Relational Database Management Systems

LAB MANUAL

Department
Of
Computer Engineering

Neotech Faculty of Diploma Engineering

Name: - _____

Branch: - _____ **Division: -** _____

Roll No: - _____ **Year: -** _____

NEOTECH FACULTY OF DIPLOMA ENGINEERING
Relational Database Management Systems(4330702)

Neotech Faculty Of

Diploma Engineering

Virod, Vadodara.

CERTIFICATE

This is to certify that

Mr./Ms. Enrollment No._____ of 3nd Semester Diploma course

In _____ has satisfactorily completed his/her termwork

In _____with in four walls of institute during the year 2022.

Place:

Date:

Subject Teacher Head of Department

Subject: Relational Database Management Systems
Subject Code : 4330702

LIST OF EXPERIMENTS

Sr. NO.	TITLE OF EXPERIMENT		Date	Sign
1.	Implement SQL Queries to perform various DDL Commands (Create minimum 5 Tables with different data type and operate upon them).			
2.	a) Implement SQL queries to perform various DML Commands (Insert minimum 10 rows using different insert methods , edit and remove data using update and delete commands). b) Retrieve data using SELECT command and various SQL operators.			
3.	Perform Queries for TCL and DCL Commands			
4.	Implement SQL queries using Data-Functions like add-month, month-between, round, nextday, truncate etc.			
5.	Implement SQL queries using Data functions like abs, ceil, power, mod, round, trunc, sqrt etc. and Character Functions like initcap, lower, upper, ltrim, rtrim, replace, substring, instr etc.			
6.	Implement SQL queries using Conversion Functions like to-char, to-date, to-number, and group Function like Avg, Min, Max, Sum, Count, Decode, etc.			
7.	Implement SQL queries using Group by, Having, and Order by clause.			
8.	Implement SQL queries using simple Case Operations and using Group Functions and Case Operations for getting summary data.			
9.	Implement SQL queries using set operators like Union, Unionall, Intersect, Minus etc.			
10.	Retrieve data spread across various table or same table using various joins.			

11.	Retrieve data from multiple tables using Subqueries (multiple Correlated).			
12.	Perform Queries to Create, alter and update view.			
13.	Implement practical-1 again with Domain Integrity, Entity Integrity and referential Constraints.			
14.	Perform Queries to create synonyms, sequence and index.			
15.	Implement PL/SQL program using control structure.			
16.	Implement PL/SQL program using cursors.			
17.	Implement PL/SQL program using exception handling.			
18.	Implement user defined procedure and functions using PL/SQL blocks.			
19.	Perform various operations on packages.			
20.	Implement various triggers.			
21.	Draw E-R diagram of the given problem statements.			
22.	Practices on Normalization – using any database perform various forms.			

Practical-1

Aim : Implement SQL Queries to perform various DDL Commands (Create minimum 5 Tables with different data type and operate upon them)

i. Table for student marks

```
CREATE TABLE `practical2`.`studentMarks`
(
        Stdname      CHAR(10) NULL,
        Sub1         CHAR(10) NULL,
        Sub2         CHAR(10) NULL,
        Sub3         CHAR(10) NULL
);
```

➔ **output**

	Time	Action	Message
1	21:00:19	CREATE TABLE `practical2`.`studentMarks` (Stdname CHAR(10) NULL, Sub1 CHAR(10) NUL...	0 row(s) affected

ii. Table for customer id

```
CREATE TABLE practical2.customer
(
        cid          varchar(3),
        name         char(10)
);
```

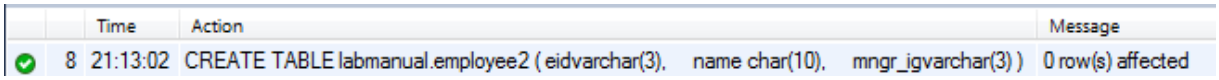
➔ **output**

	Time	Action	Message
4	21:04:56	CREATE TABLE labmanual.customer (cidvarchar(3), name char(10))	0 row(s) affected

iii. Table for Employee id

```
CREATE TABLE labmanual.employee2
(
        eid                varchar(3),
  name             char(10),
  mngr_ig          varchar(3)
);
```

➔ **output**

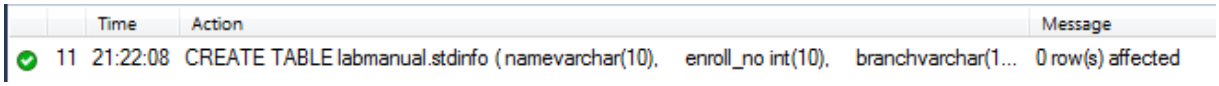


	Time	Action	Message
8	21:13:02	CREATE TABLE labmanual.employee2 (eidvarchar(3), name char(10), mngr_igvarchar(3))	0 row(s) affected

iv. Table for Employee id

```
CREATE TABLE labmanual.stdinfo
(
  name          varchar(10),
  enroll_no     int(10),
  branch        varchar(10),
  semester      int(1)
);
```

➔ **output**

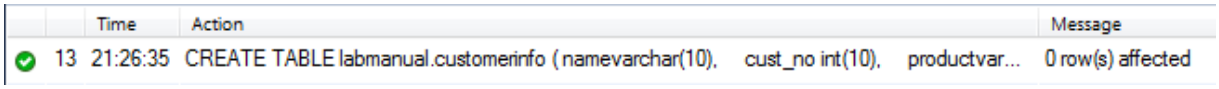


	Time	Action	Message
11	21:22:08	CREATE TABLE labmanual.stdinfo (namevarchar(10), enroll_no int(10), branchvarchar(1...	0 row(s) affected

v. Table for Employee id

```
CREATE TABLE labmanual.customerinfo
(
  name          varchar(10),
  cust_no       int(10),
  product       varchar(10),
  Quantity      int(1)
);
```

➔ **output**



	Time	Action	Message
13	21:26:35	CREATE TABLE labmanual.customerinfo (namevarchar(10), cust_no int(10), productvar...	0 row(s) affected

Practical-2

a) Implement SQL queries to perform various DML Commands (Insert minimum 10 rows using different insert methods , edit and remove data using update and delete commands)
b) Retrieve data using SELECT command and various SQL operators.

```
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('Veer',10,'computer',3);
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('Rohan',01,'computer',5);
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('Ramesh',02,'computer',3);
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('Rahul',03,'computer',1);
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('harsh',02,'computer',5);
```

➔ **Output**

	Time	Action	Message
17	21:39:34	INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('Veer',10,'comp...	1 row(s) affected
18	21:39:34	INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('Rohan',01,'co...	1 row(s) affected
19	21:39:34	INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('Ramesh',02,'co...	1 row(s) affected
20	21:39:34	INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('Rahul',03,'com...	1 row(s) affected
21	21:39:35	INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('harsh',02,'com...	1 row(s) affected

name	enroll_no	branch	semester
Veer	10	computer	3
Rohan	1	computer	5
Ramesh	2	computer	3
Rahul	3	computer	1
harsh	2	computer	5

```
DELETE FROM labmanual.stdinfo
WHERE semester = 5;
```

➔ **Output**

```
23  21:54:33  DELETE FROM labmanual.stdinfo WHERE semester = 5        2 row(s) affected
```

name	enroll_no	branch	semester
Veer	10	computer	3
Ramesh	2	computer	3
Rahul	3	computer	1

```
UPDATE labmanual.stdinfo
SET semester = 3
WHERE semester = 5;
```

➔ **Output**

```
29  22:01:52  UPDATE labmanual.stdinfo SET semester = 3 WHERE semester = 5        2 row(s) affected Rows matched: 2  Changed: 2  Warnings: 0
```

name	enroll_no	branch	semester
Veer	10	computer	3
Ramesh	2	computer	3
Rahul	3	computer	1
Rohan	1	computer	3
harsh	2	computer	3

```
DELETE FROM labmanual.stdinfo;
```

➔ **Output**

```
34  22:06:17  DELETE FROM labmanual.stdinfo        5 row(s) affected
```

name	enroll_no	branch	semester

```
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('Rahul',03,'computer',1);
INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester)
VALUES ('harsh',02,'computer',5);
```

➔ **Output**

```
39  22:12:54  INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('Rahul',03,'com...  1 row(s) affected
40  22:12:54  INSERT INTO labmanual . stdinfo(name,enroll_no,branch,semester) VALUES ('harsh',02,'com...  1 row(s) affected
```

name	enroll_no	branch	semester
Rahul	3	computer	1
harsh	2	computer	5

b) Retrieve data using SELECT command and various SQL operators.

SELECT * FROM labmanual.stdinfo;

➔ **Output**

name	enroll_no	branch	semester
Veer	10	computer	3
Rohan	1	computer	5
Ramesh	2	computer	3
Rahul	3	computer	1
harsh	2	computer	5

SELECT name , semester FROM labmanual.stdinfo;

➔ **Output**

name	semester
Veer	3
Rohan	5
Ramesh	3
Rahul	1
harsh	5

SELECT * FROM labmanual.stdinfo
WHERE name = 'veer';

➔ Output

name	enroll_no	branch	semester
Veer	10	computer	3

SELECT * FROM labmanual.stdinfo
ORDER BY enroll_no;

➔ Output

name	enroll_no	branch	semester
Rohan	1	computer	5
Ramesh	2	computer	3
harsh	2	computer	5
Rahul	3	computer	1
Veer	10	computer	3

SELECT DISTINCT semester FROM labmanual.stdinfo;

➔ Output

semester
3
5
1

Practical-3

Aim : Perform Queries for TCL and DCL Commands

➔ **TCL Commands**

COMMIT;

```
45  19:51:30  COMMIT                                    0 row(s) affected
```

ROLLBACK;

```
46  19:56:30  ROLLBACK                                  0 row(s) affected
```

SAVEPOINT majama;

```
52  19:58:39  SAVEPOINT majama                          0 row(s) affected
```

➔ **DCL Commands**

```
GRANT     ALL
ON        veer
TO        'user1'@'localhost';
```

```
53  20:01:29  GRANT ALL ONveer TO'user1'@'localhost'    0 row(s) affected
```

```
REVOKE    ALL
ON        veer
FROM      'user1'@'localhost';
```

```
54  20:04:09  REVOKE ALL ONveer FROM'user1'@'localhost'  0 row(s) affected
```

Practical-4

AIM : Implement SQL queries using Data-Functions like add-month, month-between, round, nextday, truncate etc.

➔ **SELECT sysdate()'present date' , adddate(sysdate(),3) FROM DUAL;**

	present date	adddate(sysdate(),3)
▶	2023-01-11 23:02:41	2023-01-14 23:02:41

➔ **SELECT timestampdiff(MONTH,sysdate(),'2023-09-28');**

	timestampdiff(MONTH,sysdate(),'2023-09-28')
▶	8

➔ **SELECT round(sysdate(),'DD - MON - YYYY ') FROM DUAL;**

	round(sysdate(),'DD - MON - YYYY ')
▶	20230111231215

➔ **SELECT truncate(sysdate(),'DD-MON-YYYY HH:MI:SS PM') FROM DUAL;**

	truncate(sysdate(),'DD-MON-YYYY HH:MI:SS PM')
▶	20230111231734

Practical-5

Aim : Implement SQL queries using Data functions like abs, ceil, power, mod, round, trunc, sqrt etc. and Character Functions like initcap, lower, upper, ltrim, rtrim, replace, substring, instr etc.

➔ **Data Functions**

➔ **SELECT ABS(-25) FROM DUAL;**

ABS(-25)
25

➔ **SELECT ceil(25.2),CEIL(25.7),CEIL(-25.2) FROM DUAL;**

ceil(25.2)	CEIL(25.7)	CEIL(-25.2)
26	26	-25

➔ **SELECT POWER(2,2) FROM DUAL;**

POWER(2,2)
4

➔ **SELECT MOD(5,3), MOD(2,4) FROM DUAL;**

MOD(5,3)	MOD(2,4)
2	2

➔ **SELECT round(2134.567,2),round(1234.56789),round(12345.6789,-3),round(157.732,-2) FROM DUAL;**

round(2134.567,2)	round(1234.56789)	round(12345.6789,-3)	round(157.732,-2)
2134.57	1235	12000	200

➔ **SELECT TRUNCATE(12345.6789,3),TRUNCATE(12345.6789,0),TRUNCATE(12345.6789,-2) FROM DUAL;**

TRUNCATE(12345.6789,3)	TRUNCATE(12345.6789,0)	TRUNCATE(12345.6789,-2)
12345.678	12345	12300

➔ **SELECT SQRT(25) FROM DUAL;**

	SQRT(25)
▶	5

➔ Character Functions

➔ **SELECT lower('HELLO hOw yOu dOiNg...?') FROM DUAL;**

	lower('HELLO hOw yOu dOiNg...?')
▶	hello how you doing...?

➔ **SELECT UPPER('HELLO hOw yOu dOiNg...?') FROM DUAL;**

	UPPER('HELLO hOw yOu dOiNg...?')
▶	HELLO HOW YOU DOING...?

➔ **SELECT replace('MAJAMA_123456','123456','SHANTI') FROM DUAL;**

	replace('MAJAMA_123456','123456','SHANTI')
▶	MAJAMA_SHANTI

➔ **SELECT substr('HELLO hOw yOu dOiNg...?',7,3) FROM DUAL;**

	substr('HELLO hOw yOu dOiNg...?',7,3)
▶	hOw

➔ **SELECT instr('majama','j') FROM DUAL;**

	instr('majama','j')
▶	3

Practical-6

Aim : Implement SQL queries using Conversion Functions like to-char, to-date, to-number, and group Function like Avg, Min, Max, Sum, Count, Decode, etc.

➔ Conversion Function

➔ TO NUMBER

```
SELECT TO_NUMBER('12345') FROM dual;
```

OUTPUT: 12345

➔ TO CHARACTER

```
SELECT TO_CHAR (123456,'09,99,999') FROM dual;
```

OUTPUT: 1,23,456

➔ TO DATE

```
SELECT TO_DATE ('31 DECEMBER 2012'.'DD MONTH YYYY) FROM dual;
```

OUTPUT: 31-DEC-12

➔ Group Function

➔ **SELECT MAX(salary)"max salary" FROM practical2.employee;**

	max salary
▶	25000

➔ **SELECT AVG(salary)"AVG salary" FROM practical2.employee;**

	AVG salary
▶	18333.3333

➔ **SELECT MIN(salary)"MIN salary" FROM practical2.employee;**

	MIN salary
▶	12000

➔ **SELECT SUM(salary),SUM(distinct salary) FROM practical2.employee;**

	SUM(salary)	SUM(distinct salary)
▶	110000	90000

➔ **SELECT COUNT(*) FROM practical2.employee;**

	COUNT(*)
▶	6

Practical-7

AIM : Implement SQL queries using Group by, Having, and Order by clause.

➔ Group by

➔ **SELECT**
Stdname,sum(sub1)'total_marks1',sum(sub2)'total_marks2',sum(sub3)'total_marks3' FROM practical1 . studentmarks1
GROUP BY Stdname;

Stdname	total_marks1	total_marks2	total_marks3
VEER	196	192	192
name2	47.5	98	93
Maanan	95	182	186
RISIKESH	47.5	96	97
Govind	95	184	184
Aayush	95	186	182
name3	46.5	15.3333333	97
name1	46	15.6666667	91

➔ Having

➔ **SELECT**
Stdname,sum(sub1)'total_marks1',sum(sub2)'total_marks2',sum(sub3)'total_marks3' FROM practical1 . studentmarks1
GROUP BY Stdname
HAVING Stdname = 'VEER';

Stdname	total_marks1	total_marks2	total_marks3
VEER	196	192	192

➔ Order by

➔ **SELECT * FROM labmanual.stdinfo**
ORDER BY semester;

name	enroll_no	branch	semester
Rahul	3	computer	1
Veer	10	computer	3
Ramesh	2	computer	3
Rohan	1	computer	5
harsh	2	computer	5

Practical-8

Aim : Implement SQL queries using simple Case Operations and using Group Functions and Case Operations for getting summary data.

SELECT bname,sum(balance)'Total balance' FROM practical2.account

GROUP BY bname;

Output :

BNAME	Total balance
Anand	7000
Ksad	14000
Vvn	11000

Practical-9

Aim : Implement SQL queries using set operators like Union, Unionall, Intersect, Minus etc.

➔ **SELECT name FROM practical2.customer**
UNION
SELECT name FROM practical2.employee2;

name
Riya
Jiya
Diya
Taral
Saral
Palak
Zalak
Falak

Practical-10

Aim : Retrieve data spread across various table or same table using various joins.

❖ Cross Join

➔ **SELECT * FROM practical2.customer,practical2.employee1;**

cid	name	eid	ename	bdate	salary	city
C01	Riya	E01	tulsi	2005-01-20	12000	Ahmedabad
C02	Jiya	E01	tulsi	2005-01-20	12000	Ahmedabad
C03	Diya	E01	tulsi	2005-01-20	12000	Ahmedabad
C04	Taral	E01	tulsi	2005-01-20	12000	Ahmedabad
C05	Saral	E01	tulsi	2005-01-20	12000	Ahmedabad
C01	Riya	E02	Gopi	1983-07-15	15000	Anand
C02	Jiya	E02	Gopi	1983-07-15	15000	Anand
C03	Diya	E02	Gopi	1983-07-15	15000	Anand
C04	Taral	E02	Gopi	1983-07-15	15000	Anand
C05	Saral	E02	Gopi	1983-07-15	15000	Anand
C01	Riya	E03	Rajsh...	1984-10-31	20000	Vadodara
C02	Jiya	E03	Rajsh...	1984-10-31	20000	Vadodara
C03	Diya	E03	Rajsh...	1984-10-31	20000	Vadodara
C04	Taral	E03	Rajsh...	1984-10-31	20000	Vadodara
C05	Saral	E03	Rajsh...	1984-10-31	20000	Vadodara
C01	Riya	E04	Vaishali	1985-03-23	25000	Surat
C02	Jiya	E04	Vaishali	1985-03-23	25000	Surat
C03	Diya	E04	Vaishali	1985-03-23	25000	Surat
C04	Taral	E04	Vaishali	1985-03-23	25000	Surat
C05	Saral	E04	Vaishali	1985-03-23	25000	Surat
C01	Riya	E05	Laxmi	1983-02-14	18000	Anand
C02	Jiya	E05	Laxmi	1983-02-14	18000	Anand
C03	Diya	E05	Laxmi	1983-02-14	18000	Anand
C04	Taral	E05	Laxmi	1983-02-14	18000	Anand
C05	Saral	E05	Laxmi	1983-02-14	18000	Anand
C01	Riya	E06	Shivali	1984-08-05	20000	NULL
C02	Jiya	E06	Shivali	1984-08-05	20000	NULL
C03	Diya	E06	Shivali	1984-08-05	20000	NULL
C04	Taral	E06	Shivali	1984-08-05	20000	NULL
C05	Saral	E06	Shivali	1984-08-05	20000	NULL

❖ Self Join

➔ **SELECT Emp.eid, Emp.name'EmpName', Mngr.name'MngrName' FROM practical2.employee2 Emp, practical2.employee2 Mngr**
WHERE Emp.mngr_ig = Mngr.eid;

eid	EmpName	MngrName
E01	Palak	Zalak
E03	Falak	Zalak
E04	Taral	Zalak
E05	Saral	Zalak

❖ Outer Join

➔ **SELECT id ,College.name 'NAME',department, hostel_name,room_no**
FROM college,hostel
WHERE college.name = hostel.name;

id	NAME	department	hostel_name	room_no
S02	Anisha	computer	Kaveri Hostel	K01
S03	Nisha	IT	Godavari Hostel	G07

Practical-11

Aim : Retrieve data from multiple tables using Subqueries (multiple Correlated)

➔ **select balance from practical2.account**
where ano IN (
select ano from account_holder
where cid = 'C01'
);

balance
5000

➔ **select balance from practical2.account**
where ano IN (
select ano from practical2.account_holder
where cid IN (
select cid from practical2.customer
where name = 'jiya'
)
);

balance
6000
8000

➔ **select ano , balance , bname from practical2.account acc**
where balance IN(
select max(balance) from practical2.account
where bname = acc.bname
);

ano	balance	bname
A03	7000	anand
A04	8000	ksad
A05	6000	vvn

Practical-12

Aim : Perform Queries to Create, alter and update view

- **Create view**

➔ **CREATE VIEW Acc_vvn**
AS SELECT * FROM practical2.account
WHERE bname = 'vvn';

```
24  13:55:54  CREATE VIEW Acc_vvn AS SELECT * FROM practical2.account WHERE bname = 'vvn'  0 row(s) affected
```

- **Alter view**

➔ **ALTER VIEW acc_vvn**
AS SELECT * FROM practical2.account
WHERE bname = 'ksad';

```
26  14:09:16  ALTER VIEW acc_vvn AS SELECT * FROM practical2.account WHERE bname = 'ksad'  0 row(s) affected
```

- **Update view**

➔ **UPDATE acc_vvn**
SET bname = 'vvn';

```
32  14:22:32  UPDATE acc_vvn SET bname = 'vvn'  2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0
```

Practical-13

Aim : Implement practical-1 again with Domain Integrity, Entity Integrity and referential Constraints

- **Domain Integrity**

➔
```
CREATE TABLE practical2.account2
(
        ano     char(3),
  balance       int(9) NOT NULL,
  bname         varchar(10)
);
```

```
33  15:30:14  CREATE TABLE practical2.account2 ( ano char(3),   balanceint(9) NOT NULL,   bnamevar...   0 row(s) affected
```

- **Entity Integrity**

➔
```
CREATE TABLE practical2.account3
(
        ano     char(3) UNIQUE,
  balance       int(9) ,
  bname         varchar(10)
);
```

```
34  15:48:09  CREATE TABLE practical2.account3 ( ano char(3) UNIQUE,   balanceint(9) ,   bnamevarch...   0 row(s) affected
```

- **Referential Integrity**

➔
```
CREATE TABLE practical2.account5
(
        ano     char(3) primary key,
  balance       int(9) ,
  bname         varchar(10)
  REFERENCES Branch (bname)
);
```

```
45  16:06:25  CREATE TABLE practical2.account5 ( ano char(3) primary key,   balanceint(9) ,   bnamevar...   0 row(s) affected
```

Practical-14

Aim : Perform Queries to create synonyms, sequence and index

❖ Synonym

➔

```
CREATE    SYNONYM    Cust
FOR    user1.Customer;
Synonym Created.
```

❖ Sequence

➔

```
CREATE    SEQUENCE    mySequence
START WITH        1
INCREMENT BY      1
MINVALUE          1
MAXVALUE          99
NOCYCLE ;
Sequence Created.
```

❖ Index

➔ **CREATE INDEX indcustname**
ON practical2.account(ano);

```
47  16:22:27  CREATE INDEX indcustname ON practical2.account(ano)    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

Practical-15

Aim : Implement PL/SQL program using control structure.

➔

```
DECLARE
        no    NUMBER;                    -- declare a variable to store number.
BEGIN
        -- read a number from the user...
        no := &no;
        -- check the result of the MOD function...
        IF    MOD   (no,2)   =   0   THEN
          dbms_output.put_line ('Given number '|| no || ' is EVEN.');
        ELSE
          dbms_output.put_line ('Given number '|| no || ' is ODD.');
        END IF;
END ;
```

```
Enter value for no: 7
old    5: no := &no;
new    5: no := 7;
Given number 7 is ODD.
```

Practical-16

Aim : Implement PL/SQL program using cursors.

➔

Input :

```
DECLARE
        "_ _" declare required variables...
                branch    Account.bname%TYPE;

BEGIN
        "_ _" read a number from the user...
                branch := &branch;
        "_ _" modify branch name ...
                UPDATE      Account     SET     bname = UPPER (branch)
                WHERE       bname = branch;
        "_ _" display the number of records updated, if any.
                IF          SQL%FOUND   THEN
                            dbms_output.put_line ('Total ' || SQL%ROWCOUNT ||'
                                    records are updated...');

                ELSE
                        dbms_output.put_line ('Given branch not available...');
                END IF;

END ;
/
```

Output 1 : Enter value for branch: **'surat'**

old 6: branch := &branch;

new 6: branch := 'surat';

Given branch not available...

Practical-17

Aim : Implement PL/SQL program using exception handling.

➔

```
                                    ... the required table is available. If not so, first create it.)
DECLARE
    -- declare required variables...
        no      Account.ano%TYPE;          -- Anchored data type...
        bal     Account.balance%TYPE;
        branch  Account.bname%TYPE;
BEGIN
    -- read an account number, balance and branch name for a new record...
        no      := &no;
        bal     := &bal;
        branch  := &branch;
    -- insert a record into the Account table ...
        INSERT  INTO  Account  VALUES (no, bal, branch);
    -- commit and display a message confirming insertion...
        COMMIT;
        dbms_output.put_line ('Record inserted successfully...');
EXCEPTION
    -- handle named exception...
        WHEN  DUP_VAL_ON_INDEX  THEN
          dbms_output.put_line ('Duplicate value found for Primary key.');
END ;
```

Enter value for no: 'A01'
Enter value for no: 5000
Enter value for no: 'vvn'
Record inserted successfully...

Practical-18

Aim : Implement user defined procedure and functions using PL/SQL blocks.

➔ ***Input :***

```
CREATE    OR    REPLACE    PROCEDURE    debitAcc
        -- specify parameters...
        (       no              IN      Account.ano%TYPE,
                amount      IN      NUMBER
        )
IS
        -- declare local variables...
                bal             Account.balance%TYPE;
                newBalance      Account.balance%TYPE;
BEGIN
        -- retrieve the current balance for the given account...
                SELECT balance INTO bal    FROM Account
                WHERE ano = no;
        -- calculate new balance...
                newBalance := bal - amount;
        -- update balance, without worrying about negative balance...
                UPDATE Account SET balance = newBalance
                WHERE ano = no;
        -- display a message confirming the update...
                dbms_output.put_line ('Account ' || no || ' debited...');
END ;
/
```

Output : Procedure created.

➔ **Executing a procedure**

debitAcc('A01',1000);

output : Account A01 debited

➔ **Creating function**

```
CREATE OR REPLACE FUNCTION getBalance
    ( no IN Account.ano%TYPE )
    RETURN NUMBER
    IS "– –" declare local variables...
        bal Account.balance%TYPE;
BEGIN
        "– –" retrieve the current balance for the given account...
            SELECT balance INTO bal FROM Account
            WHERE ano = no;
        "– –" return balance...
            RETURN bal;
END ;
```

➔ **Destroying Procedure and Function**

DROP PROCEDURE debitAcc;

Output: Procedure drop

DROP FUNCTION getbalance;

Output: Function drop

Practical-19

Aim : Perform various operations on packages.

➔ **Creating Package**

Input :

```
CREATE OR REPLACE PACKAGE BODY transaction
IS
-- define procedure 'debitAcc'
    PROCEDURE debitAcc (no IN Account.ano%TYPE, amount IN NUMBER )
IS
    bal          Account.balance%TYPE;
    newBalance   Account.balance%TYPE;
BEGIN
    SELECT balance INTO bal FROM Account
    WHERE ano = no;
    newBalance := bal - amount;
    UPDATE Account SET balance = newBalance
    WHERE ano = no;
    dbms_output.put_line ('Account ' || no || ' debited...');
END;
-- define function 'getBalance'
    FUNCTION getBalance ( no IN Account.ano%TYPE ) RETURN NUMBER
    IS
        bal          Account.balance%TYPE;
    BEGIN
        SELECT balance INTO bal FROM Account
        WHERE ano = no;
        RETURN bal;
    END;
END transaction;
/
```

Output : Package body created.

➔ **Referencing a package subprogram**

SELECT transaction.getBalance('A01') FROM dual;

Output : transaction.getBalance('A01')
4000

➔ **Destorying package**

DROP PACKAGE tranction;

Output : package dropped

Practical-20

Aim : Implement various triggers.

➔ **Creating Trigger**

```
CREATE OR REPLACE TRIGGER balNegative
    BEFORE INSERT
ON Account
FOR EACH ROW
BEGIN
      IF :NEW.balance < 0 THEN
            dbms_output.put_line ('Balance is negative...');
      END IF;
END;
/
```

Output : Trigger created.

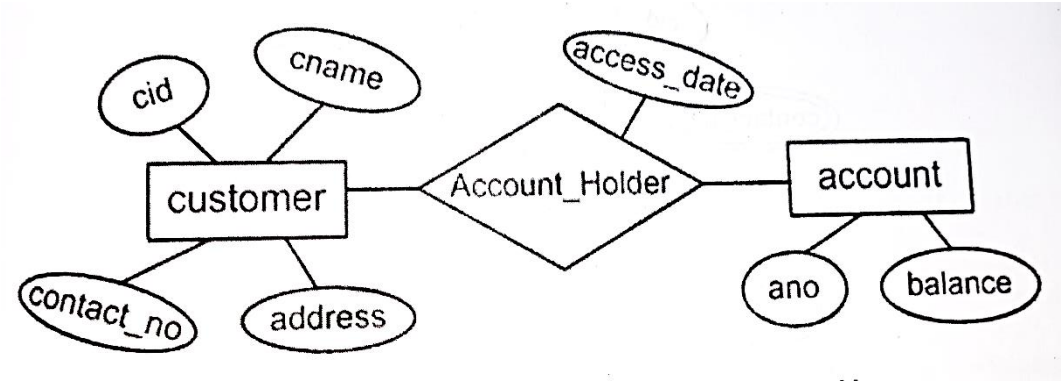
➔ **Destroying trigger**

DROP TRIGGER balNegative;

Output : Trigger dropped

Practical-21

Aim : Draw E-R diagram of the given problem statements.



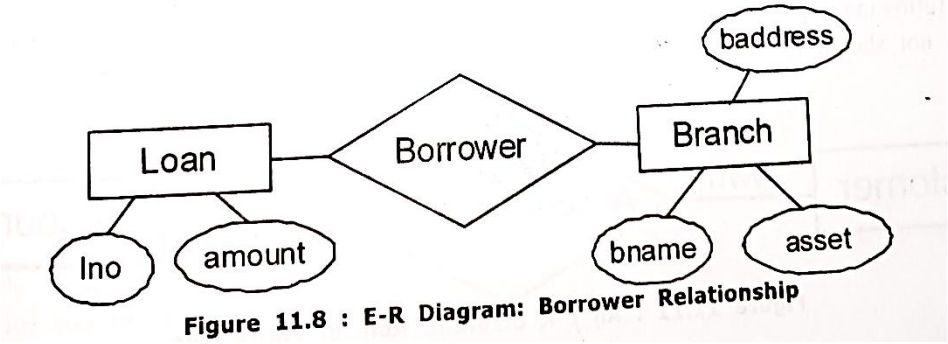


Figure 11.8 : E-R Diagram: Borrower Relationship

Practical-22

Aim : Practices on Normalization – using any database perform various forms.

Customer :

cid	name	address		contact_no
		society	city	
C01	Riya	Amul Aavas, Anand		{9876543210}
C02	Jiya	Sardar Colony, Karamsad		{232740, 25356178}
C03	Piya	Marutisadan, VVNagar		{55414,55415,55416}
C04	Diya	Saral Society, Anand		
C05	Tiya	Birla Gruh, VVNagar		{9825098250}

Customer :

cid	name	society	city
C01	Riya	Amul Aavas	Anand
C02	Jiya	Sardar Colony	Karamsad
C03	Piya	Marutisadan	VVNagar
C04	Diya	Saral Society	Anand
C05	Tiya	Birla Gruh	VVNagar

Customer_Contact :

cid	contact_no
C01	9876543210
C02	232740
C02	25356178
C03	55414
C03	55415
C03	55416
C05	9825098250

Figure 10.9 : 'Customer' relation with a solution to composite & multi-valued attribute