Ex: No: 1 **DDL COMMANDS**
Date:

AIM:
To execute a table in the database.

1. CREATE :

Purpose:
To create a table in the database.

Syntax:
- Create table <table name> (column definition);

Using primary key:
- Create table <table name>

(column 1 datatype NULL/ not NULL,column 2 datatype NULL/not NULL -----------,
Constraint constraint_name primary key column 1,column2,---(column –n));

2. DESC:

Purpose:
To display the table structure.

Syntax:
- Desc<table name>

3.ALTER:

Purpose:
To alter the structure of the table in the database.

Syntax:
- Alter table<table name> add(column definitions);
- Alter table <tablename>modify(column definitions);
- Alter table <table name> rename to<table name>;
- Alter table <table name> rename column<old column name>to<new column name>
- Alter table<table name>drop column<column name>;
- Alter table <table name>add constraint name>primary key(column 1, column2……..,column n-1);
- Alter table <table name>drop constraint<constraint name>;

4. DROP:

Purpose:
To remove the table from the database.

Syntax:

- Drop table<table name>
-

5.TRUNCATE:

Purpose:
To remove the entire content of the table from the database.

Syntax:

- Truncate tanle<table name>

EXAMPLE QUERIES:

1. Create a table called employee with followingattributes.

i. **Employee no**
ii. **Employee name**
iii. **Department number**

SQL>create table employee(eno number(3),ename varchar2(20),dno number(3));

OUTPUT:
Table created.

2. Create a table called supplier with following attributes.

i. **Supplier_id**
ii. **Supplier_name**
iii. **Contact_number**
iv. **Set supplier_id as a primary key**

SQL>Create table supplier(supplier_id numeric(10),not NULL,supplier_name varchar2(50)not NULL, contact_no number(10), constraint supplier_PK PRIMARY KEY(supplier_id));

OUTPUT:
Table created.

3. Command to give the structure of the employee table.

```
SQL> desc employee.
```

OUTPUT:

```
        NAME                               NULL? TYPE
---------------------------- -----------------------------------------------
     ENO                                  NUMBER(3)
     ENAME                                VARCHAR2(20)
     DNO                                  NUMBER(3)
```

4. Command to modify a field called employee name in employee table.

```
SQL>alter table employee modify(ename varchar2(30));
```

OUTPUT:

```
        Table altered.
```

5. Command to view the change employee table structure.

```
SQL>desc employee.
```

OUTPUT:

```
        NAME                                     NULL? Type
---------------------------------- ---------------------------------------------
     ENO                                        NUMBER(3)
     ENAME                                      VARCHAR2(30)
     DNO                                        NUMBER(3)
```

6. Command to add a field called salary in employee table.

```
SQL>alter table employee add(sal number(12,2));
```

OUTPUT:

```
    Table created .
SQL>desc employee;


            NAME                               NULL? TYPE
   ----------------------------------- --------------------------------------
        ENO                                    NUMBER(3)
        ENAME                                  VARCHAR2(30)
        DNO                                    NUMBER(3)
        SAL                                    NUMBER(12,2)
        DNAME                                  VARCHAR2(20)
```

8. Command to modify the data type of the field ENO.

```
SQL>alter table employee modify(eno number(5));
```

OUTPUT:

```
Table created.
SQL>desc employee;

                  NAME                                    NULL? TYPE
---------------------------------------- ----------------------------------------
             ENO                                         NUMBER(5)
             ENAME                                  VARCHAR2(30)
             DNO                                         NUMBER(3)
             SAL                                         NUMBER(12,2)
```

9. Command to rename the column name

```
SQL>alter table employee rename column ename to empname.
OUTPUT:
        Table altered.
SQL>desc employee;

             NAME                                    NULL? TYPE
---------------------------------------- ----------------------------------------
             ENO                                        NUMBER(5)
             EMPNAME                                    VARCHAR2(30)
             DNO                                       NUMBER(3)
             SAL                                        NUMBER(12,2)
```

10. Command to drop the employee table

```
SQL>drop table employee.
```

OUTPUT:

```
         Table droped.
```

Result:

Thus the DDL commands are executed.

Ex:No: 2 **DML COMMANDS**
Date:

Aim:
To execute DML commands.

01. INSERT:

Purpose:
To add records to the table
Syntax:
1.Direct method-only one record can be inserted in the field at a time

- Insert into<table name>values<values for all columns>

2.Null method-we can skip some field

- Insert into<table name>(column name)values(values for columns)

3.Macro method-More than one value can be inserted in the field at a time.

- Insert into<table name>values<&column names>

02. SELECT:

Purpose:
To reterieve or view records with in the table
Syntax:

- Select * from<table name>
- Select *from <table name>where(condition)
- Select (column name)from<table name>

03. UPDATE:

Purpose:
To modify records with in the table

Syntax:

- Update<table name>set(column name)=(value)
- Update<table name>set(column name)=(value)where(condition)

04. DELETE:

Purpose:
To modify records from a table
Syntax:

- Delete from<table name>
- Delete from<table name>where(condition)

<u>Example queries:</u>

01. Command to insert records into employee table.

```
SQL >.insert into employee values ( &eno,  ’&ename’ ,&dno ,&sal,    ‘&dname’);

Enter value for eno:1
Enter value for ename:D.Abinaya
Enter the value for dno:111
Enter the value sal:8000
Enter the value for dname:IT
Old 1:insert into employee values(&eno,’&ename’,&dno,&sal,’&dname’
New 1:insert into employee values(1,’D.Abinaya’,111,8000,’IT’)
```

Output:

```
        1 row created
```

02. To execute the command which is in buffer

```
  SQL>/
Enter value for eno:2
Enter value for ename: P.Ratha
Enter value for dno:111
Enter value for sal:8900
Enter value for dname:IT
Old 1: insert into employee values (&eno,’&ename’,&dno,&sal,’&dname’);
New 1: insert into employee values (2,’P.Ratha’,111,8900,’IT’)
```

Output:

```
      1 row created
```

03. Display all records from employee table

```
     SQL>select * from employee;
```

Output:

```
 ENO  ENAME              DNO    SAL       DNAME
--------------------------------------------------------
   1   D.Abinaya         111    8000        IT
   2   P.Ratha           111    8900        IT
   3   D.Geetha          222    9000        CSE
   4   K.lalitha         222    8000        CSE
   5   L.priya           333    10000       ECE
   6   K.Jaya            333    9000        ECE
   7   M.Sasi            444    8000        EEE
   8   N.Raja            444    9400        EEE
   9   G.Ganga           555    8000        MECH
  10   P.Manoj           555    9000        MECH
```

```
10 rows selected.
```

04. Find out the names of all the employees

```
SQL>select ename from employee;
```

Output:

```
ENAME
---------
D.Abinaya
P.Ratha
D.Geetha
K.Lalitha
L.Priya
K.Jaya
M.Sasi
N.Raja
G.Ganga
P.Manoj
10 rows selected
```

05. Display enmae,dno from employee where sal>9000.

```
SQL>Select ename,dno from employee where sal>9000;
```

Output:

```
ENAME                DNO
- -------- ------- -------- --------
L.Priya              333
```

Result:

Thus DML Queries were executed.

Ex:No: 3 **BASIC SELECT STATEMENT**
Date:

Aim:
To execute Basic Select statement.

Example queries:

1. Write command to list name, emp_id and sal of employee whose salary>15000.
   SQL>Select name, eno, sal from employee where (salary >15000);

   OUTPUT:

name	eno	sal
john	4	69000
mark	6	18008
p.manoj	2	19000
n.raja	10	25000
k.jaya	6	23000

2. Write command to list the emp_no and name of managers.
   SQL>Select eno, name from employee where (des= 'manager');
   OUTPUT:

employee_number	fname	lname	mname
122	john	martieni	sam
111	mark	rain	jem
333	mark	raj	jim

3. Write command to list employees who work in accounts department.
   SQL>Select * from employee where (dept = 'accounts');

   OUTPUT:

empno	fname	lname	mname	dob	dept	salary	dept
122	john	martieni	sam	20-05-17	account	25000	cse
111	mark	john	jem	15-06-11	account	35000	cse
333	mark	rain	jim	15-06-11	account	35000	cse

4. Write command to list employees whose eno = 3,5,7.
   SQL>Select * from employee where (eno = 3 and eno = 5 and eno = 7);

   OUTPUT:

name	eno	sal
john	3	69000
mark	5	18008
p.manoj	7	19000

5. Write command to list employee who do not belong to dept CSE.
   SQL>Select * from employee where (dept != ‘cse’);

   OUTPUT:

empno	fname	lname	mname	dob	salary	dept
122	john	martieni	sam	20-may-27	25000	ECE
111	mark	john	jem	15-jun-11	35000	ECE
333	mark	rain	jim	15-jun-11	35000	ECE
234	abinaya	D		15-jun-11	45000	IT
212	MARY	S	SELVAN	27-SEP-12	45000	IT

6. Write command to list employee joined before 30 Jun 17 and after 31 Dec 17.
   SQL>Select * from employee where DOJ not between 30-Jun-17 and 31-Dec-17;

   OUTPUT:

empno	fname	lname	mname	dob	salary	dept
122	john	martieni	sam	20-may-27	25000	ECE
111	mark	john	jem	15-jun-11	35000	ECE
333	mark	rain	jim	15-jun-11	35000	ECE
234	abinaya	D		15-jun-11	45000	IT
212	MARY	S	SELVAN	27-SEP-12	45000	IT

7. Write command to list employee whose salary <5000 & commission is null.
   SQL>Select * from employee where (salary<5000 and commission != 0);

   OUTPUT:

employee_number	fname	lname	mname	dob	dept	salary	department	Comm
122	john	martieni	sam	20-may-27	account	25000	cse	3000
111	mark	john	jem	15-jun-11	account	35000	cse	5000
333	mark	rain	jim	15-jun-11	account	35000	cse	2000

8. Write command to list employee whose experience > 2 years.
   SQL>Select * from employee where (exp>2);

   OUTPUT:

employee_number	fname	lname	mname	dob	Exp	salary	department
122	john	martieni	sam	20-may-15	3	25000	ECE
111	mark	john	jem	15-jun-11	7	35000	ECE
333	mark	rain	jim	15-jun-11	7	35000	ECE
234	abinaya	D		15-jun-11	7	45000	IT
212	MARY	S	SELVAN	27-SEP-12	6	45000	IT

Result:

Thus Basic Select statements were executed.

Ex:No: 4 **ADVANCED SELECT STATEMENT**
Date:

Aim:

To execute Advanced Select statement.

Example queries:

1. **Write commands to display 3rd largest salary in the employee table.**

SQL> Select max(salary) from employee where salary<(Select max(Salary) from employee where salary < (select max(salary) from employee));

OUTPUT:

max(salary)
25000

2. **Using aggregate function write query to display employee's salary who's salary is greater than average if its employees whose hire date is before 27-09-2017.**

Command: **Select** salary from employee where salary> (select avg(Salary) from employee where (hire date<'21-Feb-18'));

OUTPUT:

salary
35000
35000
45000

3. **Find the job with lowest average salary.**

SQL>Select*from employee where salary < (Select min (avg (Salary))from employee group by department);

OUTPUT:

employ	employee_number	fname	lname	mname	dob	c	salary	department
1	10	bond	john	j	20-JUL-98	hr	15000	EEE

4. **Find the employees who earn salary same as the min salary of the department.**

SQL>Select**f**name from employee where salary < (select min(avg(salary))from employee group by dept);

OUTPUT:

fname
bond

5. **Command to create new table from already existing table.**

SQL> Createtable mytable as select * from employee;
OUTPUT:

employ	employee_number	fname	lname	mname	dob	s	salary	department

6. **Command for copying record from one table to another table.**

SQL>Insert into mytable select *from empl;
OUTPUT:

employ	employee_number	fname	lname	mname	dob	c	salary	department
a90000	122	john	martieni	sam	20-may-27	Hr	25000	cse
a80000	111	mark	rain	jem	15-jun-11	Hr	35000	cse
a70000	333	mark	rain	jem	15-jun-11	co	35000	cse

7. **Create table emp1 with the employees who works in hourly basis and create another table emp2 with who works in contract basis.**

SQL> create table emp1 as select * from emp where c='Hr';
create table emp2 as select * from emp where c='co';
OUTPUT: emp1

employ	employee_number	fname	lname	mname	dob	c	salary	department
a90000	122	john	martieni	sam	20-may-27	Hr	25000	cse
a80000	111	mark	rain	jem	15-jun-11	Hr	35000	cse

emp2

employ	employee_number	fname	lname	mname	dob	c	salary	department
a70000	333	mark	rain	jem	15-jun-11	co	35000	cse

8. **Display employees name from emp1 and emp2**

SQL> select fname from emp1 union select fname from emp2;

OUTPUT:

fname
john
mark
mark

9. Write command to display the employee who have same name.

SQL>select fname from emp1 intersect select fname from emp2.

OUTPUT:

fname
mark
mark

10. Write command to concatenate first name and last name.

SQL>Select concat (fname, concat(mname,lname)) from emp;

OUTPUT:

concat(fname,concat(mname,lname))
johnsammartieni
markjemrain
markjemrain

Result:

Thus Advanced Select statements were executed.

Ex No:5 **INTEGRITY AND CONSTRAINS**
Date:

Aim:

To study the various constraints available in the SQL query language.

Description:

DOMAIN INTEGRITY CONSTRAINTS

Domain integrity validates data for a column of the table and it also means the definition of a valid set of values for an attribute.

It can be enforced using:

- Foreign key constraint.
- Check constraint.
- NOT NULL.
- Default constraint.

  These definitions ensure that a specific attribute will have a right and proper value in the database.

ENTITY INTEGRITY CONSTRAINTS

Entity Integrity can be enforced through indexes, UNIQUE constraints and PRIMARY KEY constraints.

REFERENTIAL INTEGRITY CONSTRAINTS

- The referential integrity constraint is specified between two tables and it is used to maintain the consistency among rows between the two tables.
- FOREIGN KEY and CHECK constraints are used to enforce Referential Integrity.

The rules are:

1. You can't delete a record from a primary table if matching records exist in a related table.
2. You can't change a primary key value in the primary table if that record has related records.
3. You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.

4. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.

OUTPUT:
DOMAIN INTEGRITY CONSTRAINTS
NOT NULL CONSTRAINT

```
SQL> create table empl (ename varchar2(30) not null, eid varchar2(20) not null);
Table created.
SQL> insert into empl values ('abcde',11);
1  row created.
SQL> insert into empl values ('fghij',12);
1  row created.
SQL> insert into empl values ('klmno',null);
insert into empl values ('klmno',null)
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("ITA"."EMPL"."EID")
SQL> select * from empl;
ENAME                          EID
------------------------------ --------------------
abcde                          11
fghij                          12
```

CHECK AS A COLUMN CONSTRAINT

```
SQL> create table depts ( dname varchar2(30) not null, did number(20) not null check
(did<10000));
Table created.

SQL> insert into depts values ('sales ',9876);
1  row created.
SQL> insert into depts values ('marketing',5432);
1  row created.
SQL> insert into depts values ('accounts',789645);
insert into depts values ('accounts',789645)
*
ERROR at line 1:
ORA-02290: check constraint ( ITA.SYS_C003179) violated
SQL> select * from depts;
DNAME                               DID
----------------------------- --------
sales                              9876
marketing                          5432
```

CHECK AS A TABLE CONSTRAINT

```
SQL> create table airports (aname varchar2(30) not null , aid number(20) not null, acity
varchar2(30) check( acity in ('chennai','hyderabad','bangalore')));
Table created.
SQL> insert into airports values( 'abcde', 100,'chennai');
1  row created.
SQL> insert into airports values( 'fghij', 101,'hyderabad');
1  row created.
SQL> insert into airports values( 'klmno', 102,'bangalore');
1  row created.

SQL> insert into airports values( 'pqrst', 103,'mumbai');
insert into airports values( 'pqrst', 103,'mumbai')
*
ERROR at line 1:
ORA-02290: check constraint (ITA.SYS_C003187) violated
SQL> select * from airports;
ANAME                                 AID ACITY
------------------------------ --------- ------------------------------
abcde                                100     chennai
fghij                               101      hyderabad
klmno                              102      bangalore
```

ENTITY INTEGRITY CONSTRAINTS
UNIQUE AS A COLUMN CONSTRAINT

```
SQL> create table book (bname varchar2(30) not null, bid number(20) not null unique);
Table created.
SQL> insert into book values ('fairy tales',1000);
1  row created.
SQL> insert into book values ('bedtime stories',1001);
1  row created.
SQL> insert into book values ('comics',1001);
insert into book values ('comics',1001)
*
ERROR at line 1:
ORA-00001: unique constraint (ITA.SYS_C003130) violated
SQL> select * from book;
BNAME                                 BID
------------------------------ ---------
fairy tales                         1000
bedtime stories                      1001
```

UNIQUE AS A TABLE CONSTRAINT

```
SQL> create table orders( oname varchar2(30) not null , oid number(20) not null ,
unique(oname,oid));
Table created.
```

```
SQL> insert into orders values ('chair', 2005);
1  row created.
SQL> insert into orders values ('table',2006);
1  row created.
SQL> insert into orders values ('chair',2007);
1  row created.
SQL> insert into orders values ('chair', 2005);
insert into orders values ('chair', 2005)
*
ERROR at line 1:
ORA-00001: unique constraint (ITA.SYS_C003152) violated
SQL> select * from orders;
ONAME                          OID
------------------------------ --------
chair                          2005
table                          2006
chair                          2007
```

PRIMARY KEY AS A COLUMN CONSTRAINT

```
SQL> create table custo ( cname varchar2(30) not null , cid number(20) not null primary
key);
Table created.
SQL> insert into custo values ( 'jones', 506);
1  row created.
SQL> insert into custo values ('hayden',508);
1  row created.
SQL> insert into custo values ('ricky',506);
insert into custo values ('ricky',506)
*
ERROR at line 1:
ORA-00001: unique constraint (ITA.SYS_C003165) violated
SQL> select * from custo;
CNAME                          CID
 ------------------------------ --------
jones                          506
hayden                         508
```

PRIMARY KEY AS A TABLE CONSTRAINT

```
SQL> create table branches( bname varchar2(30) not null , bid number(20) not null ,
primary key(bnam e,bid));
Table created.
SQL> insert into branches values ('anna nagar', 1005);
1  row created.
SQL> insert into branches values ('adyar',1006);
1  row created.
SQL> insert into branches values ('anna nagar',1007);
1  row created.
SQL> insert into branches values ('anna nagar', 1005);
```

```
insert into branches values ('anna nagar', 1005)
*
ERROR at line 1:
ORA-00001: unique constraint (ITA.SYS_C003173) violated
SQL> select * from branches;
BNAME                          BID
------------------------------ ---------
anna nagar                    1005
adyar                         1006
anna nagar                    1007
```

REFERENTIAL INTEGRITY CONSTRAINTS

TO CREATE 'DEPTS' TABLE

```
SQL> create table depts(city varchar2(20), dno number(5) primary key);
Table created.
SQL> insert into depts values('chennai', 11);
1  row created.
SQL> insert into depts values('hyderabad', 22);
1  row created.
```

TO CREATE 'SEMP' TABLE

```
SQL> create table semp(ename varchar2(20), dno number(5) references depts(dno));
Table created.
SQL> insert into semp values('x', 11);
1  row created.
SQL> insert into semp values('y', 22);
1  row created.
SQL> select * from semp;
ENAME                DNO
-------------------- ---------
x                   11
y                   22
```

ALTER TABLE

```
SQL> alter table semp add(eddress varchar2(20));
Table altered.
SQL> update semp set eddress='10 gandhi road' where dno=11;
1  row updated.
SQL> update semp set eddress='12 m.g. road' where dno=22;
1  row updated.
SQL> select * from semp;
ENAME                  DNO   EDDRESS
-------------------- --------- -------------------
x                       11        10 gandhi road
y                       22        12 m.g. road
```

```
SQL> select city, ename from depts, s2emp where depts.dno = s2emp.dno;
CITY               ENAME
-------------------- -------------------
chennai            x
hyderabad            y
```

Example queries:

1. **Write command to create student relationship with attributes with not null constraints.**

   Command: create table emp (id int not null,name varchar(25), dept varchar(25));

   Output:

Name	Null?	Type
ID	NOT NULL	NUMBER(38)
NAME		VARCHAR2(10)
DEPT		VARCHAR2(5)

2. **Create supply table with the attributes with composite constraints.**

   Command: Createtable supply (name varchar (10) NOT NULL, id int NOT NULL, addr varchar (10) NOT NULL, primary key(name,id));

   Output:

Name	Null?	Type
NAME	NOT NULL	VARCHAR2(10)
ID	NOT NULL	NUMBER(38)
ADDR	NOT NULL	VARCHAR2(10)

3. **Alter student table with check constraints in the attribute age where age>=18.**

   Command: Createtable student (name varchar(10) not null,ageint check(age>=18));

   Output:

Name	Null?	Type
NAME	NOT NULL	VARCHAR2(10)
AGE		NUMBER(38)

4. **Create table in the name of voter's list for a part location using default constraint.**

Command:Createtable voters(numint,loc varchar(10) default ‘chennai’);
Output:

Name	Null?	Type
NUM		NUMBER(38)
LOC		VARCHAR2(10)

5. **Show the difference in the entity integrity constraint called primary & unique key.**
   Command:
   Primary key: Createtable prime(no number(10), addr varchar(10), primary key(no, addr));

Name	Null?	Type
NO	NOT NULL	NUMBER(10)
ADDR	NOT NULL	VARCHAR2(10)

Unique Key: Createtable uniquedemo(no number(10), addr varchar(10), unique(no,addr));

Name	Null?	Type
NO	NULL	NUMBER(10)
ADDR	NULL	VARCHAR2(10)

6. **Create table customer with attributes p_id, name, location & age. Similarly another table called order with the attributes o_id, o_no&p_id.**
   Command:Createtable cust(p_id number(10) primary key, name varchar(10), loc varchar(10),age number(10));

Name	Null?	Type
P_ID	NOT NULL	NUMBER(10)
NAME		VARCHAR2(10)
LOC		VARCHAR2(10)
AGE		NUMBER(10)

Command: Createtable ordr(o_id number(10), o_no number(10),p_id number(10),constraint fkforeignkey(p_id) references cust(p_id);

Name	Null?	Type
O_ID		NUMBER(10)
O_NO		NUMBER(10)

P_ID		NUMBER(10)

7. **Write query on delete cascade.**
   Command :Createtable det1(salint , no int constraint fk references cust (p_id) on delete cascade);
   Output:

Name	Null?	Type
SAL		NUMBER(10)
NO		NUMBER(10)

Result:

Thus the various constraints available were executed.

Ex: No: 6 **JOIN OPERATIONS**
Date:

Aim:
To know the use of join operator in SQL.

Join:
A join is used to combine rows from multiple tables. A join is performed whenever two or more tables is listed in the form clause of an SQL statement.

Table structure:

```
SQL>desc suppliers;
Name        NULL?        Type
------ --------- --------- -----------------
Supplier_id                number(5)
Supplier_name              varchar2(25)

SQL>desc order;

NAME                  NULL?             TYPE
----- - --------- ------------- --------- ------- ----------
Order_id                                number(6)
Supplier_id                              number(5)
Order_date                                 date

SQL>select * from suppliers;
```

Supplier:

Supid	**supname**
12	abi
13	chindu
14	nithi
15	selvi

Order:

Oid	**supid**	**odate**
111	12	1/9/09
222	23	8/9/09
333	14	6/9/09
444	25	3/9/09
555	15	17/9/09

01. Natural joins:

```
SQL>select * from supplier,ord where supplier.supid=ord.supid;
```

Output:

```
Supid          supname             Oid        supid        odate
-------- ---------- ----------- --------------- ------------- ---------- ------
12              abi                111          12         1/9/09
14              nithi              333          14         6/9/09
15              selvi              555          15          17/9/09
```

02.Outer Join:

Left outer join:

```
SQL>select supplier.suPID ,supplier.suPNAME,ord.oDATE where
supplier.suPID(+)=ord.suPID;
```

Output:

```
SuPID          SuPNAME               Odate
--------- --------- --------- --------- -------- ----------
12              abi                   1/9/09
14             nithi                 6/9/09
15             selvi                17/9/09
                                      8/9/09
                                      3/9/09
```

Right outer join:

```
SQL>select supplier.suPID ,supplier.suPNAME,ord.oID,ord.oDATE from supplier.ord
where supplier.suPID = ord.suPID(+);
```

Output:

```
SuPID          SuPNAME        oid     Odate
-------- --------- --------- --------- -------- -------------
12              abi               111        1/9/09
14             nithi              333       6/9/09
15             selvi              555        17/9/09
13             chindu
```

Full outer join:

```
SQL>select * from supplier,ord where supplier.suPID(+) = ord.suPID(+)
Union select * from supplier,ord where supplier.suPID=ord.suPID(+);
```

Output:

```
SuPID            SuPNAME          oid     supid    Odate
--------- --------- --------- --------- -------- ------------- ----
12                   abi                111     12         1/9/09
13                   chindu
14                   nithi               333     14          6/9/09
15                   selvi               555     15         17/9/09
                                        222     23          8/9/09
                                        444     25         3/9/09
```

EXAMPLE QUERIES :

1. **Create db student with attributes & another database advisor with set of attributes & come up with a relationship table in which the db contains student name with corresponding advisor.**

   SQL>**Select** * from student crossjoin advisor;
   OUTPUT:

Stu no	fna me	lna me	mna me	dob	due s	dept	empl oy	adv_ no	fnam e	lna me	mna me	dob	sal	dept
122	ante	rtie ni	saga m	20-may -27	250 00	cse	a900 00	122	john	mart ieni	sam	20-may -17	25000	cse
122	ante	rtie ni	saga m	20-may -27	250 00	cse	a800 00	111	mark	rain	jem	15-jun-11	35000	cse
122	ante	rtie ni	saga m	20-may -27	250 00	cse	a700 00	333	mark	rain	jem	15-jun-11	35000	cse
122	ante	rtie ni	saga m	20-may -27	250 00	cse	1	10	bond	john	j	20-jul-98	15000	eee

2. **Create db employee with set of attributes similarly another db called parkinglot with set of attributes, where output should have employee name who use parking lot.**

SQL>Selectname from employee park innerjoin parking on employee.park=parking;

OUTPUT:

FNAME
JOHN
MARK
bond
dejane

3. **Display the name, city & birth month of the employees with following cosiderations:**
   1. employee address contains e_name, country & city details, e_id.
   2. employee payroll contains e_id,e_dept, e_birthdate&doj.

SQL> Selectename,doj,city from emp1 innerjoin emp2 on emp1.eid=emp2.eid;

OUTPUT:

FNAME	**DOB**	**CITY**
JOHN	20-MAY-27	Chennai
MANOJ	5-JUL-10	Bangalore
MAOPI	1-JAN-12	Kolkata
MARKIL	5-MAR-13	Delhi
MARK	1-APR-11	Mumbai
bond	20-JUL-98	Rajastan
dejane	20-JUN-93	Hyderabad

4. **List emp_id& gender for all married empls& include the names of any charity to which the employee donate via the company program.**

SQL>Selecteid,gender from employee innerjoin charity on employee.char = charity.char;

OUTPUT:

EID	**GENDER**
RA01	Male
RA03	Female
RA05	Male

5. **Get employee name, project name, order by first name from employee detail & project detail for those employee who have assigned project already.**

SQL>Selectename,project_name from emp1 inner join project on emp1.eid = project.eid;

OUTPUT:

ENAME	PROJECT_NAME
Westbrook	Machine Learning
Serena	Biometrics
Wade	Big data

6. **Get employee name, project name, order by first name from employee detail & project detail for those employee who have not assigned project.**

SQL>Selectemp.name from emp join project on emp.id, project.id where original='no';

OUTPUT:

NAME
Harden
Sumitha
Wakarimashita

7. **Display emp_name, project name from emp_detail&project_detail for all employees if not project is assigned then display no project assigned.**

SQL>select emp.name,emp.pnameproject.assign from emp join project on emp.id = project.id;

OUTPUT:

NAME	PROJECT_NAME
Aikon	Not Assigned
Ovonel	Not Assigned
Durantula	Not Assigned

8. **Get all project name even they have not matching any emp_id in the left table order by first name from emp_detail&project_detail.**

SQL> Selectproject_namefrom emp1 left outer join project on emp1.eid = project.p_id;

OUTPUT :

PROJECT_NAME
Machine learning
Big data
Cache developement

9. **Write a query to fetch emp.name & project who has assigned more than one project.**
SQL>select ename,pname from emp1.join project on emp1.eid = project.eid where pname in (Select ename from emp1 group by pname having count(*)>1);

ENAME	PROJECT_NAME
Ni san	Machine Learning
Ni san	Biometrics

Result:

Thus all joins operations were executed.

Ex No: 7 **SQL FUNCTIONS**
Date :

Aim:

To execute simple query using SQL Functions.

SQL Function:

- Set of instructions which will do some specific task
- It may or may not return the value
- It's the module.

<u>Types</u>

Single row function

1. Data function
2. Numeric function or arithmetic or mathematical Function
3. Character function
4. General function or Miscellaneous function

Group function

1. max( )
2. min( )
3. sum( )
4. avg( )
5. std dev( )
6. variance( )

<u>01 .Date functions</u>

01. Write a query to increase dates by number of months specified.

```
SQL>SELECT ADD_MONTHS (SYSDATE, 2) FROM DUAL;
ADD_MONTH
-------------------
13-APR-01
```

02. Write a query it find today's date.

```
SQL>SELECT SYSDATE FROM DUAL;
SYSDATE
--------------
13-FEB-01
```

03. Write a query to find last date in a month of date specified.

```
 SQL>SELECT SYSDATE, LAST_DAY (SYSDATE) FROM DUAL;
 SYSDATE        LAST_DAY
 -----------------------------------
13-FEB-01       28-FEB-01
```

04. Write a query to find number of months between two dates.

```
SQL>SELECT MONTHS_BETWEEN (TO_DATE ('04-MAY-2001','DD-MON-
YYYY'),                         TO_DATE ('14-FEB-2001', 'DD-MON-YYYY'))
FROM DUAL;
MONTHS_BETWEEN (TO_DATE ('04-MAY-2001', 'DD-MON-YYYY'),
TO_DATE('14-FEB-2001', 'DD-MON-YYYY'))
----------------------------------------------------------------------------------------------------
------------------------
2.6774194
```

05. Write a query to find the next day in a month of date specified

```
SQL>SELECT NEXT_DAY(SYSDATE,'SUNDAY') FROM DUAL;
NEXT_DAY(
-----------------
18-FEB-01
```

06. Write a query to round off a particular date to next year

```
SQL>SELECT ROUND(SYSDATE,'YEAR')FROM DUAL;
ROUND(SYS
----------------------
01-JAN-01
```

07. Write a query to round off a particular date to next month

```
SQL>SELECT ROUND(SYSDATE,'MONTH')FROM DUAL;
ROUND(SYS
----------------------
01-FEB-01
```

08. Write a query to round off a particular date to next day.

```
SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;
ROUND(SYS
----------------------
11-FEB-01
```

09. Write a query to round off particular date to year preceding it.

```
SQL>SELECT TRUNC SYSDATE,’YEAR’) FROM DUAL;
TRUNC (SYS
---------------------
01-JAN-01
```

10. Write a query to round off particular date to month preceding it.

```
SQL>SELECT TRUNC SYSDATE,’MONTH’) FROM DUAL;
TRUNC (SYS
---------------------
01-FEB-01
```

11. Write a query to round off particular date to day preceding it.

```
SQL>SELECT TRUNC SYSDATE,’DAY’) FROM DUAL;
TRUNC (SYS
---------------------
11-FEB-01
```

12. Write a query to find greatest date

```
SQL>SELECT GREATEST(SYSDATE,TO_DATE(’11-APR-2001’, ‘DD-MON-
YYYY’))FROM          DUAL;
GREATEST(
-------------------
11-APR-01
```

13. Write a query to add 10 days from current date

```
SQL>SELECT SYSDATE+10 FROM DUAL;
SYSDATE+10
-------------------
23-FEB-01
```

<u>02. Mathematical Functions</u>

14. Write a query to find a standard deviation of field in a table

```
SQL>SELECT STDDEV (SAL) FROM EMP;
STDDEV (SAL)
---------------------
2786.874
```

15. Write a query to find a variance of field in a table

```
SQL>SELECT VARIANCE (SAL) FROM EMP;
VARIANCE (SAL)
---------------------
7766666.7
```

16. Write a query to find the sign of a number

```
SQL>SELCT SIGN(-46) FROM DUAL;
SIGN(-46)
----------------
-1
```

17. Write a query to find a absolute value of a number

```
SQL>SELECT ABS(-89) FROM DUAL
ABS(-89)
-----------------
89
```

18. Write a query to round off a number to nearest whole number.

```
SQL>SELECT CEIL(6.2) FROM DUAL;
CEIL(6.2)
-------------
7
```

19. Write a query to retain decimal part and truncate fractional part in a number

```
SQL>SELCT FLOOR(100.7) FROM DUAL;
FLOOR(100.7)
--------------------
100
```

20. Write a query to find a value of number ’m’ raised to power ‘n’.

```
SQL>SELECT POWER(2,3) FROM DUAL;
POWER(2,3)
----------------
8
```

21. Write a query to find square root of a number.

```
SQL>SEELCT SQRT(64) FROM DUAL;
SQRT(64)
------------
8
```

22. Write a query to find remainder of a number as a result of division between two numbers

```
SQL>SELECT MODE(3,2) FROM DUAL;
MODE (3,2)
---------------
1
```

23. Write a query to find exponential of a number

```
SQL> SELECT EXP(1) FROM DUAL;
EXP(1)
-----------
2.7182818
```

24. Write a query to find sine value of a number

```
SQL>SELECT SIN(90) FROM DUAL;
SIN(90)
----------
.89399666
```

25. Write a query to find cosine value of a number

```
SQL>SELECT COS(45) FROM DUAL;
COS(45)
----------
.525332199
```

26. Write a query to find tan value of a number

```
SQL>SELECT TAN(90) FROM DUAL;
TAN(90)
----------
-1.9952
```

27. Write a query to round off a number to the specified decimal places

```
SQL>SELECT ROUND(7.235,2) FROM DUAL;
ROUND(7.235,2)
---------------------
7.24
```

28. Write a query to truncate a number

```
SQL>SELECT TRUNC(7.235,2) FROM DUAL;
TRUNC(7.235,2)
----------------------
7.23
```

29. Write a query to find the exponential of a number.

```
SQL>SELECT EXP(5) FROM DUAL;
EXP(5)
-----------
148.41316
```

<u>03. Character Functions</u>

30. Write a query to accept a character as input and return as output the initial character
in upper case .

```
SQL>SELECT INITCAP(‘anantha krishnan’) from dual ;
        INITCAP(‘ANANTHA
        --------------------
        Anantha Krishnan
```

31. Write a query to accept a character as input and return as output character in lower case.

```
SQL>select lower (‘ANITHA’) from dual ;
LOWER (
----------------
anitha
```

32. Write a query to accept a character as input and output the character in upper case.

```
SQL>select upper (‘manomani’) from dual ;
UPPER(‘MA
------------
MANOMANI
```

33. Write a query to truncate specified number of characters to left from specified string.

```
SQL>select ltrim(‘IloveIndia’,’India’) from dual ;
 LTRIM(‘IL
---------
IloveIndia
```

34. Write a query to truncate specified number of characters to right from specified string.

```
SQL>select rtrim('IloveIndia','India') from dual ;
RTRIM
-------
Ilove
```

35. Write a query to pad characters on left side of a string.

```
SQL>select Ipad('abc',5,'*') from dual ;
LPAD(
-------
**abc
```

36. Write a query to pad characters on right side of a string.

```
SQL>select rpad('abc'5,'$') from dual ;
RPAD(
-------
abc$$
```

37. Write a query to replace a particular letter in a string by a particular character.

```
SQL>select translate ('jack','j','b') from dual ;
TRAN
 ------
Back
```

38. Write a query to extract specified numbers of characters and replace them with new characters
From a specified string.

```
SQL>select replace('jack and jue','j''bl') from dual ;
REPLACE("JACKA
-------------------
Black and blue
```

39. Write a query to extract specified numbers of characters from a specified string.

```
SQL>select substr('sivapriyakumar',5,5) from dual ;
SUBST
 -------
Priya
```

40. Write a query to get the specified numbers of characters in a specified string.

```
SQL>select instr(‘abcdef’,’cd’) from dual ;
        INSTR(‘ABCDEF’ ,’CD’)
        -----------------------
              3
```

41. Write a query to convert number to character.

```
     SQL>select chr(101) from dual ;
     C
     -
      E
```

42. Write a query to convert character to number.

```
       SQL>select ascii(‘e’) from dual ;
        ASCII(‘E’)
        -----------
          101
```

<u>04.Miscellaneous functions</u>

43. Write a query to view the user id.

```
     SQL>select uid from dual ;
     UID
      ----------
      20
```

44. Write a query to view the user name.

```
     SQL>select user from dual;
     USER
     -------------
     SCOTT
```

45. Write a query to view the vertical size of the string.

```
    SQL>select vsize(‘devarajalexander’) from dual ;
     VSIZE(‘DEVARAJALEXANDER’)
      -------------------------------
                 16
```

<u>05.Group functions</u>

46. Write a query to count distinct number of records in a table.

```
SQL>select count(distinct eno) from employee1;
       COUNT(DISTINCTENO)
       ------------------------
                 7
```

47. Write a query to find the average value of an item in a column of data.

```
      SQL>select avg(sal) from employee1;
      AVG(SAL)
      ------------
      13999.875
```

48. Write a query to find sum of value of an item in a column of data .

```
        SQL>select sum(sal) from employee1;
        SUM(SAL)
        ---------------
        111999
```

49. Write a query to find the minimum value of an item in a column of data.

```
      SQL>select min(sal) from employee 1;
      MIN(SAL)
       ------------------
        8000
```

50. Write a query to find the maximum value of an item in a column of data.

```
       SQL>select max(sal) from employee 1;
       MAX(SAL)
          ------------
                 50000
```

Result:

Thus all the SQL Function queries were executed.

Ex: No: 8 **SUB QUERIES**
Date:

Aim:

To execute the Sub queries

Sub Queries:

- Query within a query is sub query
- The result will be based on innermost query

Syntax:query(query)

<u>Types:</u>

1.Single row subqueries(using single row operators like <,<=,>,>=)

2.Multiple row subqueries(using multiple row operators like in,all,any)

<u>Single row Subqueries:</u>

Display the colic's of miller

SQL>select name from emp where deptno=(select deptno from emp where name='miller');

Find 3rd most salary

SQL>select max(sal)from emp where sal<(select max(sal)from emp where sal<(select max(sal)from emp));

Using aggregating function subquery whose sal is greater than avg of its employees whose hiredate is before 1.4.81

SQL>select *from emp where sal>(select avg(sal)from emp where doj<'01-apr-81');

<u>Subqueries and Having</u>

Find the job with the lowest average salary

SQL>select job,avg(sal)from emp group by job having avg(sal)=(select min(avg(sal))from emp group by job);

Find the employee number whose salary is lower than the highest average salary

SQL>select no,avg(sal)from emp group by no having avg(sal)<(select max(avg(sal))from emp group by no);

<u>Distinct clause with subqueries</u>

To avoid single row subqueries that return more than 1 row we use distinct clause with subqueries.

List the name of the employee who are in their dept

SQL>select name from emp where deptno=(select distinct deptno from dept where deptno=emp.deptno);

<u>Sub Queries that return more than one rule</u>

Subquery returned more than one value.This is illegal when the subquery follows =,!=,<=,>=,<,>to rectify instead of ‘=’ place ‘in’.

List the names of employees who earn lowest salary in each dept

SQL>select name,sal,deptno from emp where sal in(select min(sal)from emp group by deptno);

<u>Correlated SubQuery</u>

A query which uses values from the outer query is called as correlated subquery.

SQL>select no ,deptno,name,sal from emp where sal<(select max(sal)from emp e where no=e.no);

<u>Exists Operator</u>

It checks the existence of a result of a subquery

SQL>select *from emp where exists(select *from dept where no=3 and emp.deptno=customer.deptno);

Multiple Row Subqueries

In→ equal to any member in the list

Any→compare value to each value returned by the subquery.

Any means greater than atleast one value.ie,greater than the minimum>any(1,2,3) means greater than 1

All→compare value to every value returned by the subquery

All means greater than every value.ie,greater than the maximum>any(1,2,3) means greater than 3

Find the employees who earn the same as the min sal for dept

```
SQL>select *from emp where sal in(select min(sal) from emp group by deptno);
SQL> select name from emp where sal<any(select sal from emp where deptno=30);
SQL> select name from emp where sal>all(select sal from emp where deptno=30);
```

Table structure:

```
SQL>desc emp_det;

Name                Null?        Type
------------ -------------------- ---------------
ENO                 NOT NULL     NUMBER(3)
ENAME                            VARCHAR2(25)
ADDRESS                          VARCHAR2(30)
BASIC_SALARY                     NUMBER(12,2)
JOB_STATUS                       VARCHAR2(15)
DNO                              NUMBER(3)

SQL>desc pro_det;

Name                Null?        Type
------------ -------------------- ---------------
PNO                 NOT NULL     NUMBER(3)
PNAME                            VARCHAR2(30)
NOS_OF_STAFF                     NUMBER(3)

SQL>desc work_in;

Name                Null?        Type
------------ -------------------- ---------------
PNO                                   NUMBER(3)
ENO                                   NUMBER(3)
PJOB                                  CHAR(12)
SQL>select * from emp_det;
```

Output:
```
ENO ENAME    ADDRESS     BASIC_SALARY JOB_STATUS   DNO
-------- ------------- -------------- ---------------- ------------ ---------- -----------
1       abi           erode              8000                  manager          10
2      deepak       erode               8500                  manager          10
3      anjali        raj street          10000               assistant         2
4      geetha        abc nagar          7800                 professor        3
5      shirley        kk nagar           7888                 assistant         3
6      kiruthi        kovai                10000               professor        2
7      chindu        mmnagar           7800                  professor        2
```

7 rows selected

SQL>select * from pro_det;

Output:

```
PNO           PNAME            NOS_OF_STAFF
------------- --------------- -----------------------------
1               DBMS               3
2               COMPILER         2
3                 C                  3
```

3 rows selected

SQL>select * from work_in;

Output:

```
PNO    ENO       PJOB
1           1        programmer
2           1        analyst
1           2         analyst
2           2         programmer
```

4 rows selected

Example Queries:

01.Find the names of all employees who do work in department where geetha is working.

```
SQL>select ename from emp_det where dno not in (select dno from emp_det where
ename = ‘geetha’);
```

Output:

```
ENAME
------ ------
Abi
Deepak
Anjali
Kiruthi
Chindu

4 rows selected
```

02. Find names of employees who are working in the same department with Shirley.

```
SQL>select ename,dno from emp_det where dno=(select dno from emp_det where
ename=’shirley’)order by ename;
```

Output:

```
ENAME          DNO
--------------- ---------------
Geetha           3
Shirley          3
 2 rows selected
```

03. Find the names of employees who are working in DBMS project.

```
SQL>select ename from emp_det where eno in(select eno from work_in where
pno=(select pno from pro_det where pname=’DBMS’))order by ename
```

Output:

```
ENAME
---- -------
Abi
Deepak

2 rows selected
```

04. Find names and basic salary of those employees of the department with dno2 who get more salary the highest paid employee of the department with dno 10.

```
SQL>select ename,basic_salary from emp_det where dno = 2 and basic_salary > (select max(basic_salary)from emp_det where dno=10)order by ename;
```

Output:

```
ENAME                    BASIC_SALARY
---------------- ------------------ ---------------- --------------------------
Anjali                     10000
Kiruthi                    10000

2 rows selected
```

05. Find name,job_status,basic_salary of the employees who are worked in chindu's department and also get the same salary.

```
SQL>select ename, job_status,basic_salary from emp_det
where(dno,basic_salary)in(select dno,basic_salary from emp_det where
ename='chindu');
```

Output:

```
ENAME              JOB_STATUS      BASIC_SALARY
---------------- ------------------ ---------------- --------------------------
Chindu             professor            7800
1 row selected
```

06. Find the names of all projects in which employees are working.

```
SQL>select pno,pname from pro_det where exists(select pno from work_in where
work_in.pno=pro_det.pno)
```

Output:

```
PNO   PNAME
    1     DBMS
    2   COMPILER
2 rows selected
```

07. List the employees who draw highest salary.

```
SQL>select * from emp_det where basic_salary=(select max(basic_salary)from
emp_det);
```

Output:

```
ENO   ENAME   ADDRESS     BASIC_SALARY      JOB_STAUS  DNO
--------- ---------- ------------ ------------ ----------- ------- ------------ -------------
3      anjali        raj street           10000              assistant          2
6      kiruthi        kovai                 10000              professor         2
2 rows selected
```

08. List the second maximum salary.

```
SQL>select max(basic_salary)from emp_det where basic_salary < (select
max(basic_salary)from emp_det);
```

Output:

```
      MAX(BASIC_SALARY)
      ----------- ----------------------
      8500
```

09. List the salary where basic is less than the average salary.

```
SQL>select * from emp_det where basic_salary <(select avg(basic_salary)from
emp_det);
```

Output:

```
ENO  ENAME    ADDRESS     BASIC_SALARY  JOB_STATUS   DNO
-------- ------------- -------------- ---------------- ------------ ---------- -----------
1       abi            erode               8000                  manager            10
2      deepak      erode               8500                  manager           10
4      geetha        abc nagar          7800                  professor          3
5      shirley        kk nagar           7888                  assistant           3
7      chindu        mmnagar          7800                  professor           2
```

Result:

Thus Sub queries were executed.

Ex: No: 9 **VIEWS**
Date:

Aim:

To perform operations on views.

Views:

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depend on the written SQL query to create a view.

Views, which are a type of virtual tables, allow users to do the following −

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

CREATING VIEWS

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view. To create a view, a user must have the appropriate system privilege according to the specific implementation.

SYNTAX

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

CHECK OPTION

SYNTAX

```
CREATE VIEW CUSTOMERS_VIEW AS
SELECT column1, column2.....
FROM  table_name
WHERE [condition];
WITH CHECK OPTION;
```

UPDATING A VIEW

A view can be updated under certain conditions which are given below −

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.

- The SELECT clause may not contain set functions.
- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The FROM clause may not contain multiple tables.
- The WHERE clause may not contain subqueries.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.
- All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

INSERTING ROWS INTO A VIEW

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

DELETING ROWS INTO A VIEW

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

DROPPING VIEWS

Drop the view if it is no longer needed.

SYNTAX

drop view view_name;

<u>Example Queries:</u>

1. **Create a view for salesman who belong to the city Chennai.**

   Command : create view salman as select * from salesman where(city='chennai');

   OUTPUT :

employ	employee_number	fname	lname	mname	dob	s	salary	Locaton
1	10	bond	john	j	20-jul-98	m	15000	chennai

2. **Query to find the salesman of the city Chennai who achieves the commission more than 13%.**

   Command : create view saleman as select name from salesman where (city='chennai' and commission>='13%');

   OUTPUT:

Name
bond

3. **Query to create view to get count of how customers we have at each level of a grade.**

   Command: create view cg(grade,count) as select grade, count(grade) from cust group by grade order by grade;

   OUTPUT:

GRADE	NUMBER
200	2
300	2

4. **Query to create view to keep track of no of customers ordering average amount of orders & total amount of order in a day.**

Command: Createview masters as select count(name) as count , avg(orders) as average,sum(orders) as sum from cust;

OUTPUT :

```
ord_date    | count |       avg                | sum
 2012-04-25 |   1   | 3045.6000000000000000  | 3045.60
 2012-06-27 |   1   |  250.4500000000000000  |  250.45
 2012-07-27 |   1   | 2400.6000000000000000  | 2400.60
 2012-08-17 |   3   |   95.2633333333333333  |  285.79
 2012-09-10 |   3   | 2326.3833333333333333  | 6979.15
```

Result:

Thus various Views were executed.

Ex: No: 10 **BASIC PL/SQL**
Date:

Aim:

To write PL/SQL programs using procedures and functions.

Description:

PL/SQL Programming:

- ➢ Procedural Language/Structured Query Language (PL/SQL) is an extension of SQL.
- ➢ PL/SQL is a block-structured language, meaning that PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts:

- **DECLARE-** This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.
- **EXECUTABLE COMMANDS**- This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.
- **EXCEPTION HANDLING**- This section starts with the keyword EXCEPTION. This section is again optional and contains exception(s) that handle errors in the program.

Basic Syntax of PL/SQL

```
DECLARE
/* Variables can be declared here */
BEGIN
/* Executable statements can be written here */
EXCEPTION
/* Error handlers can be written here. */
END;
```

STEPS TO WRITE & EXECUTE PL/SQL

- **As we want output of PL/SQL Program on screen, before Starting writing anything type (Only Once per session)**

```
SQL> SET SERVEROUTPUT ON
```

- **To write program, use Notepad through Oracle using ED command.**

```
SQL> ED ProName
```

- **Type the program Save & Exit.**
- **To Run the program**

```
SQL> @ProName
```

Decision Making With If Statement :-

The general syntax for the using IF—ELSE statement is

```
IF(TEST_CONDITION) THEN
SET OF STATEMENTS
ELSE
SET OF STATEMENTS
END IF;
```

For Nested IF—ELSE Statement we can use IF--ELSIF—ELSE as follows

```
IF(TEST_CONDITION) THEN
SET OF STATEMENTS
ELSIF (CONDITION)
SET OF STATEMENTS
END IF;
```

LOOPING STATEMENTS:-

- For executing the set of statements repeatedly we can use loops. The oracle supports number of looping statements like GOTO, FOR, WHILE & LOOP.
- Here is the syntax of these all the types of looping statements.

GOTO STATEMENTS

```
<<LABEL>>
SET OF STATEMENTS
GOTO LABEL;
```

FOR LOOP

```
FOR <VAR> IN [REVERSE] <INI_VALUE>..<END_VALUE>
SET OF STATEMENTS
END LOOP;
```

WHILE LOOP

```
WHILE (CONDITION) LOOP
SET OF STATEMENTS
END LOOP;
```

LOOP STATEMENT

```
LOOP
SET OF STATEMENTS
IF (CONDITION) THEN
EXIT
SET OF STATEMENTS
END LOOP;
```

While using LOOP statement, we have to take care of EXIT conditions; otherwise it may go into infinite loop.

<u>01.Procedures</u>

<u>Syntax:</u>

CREATE[OR REPLACE]PROCEDURE procedure_name

[(Parameter [,parameter])]

IS

[declaration_section]

BEGIN

executable_section

[EXCEPTION

exception _section]

END [procedure_name];

Problems:

01. Write a procedure for executing a loop.

```
SQL>Create or replace procedure pro1(n number) is
            begin
            for i in 1…n
            loop
            dbms_output.put_line(i);
end loop;
end;
```

Output:

```
SQL>set serverout on;
SQL>/
Procedure created.
SQL>exe pro1(4)
1
2
3
4
PL/SQL procedure successfully completed.
```

02.Create a table employee which has the following columns eno,ename,sal and write a procedure to display the employee details of given employee number.

Procedure:

```
   Create or  replace procedure p3(num number)is
   erec emp%rowtype;
   begin
            select eno,ename into erec.eno,erec.ename from emp where eno=num;
            dbms_ouput.put_line('Employee number:'||erec.eno);
            dbms_output.put_line ('Employee name"||erec.ename);
            exception when no_data_found then
            dbms_output.put_line('No record found');
end;
```

Main program:

```
declare
         empno emp.eno%rowtype;
begin
              empno :=&empno;
               p3(empno);
end;
```

Output:

```
SQL>set serverout on;
SQL>/
SQL>execp2(4);
Factorial of  4 is :24
PL/SQL procedure successfully completed.
```

03.Create a procedure for a swapping program.

```
   SQL>Create or replace procedure swap(a in out number, b in out number) is c number
begin
        c:=a;
        a:=b;
        b:=c;
end;
```

Main program:

```
  declare
             a number;
             b number;
begin
            a:=&a;
            b:=&b;
            dbms_output.putline('Before swapping a='||a|| b ='||b);
            swap(a,b);
            dbms_output.putline('After swapping a='||a|| b ='||b);
end;
```

Output:

```
Enter value for a: 3
Old 5:a:=&a;
New 5:a:=3;
Enter the value for b: 5
Old 6:b:=&b;
New 6:b:=5;
Before swapping a = 3 b = 5
After swapping a = 5 b = 3
PL/SQL procedure successfully completed.
```

04.Write a procedure to input a value from the user and display it.

```
SQL> set serveroutput on;
SQL> declare
2 a varchar2(20) ;
3 begin
4 a:=&a;
```

```
5 dbms_output.put_line(a);
6 end;
7 /
```

Output:

```
Enter value for a: 5
old   4: a:=&a;
new   4: a:=5;
5
PL/SQL procedure successfully completed.
```

05 .Write a procedure to find the greatest of three numbers.

```
SQL> set serveroutput on;
SQL> declare
2 a number(7);
3 b number(7);
4 c number(7);
5 begin
6 a:=&a;
7 b:=&b;
8 c:=&c;
9 if(a>b and a>c)  then
10 dbms_output.put_line (' The greatest of the three is ' || a);
11 else if (b>c) then
12 dbms_output.put_line (' The greatest of the three is ' || b);
13 else
14 dbms_output.put_line (' The greatest of the three is ' || c);
15 end if;
16 end if;
17 end;
18 /
```

Output:

```
Enter value for a: 5
old   6: a:=&a;
new   6: a:=5;
Enter value for b: 7
old   7: b:=&b;
new   7: b:=7;
Enter value for c: 1
old   8: c:=&c;
new   8: c:=1;
The greatest of the three is 7
PL/SQL procedure successfully completed.
```

06 .Write a procedure to print numbers from 1 to 5 using simple loop

```
SQL> set serveroutput on;
SQL> declare
2 a number:=1;
3 begin
4 loop
5 dbms_output.put_line (a);
6 a:=a+1;
7 exit when a>5;
8 end loop;
9 end;
10 /
```

<u>Output:</u>

```
1
2
3
4
5
PL/SQL procedure successfully completed.
```

07. Write a procedure to print numbers from 1 to 4 using while loop

```
SQL> set serveroutput on;
SQL> declare
2 a number:=1;
3 begin
4 while(a<5)
5 loop
6 dbms_output.put_line (a);
7 a:=a+1;
8 end loop;
9 end;
10 /
```

<u>Output:</u>

```
1
2
3
4
PL/SQL procedure successfully completed.
```

08 . Write a procedure to print numbers from 1 to 5 using for loop

```
SQL> set serveroutput on;
SQL> declare
2 a number:=1;
```

```
3 begin
4 for a in 1..5
5 loop
6 dbms_output.put_line (a);
7 end loop;
8 end;
9 /
```

Output:

```
1
2
3
4
5
PL/SQL procedure successfully completed.
```

02. Functions:

Syntax:

```
 CREATE[OR REPLACE] FUNCTION function_name
[(parameter[,parameter])]
RETURN return_datatype
IS | AS
[declaration_section]
BEGIN
executable_section
[EXCEPTION
      Exception_section]
END[function_name];
```

01.Create a function to display the salary employee number.

```
SQl>Create or replace function fun1 (num emp.eno%type)return emp.sal%type is esal
emp.sal%type;
begin
        select sal into esal from emp where eno=num;
        return(esal);
        exception when no_data_found then
                return(-1);
end;
```

Main program:

```
declare
       num emp.eno%type;
       salary emp.sal%type;
```

```
begin
 num :=&num;
 salary:=fun 1(num);
 if(salary =-1)then
 dbms_output.putline('No records available with employee number: '|| num);
else
dbms_output.putline('Salary of employee number '||num ||'is:'||salary);
end if;
end;
```

Output:

```
SQL>/
Function created
SQL>/
Enter the value for num:124
Old 5:    num:=&num;
new 5:   num:124;
Salary of employee number 124 is :22312

PL/SQL procedure successfully completed.
```

02. Create a function to calculate the factorial of given number.

```
SQL> create or replace function fact(n in number)
2 return number
3 is
4 s number
5 begin
6 s:=1;
7 if n = 0 then
8 return 1;
9 end if;
10 for I in l..n loop
11 s:=s*I;
12 end loop;
13 return s;
14 end;
15 /

Function created.

SQL>select fact(10) from dual;
```

Output:

```
FACT(10)

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

3628800
```

03. Create a function to calculate the sum of Odd numbers.

```
SQL>set serveroutput on
SQL> declare
2 n number;
3 Sum1 nmber default 0;
4 Endvalue number;
 5 Begin
6 endvalue:=&endvalue;
7 n:=1;
8 while(n<endvalue)
9 loop
10sum1: = sum1 + n;
11 n:=n+2;
12 end loop
13 dbms_output.put_line(‘sum of odd numbers between 1 and ‘||endvalue|| ‘ is ‘|| sum1);
14 end;
15 /
```

Output:

```
Enter value for endvalue: 20
Old 6: endvalue :=&endvalue;
New 6:endvalue:=20;
Sum of odd numbers between 1 and 20 is 100
PL/SQL procedure successfully completed.
```

03. Create a function to display the table details.

```
SQL> create table pupil (name varchar2(10), age number(3));
Table created.
SQL> select * from pupil
NAME       AGE
------------- ----------
SOUND         18
Shwetha       19
```

```
SQL> create or replace function pup
2 (
3 p_age in number
4 )
5 Return varchar
6 is
7 pupil.age
8 begin
9 slect age into i
10 from pupil
11 where age p_age;
12 Return ‘exist’
13 Exception
14 when w_data_found
15 then
16 return ‘not exist’;
17 End
19 /
Function created
SQL> select pup(18) from dual
PUP(18)
----------------
exist
SQL>select pup(20) from dual
PUP(18)
Not exist
```

Result:

Thus procedure and functions were executed.

Ex.No:11a **DESIGN AND IMPLEMENTATION OF STUDENTS INFORMATION SYSTEM**
Date:

Aim:

To design a simple form in Visual Basic using Oracle as backend.

Table creation:

```
SQL> create table vb1(Name varchar2(20),Rollno number(3),Maths number(3),English number(3),Physics number(3),Chemistry number(3));
Table created.

SQL> desc vb1;
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
NAME                                               VARCHAR2(20)
ROLLNO                                             NUMBER(3)
MATHS                                              NUMBER(3)
ENGLISH                                            NUMBER(3)
PHYSICS                                            NUMBER(3)
CHEMISTRY                                          NUMBER(3)
```

2. **Insert** all the possible values into the vb1 table.
3. Enter **commit** command.

Algorithm for ADO Connection:

After creating the table in Oracle, Go to start menu.

1. Start→ControlPanel→AdministrativeTools→DataSources(ODBC)→User DSN→Add
   → Select **Microsoft ODBC for Oracle**→Finish→OK
2. One new window will appear. In that window type **<u>Data Source Name</u>** as table name created in Oracle. Type user name as the user name entered in SQL+, Server as 172.31.4.4 and then click O.K.

Algorithm for ADODC in Visual Basic:

1. In Visual Basic create the labels, command buttons and their text boxes.
2. In Visual Basic go to Project menu→Components→**Microsoft ADO Data Control 6.0 for OLEDB**→OK
3. Now drag and drop ADODC Data Control available in toolbox into the form.
4. Right click in ADODC Data Control then click the **ADODC** properties.
5. In the new window, choose G**eneral** tab and use**ODBC Data source name** as the table name created in Oracle(select table name in drop down menu).
6. Choose **Use Connection String:** Orcl→click Build→select **Oracle provider for OLE**
   - Click→Next
   - Data Source: Orcl
   - User Name: as entered in SQL+ login
   - Password: student

   Click ok.
7. Choose A**uthentication** tab and select username, password as entered for SQL+.

8. Choose **Record source**→select Command type as **adcmdTable**.
9. Select **Table or Stored procedure name** as table created in Oracle.
10. Click Apply→O.K
11. Select the **Data Source** as **ADODC1**
12. Select the **Data Field** and set the required **field name** created in table

Coding:

1.ADD

```
Private Sub ADD_Click()
Text1.SetFocus
Adodc1.Recordset.AddNew
End Sub
```

2.DELETE

```
Private Sub DELETE_Click()
If MsgBox("DELETE IT?", vbOKCancel) = vbOK Then
Adodc1.Recordset.DELETE
MsgBox "One row deleted"
End If
Text1.Text = " "
Text2.Text = " "
Text3.Text = " "
Text4.Text = " "
Text5.Text = " "
Text6.Text = " "
End Sub
```

3.EXIT

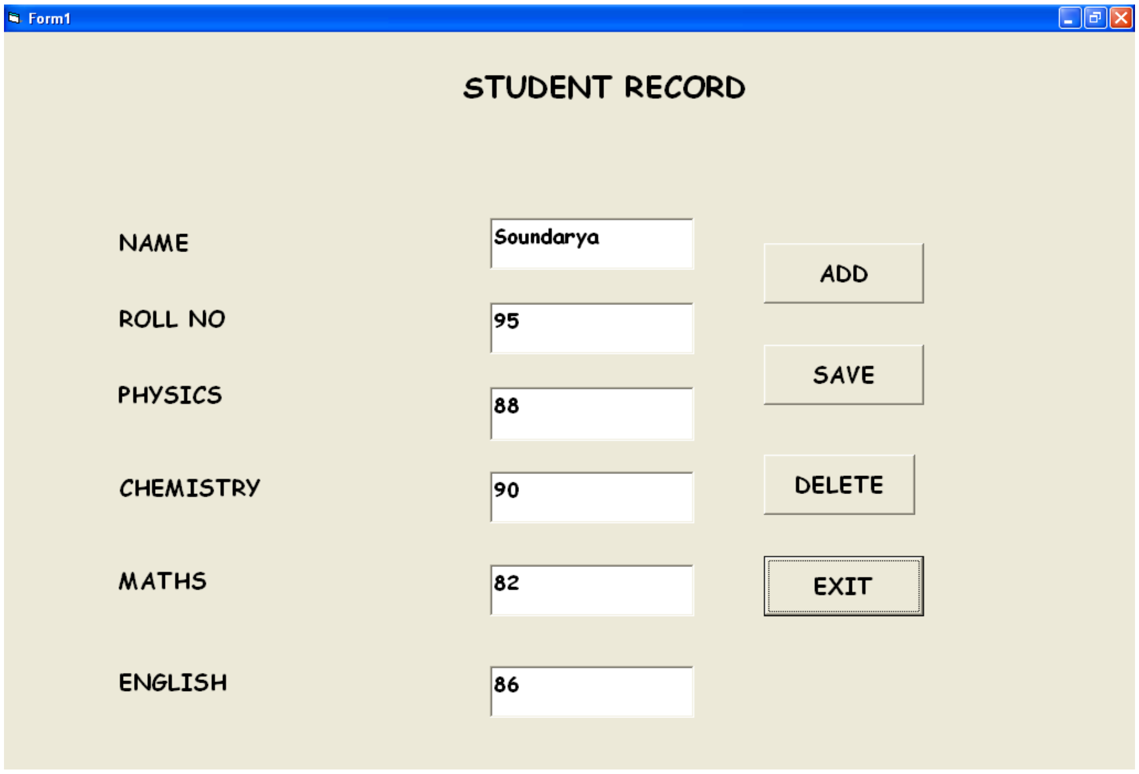
```
Private Sub EXIT_Click()
Unload Me
End Sub
```

4.SAVE

```
Private Sub SAVE_Click()
If MsgBox("SAVE IT?", vbOKCancel) = vbOK Then
Adodc1.Recordset.Update
Else
Adodc1.Recordset.CancelUpdate
End If
End Sub
```

OUTPUT:

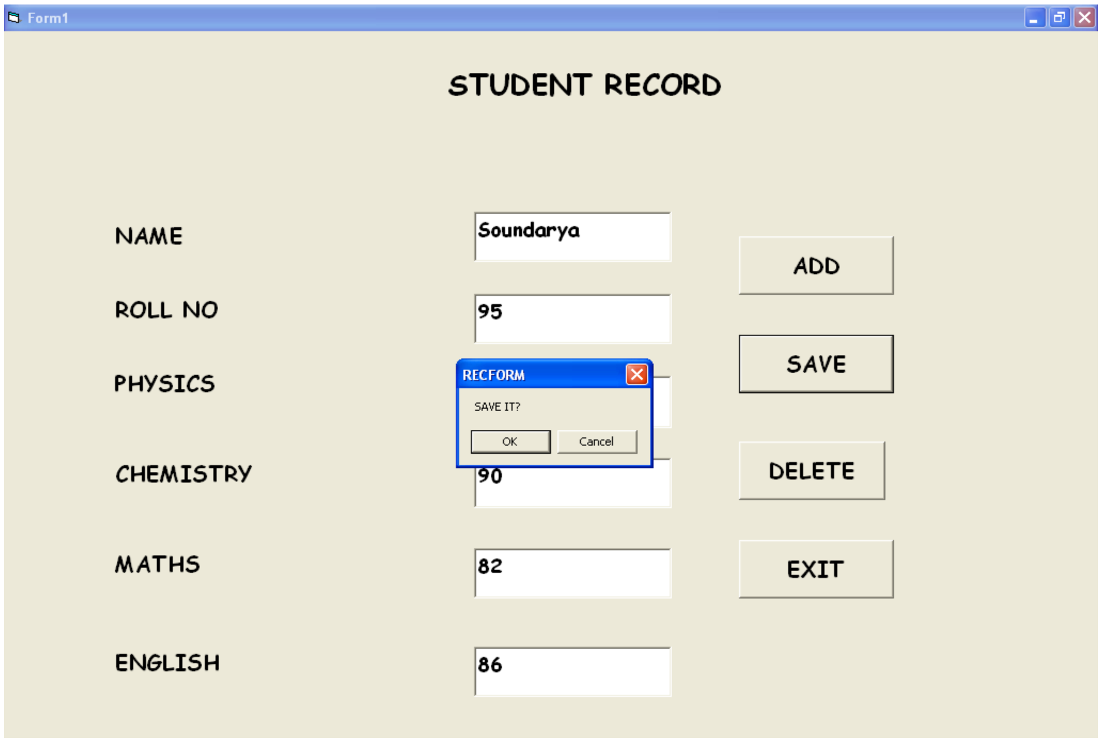
INSERTING VALUES INTO TABLE



Form1

STUDENT RECORD

NAME	Soundarya
ROLL NO	95
PHYSICS	88
CHEMISTRY	90
MATHS	82
ENGLISH	86



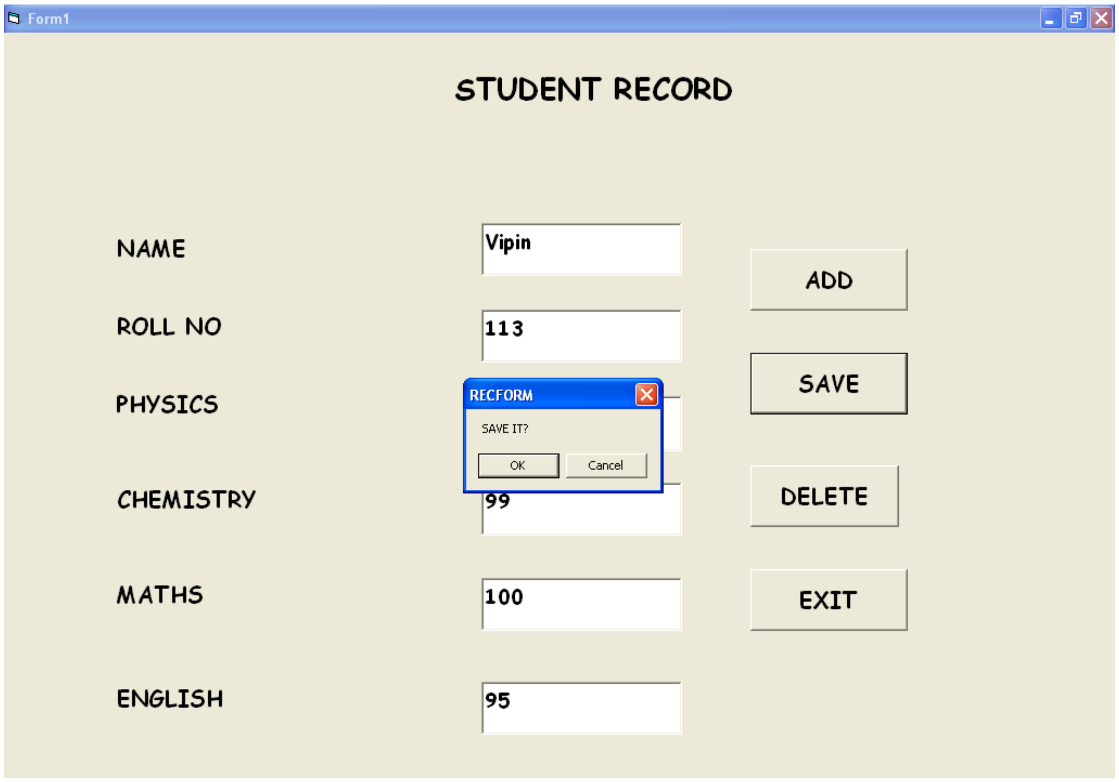
Form1

STUDENT RECORD

NAME	Soundarya
ROLL NO	95
PHYSICS	
CHEMISTRY	90
MATHS	82
ENGLISH	86

RECFORM

SAVE IT?



STUDENT RECORD

NAME Vipin

ROLL NO 113

PHYSICS

CHEMISTRY 99

MATHS 100

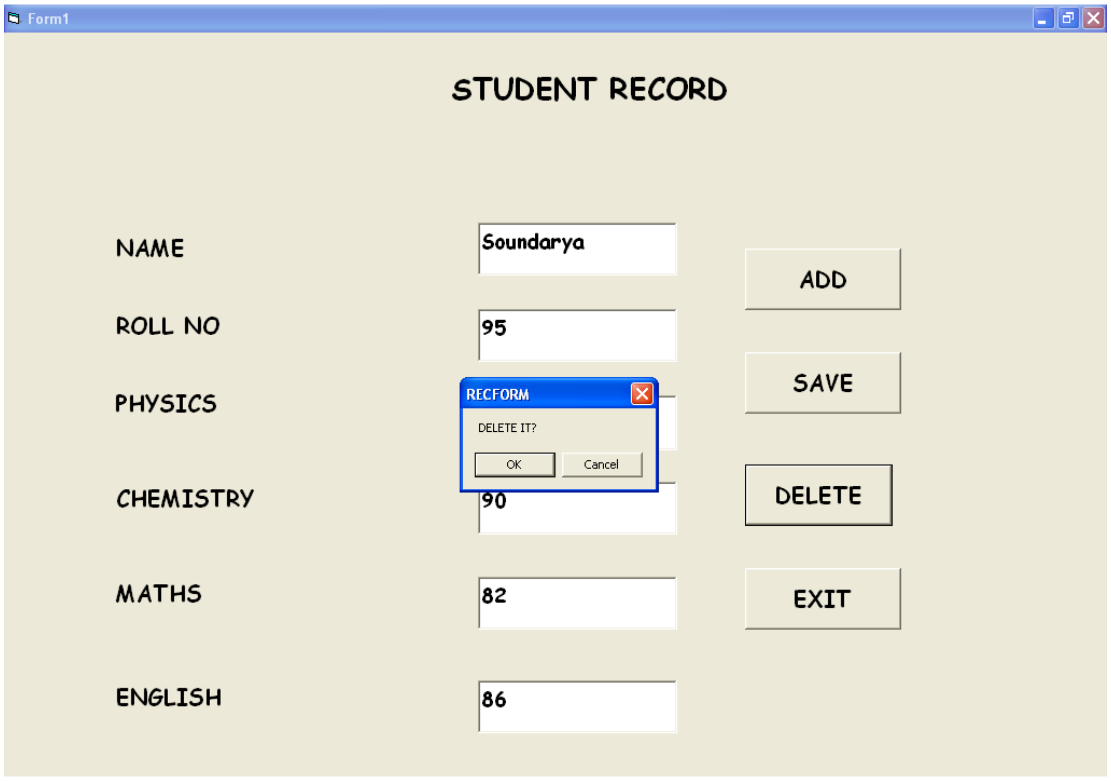
ENGLISH 95

RECFORM

SAVE IT?

```
SQL> select * from vb1;
NAME                 ROLLNO     MATHS    ENGLISH    PHYSICS  CHEMISTRY
--------------------------------------------------------------------------------------------------------
Vipin                   113              100          95               96        99
Soundarya                  95                         82          86              88         90
```

DELETING ONE ROW FROM TABLE



NAME: Soundarya
ROLL NO: 95
CHEMISTRY: 90
MATHS: 82
ENGLISH: 86

DELETE IT?

```
SQL> select * from vb1;

NAME                ROLLNO     MATHS    ENGLISH    PHYSICS   CHEMISTRY
----------------------------------------------------------------------------------------------------
Vipin                    113          100            95             96         99
```

Result:

Thus the Students Information System using Visual Basic has been created.

Ex.No:11b **DESIGN AND IMPLEMENTATION OF LIBRARY**

Date: **INFORMATION SYSTEM**

Aim:

To design the library details in Visual Basic using Oracle as backend.

Algorithm for creating table:

1. **Create** library table with following fields

```
Name                    Type
------------------------------ -------- ----
BOOK NO                 NUMBER (15)
BOOKNAME                VARCHAR2 (15)
BOOKAUTHOR              VARCHAR2 (15)
BOOKRATE                NUMBER (8)
NCOPY                   NUMBER (8)
ISSUE DATE              DATE
RETURN DATE             DATE
```

2. **Insert** all the possible values into the library table.
3. Enter **commit** command.

Algorithm for ADO Connection:

After creating the table in Oracle, Go to start menu.

1. Start→ Control Panel →Administrative Tools→Data Sources (ODBC)→User DSN→Add→ Select **Microsoft ODBC for Oracle**→Finish→OK
2. One new window will appear. In that window type **Data Source Name** as table name created in Oracle. Type user name as the user name entered in SQL+, Server as 172.31.4.4 and then click O.K.

Algorithm for ADODC in Visual Basic:

1. In Visual Basic create the labels, command buttons and their text boxes.
2. In Visual Basic go to Project menu→Components→**Microsoft ADO Data Control 6.0 for OLEDB**→OK
3. Now drag and drop ADODC Data Control available in toolbox into the form.
4. Right click in ADODC Data Control then click the ADODC properties.
5. Choose G**eneral** tab, use**ODBC Data source name** as the table name created in Oracle.
6. Choose **Use Connection String:** Orcl→click Build→select **Oracle provider for OLE**
    - Click→Next
    - Data Source: Orcl
    - User Name: as entered in SQL+ login
    - Password: student
7. Choose A**uthentication** tab and select username, password as entered for SQL+.
8. Choose **Record source**→select Command type as **adcmdTable**.
9. Select **Table or Stored procedure name** as table created in Oracle.
10. Click Apply→O.K
11. Select the **Data Source** as **ADODC1**
12. Select the **Data Field** and set the required **field name** created in table

Coding:

1. **ADD**
```
Private Sub ADD_Click ()
Text1.SetFocus
Adodc1.Recordset.AddNew
End Sub
```
2. **SAVE**
```
Private Sub SAVE_Click ()
If MsgBox ("SAVE IT?", vbOKCancel) = vbOK Then
Adodc1.Recordset.Update
Else
Adodc1.Recordset.CancelUpdate
End If
End Sub
```
3. **DELETE**
```
Private Sub DELETE_Click ()
If MsgBox ("DELETE IT?", vbOKCancel) = vbOK Then
Adodc1.Recordset.Delete
MsgBox "One row deleted"
End If
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
End Sub
```
4. **FIND**
```
Private Sub FIND_Click ()
Dim N As String
N = InputBox ("Enter the book")
Adodc1.Recordset.Find "book_no=" & N
If Adodc1.Recordset.BOF or Adodc1.Recordset.EOF Then
MsgBox "Record not found"
End If
End Sub
```
5. **MOVE**

FIRST
```
Private Sub FIRST_Click ()
Adodc1.Recordset.MoveFirst
```

```
End Sub
```

PREVIOUS

```
Private Sub PREVIOUS_Click ()
Adodc1.Recordset.MovePrevious
End Sub
```

NEXT

```
Private Sub NEXT_Click ()
Adodc1.Recordset.MoveNext
End Sub
```

LAST

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

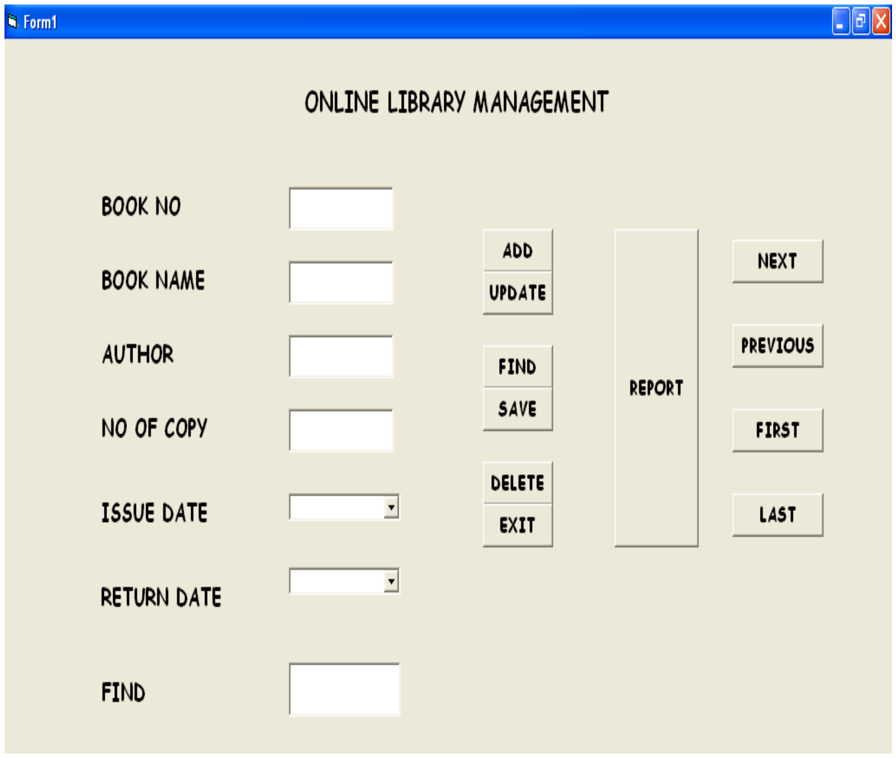
6. **UPDATE**

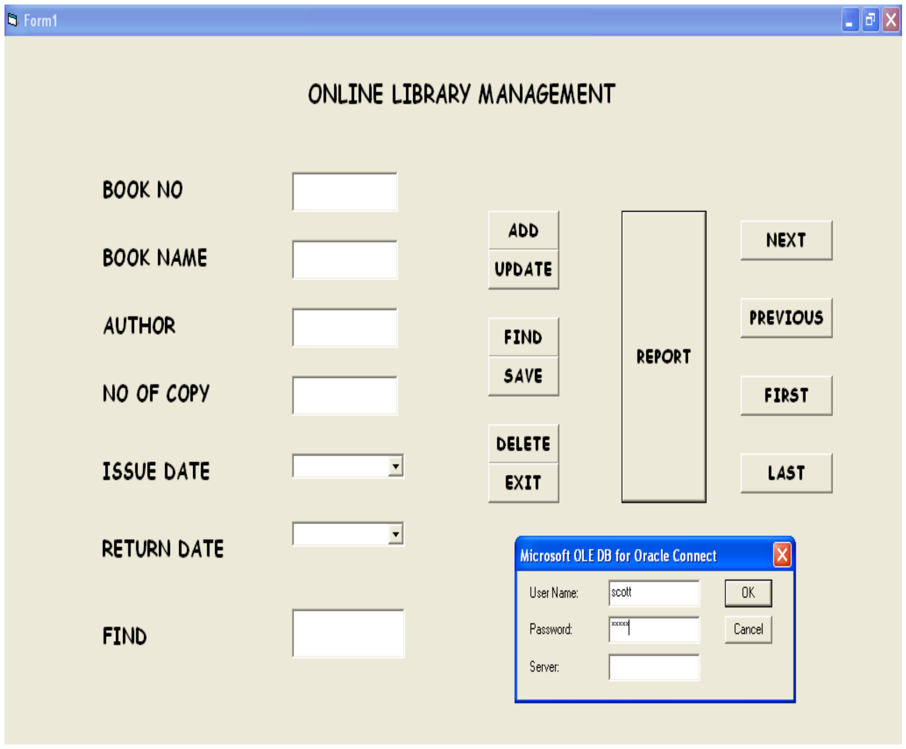
```
Private Sub UPDATE_Click ()
Adodc1.Recordset.Update
End Sub
```

7. **EXIT**

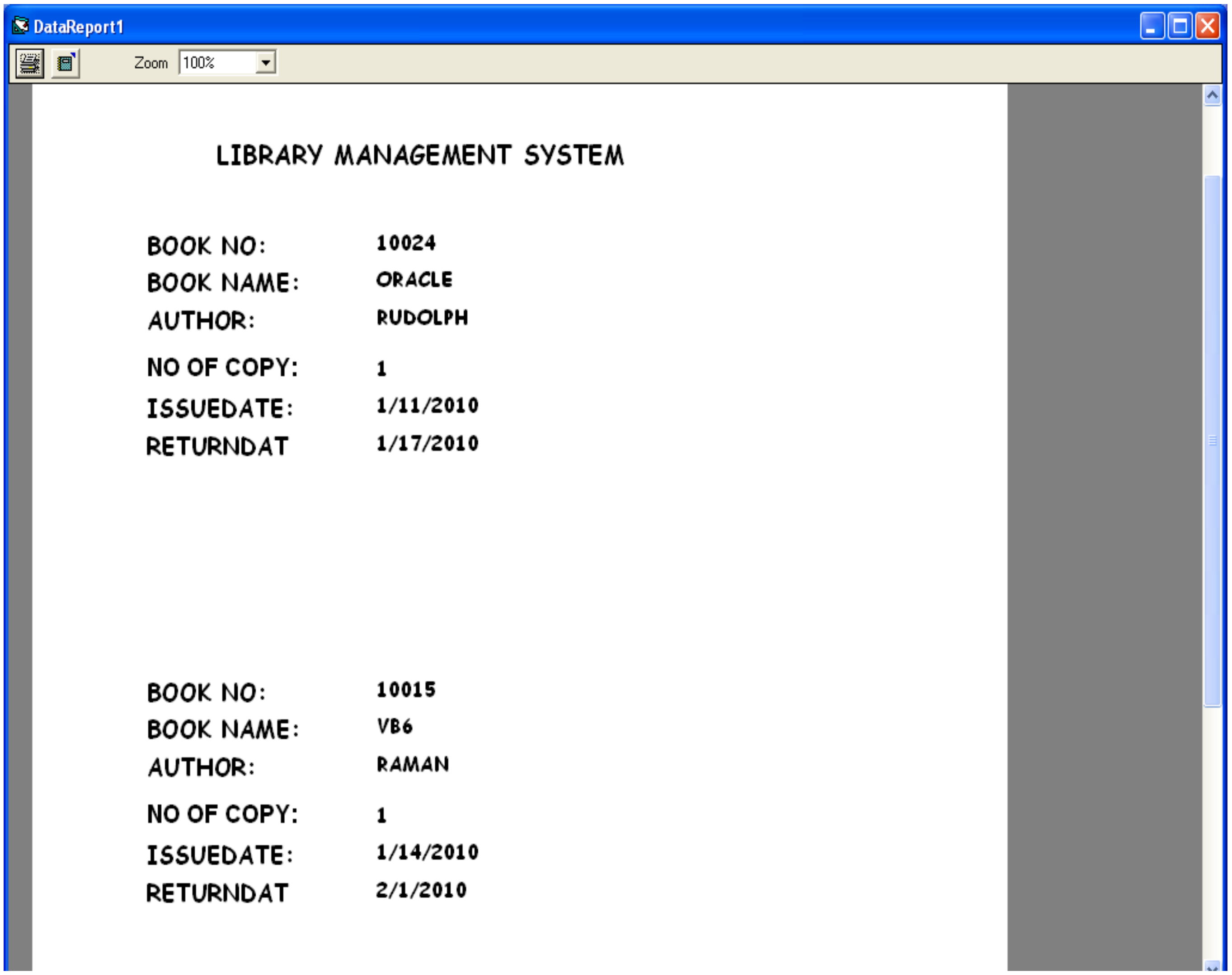
```
Private Sub EXIT_Click ()
Unload Me
End Sub
```

OUTPUT:





User Name: scott
Password: *****
Server:



LIBRARY MANAGEMENT SYSTEM

BOOK NO:	10024
BOOK NAME:	ORACLE
AUTHOR:	RUDOLPH
NO OF COPY:	1
ISSUEDATE:	1/11/2010
RETURNDAT	1/17/2010

BOOK NO:	10015
BOOK NAME:	VB6
AUTHOR:	RAMAN
NO OF COPY:	1
ISSUEDATE:	1/14/2010
RETURNDAT	2/1/2010

Result:

Thus the library details were designed in Visual Basic using Oracle as backend and executed successfully.