VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Faculty	:	**Dr V VANI**	Slot	:	**L55+L56**
Name	:	**P VISHNU MOHAN KRISHNA**	Regno	:	**21MIS1118**

Ex no.1 18.2.22

SQL-TABLE CREATION-DDL

Aim : To create database and perform DDL operations

Lab Exercise

1. **Create and describe the following table.**

A) **NAME**: branch

FIELDS	**DATATYPE**
branch_name	varchar2(30)
branch_city	varchar2(30)
assets	number(8,2)

```
SQL> create table branch(branch_name varchar(30),branch_city varchar(30),assets number(8));

Table created.

SQL> desc branch;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NAME                                        VARCHAR2(30)
 BRANCH_CITY                                        VARCHAR2(30)
 ASSETS                                             NUMBER(8)
```

2. **Describe the structure of database schema.**

```
SQL> desc branch;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NAME                                        VARCHAR2(30)
 BRANCH_CITY                                        VARCHAR2(30)
 ASSETS                                             NUMBER(8)
```

3. **Rename the table with existing name followed by last four digits of your register number.**

```
SQL> rename branch to branch1118;

Table renamed.

SQL> desc branch1118;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NAME                                        VARCHAR2(30)
 BRANCH_CITY                                        VARCHAR2(30)
 ASSETS                                             NUMBER(8)
```

4. **Drop the table.**

```
SQL> drop table branch1118;

Table dropped.

SQL> desc branch1118;
ERROR:
ORA-04043: object branch1118 does not exist
```

B) **NAME**: account

FIELDS	**DATATYPE**
account_no	varchar2(11)
branch_name	varchar2(30)
balance	number(8)

1. **Create and describe the following table.**

```
SQL>  create table account(account_no varchar(11),branch_name varchar(20),balance number(8));

Table created.

SQL> desc account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NO                                         VARCHAR2(11)
 BRANCH_NAME                                        VARCHAR2(20)
 BALANCE                                            NUMBER(8)
```

2. **Describe the structure of database schema.**

```
SQL> desc account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NO                                         VARCHAR2(11)
 BRANCH_NAME                                        VARCHAR2(20)
 BALANCE                                            NUMBER(8)
```

3. **Rename the table with existing name followed by last four digits of your register number.**

```
SQL> rename account to account1118;

Table renamed.

SQL> desc account1118
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NO                                         VARCHAR2(11)
 BRANCH_NAME                                        VARCHAR2(20)
 BALANCE                                            NUMBER(8)
 ACCOUNT_OPENING_DATE                               VARCHAR2(10)
```

4. **Drop the table.**

```
SQL> drop table account1118;

Table dropped.

SQL> desc account1118;
ERROR:
ORA-04043: object account1118 does not exist
```

C) **NAME**: customer

FIELD	**DATATYPE**
customer_id	varchar2(11)
customer_name	varchar2(20)
customer_street	varchar2(15)
customer_city	varchar2(15)

1. **Create and describe the following table.**

```
SQL> create table customer(customer_id varchar(11),customer_name varchar(20),customer_street varchar(15),customer_city
 varchar(15));

Table created.

SQL> desc customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 CUSTOMER_NAME                                      VARCHAR2(20)
 CUSTOMER_STREET                                    VARCHAR2(15)
 CUSTOMER_CITY                                      VARCHAR2(15)
```

2. **Describe the structure of database schema.**

```
SQL> desc customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 CUSTOMER_NAME                                      VARCHAR2(20)
 CUSTOMER_STREET                                    VARCHAR2(15)
 CUSTOMER_CITY                                      VARCHAR2(15)
```

3. **Rename the table with existing name followed by last four digits of your register number.**

```
SQL> rename customer to customer1118;

Table renamed.

SQL> desc customer1118;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 CUSTOMER_NAME                                      VARCHAR2(20)
 CUSTOMER_STREET                                    VARCHAR2(20)
 CUSTOMER_CITY                                      VARCHAR2(15)
```

4. Drop the table.

```
SQL> drop table customer1118;

Table dropped.

SQL> desc customer1118;
ERROR:
ORA-04043: object customer1118 does not exist
```

D) **NAME**: depositor

FIELD	**DATATYPE**
customer_id	varchar2(11)
account_no	varchar2(11)

1. Create and describe the following table.

```
SQL> create table depositor(customer_id varchar(11),account_no varchar(11));

Table created.

SQL> desc depositor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 ACCOUNT_NO                                         VARCHAR2(11)
```

2. Describe the structure of database schema.

```
SQL> desc depositor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 ACCOUNT_NO                                         VARCHAR2(11)
```

3. Rename the table with existing name followed by last four digits of your register number.

```
SQL> rename depositor to depositor1118;

Table renamed.

SQL> desc depositor1118;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 ACCOUNT_NO                                         VARCHAR2(11)
```

4. Drop the table.

```
SQL> drop table depositor1118;

Table dropped.

SQL> desc depositor1118;
ERROR:
ORA-04043: object depositor1118 does not exist
```

E) **NAME**: loan

FIELDS	**DATATYPE**
loan_no	varchar2(4)
branch_name	varchar2(30)
amount	number(8,2)

1. Create and describe the following table.

```
SQL> create table loan(loan_no varchar(4),branch_name varchar(30),amount number(8));

Table created.

SQL> desc loan;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOAN_NO                                            VARCHAR2(4)
 BRANCH_NAME                                        VARCHAR2(30)
 AMOUNT                                             NUMBER(8)
```

2. Describe the structure of database schema.

```
SQL> desc loan;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOAN_NO                                            VARCHAR2(4)
 BRANCH_NAME                                        VARCHAR2(30)
 AMOUNT                                             NUMBER(8)
```

3. **Rename the table with existing name followed by last four digits of your register number.**

```
SQL> rename loan to loan1118;

Table renamed.

SQL> desc loan1118;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOAN_NO                                            VARCHAR2(4)
 BRANCH_NAME                                        VARCHAR2(30)
 AMOUNT                                             NUMBER(8)
```

4. **Drop the table.**

```
SQL> drop table loan1118;

Table dropped.

SQL> desc loan1118;
ERROR:
ORA-04043: object loan1118 does not exist
```

F) **NAME**: borrower

FIELDS	**DATATYPE**
customer_id	varchar2(11)
loan_no	varcahr2(4)

1. **Create and describe the following table.**

```
SQL> create table borrower(customer_id varchar(11),loan_no varchar(4));

Table created.

SQL> desc borrower;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 LOAN_NO                                            VARCHAR2(4)
```

2. **Describe the structure of database schema.**

```
SQL> desc borrower;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 LOAN_NO                                            VARCHAR2(4)
```

3. **Rename the table with existing name followed by last four digits of your register number.**

```
SQL> rename borrower to borrower1118;

Table renamed.

SQL> desc borrower1118;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 LOAN_NO                                            VARCHAR2(4)
```

4. **Drop the table.**

```
SQL> drop table borrower1118;

Table dropped.

SQL> desc borrower1118;
ERROR:
ORA-04043: object borrower1118 does not exist
```

Alter the structure of the Database

a. Add a new column ‘account opening date’ in the account table, which is of type Date.

```
SQL> alter table account add(account_opening_date varchar(10));

Table altered.

SQL> desc account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NO                                         VARCHAR2(11)
 BRANCH_NAME                                        VARCHAR2(20)
 BALANCE                                            NUMBER(8)
 ACCOUNT_OPENING_DATE                               VARCHAR2(10)
```

b. Increase the width of the column customer_street in table customer to 20.

```
SQL> alter table customer modify(customer_street varchar(20));

Table altered.

SQL> desc customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 CUSTOMER_NAME                                      VARCHAR2(20)
 CUSTOMER_STREET                                    VARCHAR2(20)
 CUSTOMER_CITY                                      VARCHAR2(15)
```

Result: Thus the DDL commands were executed successfully for table creation, altering, renaming and dropping.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Faculty	:	**Dr V VANI**	Slot	:	**L55+L56**
Name	:	**P VISHNU MOHAN KRISHNA**	Reg.no	:	**21MIS1118**

Ex.No.2 25.2.22

Data Manipulation Commands for Inserting.

Aim : To create database and perform DML and TCL queries to retrieve information from the database.

Exercises:

Insert the following values into the tables

1. **branch :**

BRANCH_NAME	BRANCH_CITY	ASSETS
Perryridge	**Rye**	**5000000**
Downtown	**Stamford**	**1000000**
Brighton	**Paloalto**	**2500000**
Redwood	**Harrison**	**1500000**
Mianus	**Pitsfield**	**4500000**
Roundhill	**Princeton**	**1500000**

Syntax:

```
SQL> desc branch;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRANCH_NAME                                        VARCHAR2(30)
 BRANCH_CITY                                        VARCHAR2(30)
```

```
 ASSETS                                    NUMBER(8)

SQL> insert into branch values('&branch-name','&branch_city',&assets);
Enter value for branch: Perryridge
Enter value for branch_city: Rye
Enter value for assets: 5000000
old   1: insert into branch values('&branch-name','&branch_city',&assets)
new   1: insert into branch values('Perryridge-name','Rye',5000000)

1 row created.

SQL> /
Enter value for branch: Downtown
Enter value for branch_city: Stamford
Enter value for assets: 1000000
old   1: insert into branch values('&branch-name','&branch_city',&assets)
new   1: insert into branch values('Downtown-name','Stamford',1000000)

1 row created.

SQL> /
Enter value for branch: Brighton
Enter value for branch_city: Paloalto
Enter value for assets: 2500000
old   1: insert into branch values('&branch-name','&branch_city',&assets)
new   1: insert into branch values('Brighton-name','Paloalto',2500000)

1 row created.

SQL> /
Enter value for branch: Redwood
Enter value for branch_city: harrison
Enter value for assets: 1500000
old   1: insert into branch values('&branch-name','&branch_city',&assets)
new   1: insert into branch values('Redwood-name','harrison',1500000)

1 row created.

SQL> /
Enter value for branch: Mianus
Enter value for branch_city: Pitsfield
Enter value for assets: 4500000
old   1: insert into branch values('&branch-name','&branch_city',&assets)
new   1: insert into branch values('Mianus-name','Pitsfield',4500000)

1 row created.
```

```
SQL> /
Enter value for branch: Roundhill
Enter value for branch_city: Princeton
Enter value for assets: 1500000
old   1: insert into branch values('&branch-name','&branch_city',&assets)
new   1: insert into branch values('Roundhill-name','Princeton',1500000)

1 row created.
```

Output:

```
SQL> select * from branch;

BRANCH_NAME                    BRANCH_CITY                       ASSETS
------------------------------ ------------------------------ ----------
Perryridge-name                Rye                                5000000
Downtown-name                  Stamford                           1000000
Brighton-name                  Paloalto                           2500000
Redwood-name                   harrison                           1500000
Mianus-name                    Pitsfield                          4500000
Roundhill-name                 Princeton                          1500000

6 rows selected.
```

2. account :

ACCOUNT_NO	BRANCH_NAME	BALANCE
019_28_3746	**Perryridge**	**15000**
182_73_6091	**Downtown**	**23000**
192_83_7465	**Brighton**	**18000**
321_12_3123	**Redwood**	**5000**
336_66_9999	**Mianus**	**5000**
963_96_3963	**Roundhill**	**5000**
376_66_9999	**Mianus**	**9000**
963_96_3964	**Mianus**	**13000**

Syntax:

```
SQL> desc account;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACCOUNT_NO                                          VARCHAR2(11)
 BRANCH_NAME                                         VARCHAR2(30)
 BALANCE                                             NUMBER(8)
```

```
SQL> insert into account values('&account_no','&branch_name',&balance);
Enter value for account_no: 019_28_3746
Enter value for branch_name: Perryridge
Enter value for balance: 15000
old   1: insert into account values('&account_no','&branch_name',&balance)
new   1: insert into account values('019_28_3746','Perryridge',15000)

1 row created.

SQL> /
Enter value for account_no: 182_73_6091
Enter value for branch_name: Downtown
Enter value for balance: 23000
old   1: insert into account values('&account_no','&branch_name',&balance)
new   1: insert into account values('182_73_6091','Downtown',23000)

1 row created.

SQL> /
Enter value for account_no: 192_83_7465
Enter value for branch_name: Brighton
Enter value for balance: 18000
old   1: insert into account values('&account_no','&branch_name',&balance)
new   1: insert into account values('192_83_7465','Brighton',18000)

1 row created.

SQL> /
Enter value for account_no: 321_12_3123
Enter value for branch_name: Redwood
Enter value for balance: 5000
old   1: insert into account values('&account_no','&branch_name',&balance)
new   1: insert into account values('321_12_3123','Redwood',5000)

1 row created.

SQL> /
Enter value for account_no: 336_66_9999
Enter value for branch_name: Mianus
Enter value for balance: 5000
old   1: insert into account values('&account_no','&branch_name',&balance)
new   1: insert into account values('336_66_9999','Mianus',5000)

1 row created.

SQL> /
```

Enter value for account_no: 963_96_3963
Enter value for branch_name: Roundhill
Enter value for balance: 5000
old 1: insert into account values('&account_no','&branch_name',&balance)
new 1: insert into account values('963_96_3963','Roundhill',5000)

1 row created.

SQL> /
Enter value for account_no: 376_66_9999
Enter value for branch_name: Mianus
Enter value for balance: 9000
old 1: insert into account values('&account_no','&branch_name',&balance)
new 1: insert into account values('376_66_9999','Mianus',9000)

1 row created.

SQL> /
Enter value for account_no: 963_96_3964
Enter value for branch_name: Mianus
Enter value for balance: 13000
old 1: insert into account values('&account_no','&branch_name',&balance)
new 1: insert into account values('963_96_3964','Mianus',13000)

1 row created.

Output:

```
SQL> select * from account;

ACCOUNT_NO  BRANCH_NAME                      BALANCE
----------- ------------------------------ ----------
019_28_3746 Perryridge                          15000
182_73_6091 Downtown                            23000
192_83_7465 Brighton                            18000
321_12_3123 Redwood                              5000
336_66_9999 Mianus                               5000
963_96_3963 Roundhill                            5000
376_66_9999 Mianus                               9000
963_96_3964 Mianus                              13000

8 rows selected.
```

3. loan :

LOAN	BRANCH_NAME	AMOUNT
1_11	Roundhill	9000
1_14	Downtown	15000
1_15	Perryridge	15000
1_16	Perryridge	13000
1_17	Downtown	10000
1_23	Redwood	20000
1_93	Mianus	500

Syntax:

```
SQL> desc loan;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOAN_NO                                            VARCHAR2(4)
 BRANCH_NAME                                        VARCHAR2(30)
 AMOUNT                                             NUMBER(8)

SQL> insert into loan values('&loan_no','&branch_name',&amount);
Enter value for loan_no: 1_11
Enter value for branch_name: Roundhill
Enter value for amount: 9000
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1_11','Roundhill',9000)

1 row created.

SQL> /
Enter value for loan_no: 1-14
Enter value for branch_name: Downtown
Enter value for amount: 15000
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1-14','Downtown',15000)

1 row created.

SQL> /
Enter value for loan_no: 1_15
Enter value for branch_name: Perryridge
Enter value for amount: 15000
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1_15','Perryridge',15000)

1 row created.
```

```
SQL> /
Enter value for loan_no: 1_16
Enter value for branch_name: Perryridge
Enter value for amount: 13000
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1_16','Perryridge',13000)

1 row created.

SQL> /
Enter value for loan_no: 1_17
Enter value for branch_name: Downtown
Enter value for amount: 10000
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1_17','Downtown',10000)

1 row created.

SQL> /
Enter value for loan_no: 1_23
Enter value for branch_name: Redwood
Enter value for amount: 20000
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1_23','Redwood',20000)

1 row created.

SQL> /
Enter value for loan_no: 1_93
Enter value for branch_name: Mianus
Enter value for amount: 500
old   1: insert into loan values('&loan_no','&branch_name',&amount)
new   1: insert into loan values('1_93','Mianus',500)

1 row created.
```

Output:

```
SQL> select * from loan;

LOAN BRANCH_NAME                        AMOUNT
---- ------------------------------ ----------
1_11 Roundhill                            9000
1-14 Downtown                            15000
1_15 Perryridge                          15000
1_16 Perryridge                          13000
1_17 Downtown                            10000
1_23 Redwood                             20000
1_93 Mianus                                500

7 rows selected.
```

4. depositor

CUSTOMER_ID	ACCOUNT_NO
c_08	**182_73_6091**
c_03	**192_83_7465**
c_05	**321_12_3123**
c_07	**336_66_9999**
c_08	**963_96_3963**
c_02	**376_66_9999**

Syntax:

```
SQL> desc depositor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 ACCOUNT_NO                                         VARCHAR2(11)

SQL> insert into depositor values('&customer_id','&account_no');
Enter value for customer_id: c_08
Enter value for account_no: 182_73_6091
old   1: insert into depositor values('&customer_id','&account_no')
new   1: insert into depositor values('c_08','182_73_6091')

1 row created.

SQL> /
Enter value for customer_id: c+03
Enter value for account_no: 192_83_7465
old   1: insert into depositor values('&customer_id','&account_no')
new   1: insert into depositor values('c+03','192_83_7465')
```

1 row created.

SQL> /
Enter value for customer_id: c_05
Enter value for account_no: 321
old 1: insert into depositor values('&customer_id','&account_no')
new 1: insert into depositor values('c_05','321')

1 row created.

SQL> /
Enter value for customer_id: c_07
Enter value for account_no: 336_66_9999
old 1: insert into depositor values('&customer_id','&account_no')
new 1: insert into depositor values('c_07','336_66_9999')

1 row created.

SQL> /
Enter value for customer_id: c_08
Enter value for account_no: 963_96_3963
old 1: insert into depositor values('&customer_id','&account_no')
new 1: insert into depositor values('c_08','963_96_3963')

1 row created.

SQL> /
Enter value for customer_id: c_02
Enter value for account_no: 376_66_9999
old 1: insert into depositor values('&customer_id','&account_no')
new 1: insert into depositor values('c_02','376_66_9999')

1 row created.

Output:

```
SQL> select * from depositor;

CUSTOMER_ID ACCOUNT_NO
----------- -----------
c_08        182_73_6091
c+03        192_83_7465
c_05        321
c_07        336_66_9999
c_08        963_96_3963
c_02        376_66_9999

6 rows selected.
```

5. customer

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_STREET	CUSTOMER_CITY
c_01	**smith**	**north**	**rye**
c_02	**turner**	**putnam**	**stamford**
c_03	**johnson**	**alma**	**palo alto**
c_04	**curry**	**north**	**rye**
c_05	**jones**	**main**	**harrisdon**
c_06	**adoms**	**spring**	**pittsfield**
c_07	**lindsay**	**park**	**pittsfield**
c_08	**hayes**	**main**	**harrison**
c_09	**williams**	**nassau**	**Princeton**

Syntax:

```
SQL>  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city');
Enter value for customer_id: c_01
Enter value for customer_name: north
Enter value for customer_street: smith
Enter value for customer_city: rye
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_01','north','smith','rye')

1 row created.

SQL> /
Enter value for customer_id: c_02
Enter value for customer_name: turner
Enter value for customer_street: putnam
Enter value for customer_city: stamford
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_02','turner','putnam','stamford')

1 row created.

SQL> /
Enter value for customer_id: c_03
Enter value for customer_name: johnson
Enter value for customer_street: alma
Enter value for customer_city: palo alto
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_03','johnson','alma','palo alto')
```

```
1 row created.

SQL> /
Enter value for customer_id: c_04
Enter value for customer_name: curry
Enter value for customer_street: north
Enter value for customer_city: rye
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_04','curry','north','rye')

1 row created.

SQL> /
Enter value for customer_id: c_05
Enter value for customer_name: jones
Enter value for customer_street: main
Enter value for customer_city: harrisdon
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_05','jones','main','harrisdon')

1 row created.

SQL> /
Enter value for customer_id: c_06
Enter value for customer_name: adoms
Enter value for customer_street: spring
Enter value for customer_city: pittsfield
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_06','adoms','spring','pittsfield')

1 row created.

SQL> /
Enter value for customer_id: c_07
Enter value for customer_name: lindsay
Enter value for customer_street: park
Enter value for customer_city: pittsfiled
old   1:  insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new   1:  insert into customer values('c_07','lindsay','park','pittsfiled')

1 row created.
```

SQL> /
Enter value for customer_id: c_08
Enter value for customer_name: hayes
Enter value for customer_street: main
Enter value for customer_city: harrison
old 1: insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new 1: insert into customer values('c_08','hayes','main','harrison')

1 row created.

SQL> /
Enter value for customer_id: c_09
Enter value for customer_name: williams
Enter value for customer_street: nassan
Enter value for customer_city: Princeton
old 1: insert into customer
values('&customer_id','&customer_name','&customer_street','&customer_city')
new 1: insert into customer values('c_09','williams','nassan','Princeton')

1 row created.

Output:

```
SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_STREET CUSTOMER_CITY
----------- -------------------- --------------- ---------------
c_01        north                smith           rye
c_02        turner               putnam          stamford
c_03        johnson              alma            palo alto
c_04        curry                north           rye
c_05        jones                main            harrisdon
c_06        adoms                spring          pittsfield
c_07        lindsay              park            pittsfiled
c_08        hayes                main            harrison
c_09        williams             nassan          Princeton

9 rows selected.
```

6. borrower

CUSTOMER_ID	LOAN_NO
c_01	1_11
c_01	1_23
c_03	1_93
c_05	1_17
c_03	1_16
c_05	1_14

Syntax:

```
SQL> desc borrower;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTOMER_ID                                        VARCHAR2(11)
 LOAN_NO                                            VARCHAR2(4)

SQL> insert into borrower values('&customer_id','&loan_no');
Enter value for customer_id: c_01
Enter value for loan_no: 1_11
old   1: insert into borrower values('&customer_id','&loan_no')
new   1: insert into borrower values('c_01','1_11')

1 row created.

SQL> /
Enter value for customer_id: c_01
Enter value for loan_no: 1_23
old   1: insert into borrower values('&customer_id','&loan_no')
new   1: insert into borrower values('c_01','1_23')

1 row created.

SQL> /
Enter value for customer_id: c_03
Enter value for loan_no: 1_93
old   1: insert into borrower values('&customer_id','&loan_no')
new   1: insert into borrower values('c_03','1_93')

1 row created.

SQL> /
Enter value for customer_id: c_05
Enter value for loan_no: 1_17
old   1: insert into borrower values('&customer_id','&loan_no')
new   1: insert into borrower values('c_05','1_17')

1 row created.

SQL> /
Enter value for customer_id: c_03
Enter value for loan_no: 1_16
old   1: insert into borrower values('&customer_id','&loan_no')
new   1: insert into borrower values('c_03','1_16')
```

1 row created.

SQL> /
Enter value for customer_id: c_05
Enter value for loan_no: 1_14
old 1: insert into borrower values('&customer_id','&loan_no')
new 1: insert into borrower values('c_05','1_14')

1 row created.

Output:

```
SQL> select * from borrower;

CUSTOMER_ID LOAN
----------- ----
c_01        1_11
c_01        1_23
c_03        1_93
c_05        1_17
c_03        1_16
c_05        1_14

6 rows selected.
```

Result: Thus, the DML command was executed successfully for inserting values into tables.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Faculty	:	**Dr V VANI**	Slot	:	**L55+L56**
Name	:	**P VISHNU MOHAN KRISHNA**	Reg.no	:	**21MIS1118**

Ex.No.3 26.2.22

Data Manipulation Commands for inserting, deleting, updating and retrieving Tables and Transaction Control Commands

Aim : To create database and perform DML and TCL queries to retrieve information from the database.

Exercises: Update, Delete & Retrieval operation-

a. Modify the balance attribute alone such that it decreases the amount by 10% for the account table

```
SQL> select * from account;

ACCOUNT_NO  BRANCH_NAME                        BALANCE
----------- ------------------------------ ----------
019_28_3746 Perryridge                          15000
182_73_6091 Downtown                            23000
192_83_7465 Brighton                            18000
321_12_3123 Redwood                              5000
336_66_9999 Mianus                               5000
963_96_3963 Roundhill                            5000
376_66_9999 Mianus                               9000
963_96_3964 Mianus                              13000

8 rows selected.

SQL> update account set balance=balance-(balance*0.1);

8 rows updated.

SQL> select * from account;

ACCOUNT_NO  BRANCH_NAME                        BALANCE
----------- ------------------------------ ----------
019_28_3746 Perryridge                          13500
182_73_6091 Downtown                            20700
192_83_7465 Brighton                            16200
321_12_3123 Redwood                              4500
336_66_9999 Mianus                               4500
963_96_3963 Roundhill                            4500
376_66_9999 Mianus                               8100
963_96_3964 Mianus                              11700

8 rows selected.
```

b. Delete all the account tuples in the ‘Redwood’ branch.

```
SQL> select * from account;

ACCOUNT_NO  BRANCH_NAME                        BALANCE
----------- ------------------------------ ----------
019_28_3746 Perryridge                          13500
182_73_6091 Downtown                            20700
192_83_7465 Brighton                            16200
321_12_3123 Redwood                              4500
336_66_9999 Mianus                               4500
963_96_3963 Roundhill                            4500
376_66_9999 Mianus                               8100
963_96_3964 Mianus                              11700

8 rows selected.

SQL>  delete from account where branch_name='Redwood';

1 row deleted.

SQL> select * from account;

ACCOUNT_NO  BRANCH_NAME                        BALANCE
----------- ------------------------------ ----------
019_28_3746 Perryridge                          13500
182_73_6091 Downtown                            20700
192_83_7465 Brighton                            16200
336_66_9999 Mianus                               4500
963_96_3963 Roundhill                            4500
376_66_9999 Mianus                               8100
963_96_3964 Mianus                              11700

7 rows selected.
```

c. Delete all loans with loan amounts between 15000 to 20000.

```
SQL> select * from loan;

LOAN BRANCH_NAME                        AMOUNT
---- ------------------------------ ----------
1_11 Roundhill                            9000
1-14 Downtown                            15000
1_15 Perryridge                          15000
1_16 Perryridge                          13000
1_17 Downtown                            10000
1_23 Redwood                             20000
1_93 Mianus                                500

7 rows selected.

SQL> delete from loan where amount between 15000 and 20000;

3 rows deleted.

SQL> select * from loan;

LOAN BRANCH_NAME                        AMOUNT
---- ------------------------------ ----------
1_11 Roundhill                            9000
1_16 Perryridge                          13000
1_17 Downtown                            10000
1_93 Mianus                                500
```

d. Find the names of all branches in the loan relation.

```
SQL> select * from loan;

LOAN BRANCH_NAME                        AMOUNT
---- ------------------------------ ----------
1_11 Roundhill                            9000
1_16 Perryridge                          13000
1_17 Downtown                            10000
1_93 Mianus                                500

SQL> select branch_name from loan;

BRANCH_NAME
------------------------------
Roundhill
Perryridge
Downtown
Mianus
```

e. Display all the Customer names whose come from either pittsfield or stamford.

```
SQL>  select * from customer;

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_STREET CUSTOMER_CITY
----------- -------------------- --------------- ---------------
c_01        north                smith           rye
c_02        turner               putnam          stamford
c_03        johnson              alma            palo alto
c_04        curry                north           rye
c_05        jones                main            harrisdon
c_06        adoms                spring          pittsfield
c_08        hayes                main            harrison
c_09        williams             nassan          Princeton
c_07        lindsay              park            pittsfield

9 rows selected.

SQL> select customer_name from customer where customer_city in ('pittsfield','stamford');

CUSTOMER_NAME
-------------------
turner
adoms
lindsay
```

f. Find all loan numbers for loans made at the 'Perryridge' branch with loan amount greater than 1200.

```
SQL> select * from loan;

LOAN BRANCH_NAME                       AMOUNT
---- ------------------------------ ----------
1_16 Perryridge                          13000
1_17 Downtown                            10000
1_93 Mianus                                500
1_11 Roundhill                            9000

SQL> select loan_no from loan where branch_name='Perryridge' and amount>1200;

LOAN
----
1_16
```

g. Find loan numbers of those loans with loan amount between 10000 and 20000.

```
SQL>  select * from loan;

LOAN BRANCH_NAME                       AMOUNT
---- ------------------------------ ----------
1_16 Perryridge                          13000
1_17 Downtown                            10000
1_93 Mianus                                500
1_11 Roundhill                            9000

SQL>  select loan_no from loan where amount between 10000 and 20000;

LOAN
----
1_16
1_17
```

h. Display the customer name in alphabetical order.

```
SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_STREET CUSTOMER_CITY
----------- -------------------- --------------- ---------------
c_01        north                smith           rye
c_02        turner               putnam          stamford
c_03        johnson              alma            palo alto
c_04        curry                north           rye
c_05        jones                main            harrisdon
c_06        adoms                spring          pittsfield
c_08        hayes                main            harrison
c_09        williams             nassan          Princeton
c_07        lindsay              park            pittsfield

9 rows selected.

SQL> select customer_name from customer order by customer_name asc;

CUSTOMER_NAME
--------------------
adoms
curry
hayes
johnson
jones
lindsay
north
turner
williams

9 rows selected.
```

i. Display all the customer names ordered by customer city.

```
SQL> select * from customer;

CUSTOMER_ID CUSTOMER_NAME        CUSTOMER_STREET CUSTOMER_CITY
----------- -------------------- --------------- ---------------
c_01        north                smith           rye
c_02        turner               putnam          stamford
c_03        johnson              alma            palo alto
c_04        curry                north           rye
c_05        jones                main            harrisdon
c_06        adoms                spring          pittsfield
c_08        hayes                main            harrison
c_09        williams             nassan          Princeton
c_07        lindsay              park            pittsfield

9 rows selected.

SQL> select customer_name from customer order by customer_city asc;

CUSTOMER_NAME
--------------------
williams
jones
hayes
johnson
lindsay
adoms
curry
north
turner

9 rows selected.
```

j. Give a count of how many account holds are in each branch.

```
SQL> select * from account;

ACCOUNT_NO  BRANCH_NAME                      BALANCE
---------- ---------------------------- ----------
019_28_3746 Perryridge                        13500
182_73_6091 Downtown                          20700
192_83_7465 Brighton                          16200
336_66_9999 Mianus                             4500
963_96_3963 Roundhill                          4500
376_66_9999 Mianus                             8100
963_96_3964 Mianus                            11700

7 rows selected.

SQL> select branch_name,count(branch_name) from account group by branch_name;

BRANCH_NAME                    COUNT(BRANCH_NAME)
------------------------------ ------------------
Mianus                                          3
Roundhill                                       1
Perryridge                                      1
Brighton                                        1
Downtown                                        1
```

Result: Thus the DML and TCL commands are executed successfully for updating, deleting and retrieving the tables.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Faculty	:	**Dr V VANI**	Slot	:	**L55+L56**
Name	:	**P VISHNU MOHAN KRISHNA**	Regno	:	**21MIS1118**

Ex.No.4 04.03.2022

SQL- Aggregate Functions

Aim : To perform aggregate functions to retrieve information from the database.

Exercise:

Note: Create the tables with key constraints. All further exercises will be asked from the following schema only. Kindly keep the table with proper records.

student(regno, sname, CGPA)

course(ccode, cname, credits)

faculty(empid, fname, designation, salary)

course_student(regno, ccode)

course_faculty(fid, ccode)

Tables Created:

1. **Student Table**

```
SQL> select * from student;

REGNO      SNAME                      CGPA SCHOOL
---------- -------------------- ---------- ----------
A100       Arya                         10 Navodaya
A101       Anjaneya                      9 KVS
A102       Vishnu                       10 Navodaya
A103       Hemanth                       8 KVS
A104       Shiva                         8 Navodaya
A105       Ramana                        7 KVS

6 rows selected.
```

2. **Course Table**

```
SQL> select * from course;

CCODE CNAME                   CREDITS
----- -------------------- ----------
C1    Maths                         4
C2    Physics                       3
C4    DBMS                          5
C5    CSE                           3
C6    EVS                           2
C3    Soft skills                   1

6 rows selected.
```

3. **Faculty Table**

```
SQL> select * from faculty
  2  ;

EMPID FNAME                DESIGNATIO     SALARY
----- -------------------- ---------- ----------
1001  Pradeep              Senior         350000
1002  Anish                Senior         390000
1003  Rakesh               Junior         400000
1004  Ravindra             Assistant       50000
1005  Anila                Assistant       60000
1006  Ravikanth            Senior         500000

6 rows selected.
```

4. course_student table

```
SQL> select * from course_student;

REGNO      CCODE
---------- -----
A100       C1
A101       C2
A102       C3
A103       C4
A104       C5
A105       C6

6 rows selected.
```

5. course_faculty table

```
SQL> select * from course_faculty;

FID        CCODE
---------- -----
5001       C1
5002       C2
5003       C3
5004       C4
5005       C5
5006       C6

6 rows selected.
```

1. Add school attribute to student relation and update the value for all students.

Syntax:

```
SQL> select * from student;

REGNO      SNAME                     CGPA
---------- -------------------- ----------
A100       Arya                         10
A101       Anjaneya                      9
A102       Vishnu                       10
A103       Hemanth                       8
A104       Shiva                         8
A105       Ramana                        7

6 rows selected.

SQL> alter table student add(school varchar(10));

Table altered.
```

```
SQL> update student set school='Navodaya' where sname='Arya';

1 row updated.

SQL> update student set school='Navodaya' where sname='Vishnu';

1 row updated.

SQL> update student set school='Navodaya' where sname='Shiva';

1 row updated.

SQL> update student set school='Kendriya Vidyalaya' where sname='Anjaneya';
update student set school='Kendriya Vidyalaya' where sname='Anjaneya'
                          *
ERROR at line 1:
ORA-12899: value too large for column "SYSTEM"."STUDENT"."SCHOOL" (actual: 18,
maximum: 10)


SQL> update student set school='KVS' where sname='Anjaneya';

1 row updated.

SQL> update student set school='KVS' where sname='Hemanth';

1 row updated.

SQL> update student set school='KVS' where sname='Ramana';

1 row updated.
```

Output:

```
SQL> select * from student;

REGNO      SNAME                     CGPA SCHOOL
---------- -------------------- ---------- ----------
A100       Arya                         10 Navodaya
A101       Anjaneya                      9 KVS
A102       Vishnu                       10 Navodaya
A103       Hemanth                       8 KVS
A104       Shiva                         8 Navodaya
A105       Ramana                        7 KVS

6 rows selected.
```

2. Display the total credits of all the courses.

```
SQL> select * from course;

CCODE CNAME                  CREDITS
----- -------------------- ----------
C1    Maths                         4
C2    Physics                       3
C4    DBMS                          5
C5    CSE                           3
C6    EVS                           2
C3    Soft skills                   1

6 rows selected.

SQL> select sum(credits) from course;

SUM(CREDITS)
------------
          18
```

3. Identify the course(s) with maximum credits.

```
SQL> select cname from course where credits=(select max(credits) from course);

CNAME
--------------------
DBMS
```

4. List the name of the faculty who receive minimum salary.

```
SQL> select fname from faculty where salary=(select min(salary) from faculty);

FNAME
--------------------
Ravindra
```

5. List the count of faculty who receive more than 1 lakh as salary.

```
SQL> select count(salary) from faculty where salary>100000;

COUNT(SALARY)
-------------
            4
```

6. Display the average CGPA for each school.

```
SQL> select avg(cgpa),school from student group by school;

 AVG(CGPA) SCHOOL
---------- ----------
9.33333333 Navodaya
         8 KVS
```

7. Display the maximum and minimum CGPA for each school.

```
SQL> select max(cgpa),school from student group by school;

 MAX(CGPA) SCHOOL
---------- ----------
        10 Navodaya
         9 KVS

SQL> select min(cgpa),school from student group by school;

 MIN(CGPA) SCHOOL
---------- ----------
         8 Navodaya
         7 KVS
```

8. Display the maximum of average CGPA for each school.

```
SQL> select max (avg(cgpa)) from student group by school;

MAX(AVG(CGPA))
--------------
    9.33333333
```

9. Display the number of students whose credit is >9.
10. Display the number of students whose credit is >9 for each school

11. Display the average salary for each designation if the average exceeds 3 lakhs

```
SQL> select * from faculty;

EMPID FNAME                DESIGNATIO     SALARY
----- -------------------- ---------- ----------
1001  Pradeep              Senior         350000
1002  Anish                Senior         390000
1003  Rakesh               Junior         400000
1004  Ravindra             Assistant       50000
1005  Anila                Assistant       60000
1006  Ravikanth            Senior         500000

6 rows selected.

SQL> select designation,avg(salary) from faculty where salary>300000 group by designation;

DESIGNATIO AVG(SALARY)
---------- -----------
Senior      413333.333
Junior          400000
```

12. Display all students whose name start with ‘A’ and third character is ‘a’.

```
SQL> select sname from student where sname LIKE 'A_a%';

no rows selected
```

13. Display name, number and salary of those faculty whose name is 5 characters long and first three characters are ‘Ani’.

```
SQL> select empid,fname,designation,salary from faculty where fname LIKE 'Ani__%' and length(fname)=5;

EMPID FNAME                DESIGNATIO     SALARY
----- -------------------- ---------- ----------
1002  Anish                Senior         390000
1005  Anila                Assistant       60000
```

14. Display the non-null values of faculty and also faculty name second character should be ‘n’ and string should be 5 character long.

```
SQL> select empid,fname,designation,salary from faculty where fname is NOT NULL and
fname LIKE '_n%' and length(fname)=5;

EMPID FNAME                DESIGNATIO     SALARY
----- -------------------- ---------- ----------
1002  Anish                Senior         390000
1005  Anila                Assistant       60000
```

15. What will be output if you are giving LIKE predicate as ‘%_%’ ESCAPE ‘\’

```
SQL> select * from student where sname like '%\%%'ESCAPE'\';

no rows selected
```

Result: Thus, aggregate functions were executed successfully to retrieve information from the database.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Faculty	:	**Dr V VANI**	Slot	:	**L55+L56**
Name	:	**P VISHNU MOHAN KRISHNA**	REG	:	**21MIS1118**

Aim:
To add primary and foreign keys to tables.

Adding Constraints to the table

Constraints in SQL:

- NULL
- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL Constraints

- The NOT NULL constraint enforces a column to NOT accept NULL values.
- The NOT NULL constraint enforces a field to always contain a value. The following SQL enforces the "P_Id" column and the "LastName" column to not accept NULL values:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(25)
)
```

UNIQUE Constraints

- The UNIQUE constraint uniquely identifies each record in a database table.
- The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.
- Note that you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Eg

```
CREATE TABLE Persons
(
P_Id int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

To create a UNIQUE constraint on the "P_Id" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD UNIQUE (P_Id)
```

CHECK Constraints

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a single column it allows only certain values for this column.
- The following SQL creates a CHECK constraint on the "P_Id" column when the "Persons" table is created. The CHECK constraint specifies that the column "P_Id" must only include integers greater than 0.

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P_Id>0)
)
```

To add a CHECK constraint, use the following SQL:

```
ALTER TABLE Persons
ADD CHECK (P_Id>0)
```

DEFAULT Constraints

- The DEFAULT constraint is used to insert a default value into a column.
- The following SQL creates a DEFAULT constraint on the "City" column when the "Persons" table is created:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'chennai'
)
```

To create a DEFAULT constraint on the "City" column when the table is already created without city column, use the following SQL:

```
ALTER TABLE Persons
add (City varchar(25) DEFAULT 'Chennai')
```

To drop a DEFAULT constraint, use the following SQL:

```
ALTER TABLE Persons
ALTER City DROP DEFAULT
```

PRIMARY KEY Constraints

- The PRIMARY KEY constraint uniquely identifies each record in a database table.
- Each table should have a primary key, and each table can have only one primary key.
- The following SQL creates a PRIMARY KEY on the "P_Id" column when the "Persons" table is created:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
);
```

To add a PRIMARY KEY constraint, use the following SQL:

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id)
```

Note: If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared to not contain NULL values (when the table was first created).

To drop a PRIMARY KEY constraint, use the following SQL:

```
ALTER TABLE Persons
DROP PRIMARY KEY
```

FOREIGN KEY Constraints

- The "P_Id" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.
- The FOREIGN KEY constraint is used to prevent actions that would destroy link between tables.
- The FOREIGN KEY constraint also prevents that invalid data is inserted into the foreign key column, because it has to be one of the values contained in the table it points to
- A FOREIGN KEY in one table points to a PRIMARY KEY in another table.
- Let's illustrate the foreign key with an example. Look at the following two tables:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The "Orders" table:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Note that the "P_Id" column in the "Orders" table points to the "P_Id" column in the "Persons" table. The "P_Id" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

To create a FOREIGN KEY constraint on the "P_Id" column when the "Orders" table is already created, use the following SQL:

```
ALTER TABLE Orders
ADD FOREIGN KEY
REFERENCES Persons(P_Id)
```

Adding & Deleting the Constraints to the Table using constraint Names:

This example demonstrates **how to specify constraints with constraint name** during table creation

```
create table salary_table (
    fiscal_year varchar2(4) constraint nn_sal_year not null,
    salary number constraint chk_sal_sal check (salary > 0 ),
    fullname varchar2(64) constraint unq_sal_fullname unique,
    emp_id varchar2(12) constraint fk_sal_emp_id references emp1(eno));
```

To **disable a constraint**, use the alter table command. To **enable a disabled constraint**, again use the alter table command. The following examples disables and then re-enables the salary check condition

```
alter table salary_table disable constraint chk_salary_salary;
alter table salary_table enable constraint chk_salary_salary;
```

The following example demonstrates **how to drop the constraints using constraint name**,

```
alter table salary_table drop constraint unq_sal_fullname;
alter table salary_table drop constraint chk_sal_sal;
alter table salary_table drop constraint nn_sal_year;
alter table salary_table drop constraint fk_sal_emp_id;
```

DELETE COMMAND

```
SQL> delete from person where lastname='Kumar';
2 rows deleted.
SQL> select * from person;
    PID LASTNAME FIRSTNAME ADDRESS AGE
    ---- ---------- ---------- ---------
    1 Prettina Anne BAngalore 14
    2 Benitto Anish Trichy 24
    3 Raj Anita Chennai 27
    4 Hinn Benny Britain 55
    5 Prakash Bhaskar Assam 40

SQL> rollback;
Rollback complete.

SQL> select * from person;
    PID LASTNAME FIRSTNAME ADDRESS AGE
    ----- ---------- ---------- ---------- -----
```

```
        1 Prettina Anne BAngalore 14
        2 Benitto Anish Trichy 24
        3 Raj Anita Chennai 27
        4 Kumar Ashok Coimbatore 30
        5 Hinn Benny Britain 55
        6 Prakash Bhaskar Assam 40
        7 Kumar Chander Coimbatore 45
7 rows selected.
SQL> savepoint s1;
Savepoint created.
SQL> delete from person;
7 rows deleted.
SQL> select * from person;
no rows selected
SQL> rollback to savepoint s1;
Rollback complete.
SQL> select * from person;
        PID LASTNAME FIRSTNAME ADDRESS AGE
        ---- ---------- ---------- ----- -----
        1 Prettina Anne BAngalore 14
        2 Benitto Anish Trichy 24
        3 Raj Anita Chennai 27
        4 Kumar Ashok Coimbatore 30
        5 Hinn Benny Britain 55
        6 Prakash Bhaskar Assam 40
        7 Kumar Chander Coimbatore 45
    7   rows selected.
```

1. Add primary keys to all the tables for the specified attributes

A) **NAME**: branch

FIELDS	**DATATYPE**
branch_name	varchar2(30) primary key
branch_city	varchar2(30)
assets	number(8,2)

```
SQL> alter table branch1059 add primary key(branch_name);

Table altered.
```

B) **NAME**: account

FIELDS	**DATATYPE**
account_no	varchar2(11) primary key
branch_name	varchar2(30)
balance	number(8)

```
SQL> alter table account1059 add primary key(account_no);
```

```
Table altered.
```

C) **NAME**: customer

FIELD	**DATATYPE**
customer_id	varchar2(11) primary key
customer_name	varchar2(20)
customer_street	varchar2(15)
customer_city	varchar2(15)

```
SQL> alter table customer1059 add primary key(account_no);

Table altered.
```

D) **NAME**: loan

FIELDS	**DATATYPE**
loan_no	varchar2(4) primary key
branch_name	varchar2(30)
amount	number(8,2)

```
SQL> alter table loan1059 add primary key(loan_no);

Table altered.
```

2. **Add foreign keys to the following tables for the specified attributes with mentioned reference table**

B) **NAME**: account

FIELDS	**DATATYPE**
account_no	varchar2(11) primary key
branch_name	varchar2(30) references branch(branch_name)
balance	number(8)

```
SQL> alter table account1059 add foreign key (branch_name)
references branch (branch_name);

Table altered.
```

C) **NAME**: depositor

FIELD	**DATATYPE**
customer_id	varchar2(11)references customer (customer_id)
account_no	varchar2(11)references account (account_no)

```
SQL> alter table depositor1059 add foreign key (customer_id)
references customer1059 (customer_id);

Table altered.

SQL> alter table depositor1059 add foreign key (customer_id)
references account1059 (account_no);
```

```
Table altered.
```

D) **NAME**: loan

FIELDS	**DATATYPE**
loan_no	varchar2(4) primary key
branch_name	varchar2(30) references branch (branch_name) (Create constraint with constraint name)
amount	number (8,2)

```
 SQL>alter table loan1059 add constraint branch_name foreign key
(branch_name) references branch1059 (branch_name);

Table altered.
```

3. Drop foreign key constraint from loan table

```
SQL> alter table loan1104 drop constraint branch_name;

Table altered.
```

4. Set loan_no attribute of borrower table as foreign key with cascade deletion, which refers to loan table loan_no column.

5. Add foreign key for the customer_id of borrower table which refers to customer table with constraint name.

6. Create the Database Schema for a Employee-pay scenario

a) employee(emp_id : integer, emp_name: string, address: string, city: string)

```
SQL> create table employee(emp_id number(20), emp_name
varchar(20),address varchar(30),city varchar(30));



Table created.
```

b) department(dept_id: integer, dept_name:string)

```
SQL> create table department(dep_id number(10),dep_name
varchar(40));

Table created.
```

c) paydetails(emp_id : integer, dept_id: integer, basic: integer, deductions: integer, additions: integer, DOJ: date)

```
SQL> create table paydetails(emp_id number(10),dep_is
number(10),basic number(10),education number(10),addition
number(10));



Table created.
```

d) payroll(emp_id : integer, pay_date: date)

```
SQL> create table payroll(emp_id number(10), pay_date date);

Table created.
```

For the above schema, perform the following:

7. Create PRIMARY KEY for employee(emp_id) and department(dept_id).

```
SQL> alter table employee add primary key(emp_id);



Table altered.
```

8. Creates a DEFAULT constraint on the "City" column of employee table.

```
SQL> alter table employee modify (city varchar(30) default
'chennai');



Table altered.
```

9. Create NOT NULL for dept_id on department table.

```
SQL> alter table department modify (dep_id not null);



Table altered.
```

10. Create NOT NULL for basic in pay details.

```
SQL> alter table paydetails modify(basic not null);

Table altered.
```

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Name	:	**P VISHNU MOHAN KRISHNA**	Regno	:	**21MIS1118**

Ex.No.6 06.04.2022

Set Operations and Sub Queries

Aim : To perform set operations and sub-queries to retrieve information from the database.

Tables created:

1) student(regno, sname, cgpa)

```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGNO                                     NOT NULL VARCHAR2(4)
 SNAME                                              VARCHAR2(20)
 CGPA                                               VARCHAR2(3)

SQL> select * from student;

REGN SNAME                CGP
---- -------------------- ---
1001 ravi                 9.5
1002 raju                 9.3
1003 rakesh               8.1
1004 rajesh               8.6
1005 ramesh               7.5
```

2) course(ccode, cname, credits)

```
SQL> desc course;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CCODE                                     NOT NULL VARCHAR2(7)
 CNAME                                              VARCHAR2(20)
 CREDITS                                            NUMBER(2)

SQL> select * from course;

CCODE   CNAME                  CREDITS
------- -------------------- ----------
CSE3001 CSE                           4
CSE3002 CSE                           4
CSE3003 CSE                           3
CSE3004 CSE                           3
CSE3005 CSE                           4
```

3) faculty(empid, fname, designation)

```
SQL> desc faculty;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPID                                     NOT NULL VARCHAR2(4)
 FNAME                                              VARCHAR2(20)
 DESIGNATION                                        VARCHAR2(20)

SQL> select * from faculty;

EMPI FNAME                DESIGNATION
---- -------------------- --------------------
5001 avinash              professor
5002 arun                 associate
5003 vani                 senior
5004 sukesh               senior
5005 rahul                senior
```

4) course_student(regno, ccode)

```
SQL> desc course_student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGNO                                              VARCHAR2(4)
 CCODE                                              VARCHAR2(7)

SQL> select * from course_student;

REGN CCODE
---- -------
1001 CSE3001
1002 CSE3002
1003 CSE3003
1004 CSE3004
1005 CSE3005
```

5) course_faculty(empid, ccode)

```
SQL> desc course_faculty;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPID                                              VARCHAR2(4)
 CCODE                                              VARCHAR2(7)

SQL> select * from course_faculty;

EMPI CCODE
---- -------
5001 CSE3001
5002 CSE3002
5003 CSE3003
5004 CSE3004
5005 CSE3005
```

6) student_faculty(regno,empid,designation)

```
SQL> desc student_faculty;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REGNO                                              VARCHAR2(4)
 EMPID                                              VARCHAR2(4)
 DESIGNATION                                        VARCHAR2(20)

SQL> select * from student_faculty;

REGN EMPI DESIGNATION
---- ---- --------------------
1001 5001 professor
1002 5002 associate
1003 5003 senior
1004 5004 senior
1005 5005 senior
```

Write SQL Queries for the following:

Part A – Set Operations

1. List the courses which are handled by a faculty and have students registered it.

```
SQL> select ccode from course_faculty intersect select ccode from course_student;

CCODE
-------
CSE3001
CSE3002
CSE3003
CSE3004
CSE3005
```

2. List the courses which are handled by a faculty and have no students registered in it.

```
SQL> select ccode from course_faculty minus select ccode from course_student;

no rows selected
```

3. List the courses which are not handled by any faculty.

```
SQL> select ccode from course minus select ccode from course_faculty;

no rows selected
```

4. List the students who haven't registered for any course.

```
SQL> select regno from student minus select regno from course_student;

no rows selected
```

5. List the faculty who are not handling any courses.

```
SQL> select empid from faculty minus select empid from course_faculty;

no rows selected
```

Part-B Sub Queries

1. List the students' regno who took courses with credits more than 3.

```
SQL> select regno from course_student where ccode in (select ccode from course where credits>3);

REGN
----
1001
1002
1005
```

2. List the faculty details who are not Professors and Associate Professors

```
SQL> select * from faculty where designation not in ('professor','associate');

EMPI FNAME                DESIGNATION
---- -------------------- --------------------
5003 vani                 senior
5004 sukesh               senior
5005 rahul                senior
```

3. Increase the CGPA by 0.5 for the students who have taken CSE3002

Before updating:

```
SQL> select * from student;

REGN SNAME                CGP
---- -------------------- ---
1001 ravi                 9.5
1002 raju                 9.3
1003 rakesh               8.1
1004 rajesh               8.6
1005 ramesh               7.5
```

After updating:

```
SQL> update student set cgpa=cgpa+0.5 where regno in(select regno from course_student where ccode='CSE3002');

1 row updated.

SQL> select * from student;

REGN SNAME                CGP
---- -------------------- ---
1001 ravi                 9.5
1002 raju                 9.8
1003 rakesh               8.1
1004 rajesh               8.6
1005 ramesh               7.5
```

4. Update the credits of the courses except CSE3002.

Before Updating:

```
SQL> select * from course;

CCODE   CNAME                  CREDITS
------- -------------------- ----------
CSE3001 CSE                         4
CSE3002 CSE                         4
CSE3003 CSE                         3
CSE3004 CSE                         3
CSE3005 CSE                         4
```

After updating:

```
SQL> update course set credits=3 where ccode in (select ccode from course where ccode!='CSE3002');

4 rows updated.

SQL> select * from course;

CCODE   CNAME                  CREDITS
------- -------------------- ----------
CSE3001 CSE                         3
CSE3002 CSE                         4
CSE3003 CSE                         3
CSE3004 CSE                         3
CSE3005 CSE                         3
```

5. List the student register number with maximum CGPA.

```
SQL> select regno from student where cgpa=(select max(cgpa) from student);

REGN
----
1002
```

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Name	:	**P VISHNU MOHAN KRISHNA**	Regno	:	**21MIS1118**

Ex.No.7 09.04.2022

PL/SQL Basics

Aim : To develop and execute PL/SQL .

LAB EXCERCISES:

1. Write a PL/SQL structure to display 'Hello World'.

```
SQL> set serveroutput on;
SQL> begin
  2  dbms_output.put_line('Hello World');
  3  end;
  4  /
Hello World

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL structure to find sum of two numbers.

```
SQL> Declare
  2  a integer;
  3  b integer;
  4  c integer;
  5  Begin
  6  a:=&a;
  7  b:=&b;
  8  c:=a+b;
  9  dbms_output.put_line(c);
 10  End;
 11  /
Enter value for a: 10
old   6: a:=&a;
new   6: a:=10;
Enter value for b: 15
old   7: b:=&b;
new   7: b:=15;
25

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL structure to find the greatest number among three numbers using if-else.

```
PL/SQL procedure successfully completed.

SQL> Declare
  2  a number;
  3  b number;
  4  c number;
  5  Begin
  6  a:=&a;
  7  b:=&b;
  8  c:=&c;
  9  if(a>b and a>c) then
 10  dbms_output.put_line('a is greater');
 11  elsif(b>a and b>c) then
 12  dbms_output.put_line('b is greater');
 13  else
 14  dbms_output.put_line('c is greater');
 15  end if;
 16  End;
 17  /
Enter value for a: 10
old   6: a:=&a;
new   6: a:=10;
Enter value for b: 15
old   7: b:=&b;
new   7: b:=15;
Enter value for c: 20
old   8: c:=&c;
new   8: c:=20;
c is greater

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL structure to find addition, subtraction, multiplication and division of two numbers using 'case'.

```
SQL> Declare
  2  a number:=20;
  3  b number:=10;
  4  c number;
  5  ch varchar(20);
  6  Begin
  7  ch:='&ch';
  8  case ch
  9  when 'add' then
 10  c:=a+b;
 11  dbms_output.put_line('Addition is '||c);
 12  when 'subtract' then
 13  c:=a-b;
 14  dbms_output.put_line('Subtraction is '||c);
 15  when 'multiply' then
 16  c:=a*b;
 17  dbms_output.put_line('Multiplication is '||c);
 18  when 'divide' then
 19  c:=a/b;
 20  dbms_output.put_line('Division is '||c);
 21  end case;
 22  END;
 23  /
```

```
SQL> set serveroutput on;
SQL> /
Enter value for ch: add
old   7: ch:='&ch';
new   7: ch:='add';
Addition is 30

PL/SQL procedure successfully completed.

SQL> /
Enter value for ch: subtract
old   7: ch:='&ch';
new   7: ch:='subtract';
Subtraction is 10

PL/SQL procedure successfully completed.

SQL> /
Enter value for ch: multiply
old   7: ch:='&ch';
new   7: ch:='multiply';
Multiplication is 200

PL/SQL procedure successfully completed.

SQL> /
Enter value for ch: divide
old   7: ch:='&ch';
new   7: ch:='divide';
Division is 2

PL/SQL procedure successfully completed.
```

Result: Hence, PL/SQL programs are developed and executed successfully.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Name	:	**P VISHNU MOHAN KRISHNA**	Regno	:	**21MIS1118**

Ex.No.8 23.04.2022

PL/SQL –Cursors

Aim : To develop and execute PL/SQL-concept of cursors .

Basic Information:

A cursor **holds the rows (one or more) returned by a SQL statement**. The set of rows the cursor holds is referred to as the active set. You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time.

Cursors are used to store Database Tables. There are 2 types of Cursors:

Implicit Cursors:
Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

Explicit Cursors :
Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

1. **Declare Cursor Object.**
   Syntax : DECLARE cursor_name CURSOR FOR SELECT * FROM table_name

   DECLARE s1 CURSOR FOR SELECT * FROM studDetails

2. **Open Cursor Connection.**
   Syntax : OPEN cursor_connection

OPEN s1

3. **Fetch Data from cursor.**

There are total 6 methods to access data from cursor. They are as follows :
FIRST is used to fetch only the first row from cursor table.
LAST is used to fetch only last row from cursor table.
NEXT is used to fetch data in forward direction from cursor table.
PRIOR is used to fetch data in backward direction from cursor table.
ABSOLUTE n is used to fetch the exact n^{th} row from cursor table.
RELATIVE n is used to fetch the data in incremental way as well as decremental way.

Syntax : FETCH NEXT/FIRST/LAST/PRIOR/ABSOLUTE n/RELATIVE n FROM cursor_name

FETCH FIRST FROM s1

FETCH LAST FROM s1

FETCH NEXT FROM s1

FETCH PRIOR FROM s1

FETCH ABSOLUTE 7 FROM s1

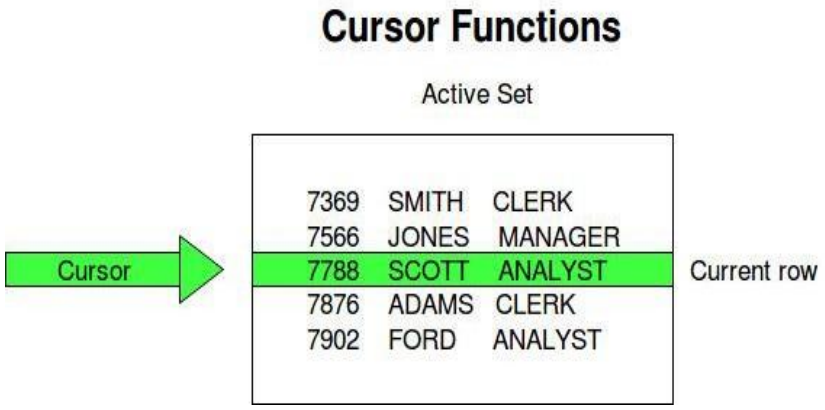
FETCH RELATIVE -2 FROM s1

4. **Close cursor connection.**
   Syntax : CLOSE cursor_name

CLOSE s1

5. **cursor memory.**
   Syntax : DEALLOCATE cursor_name

DEALLOCATE s1



LAB EXCERCISES:

1. Write a PL/SQL –cursor program to list the name of the students who have more than 9 CGPA.

```
SQL> select * from student;

REGN SNAME                CGP
---- -------------------- ---
1001 ravi                 10
1002 raju                 9.8
1003 rakesh               8.1
1004 rajesh               8.6
1005 ramesh               7.5
```

```
SQL> set serveroutput on;
SQL> DECLARE
  2  s_sname student.sname%type;
  3  CURSOR s_student is SELECT sname FROM student    WHERE cgpa>9;
  4  BEGIN
  5  OPEN s_student;
  6  LOOP
  7  FETCH s_student into s_sname;
  8  EXIT WHEN s_student%notfound;
  9  dbms_output.put_line(s_sname);
 10  END LOOP;
 11  CLOSE s_student;
 12  END;
 13  /
ravi
raju

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL Cursor to update the CGPA of the student with regno 1003.

<u>Before updating</u>:

```
SQL> select * from student;

REGN SNAME                CGP
---- -------------------- ---
1001 ravi                 10
1002 raju                 9.8
1003 rakesh               8.1
1004 rajesh               8.6
1005 ramesh               7.5
```

After updating CGPA:

```
SQL> declare
  2  s_cgpa student.cgpa%type;
  3  cursor s_student is select cgpa from student where regno=1003 for update of cgpa;
  4  begin
  5  open s_student;
  6  loop
  7  fetch s_student into s_cgpa;
  8  exit when s_student%notfound;
  9  update student set cgpa=10 where current of s_student;
 10  end loop;
 11  close s_student;
 12  end;
 13  /

PL/SQL procedure successfully completed.

SQL> select * from student;

REGN SNAME                CGP
---- -------------------- ---
1001 ravi                 9.5
1002 raju                 9.8
1003 rakesh               10
1004 rajesh               8.6
1005 ramesh               7.5
```

CGPA of regno 1003 is updated from **8.1 to 10**

Result: PL/SQL programs are developed and executed successfully for cursors concept.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Programme	:	**MIS**	Semester	:	**Winter 21-22**
Course	:	**Database Management Systems (Embedded Lab)**	Code	:	**SWE1004**
Name	:	**P VISHNU MOHAN KRISHNA**	Regno	:	**21MIS1118**

Practice Session 21.05.2022

Tables Created:

1. **Students Table**

```
SQL> desc students;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDID                                             VARCHAR2(20)
 NAME                                               VARCHAR2(20)
 DEPT                                               VARCHAR2(20)
 GRADE                                              NUMBER(3)

SQL> select * from students;

STUDID               NAME                 DEPT                      GRADE
-------------------- -------------------- -------------------- ----------
1001                 vikas                CSE                           9
1002                 vishnu               EEE                          10
1003                 arnab                CSE                           8
1004                 arun                 EEE                           9
1005                 saravana             CSE                          10
```

2. Employees Table

```
SQL> desc employees;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPID                                              VARCHAR2(20)
 EMPNAME                                            VARCHAR2(20)
 EMPDEPT                                            VARCHAR2(20)
 SALARY                                             NUMBER(8)
 EXPEREINECE                                        NUMBER(3)
 PROJECT                                            VARCHAR2(8)

SQL> select * from employees;

EMPID                EMPNAME              EMPDEPT                  SALARY
-------------------- -------------------- -------------------- ----------
EXPEREINECE PROJECT
----------- --------
5001                 aravind              CSE                       25000
         11 1

5002                 ashok                CSE                       30000
         12 2

5003                 aakash               EEE                       40000
          9 1


EMPID                EMPNAME              EMPDEPT                  SALARY
-------------------- -------------------- -------------------- ----------
EXPEREINECE PROJECT
----------- --------
5004                 alok                 Mechanical                25000
         11 2

5005                 nagesh               ECE                       20000
          8 1
```

1. Display the student details of CSE department.

```
SQL> set serveroutput on;
SQL> DECLARE
  2  CURSOR c is select * from students where dept='CSE';
  3  BEGIN
  4  for i in c
  5  LOOP
  6  dbms_output.put_line('Student '||i.name||' ('||i.studID||') '||'belongs to '||i.dept||' have '||i.grade||' Grade');
  7  END LOOP;
  8  END;
  9  /
Student vikas (1001) belongs to CSE have 9 Grade
Student arnab (1003) belongs to CSE have 8 Grade
Student saravana (1005) belongs to CSE have 10 Grade

PL/SQL procedure successfully completed.
```

2. display the details of students whose grade >9

```
SQL> set serveroutput on;
SQL> DECLARE
  2  CURSOR c is select * from students where grade>9;
  3  BEGIN
  4  for i in c
  5  LOOP
  6  dbms_output.put_line('Student '||i.name||' ('||i.studID||') '||'belongs to '||i.dept||' have '||i.grade||' Grade');
  7  END LOOP;
  8  END;
  9  /
Student vishnu (1002) belongs to EEE have 10 Grade
Student saravana (1005) belongs to CSE have 10 Grade

PL/SQL procedure successfully completed.
```

3. Update the department name from “CSE” to “Computer Science”

<u>Before Updating:</u>

```
SQL> select * from students;

STUDID               NAME                 DEPT                      GRADE
-------------------- -------------------- -------------------- ----------
1001                 vikas                CSE                           9
1002                 vishnu               EEE                          10
1003                 arnab                CSE                           8
1004                 arun                 EEE                           9
1005                 saravana             CSE                          10
```

<u>After Updating:</u>

```
SQL> set serveroutput on;
SQL> DECLARE
  2  s_dept students.dept%type;
  3  CURSOR s_students is select dept from students where dept='CSE' for update of dept;
  4  BEGIN
  5  OPEN s_students;
  6  LOOP
  7  FETCH s_students into s_dept;
  8  EXIT WHEN s_students%notfound;
  9  update students set dept='Computer Science' where current of s_students;
 10  END LOOP;
 11  CLOSE s_students;
 12  END;
 13  /

PL/SQL procedure successfully completed.

SQL> select * from students;

STUDID               NAME                 DEPT                      GRADE
-------------------- -------------------- -------------------- ----------
1001                 vikas                Computer Science              9
1002                 vishnu               EEE                          10
1003                 arnab                Computer Science              8
1004                 arun                 EEE                           9
1005                 saravana             Computer Science             10
```

4. create a trigger for the student table on insert

```
SQL> create or replace trigger t1 after insert on student for each row
  2  BEGIN
  3  dbms_output.put_line('Inserted Successfully');
  4  END;
  5  /

Trigger created.
```

5. increase the salary of employees by 5000 whose experience >10

```
SQL> declare
  2  e_salary employees.salary%type;
  3  cursor e_employees is select salary from employees where expereinece>10 for update of salary;
  4  begin
  5  open e_employees;
  6  loop
  7  fetch e_employees into e_salary;
  8  exit when e_employees%notfound;
  9  update employees set salary=salary+5000 where current of e_employees;
 10  end loop;
 11  close e_employees;
 12  end;
 13  /

PL/SQL procedure successfully completed.
```

6. display the details of the rows affected in the first query

<u>Before updating:</u>

```
SQL> select * from employees;

EMPID                EMPNAME              EMPDEPT                  SALARY
-------------------- -------------------- -------------------- ----------
EXPEREINECE PROJECT
----------- --------
5001                 aravind              CSE                       25000
         11 1

5002                 ashok                CSE                       30000
         12 2

5003                 aakash               EEE                       40000
          9 1


EMPID                EMPNAME              EMPDEPT                  SALARY
-------------------- -------------------- -------------------- ----------
EXPEREINECE PROJECT
----------- --------
5004                 alok                 Mechanical                25000
         11 2

5005                 nagesh               ECE                       20000
          8 1
```

After updating:

```
SQL> select * from employees;

EMPID                EMPNAME              EMPDEPT                  SALARY
-------------------- -------------------- -------------------- ----------
EXPEREINECE PROJECT
----------- --------
5001                 aravind              CSE                       30000
         11 1

5002                 ashok                CSE                       35000
         12 2

5003                 aakash               EEE                       40000
          9 1


EMPID                EMPNAME              EMPDEPT                  SALARY
-------------------- -------------------- -------------------- ----------
EXPEREINECE PROJECT
----------- --------
5004                 alok                 Mechanical                30000
         11 2

5005                 nagesh               ECE                       20000
          8 1
```

7. display details of employees working on Project1

```
SQL> set serveroutput on;
SQL> DECLARE
  2  CURSOR c is select * from employees where project=1;
  3  BEGIN
  4  for i in c
  5  LOOP
  6  dbms_output.put_line(i.empid||' '||i.empname||' '||i.empdept||' '||i.salary||' '||i.expereinece||' '||i.project);
  7  END LOOP;
  8  END;
  9  /
5001 aravind CSE 25000 11 1
5003 aakash EEE 40000 9 1
5005 nagesh ECE 20000 8 1
```

8. create a trigger for table employee on delete.

```
SQL> set serveroutput on;
SQL> create or replace trigger t after delete on employees for each row
  2  begin
  3  dbms_output.put_line('Deleted Successfully');
  4  end;
  5  /

Trigger created.

SQL> delete from employees where empid='5005';
Deleted Successfully

1 row deleted.
```

SWE 1004 Database Management Systems Lab

Ex. No: 10

Write a PL/SQL using VARRAY to store the names of 5 cities. Initialize VARRAY with size 10. Use the extend function when you store the values for 5 cities. Print the no of elements, first element, last element in the array. Remove the last element from the array. Insert two more cities into the array and print the final list of cities in the array.

PL/SQL Code and Output:

Run SQL Command Line

```
SQL> set serveroutput on;
SQL> declare
  2  type cities is varray(10) of varchar(25);
  3  city_name cities := cities();
  4  begin
  5  city_name.extend(5);
  6  city_name(1) := 'Eluru';
  7  city_name(2) := 'Kakinada';
  8  city_name(3) := 'Varanasi';
  9  city_name(4) := 'Hyderabad';
 10  city_name(5) := 'Vijayawada';
 11  dbms_output.put_line('The no.of elements in varray are '||city_name.count);
 12  dbms_output.put_line('The First element in varray is '||city_name(city_name.first));
 13  dbms_output.put_line('The Last element in varray is '||city_name(city_name.last));
 14  dbms_output.put_line('Elements before removing last element are: ');
 15  for i in 1..city_name.count
 16  loop
 17  dbms_output.put(city_name(i)||', ');
 18  end loop;
 19  dbms_output.new_line;
 20  city_name.trim;
 21  dbms_output.put_line('Elements after removing last element are: ');
 22  for i in 1..city_name.count
 23  loop
 24  dbms_output.put(city_name(i)||', ');
 25  end loop;
 26  dbms_output.new_line;
 27  city_name.extend(2);
 28  city_name(5) := 'Vellore';
 29  city_name(6) := 'Chennai';
 30  for i in 1..city_name.count
 31  loop
 32  dbms_output.put(city_name(i)||', ');
 33  end loop;
 34  dbms_output.new_line;
 35  end;
 36  /
The no.of elements in varray are 5
The First element in varray is Eluru
The Last element in varray is Vijayawada
Elements before removing last element are:
Eluru, Kakinada, Varanasi, Hyderabad, Vijayawada,
Elements after removing last element are:
Eluru, Kakinada, Varanasi, Hyderabad,
Eluru, Kakinada, Varanasi, Hyderabad, Vellore, Chennai,

PL/SQL procedure successfully completed.
```