHERE issue.rollno is null;
```

OUTPUT:



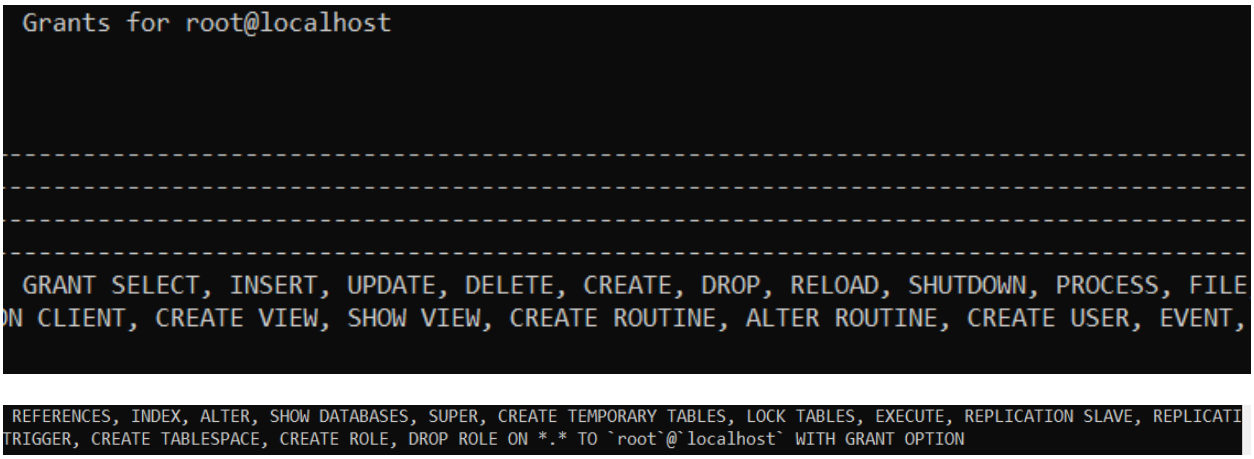
Lab Assignment 9

Program 1: Log in to MySQL and see all the privileges for the root.

Source Code:

```
SHOW GRANTS FOR 'root'@'localhost';
```

Output:



Program 2: a) Create a new user named Cust_Acct and perform the following queries:

● Show current user and the current databases which user can access.

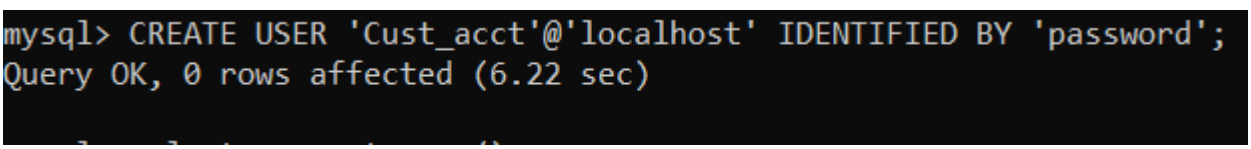
● Login into the User Account (Cust_Acct) and check for the current user and the databases accessible by the Cust_Acct.

● Switch back to the root account and Grant all privileges on BRANCH_MSTR and EMP_MSTR of Bank Database to Cust_Acct.

Source Code:

```
CREATE USER 'Cust_acct'@'localhost' IDENTIFIED BY 'password';
```

Output:



Source Code:

mysql -u Cust_Acct –p

password

select user();

show databases;

Output:

```
mysql> select user();
+--------------------+
| user()             |
+--------------------+
| Cust_Acc@localhost |
+--------------------+
1 row in set (0.00 sec)
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
+--------------------+
2 rows in set (0.00 sec)
```

(b) Insert a new row into the table.

● Display all the data from the table.

● Delete the newly inserted rows from the table and show if they are removed or not.

● Show the results produced on those tables.

Source Code:

insert into book (ISBN,title,author,publisher) values(910,'coa','morris','aaa');

Output:

ISBN	title	author	publisher
908	dbms	micheal	abc
907	ML	john	jkl
906	DAA	robb	pqr
905	MPMC	steve	xyz

ISBN	title	author	publisher
908	dbms	micheal	abc
907	ML	john	jkl
906	DAA	robb	pqr
905	MPMC	steve	xyz
910	coa	morris	aaa

Source Code:

Delete from book where ISBN= 910;

Output:

ISBN	title	author	publisher
908	dbms	micheal	abc
907	ML	john	jkl
906	DAA	robb	pqr
905	MPMC	steve	xyz

Program 3: ● Revoke the permissions/privileges of inserting and deleting data from the table

Source Code:

REVOKE INSERT, UPDATE, DELETE ON library.* FROM 'root'@'localhost';

Output:

```
mysql> REVOKE INSERT, UPDATE, DELETE ON library.* FROM 'root'@'localhost';
ERROR 1141 (42000): There is no such grant defined for user 'root' on host 'localhost'
mysql>
```

Program 4: Create a role named ‘Manager’ on table EMP_MSTR which is capable of displaying ,inserting, and deleting the data from the table.

Source Code:

CREATE ROLE `group`;

GRANT SELECT, INSERT, DELETE ON library.book TO `group`;

SHOW GRANTS FOR `group`;

Output:

```
mysql> CREATE ROLE `group`;
Query OK, 0 rows affected (0.45 sec)

mysql> GRANT SELECT, INSERT, DELETE ON library.book TO `group`;
Query OK, 0 rows affected (0.23 sec)

mysql> SHOW GRANTS FOR `group`;
+-----------------------------------------------------------------------+
| Grants for group@%                                                    |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `group`@`%`                                     |
| GRANT SELECT, INSERT, DELETE ON `library`.`book` TO `group`@`%`       |
+-----------------------------------------------------------------------+
```

Assignment 10

Program 1: Practice all the 4 programs given in presentation.

Source Code:

```
BEGIN
dbms_output.put_line('Hello');
END;
```

Output:

```
Hello

Statement processed.


0.02 seconds
```

```
DECLARE
a number;
c number;
BEGIN
a:= 10;
c:= a + a;
dbms_output.put_line(c);
END;
```

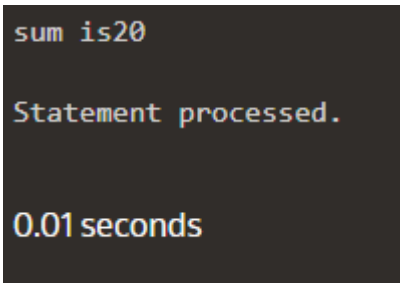
Output:

```
20

Statement processed.


0.02 seconds
```

Source Code:

```
Declare
a number;
c number;
begin
a:= 10;
c:= a + a;
dbms_output.put_line('sum is' || c);
end;
```

Output:



Source Code:

```
DECLARE
a NUMBER;
b NUMBER;
BEGIN
a:= :a; --this will take input from user
b:= :b;
DBMS_OUTPUT.PUT_LINE('a = '|| a);
DBMS_OUTPUT.PUT_LINE('b = '|| b);
END;
```

Output:

```
a = 18
b = 15

Statement processed.


0.01 seconds
```

Program 2: WAP to find the sum of two numbers.

Source Code:

```
DECLARE
   a NUMBER;
   b NUMBER;
   c NUMBER; -- Corrected data type declaration for variable c
BEGIN
   a := :a; -- this will take input from user
   b := :b;
   c := a + b; -- corrected assignment operator
   DBMS_OUTPUT.PUT_LINE('sum a and b is = ' || c);
END;
```

Output:

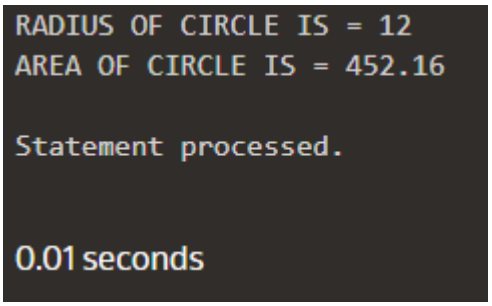
```
sum a and b is = 33

Statement processed.


0.01 seconds
```

Program 3: WAP to find the area of circle, get radius as an input from user.

Source Code:

```
DECLARE
  R NUMBER;
  ANS NUMBER;
BEGIN
  R := :R;
  ANS := 3.14 * R * R;
  DBMS_OUTPUT.PUT_LINE('RADIUS OF CIRCLE IS = ' || R);
  DBMS_OUTPUT.PUT_LINE('AREA OF CIRCLE IS = ' || ANS);
END;
```

Output:



Program 4: WAP to perform various arithmetic operations in one program.

Source Code:

```
DECLARE
  a NUMBER;
  b NUMBER;
  ADD_RESULT NUMBER;
  SUBTRACT_RESULT NUMBER;
  MUL_RESULT NUMBER;
 DIV_RESULT NUMBER;
  MOD_RESULT NUMBER;
BEGIN
  a := :a;
```

```
 b := :b;
 ADD_RESULT := a + b;
 SUBTRACT_RESULT := a - b;
 MUL_RESULT := a * b;
 DIV_RESULT := a / b;
 MOD_RESULT := MOD(a, b); -- Corrected modulus calculation
  DBMS_OUTPUT.PUT_LINE('FIRST NUMBER IS = ' || a);
   DBMS_OUTPUT.PUT_LINE('SECOND NUMBER IS = ' || b);
 DBMS_OUTPUT.PUT_LINE('SUM OF TWO NUMBERS = ' || ADD_RESULT);
 DBMS_OUTPUT.PUT_LINE('SUBTRACTION OF TWO NUMBERS = ' || SUBTRACT_RESULT);
 DBMS_OUTPUT.PUT_LINE('MULTIPLICATION OF TWO NUMBER = ' || MUL_RESULT);
 DBMS_OUTPUT.PUT_LINE('DIVISION OF TWO  NUMBER = ' || DIV_RESULT);
 DBMS_OUTPUT.PUT_LINE('MODULUS OF TWO NUMBER = ' || MOD_RESULT);
END;
```

Output:

```
FIRST NUMBER IS = 30
SECOND NUMBER IS = 10
SUM OF TWO NUMBERS = 40
SUBTRACTION OF TWO NUMBERS = 20
MULTIPLICATION OF TWO NUMBER = 300
DIVISION OF TWO  NUMBER = 3
MODULUS OF TWO NUMBER = 0
```

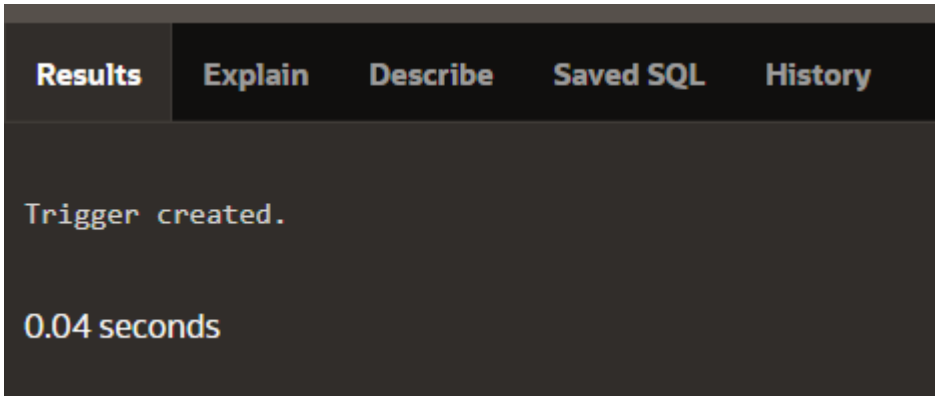
LAB-11

1.Create a trigger to display the salary difference between the old values and new values.

CODE:

```
CREATE TRIGGER trg_SalaryDifference
AFTER UPDATE OF Salary ON Employees
FOR EACH ROW
BEGIN
  -- Display the difference in salary
  DBMS_OUTPUT.PUT_LINE(
    'Salary difference: ' || TO_CHAR(:NEW.Salary - :OLD.Salary)
  );
END;
```

OUTPUT:

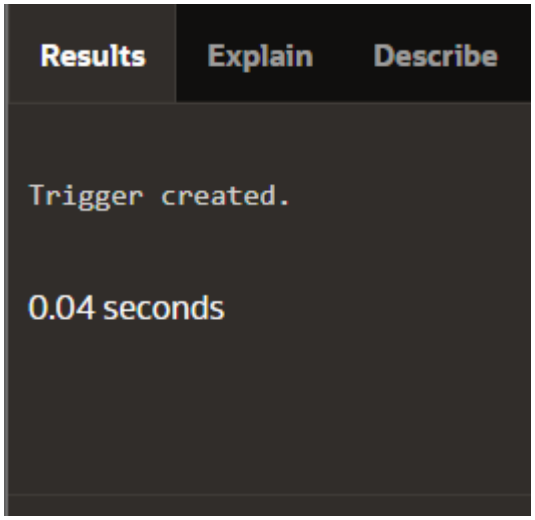


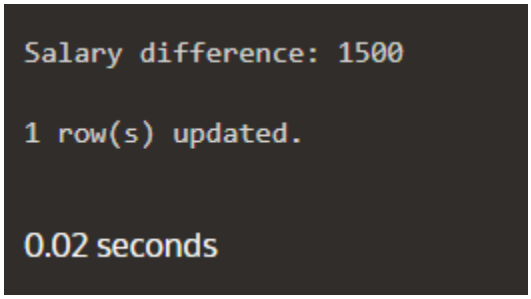
2. Create a trigger to display the increase in salary by 1500.

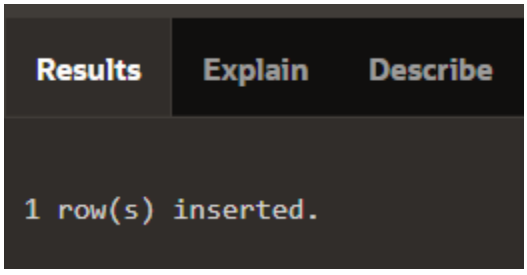
CODE:

```
CREATE TRIGGER trg_SalaryIncrease1500
AFTER UPDATE OF Salary ON Employees
FOR EACH ROW
BEGIN
  IF :NEW.Salary - :OLD.Salary = 1500 THEN
    DBMS_OUTPUT.PUT_LINE('Salary increased by exactly 1500');
  END IF;
END;
```

OUTPUT:







3. The price of a product changes constantly. It is important to maintain the history of the prices of the products. Create a trigger to update the ‘product_price_history’ table when the price of product is updated in the product table.

(Create a trigger to display the old values of the products whenever update statement runs.)

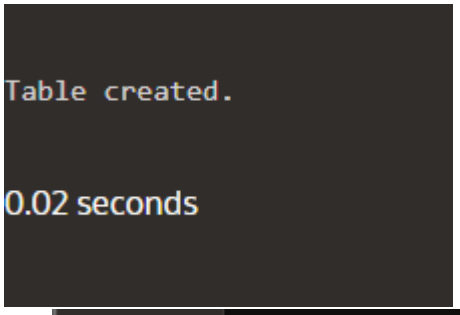
Database –

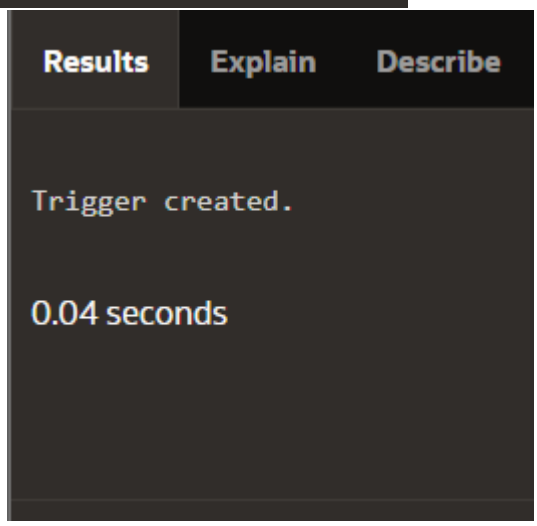
A. product_price_history (product_id, product_name, supplier_name, unit_price)

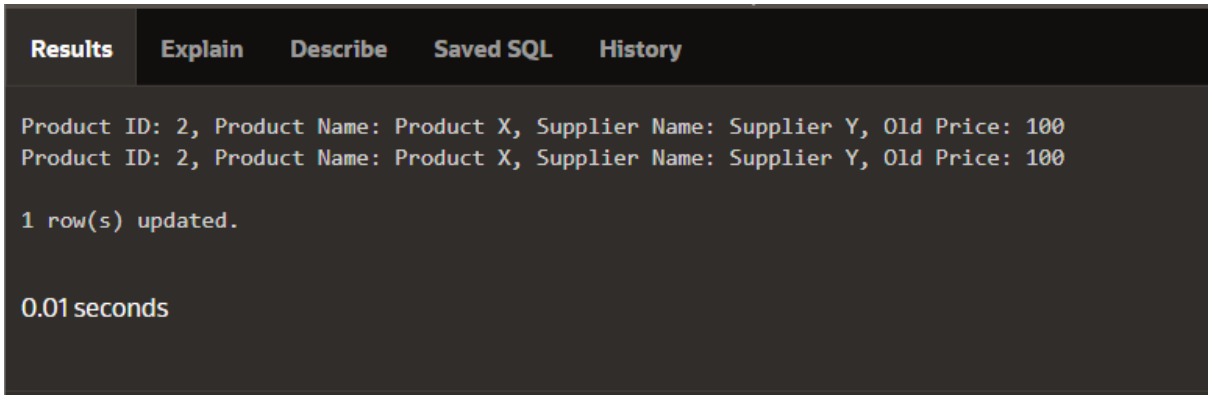
CODE:

```
CREATE TABLE product_price_history (
  product_id INT,
  product_name VARCHAR(50),
  supplier_name VARCHAR(50),
  unit_price DECIMAL(10, 2),
  change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TRIGGER trg_PriceHistory
BEFORE UPDATE OF unit_price ON product
FOR EACH ROW
BEGIN
  -- Insert the old product price details into the history table
  INSERT INTO product_price_history (product_id, product_name, supplier_name, unit_price, change_date)
  VALUES (:OLD.product_id, :OLD.product_name, :OLD.supplier_name, :OLD.unit_price, CURRENT_TIMESTAMP);
END;
```

OUTPUT:







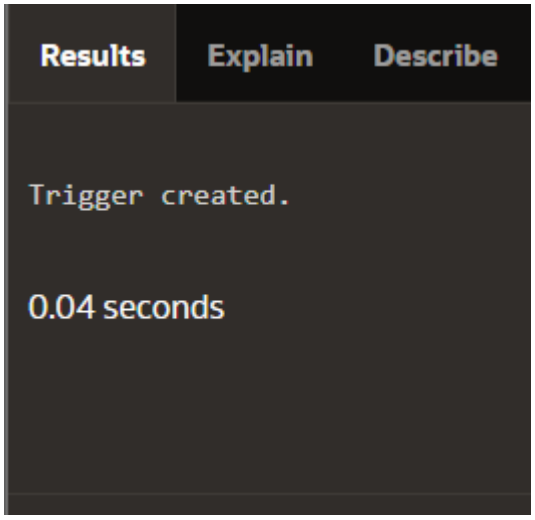
B. product (product_id, product_name, supplier_name, unit_price)

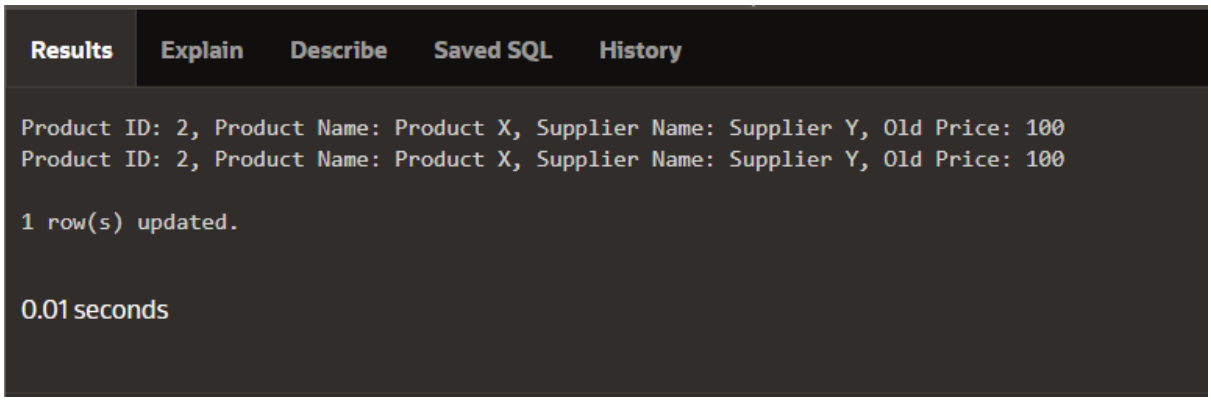
CODE:

```
CREATE TRIGGER trg_DisplayOldProductValues
BEFORE UPDATE ON product
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE(
    'Old Product Info - ID: ' || :OLD.product_id ||
```

```
    ', Name: ' || :OLD.product_name ||
    ', Supplier: ' || :OLD.supplier_name ||
    ', Price: ' || TO_CHAR(:OLD.unit_price)
  );
END;
```

OUTPUT:





4. Create a table ‘product’ which can use to store message when triggers are fired.

CODE:

```
CREATE TABLE trigger_messages (
  message_id INT PRIMARY KEY,
  message_text VARCHAR(255),
  trigger_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- Trigger to store a message in 'product' table when an update occurs
CREATE OR REPLACE TRIGGER trg_StoreTriggerMessage
AFTER UPDATE ON employees  -- This trigger is activated after an update on the 'employees' table
FOR EACH ROW
BEGIN
  INSERT INTO product (message_id, message_text, trigger_time)  -- Insert into the 'product' table
  VALUES (
    product_seq.NEXTVAL,  -- Ensure you have a sequence for auto-incrementing message_id
    'Employee ID: ' || :NEW.employee_id || ' was updated',  -- Example message
    CURRENT_TIMESTAMP  -- Record the current timestamp
  );
END;
```

OUTPUT:

Not working