IT5312
DATABASE MANAGEMENT SYSTEM
(DBMS) LABORATORY

A PRACTICAL RECORD

Submitted by

Gowtham Rajasekaran
2022506084
B. Tech (3/8)

DEPARTMENT OF INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY- CHENNAI

CHENNAI - 600 044.

November / December 2023

BONAFIDE CERTIFICATE

Name	: Gowtham Rajasekaran
Reg. No	: 2022506084
Subject	: IT5312 Database Management System Laboratory
Department	: Information Technology

Certified to the bonafide record of practical work done by Mr Gowtham Rajasekaran

in the **IT5312 DBMS Laboratory** during the period August 2023 to November 2023.

Date: 26/12/2023

Dr. D. Vivekanandan
STAFF-IN-CHARGE

Submitted for the Practical Examination held on: 26/12/2023

Internal Examiner External Examiner

IT5312 DBMS LABORATORY (R2019)

COURSE OBJECTIVES

CO1	To learn and implement important commands in SQL.
CO2	To learn the usage of nested and joint queries.
CO3	To understand functions, procedures, and procedural extensions of databases.
CO4	To be familiar with the use of a front-end tool for GUI based application development.

COURSE OUTOCOMES

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML and DCL commands.
CO3	Realize database design using 3NFand BCNF.
CO4	Use advanced features such as stored procedures and triggers and incorporate in GUI based application development.
CO5	Create XML database and validate with meta – data (XML schema).
CO6	Create and manipulate data using NOSQL database.

CO-PO MATRIX

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3	3	3	3	3	-	-	-	1	-	-	2	1	1	1
CO2	3	3	3	3	3	-	-	-	1	-	-	2	1	1	1
CO3	3	3	3	2	2	1	-	1	3	2	2	2	3	3	1
CO4	3	3	3	3	3	-	-	-	1	-	-	2	2	2	2
CO5	3	3	3	3	3	-	-	-	1	-	-	2	1	1	1
CO6	3	3	3	3	3	-	-	-	1	-	-	2	2	2	2

List of Experiments

Exp No	01
Date	05/09/2023

INTRODUCTION TO DATABASE DESIGN

ENTITY RELATIONSHIP DIAGRAM & SCHEMAS

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML and DCL commands.
PO1	Engineering Knowledge.
PO2	Problem Analysis.
PO3	Design and Development of solution.
PO9	Individual and TeamWork.
PO10	Communication.

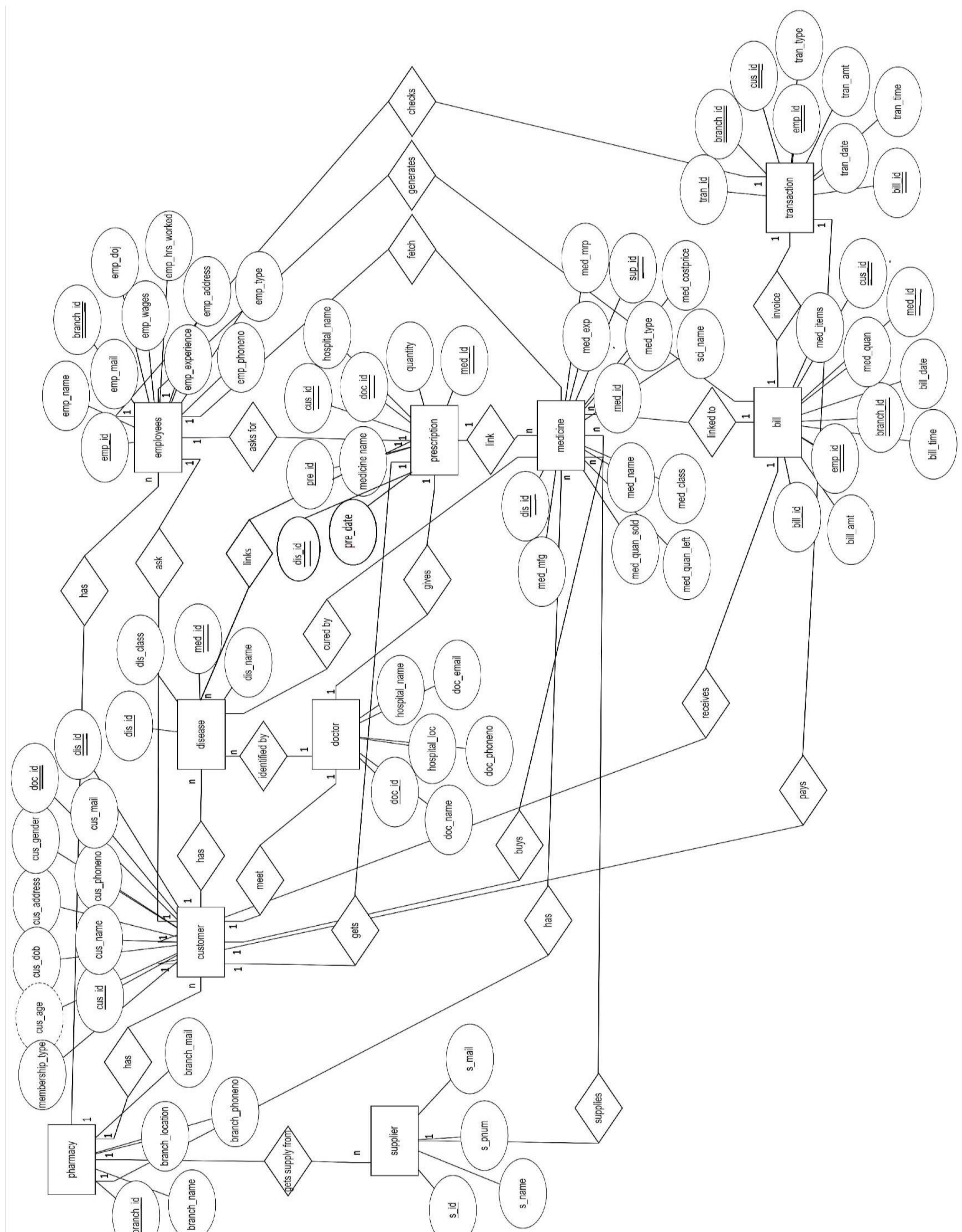
AIM :

To construct an ER diagram and table schema for offline pharmacy database management system.

STEPS :

1. Build a pharmacy.
2. Ask suppliers to supply us with medicine.
3. Pharmacy has customers who have diseases.
4. Customer meets doctors who identify the diseases and give prescription.
5. Pharmacy has employees.
6. Patients ask employees to get them the medicine.
7. Employee gives the medicine, generates bill and gives it to customers.
8. Transaction takes place where the customers pay to the pharmacy through employees for the medicines they buy.

ER DIAGRAM ABOUT PHARMACY MANAGEMENT SYSTEM :



RELATIONAL SCHEMAS:

1. PHARMACY (branch_id, branch_location, branch_name, branch_mail, branch_phoneno)
2. SUPPLIER (s_id, s_name, s_pnum, s_mai l)
3. CUSTOMER (cus_id, cus_name, cus_phoneno, cus_mail, doc_id, cus_gender, cus_dob, cus_age, membership_type, dis_id, cus_address)
4. DOCTOR (doc_id, doc_name, doc_phoneno, doc_email, hospital_name, hospital_location)
5. DISEASE (dis_id, dis_class, med_id, dis_name)
6. PRESCRIPTION (pre_id, medicine_name, cus_id, doc_id, quantity, med_id, hospital_name, dis_id, pre_date)
7. EMPLOYEE (emp_id, emp_name, emp_mail, emp_wages, emp_doj, emp_hrs_worked, emp_experience, emp_address, emp_phoneno, emp_type, branch_id)
8. MEDICINE (med_id, med_name, med_class, med_quan_sold, med_quan_left, med_mfg, med_exp, med_mrp, sup_id, med_costprice, sci_name, med_type)
9. BILL (bill_id, bill_amt, emp_id, branch_id, bill_time, bill_date, med_quan, med_items, med_id, cus_id)
10. TRANSACTION (tran_id, branch_id, cus_id, tran_type, emp_id, tran_date, tran_amt, tran_time, bill_id)

DATABASE SCHEMA:

OFFLINE PHARMACY MANAGEMENT SYSTEM (PHARMACY, SUPPLIER, CUSTOMER, DOCTOR, DISEASE, PRESCRIPTION, EMPLOYEE, MEDICINE, BILL, TRANSACTION)

RESULT:

Hence ER diagram and table schema for offline pharmacy management system is successfully constructed.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	02
Date	12/09/2023

INTRODUCTION TO DATABASE DESIGN

BASIC SQL DDL AND DML COMMANDS

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML and DCLcommands.
PO1	Engineering Knowledge.
PO2	Problem Analysis.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and TeamWork.
PO10	Communication.

AIM:

To create a database table for customer and theatre, add constraints (primary key, unique, check, not null), insert rows, update and delete rows using SQL DDL and DML commands.

EXAMPLE FOR SUPPLIERS ENTITY

SQL COMMANDS:

```
SET LINESIZE 100;
SET AUTOCOMMIT ON;
```

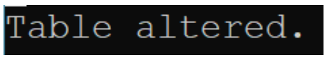
CREATING TABLE (DDL): (NOT NULL, UNIQUE, CHECK CONSTRAINTS)

```
CREATE TABLE suppliers (
    s_id NUMBER(30) PRIMARY KEY,
    s_name VARCHAR2(100) NOT NULL,
    s_pnum VARCHAR2(15) NOT NULL,
    s_mail VARCHAR2(100) UNIQUE,
    CHECK (s_id>0)
  );
```

```
Table created.
```

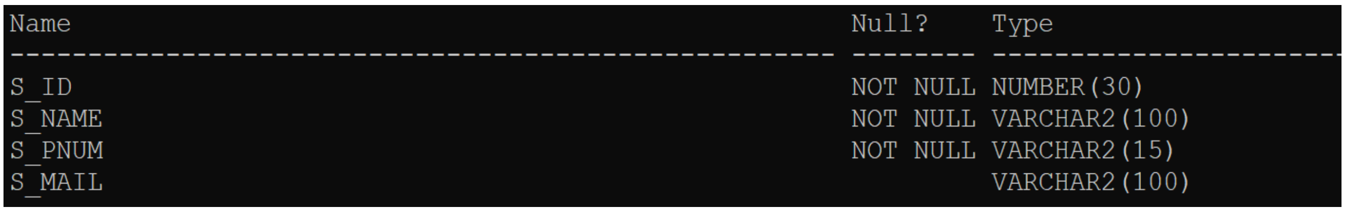
ADDING CONSTRAINT :

ALTER TABLE suppliers
ADD CONSTRAINT unique_phone_email_pair
UNIQUE (s_pnum, s_mail);



```
Table altered.
```

DESC suppliers;

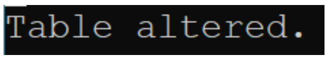


```
Name                                                      Null?    Type
 --------------------------------------------------------- -------- ----------------------
S_ID                                                      NOT NULL NUMBER(30)
S_NAME                                                    NOT NULL VARCHAR2(100)
S_PNUM                                                    NOT NULL VARCHAR2(15)
S_MAIL                                                             VARCHAR2(100)
```

ALTERING TABLE (DDL) :

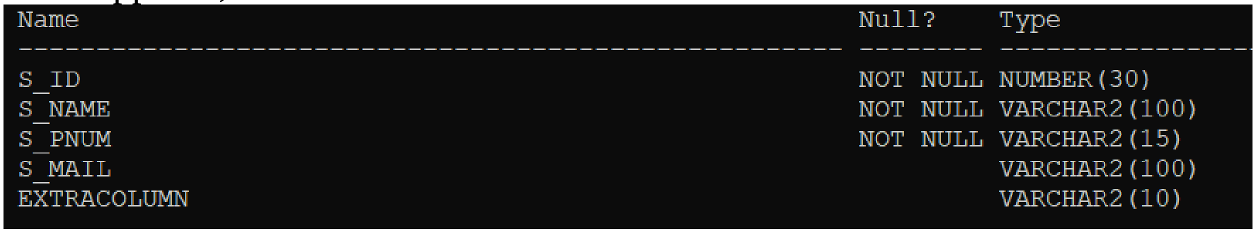
ADD:

ALTER TABLE suppliers ADD extracolumn varchar2(10);



```
Table altered.
```

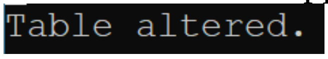
desc suppliers;



```
Name                                                      Null?    Type
 --------------------------------------------------------- -------- ------------------
S_ID                                                      NOT NULL NUMBER(30)
S_NAME                                                    NOT NULL VARCHAR2(100)
S_PNUM                                                    NOT NULL VARCHAR2(15)
S_MAIL                                                             VARCHAR2(100)
EXTRACOLUMN                                                        VARCHAR2(10)
```

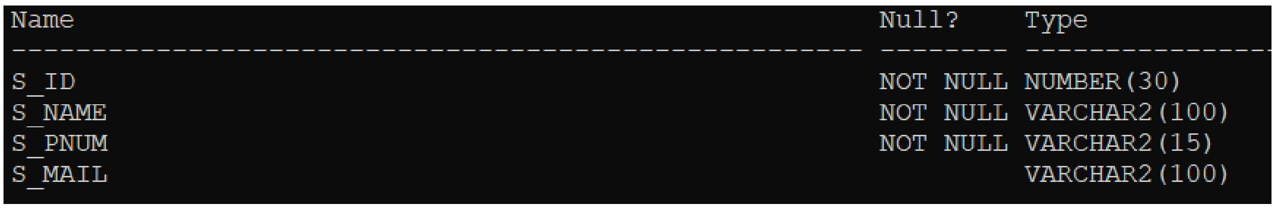
DROP:

ALTER TABLE suppliers DROP COLUMN extracolumn;



```
Table altered.
```

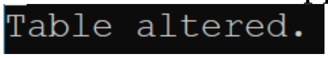
desc suppliers;



```
Name                                                      Null?    Type
 --------------------------------------------------------- -------- ------------------
S_ID                                                      NOT NULL NUMBER(30)
S_NAME                                                    NOT NULL VARCHAR2(100)
S_PNUM                                                    NOT NULL VARCHAR2(15)
S_MAIL                                                             VARCHAR2(100)
```

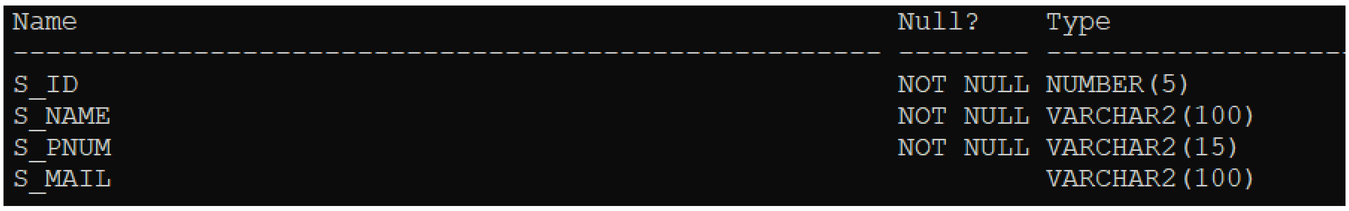
MODIFY :

ALTER TABLE suppliers MODIFY s_id NUMBER(5);



```
Table altered.
```

desc suppliers;



```
 Name                                                      Null?    Type
 --------------------------------------------------------- -------- --------------------
 S_ID                                                      NOT NULL NUMBER(5)
 S_NAME                                                    NOT NULL VARCHAR2(100)
 S_PNUM                                                    NOT NULL VARCHAR2(15)
 S_MAIL                                                             VARCHAR2(100)
```

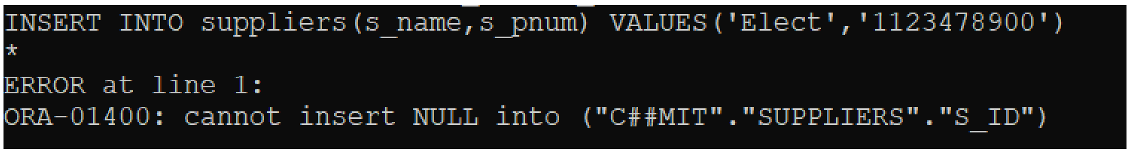
<u>INSERTING DATA IN TABLE (DML) :</u>

GETTING DATA FROM THE USER :

INSERT INTO suppliers VALUES(&s_id,'&s_name','&s_pnum', ‘&s_mail’);

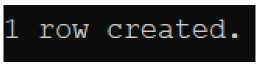
BULK INSERT AND <u>CONSTRAINT CHECK</u> :

INSERT INTO suppliers(s_name,s_pnum) VALUES('Elect','1123478900');



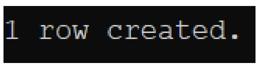
```
INSERT INTO suppliers(s_name,s_pnum) VALUES('Elect','1123478900')
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("C##MIT"."SUPPLIERS"."S_ID")
```

INSERT INTO suppliers(s_id,s_name,s_pnum) VALUES(5,'Elective','1123400900');



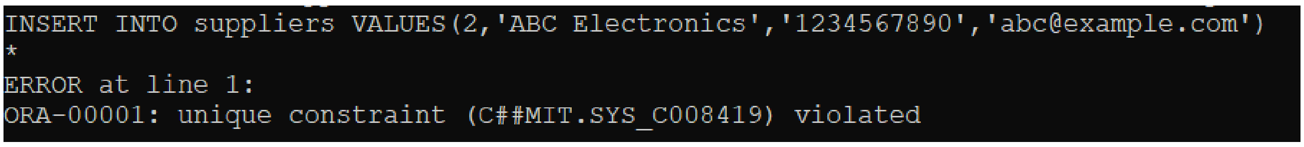
```
1 row created.
```

INSERT INTO suppliers VALUES(1,'Electron','1123400900','abc@example.com');



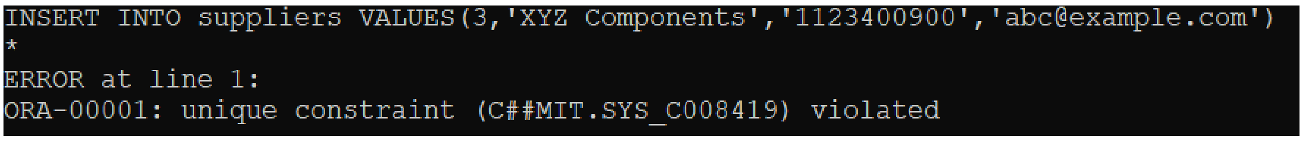
```
1 row created.
```

INSERT INTO suppliers VALUES(2,'ABC Electronics','1234567890','abc@example.com');



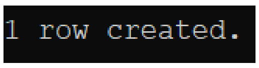
```
INSERT INTO suppliers VALUES(2,'ABC Electronics','1234567890','abc@example.com')
*
ERROR at line 1:
ORA-00001: unique constraint (C##MIT.SYS_C008419) violated
```

INSERT INTO suppliers VALUES(3,'XYZ Components','1123400900','abc@example.com');



```
INSERT INTO suppliers VALUES(3,'XYZ Components','1123400900','abc@example.com')
*
ERROR at line 1:
ORA-00001: unique constraint (C##MIT.SYS_C008419) violated
```

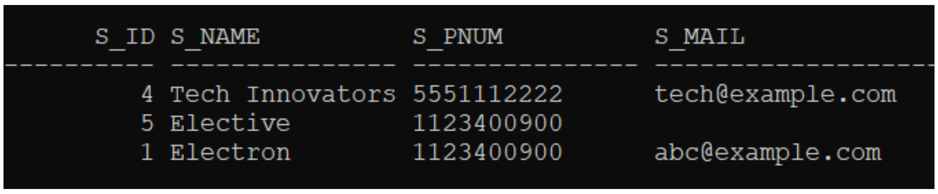
INSERT INTO suppliers VALUES(4,'Tech Innovators','5551112222','tech@example.com');



```
1 row created.
```

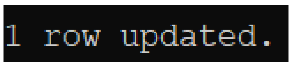
SELECT (DQL) , UPDATING (DML)DATA IN TABLE:

SELECT * FROM suppliers;



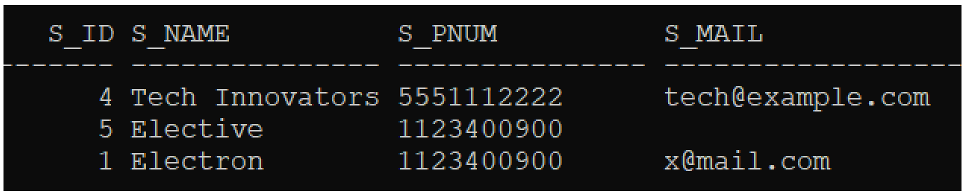
```
      S_ID S_NAME          S_PNUM          S_MAIL
---------- --------------- --------------- -------------------
         4 Tech Innovators 5551112222      tech@example.com
         5 Elective        1123400900
         1 Electron        1123400900      abc@example.com
```

UPDATE suppliers SET s_mail='x@mail.com' WHERE s_id = 1;



```
1 row updated.
```

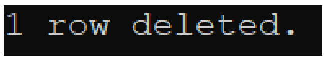
SELECT * FROM suppliers;



```
   S_ID S_NAME          S_PNUM          S_MAIL
------- --------------- --------------- -------------------
      4 Tech Innovators 5551112222      tech@example.com
      5 Elective        1123400900
      1 Electron        1123400900      x@mail.com
```

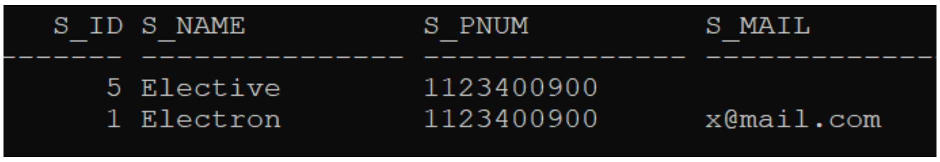
DELETING DATA FROM TABLE (DML) :

DELETE FROM suppliers WHERE s_id=4;



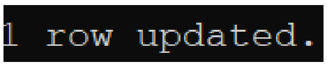
```
1 row deleted.
```

SELECT * FROM suppliers;



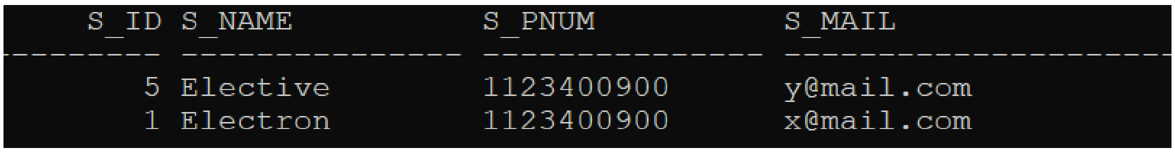
```
   S_ID S_NAME          S_PNUM          S_MAIL
------- --------------- --------------- --------------
      5 Elective        1123400900
      1 Electron        1123400900      x@mail.com
```

UPDATE suppliers SET s_mail='y@mail.com' WHERE s_id = 5 AND s_name='Elective';



```
1 row updated.
```

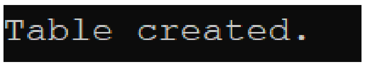
SELECT * FROM suppliers;



```
      S_ID S_NAME          S_PNUM          S_MAIL
---------- --------------- --------------- ----------------------
         5 Elective        1123400900      y@mail.com
         1 Electron        1123400900      x@mail.com
```

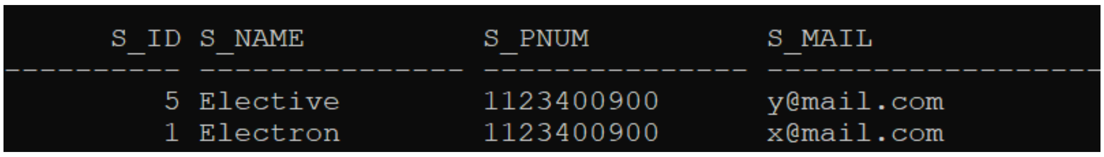
TABLE CLONING :

CREATE TABLE sup AS SELECT * FROM suppliers;



```
Table created.
```

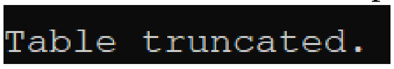
SELECT * FROM sup;



```
      S_ID S_NAME          S_PNUM          S_MAIL
---------- --------------- --------------- --------------------
         5 Elective        1123400900      y@mail.com
         1 Electron        1123400900      x@mail.com
```

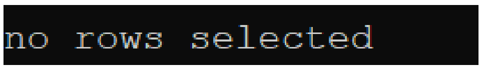
TRUNCATE (DDL) :

TRUNCATE TABLE sup;



```
Table truncated.
```

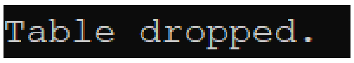
SELECT * FROM sup;



```
no rows selected
```

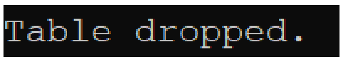
DROPTABLE (DDL) :

DROP TABLE sup;



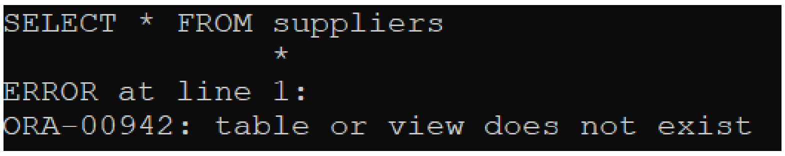
```
Table dropped.
```

DROP TABLE suppliers;



```
Table dropped.
```

SELECT * FROM suppliers;



```
SELECT * FROM suppliers
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

RESULT:

Hence database table for suppliers have been created successfully using constraints (primary key, unique, check, not null, null, default, index), SQL DDL and DML commands.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	03
Date	19/09/2023

INTRODUCTION TO DATABASE DESIGN FOREIGN KEY CONSTRAINT AND INCORPORATE REFERENTIAL INTEGRITY

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML and DCL commands.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM :

To create a database table for suppliers and medicines then add foreign key constraints and incorporate referential integrities.

SQL COMMANDS:

```
SET LINESIZE 100;
SET AUTOCOMMIT ON;
```

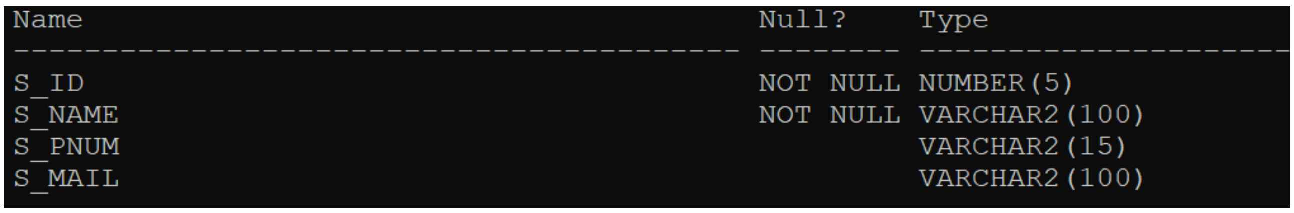
CREATING TABLE WITH FOREIGN KEY CONSTRAINT :

TABLE ABOUT SUPPLIERS:

```
CREATE TABLE suppliers (
 s_id NUMBER(5) PRIMARY KEY,
 s_name VARCHAR2(100) NOT NULL,
 s_pnum VARCHAR2(15),
 s_mail VARCHAR2(100) UNIQUE);
```

```
Table created.
```

DESC suppliers;



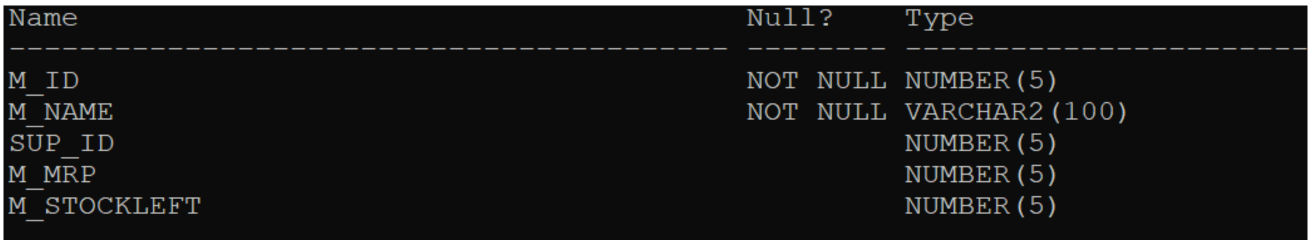
```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------
S_ID                                      NOT NULL NUMBER(5)
S_NAME                                    NOT NULL VARCHAR2(100)
S_PNUM                                             VARCHAR2(15)
S_MAIL                                             VARCHAR2(100)
```

TABLE ABOUT medicines :

```
CREATE TABLE medicines (
m_id NUMBER(5) PRIMARY KEY,
m_name VARCHAR2(100) NOT NULL,
sup_id NUMBER(5),
m_mrp NUMBER(5),
m_stockleft NUMBER(5),
FOREIGN KEY (sup_id) REFERENCES suppliers(s_id));
```

```
Table created.
```

DESC medicines;

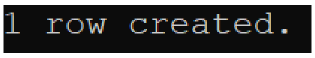


```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------
M_ID                                      NOT NULL NUMBER(5)
M_NAME                                    NOT NULL VARCHAR2(100)
SUP_ID                                             NUMBER(5)
M_MRP                                              NUMBER(5)
M_STOCKLEFT                                        NUMBER(5)
```

INSERTING DATA INTO THE CREATED TABLE :

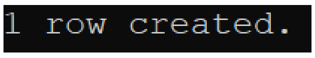
SUPPLIERS:

INSERT INTO suppliers VALUES (1,'ABC Sharma', '9638527410', 'john@abcpharma.com');



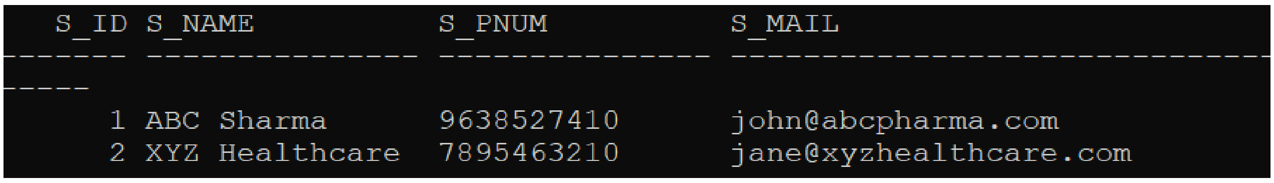
```
1 row created.
```

INSERT INTO suppliers VALUES (2,'XYZ Healthcare', '7895463210', 'jane@xyzhealthcare.com');



```
1 row created.
```

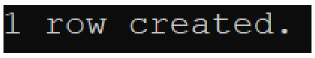
SELECT * FROM suppliers;



```
   S_ID S_NAME          S_PNUM          S_MAIL
------- --------------- --------------- ------------------------------
-----
      1 ABC Sharma      9638527410      john@abcpharma.com
      2 XYZ Healthcare  7895463210      jane@xyzhealthcare.com
```

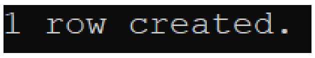
MEDICINES:

INSERT INTO medicines VALUES (101,'PainAway',1,25,120);



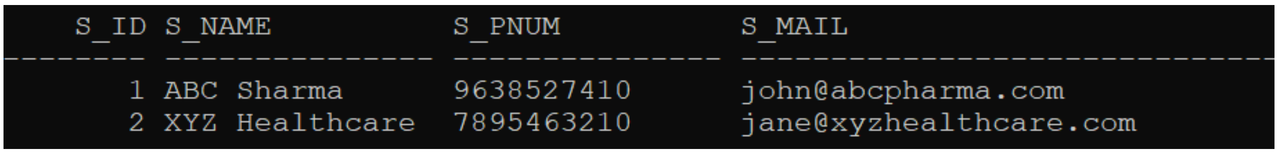
```
1 row created.
```

INSERT INTO medicines VALUES (102,'CoughRelief',2,10,122);



```
1 row created.
```

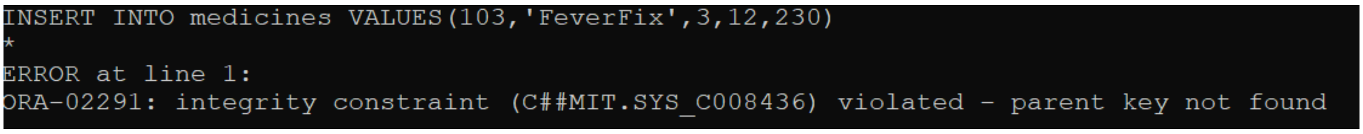
SELECT * FROM medicines;



```
    S_ID S_NAME          S_PNUM          S_MAIL
-------- --------------- --------------- ------------------------------
       1 ABC Sharma      9638527410      john@abcpharma.com
       2 XYZ Healthcare  7895463210      jane@xyzhealthcare.com
```

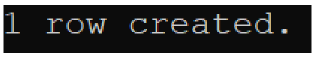
CHECKING REFERENTIAL INTEGRITY:

INSERT INTO medicines VALUES (103,'FeverFix',3,12,230);



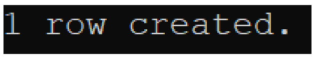
```
INSERT INTO medicines VALUES(103,'FeverFix',3,12,230)
*
ERROR at line 1:
ORA-02291: integrity constraint (C##MIT.SYS_C008436) violated - parent key not found
```

INSERT INTO suppliers VALUES (3,'thorcare','7897453210','e@thorcare.com');



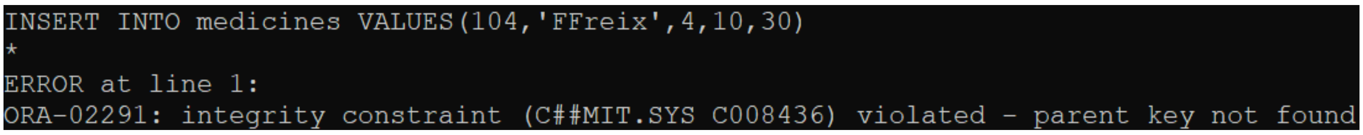
```
1 row created.
```

INSERT INTO medicines VALUES (103,'FeverFix',3,12,230);



```
1 row created.
```

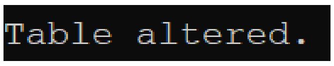
INSERT INTO medicines VALUES (104,'FFreix',4,10,30);



```
INSERT INTO medicines VALUES(104,'FFreix',4,10,30)
*
ERROR at line 1:
ORA-02291: integrity constraint (C##MIT.SYS_C008436) violated - parent key not found
```

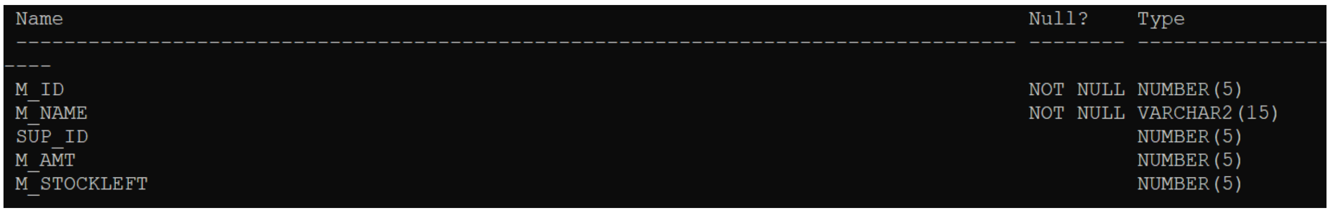
ALTER TABLE:

ALTER TABLE medicines RENAME COLUMN m_mrp TO m_amt;



```
Table altered.
```

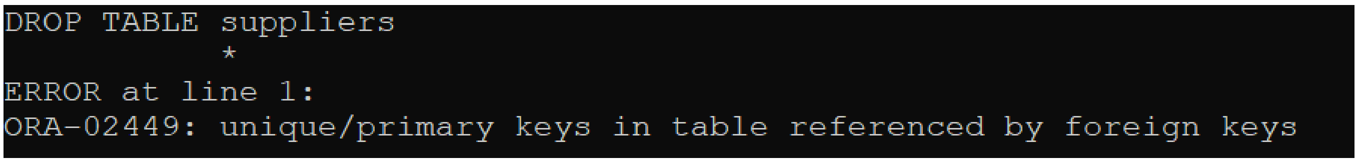
DESC medicines;



```
 Name                                                                              Null?    Type
 --------------------------------------------------------------------------------- -------- ----------------
----
 M_ID                                                                              NOT NULL NUMBER(5)
 M_NAME                                                                            NOT NULL VARCHAR2(15)
 SUP_ID                                                                                     NUMBER(5)
 M_AMT                                                                                      NUMBER(5)
 M_STOCKLEFT                                                                                NUMBER(5)
```

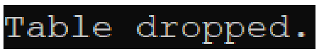
DROP TABLE:

DROP TABLE suppliers;



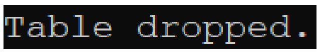
```
DROP TABLE suppliers
           *
ERROR at line 1:
ORA-02449: unique/primary keys in table referenced by foreign keys
```

DROP TABLE medicines;



```
Table dropped.
```

DROP TABLE suppliers;



```
Table dropped.
```

RESULT:

Hence database table for suppliers, medicines have been created successfully using foreign key constraints and incorporate referential integrity.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	04
Date	26/09/2023

INTRODUCTION TO DATABASE DESIGN WHERE CLAUSE CONDITION AND AGGREGATE FUNCTIONS

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML andDCL commands.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM :

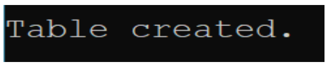
To create perform basic SQL commands using different ‘where’ clause conditions and also implement aggregate functions in created table.

SQL COMMANDS:

```
SET LINESIZE 100;
SET AUTOCOMMIT ON;
```

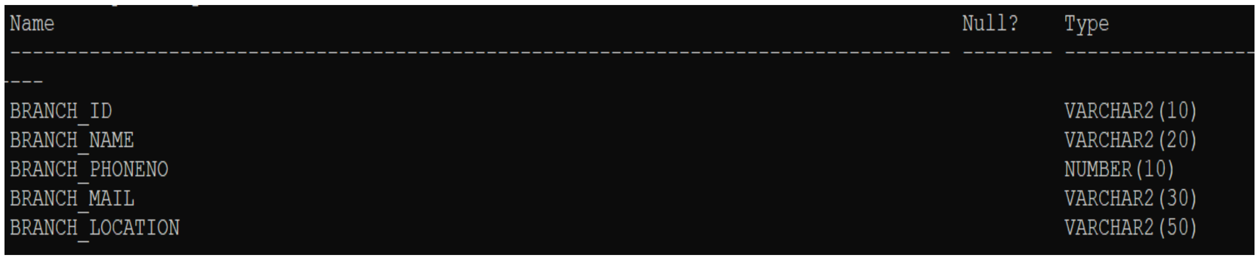
CREATING TABLE :

```
CREATE TABLE pharmacy
(
branch_id VARCHAR2(5),
branch_name VARCHAR2(20),
branch_phoneno NUMBER(10),
branch_mail VARCHAR2(30),
branch_location VARCHAR2(50)
);
```



```
Table created.
```

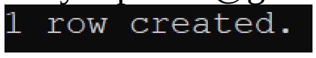
DESC suppliers;



```
Name                                                                               Null?    Type
---------------------------------------------------------------------------------- -------- ----------------
---
BRANCH_ID                                                                                   VARCHAR2(10)
BRANCH_NAME                                                                                 VARCHAR2(20)
BRANCH_PHONENO                                                                              NUMBER(10)
BRANCH_MAIL                                                                                 VARCHAR2(30)
BRANCH_LOCATION                                                                             VARCHAR2(50)
```

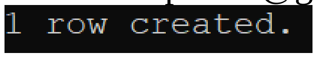
<u>INSERTING DATA INTO CREATED TABLE :</u>

INSERT into pharmacy VALUES (1000,' Adayarpharm', 314567, 'adayar.pharm@gmail.com', 'xxx street,PASS road,chennai-600987');



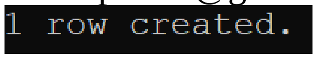
```
1 row created.
```

INSERT into pharmacy VALUES (1900, 'Arakonampharm', 376567, 'arakonam.pharm@gmail.com', 'xxx street,madurai');



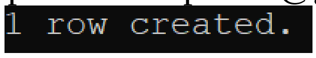
```
1 row created.
```

INSERT into pharmacy VALUES (1070, 'Avadipharm', 314357, 'avadi.pharm@gmail.com', 'tiruchi');



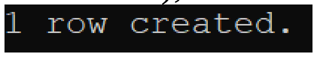
```
1 row created.
```

INSERT into pharmacy VALUES (1670, 'Pallavarampharm', 123457, 'pallavaram.pharm@gmail.com', 'xxx street,kodai');



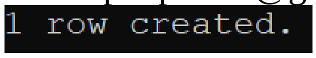
```
1 row created.
```

INSERT into pharmacy VALUES (1230, 'Tirupharm', 310007, 'tiru.pharm@gmail.com', 'xxx street');



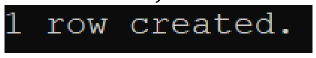
```
1 row created.
```

INSERT into pharmacy VALUES (3450, 'Chrompetpharm', 309767, 'chrompet.pharm@gmail.com', 'xxx street,PASS road,chennai-600987');



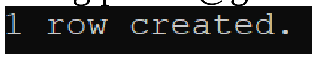
```
1 row created.
```

INSERT into pharmacy VALUES (7540, 'OMRpharm', 354767, 'omr.pharm@gmail.com', 'xxx street,PASS road,chennai-600987');



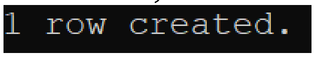
```
1 row created.
```

INSERT into pharmacy VALUES (1450, 'Mangulampharm', 300067, 'mang.pharm@gmail.com', 'xxx street,PASS road,chennai-600987');



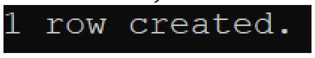
```
1 row created.
```

INSERT into pharmacy VALUES (1090, 'Hillpharm', 318760, 'hill.pharm@gmail.com', 'xxx street,PASS road,chennai-600987');



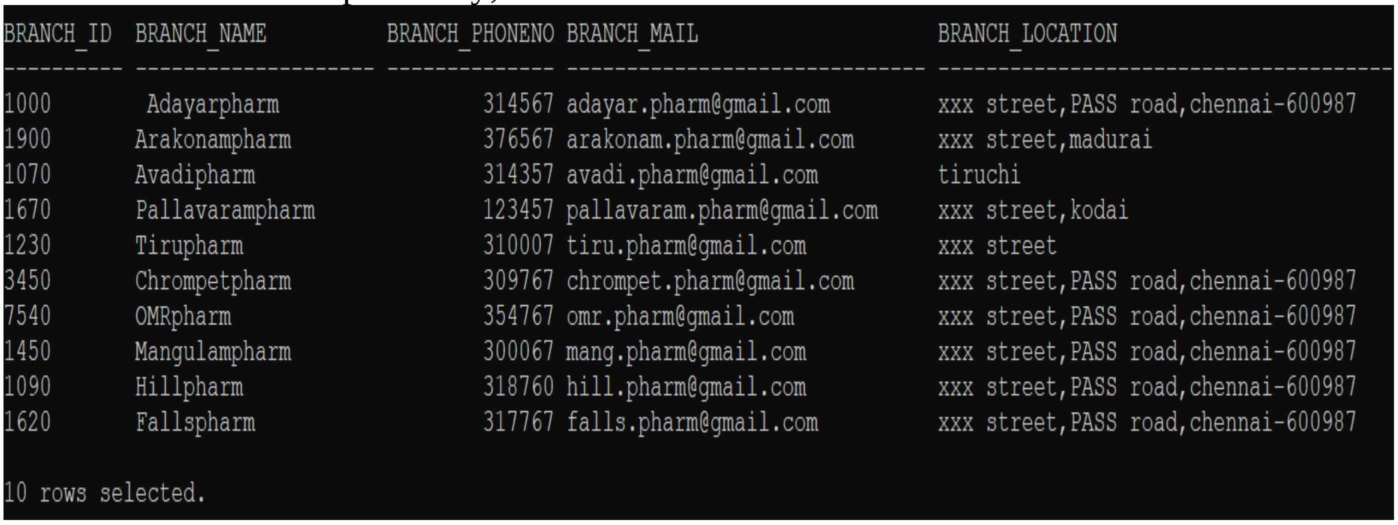
```
1 row created.
```

INSERT into pharmacy VALUES (1620, 'Fallspharm', 317767, 'falls.pharm@gmail.com', 'xxx street,PASS road,chennai-600987');



```
1 row created.
```

SELECT * FROM pharmacy;



```
BRANCH_ID  BRANCH_NAME          BRANCH_PHONENO BRANCH_MAIL                    BRANCH_LOCATION
---------- -------------------- -------------- ------------------------------ ----------------------------------------
1000        Adayarpharm                 314567 adayar.pharm@gmail.com         xxx street,PASS road,chennai-600987
1900       Arakonampharm                376567 arakonam.pharm@gmail.com       xxx street,madurai
1070       Avadipharm                   314357 avadi.pharm@gmail.com          tiruchi
1670       Pallavarampharm              123457 pallavaram.pharm@gmail.com     xxx street,kodai
1230       Tirupharm                    310007 tiru.pharm@gmail.com           xxx street
3450       Chrompetpharm                309767 chrompet.pharm@gmail.com       xxx street,PASS road,chennai-600987
7540       OMRpharm                     354767 omr.pharm@gmail.com            xxx street,PASS road,chennai-600987
1450       Mangulampharm                300067 mang.pharm@gmail.com           xxx street,PASS road,chennai-600987
1090       Hillpharm                    318760 hill.pharm@gmail.com           xxx street,PASS road,chennai-600987
1620       Fallspharm                   317767 falls.pharm@gmail.com          xxx street,PASS road,chennai-600987

10 rows selected.
```

DIFFERENT WHERE CLAUSE CONDITIONS :

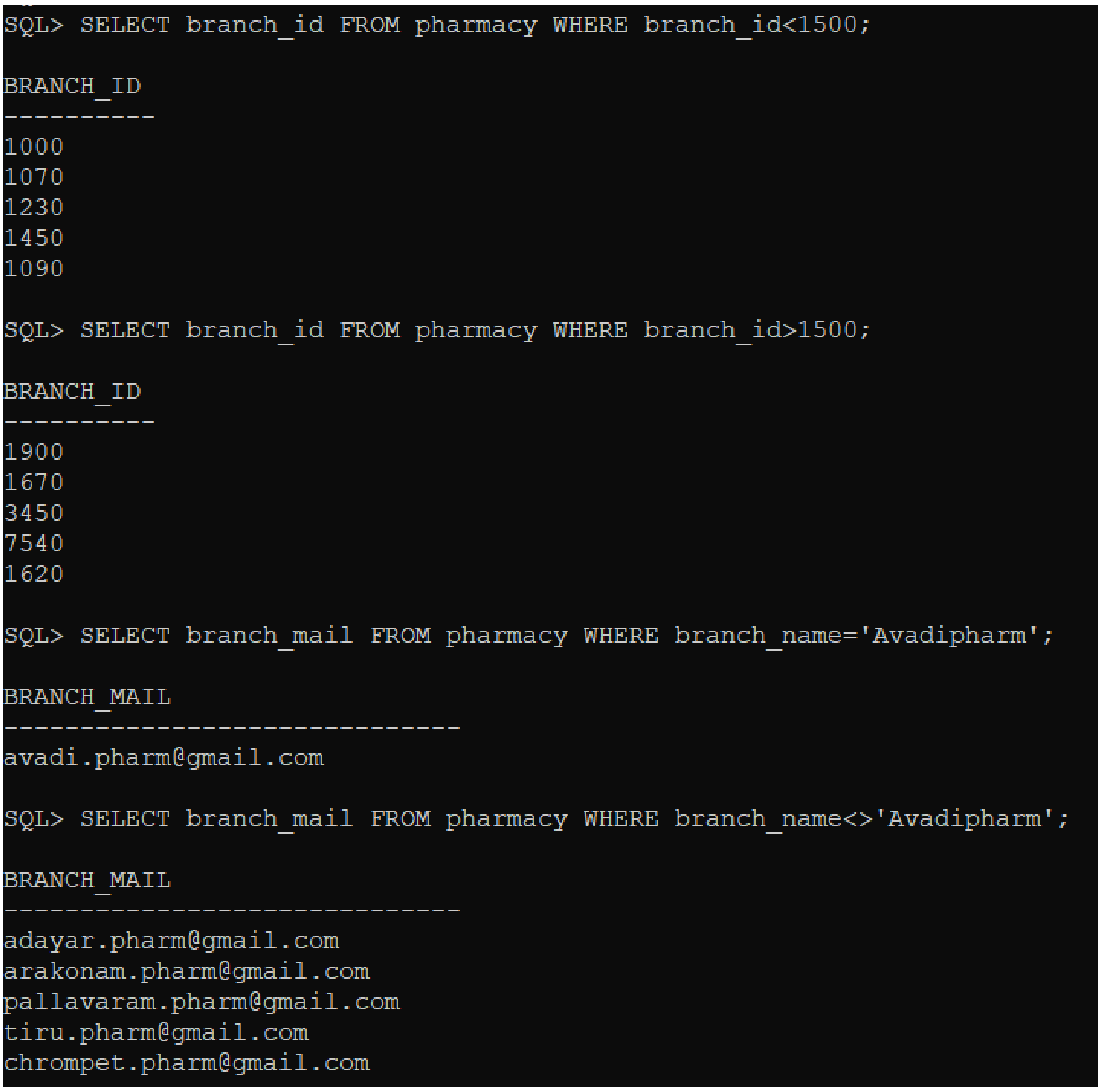
COMPARISON OPERATORS ARE USED IN CREATED TABLES :

LESS THAN:
SELECT branch_id FROM pharmacy WHERE branch_id<1500;

GREATER THAN:
SELECT branch_id FROM pharmacy WHERE branch_id>1500;

EQUAL TO:
SELECT branch_mail FROM pharmacy WHERE branch_name='Avadipharm';

NOT EQUAL TO:
SELECT branch_mail FROM pharmacy WHERE branch_name<>'Avadipharm';

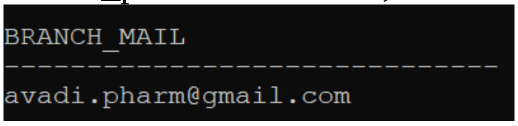


```
SQL> SELECT branch_id FROM pharmacy WHERE branch_id<1500;

BRANCH_ID
----------
1000
1070
1230
1450
1090

SQL> SELECT branch_id FROM pharmacy WHERE branch_id>1500;

BRANCH_ID
----------
1900
1670
3450
7540
1620

SQL> SELECT branch_mail FROM pharmacy WHERE branch_name='Avadipharm';

BRANCH_MAIL
------------------------------
avadi.pharm@gmail.com

SQL> SELECT branch_mail FROM pharmacy WHERE branch_name<>'Avadipharm';

BRANCH_MAIL
------------------------------
adayar.pharm@gmail.com
arakonam.pharm@gmail.com
pallavaram.pharm@gmail.com
tiru.pharm@gmail.com
chrompet.pharm@gmail.com
```

LOGICAL OPERATORS ARE USED IN CREATED TABLES :

AND OPERATOR:

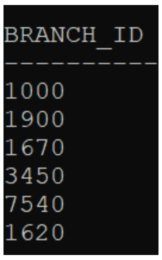
SELECT branch_mail FROM pharmacy WHERE branch_name='Avadipharm' AND branch_phoneno=314357;



```
BRANCH_MAIL
------------------------------
avadi.pharm@gmail.com
```

OR OPERATOR:

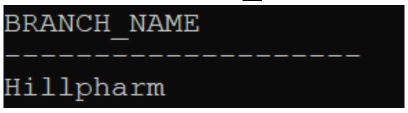
SELECT branch_id FROM pharmacy WHERE branch_id>1500 OR branch_phoneno=314567;



```
BRANCH_ID
----------
1000
1900
1670
3450
7540
1620
```

LIKE:

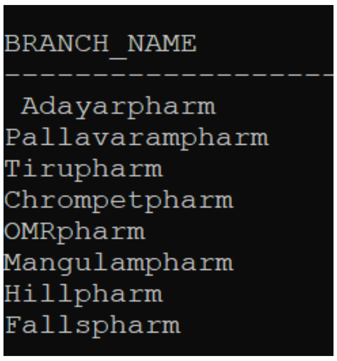
SELECT branch_name FROM pharmacy WHERE branch_name LIKE 'H%';



```
BRANCH_NAME
--------------------
Hillpharm
```

NOT LIKE:

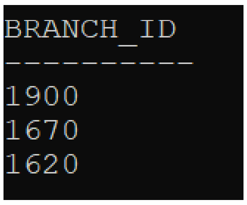
SELECT branch_name FROM pharmacy WHERE branch_name NOT LIKE 'A%';



```
BRANCH_NAME
--------------------
 Adayarpharm
Pallavarampharm
Tirupharm
Chrompetpharm
OMRpharm
Mangulampharm
Hillpharm
Fallspharm
```

BETWEEN :

SELECT branch_id FROM pharmacy WHERE branch_id BETWEEN 1500 AND 1920;

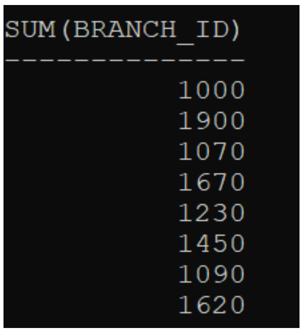


```
BRANCH_ID
----------
1900
1670
1620
```

AGGREGATE FUNCTIONS :

SUM:

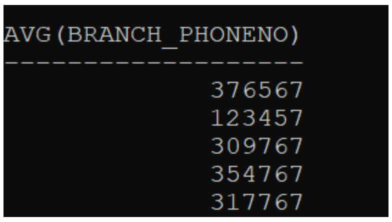
SELECT sum(branch_id) FROM pharmacy GROUP BY branch_id having branch_id<3450;



```
SUM(BRANCH_ID)
--------------
          1000
          1900
          1070
          1670
          1230
          1450
          1090
          1620
```

AVERAGE:

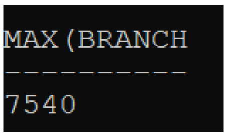
SELECT avg(branch_phoneno) FROM pharmacy WHERE branch_id>1540 GROUP BY branch_id;



```
AVG(BRANCH_PHONENO)
-------------------
             376567
             123457
             309767
             354767
             317767
```

MAX:

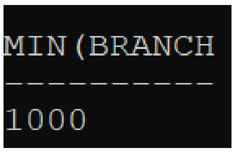
SELECT MAX(branch_id) FROM pharmacy;



```
MAX(BRANCH
----------
7540
```

MIN:

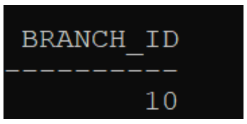
SELECT MIN(branch_id) FROM pharmacy;



```
MIN(BRANCH
----------
1000
```

COUNT:

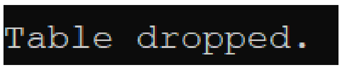
SELECT COUNT(*) AS branch_id FROM pharmacy;



```
 BRANCH_ID
----------
        10
```

DROP TABLE:

DROP TABLE pharmacy;



```
Table dropped.
```

RESULT :

Hence database table for pharmacy have been created successfully created and used ‘where’ clause conditions and aggregate functions.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	05
Date	03/09/2023

INTRODUCTION TO DATABASE DESIGN SUBQUERIES AND SIMPLE JOIN OPERATIONS

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML and DCL commands.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

To create a database table for suppliers and medicines entity to explore subqueries and simple join operations.

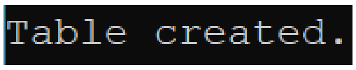
SQL COMMANDS:

```
SET LINESIZE 100;
SET AUTOCOMMIT ON;
```

SUPPLIERS AND MEDICINE ENTITY

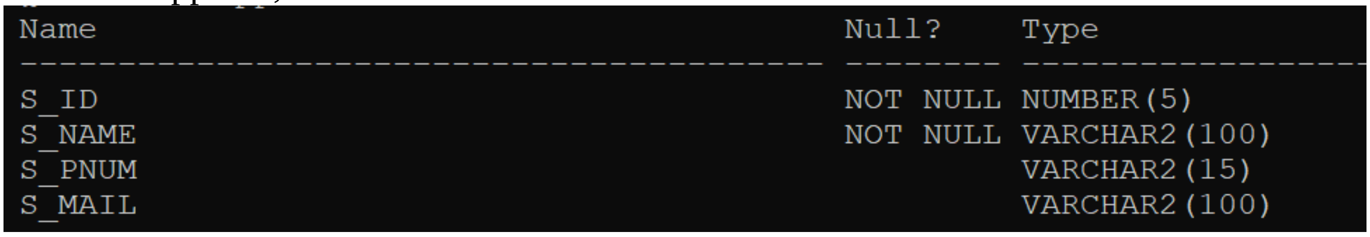
CREATING SUPPLIER TABLE :

```
CREATE TABLE suppliers (
   s_id NUMBER(5) PRIMARY KEY,
   s_name VARCHAR2(100) NOT NULL,
   s_pnum VARCHAR2(15),
   s_mail VARCHAR2(100) UNIQUE);
```



```
Table created.
```

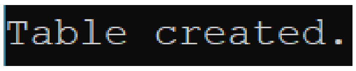
DESC suppliers;



```
Name                                      Null?    Type
----------------------------------------- -------- ------------------
S_ID                                      NOT NULL NUMBER(5)
S_NAME                                    NOT NULL VARCHAR2(100)
S_PNUM                                             VARCHAR2(15)
S_MAIL                                             VARCHAR2(100)
```

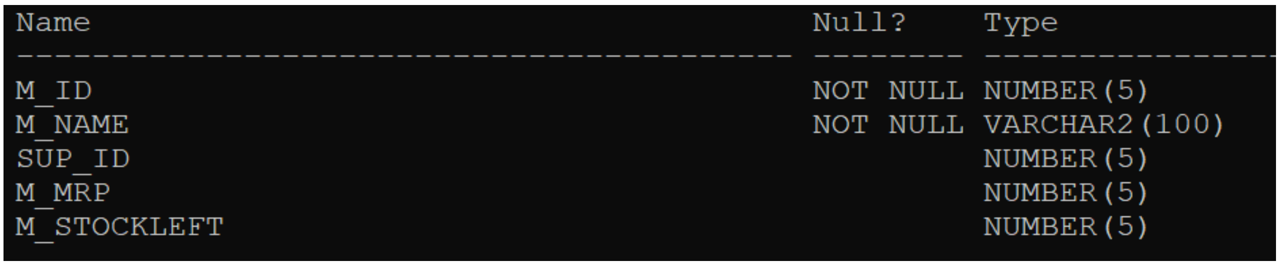
<u>CREATING MEDICINE TABLE :</u>

```
CREATE TABLE medicines (
    m_id NUMBER(5) PRIMARY KEY,
    m_name VARCHAR2(100) NOT NULL,
    sup_id NUMBER(5),
    m_mrp NUMBER(5),
    m_stockleft NUMBER(5),
    FOREIGN KEY (sup_id) REFERENCES suppliers(s_id));
```



```
Table created.
```

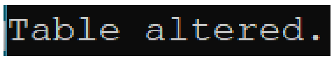
DESC medicines;



```
Name                                      Null?    Type
----------------------------------------- -------- ----------------
M_ID                                      NOT NULL NUMBER(5)
M_NAME                                    NOT NULL VARCHAR2(100)
SUP_ID                                             NUMBER(5)
M_MRP                                              NUMBER(5)
M_STOCKLEFT                                        NUMBER(5)
```

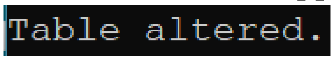
<u>ALTERING THE TABLE:</u>

ALTER TABLE medicines MODIFY m_name VARCHAR2(20);



```
Table altered.
```

ALTER TABLE suppliers modify s_name VARCHAR2(20);

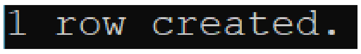


```
Table altered.
```

<u>INSERTING DATA INTO CREATED TABLE :</u>

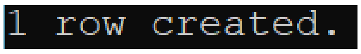
SUPPLIERS TABLE:

INSERT INTO suppliers VALUES (1,'ABC','9344002774','a@gmail.com');



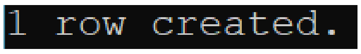
```
1 row created.
```

INSERT INTO suppliers VALUES (2,'DEF','8344005793','b@gmail.com');



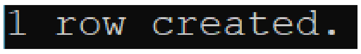
```
1 row created.
```

INSERT INTO suppliers VALUES (3,'FIH','5342027774','c@gmail.com');



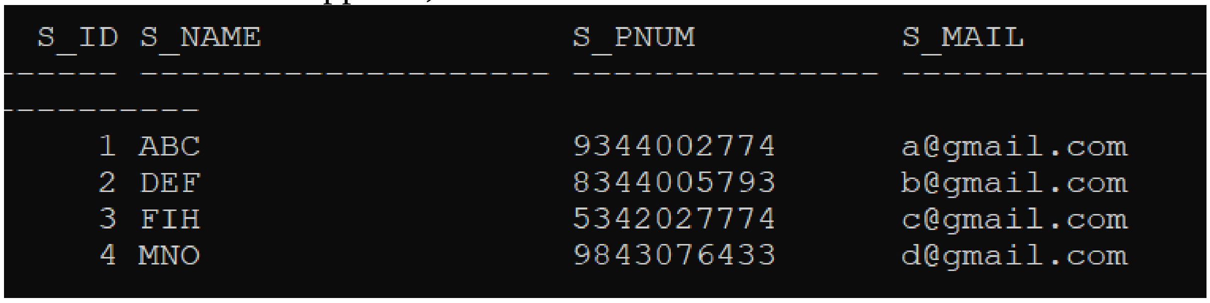
```
1 row created.
```

INSERT INTO suppliers VALUES (4,'MNO','9843076433','d@gmail.com');



```
1 row created.
```

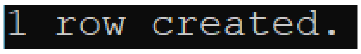
SELECT * FROM suppliers;



```
  S_ID S_NAME                S_PNUM          S_MAIL
------ --------------------- --------------- ---------------
----------
     1 ABC                   9344002774      a@gmail.com
     2 DEF                   8344005793      b@gmail.com
     3 FIH                   5342027774      c@gmail.com
     4 MNO                   9843076433      d@gmail.com
```

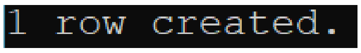
MEDICINES TABLE:

INSERT INTO medicines VALUES (101,'PainAway', 1, 25, 120);



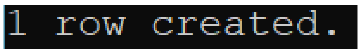
```
1 row created.
```

INSERT INTO medicines VALUES (102,'CoughRelief', 2, 10, 122);



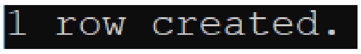
```
1 row created.
```

INSERT INTO medicines VALUES (103,'thorcare', 1, 12, 25);



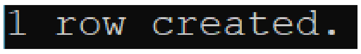
```
1 row created.
```

INSERT INTO medicines VALUES (104,'medicure', 3, 5, 12);



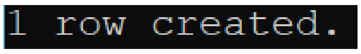
```
1 row created.
```

INSERT INTO medicines VALUES (105,'Paramol', 3, 15, 21);



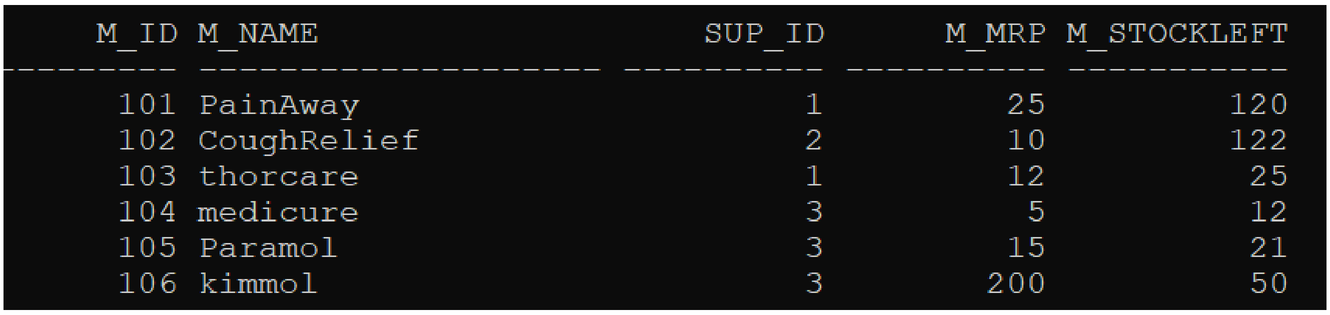
```
1 row created.
```

INSERT INTO medicines VALUES (106,'kimmol', 3, 200, 50);



```
1 row created.
```

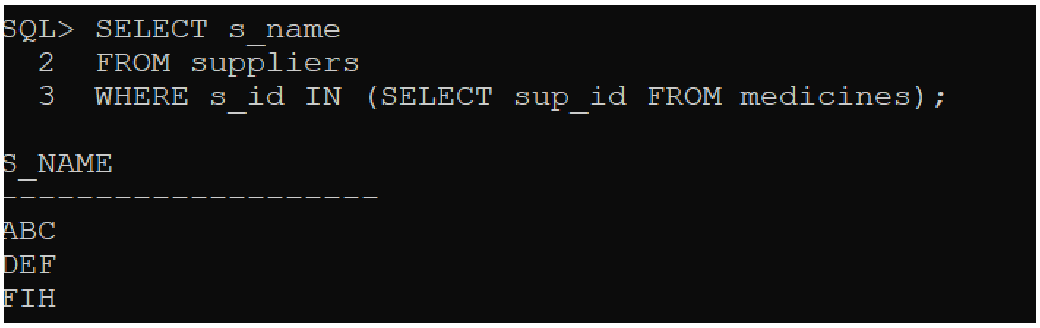
SELECT * FROM medicines;



```
      M_ID M_NAME                   SUP_ID      M_MRP M_STOCKLEFT
---------- -------------------- ---------- ---------- -----------
       101 PainAway                      1         25         120
       102 CoughRelief                   2         10         122
       103 thorcare                      1         12          25
       104 medicure                      3          5          12
       105 Paramol                       3         15          21
       106 kimmol                        3        200          50
```

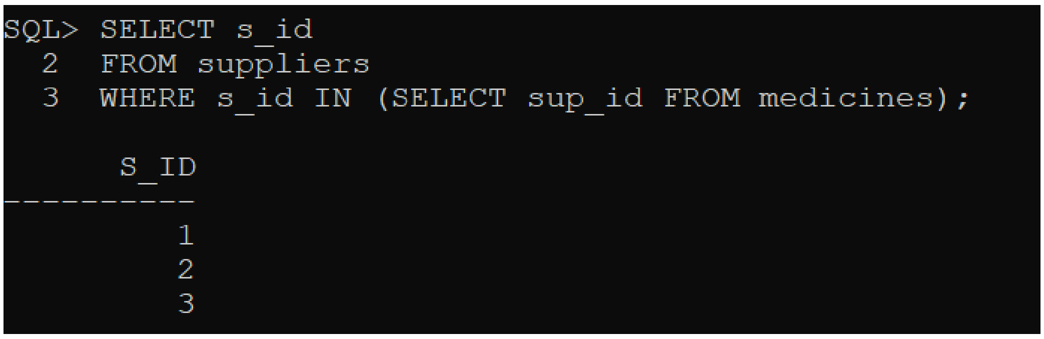
<u>SUB QUERIES ARE USED IN CREATED TABLES</u> :

SELECT s_name
FROM suppliers
WHERE s_id IN (SELECT sup_id FROM medicines);



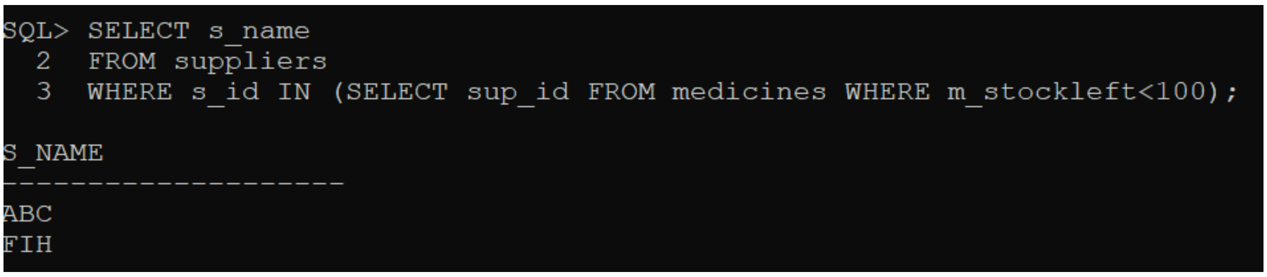
```
SQL> SELECT s_name
  2  FROM suppliers
  3  WHERE s_id IN (SELECT sup_id FROM medicines);

S_NAME
--------------------
ABC
DEF
FIH
```

SELECT s_id
FROM suppliers
WHERE s_id IN (SELECT sup_id FROM medicines);



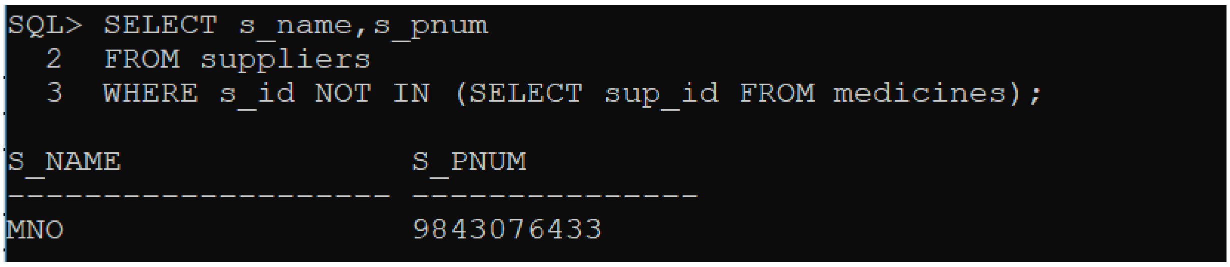
```
SQL> SELECT s_id
  2  FROM suppliers
  3  WHERE s_id IN (SELECT sup_id FROM medicines);

      S_ID
----------
         1
         2
         3
```

SELECT s_name
FROM suppliers
WHERE s_id IN (SELECT sup_id FROM medicines WHERE m_stockleft<100);



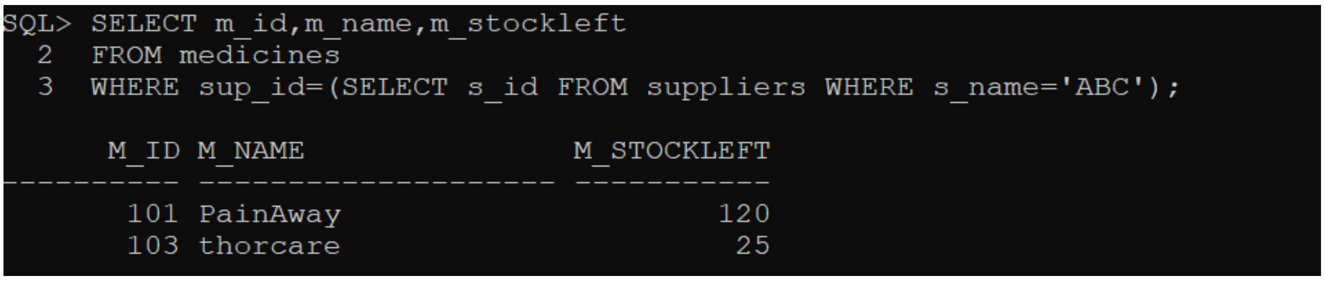
```
SQL> SELECT s_name
  2  FROM suppliers
  3  WHERE s_id IN (SELECT sup_id FROM medicines WHERE m_stockleft<100);

S_NAME
--------------------
ABC
FIH
```

SELECT s_name,s_pnum
FROM suppliers
WHERE s_id NOT IN (SELECT sup_id FROM medicines);



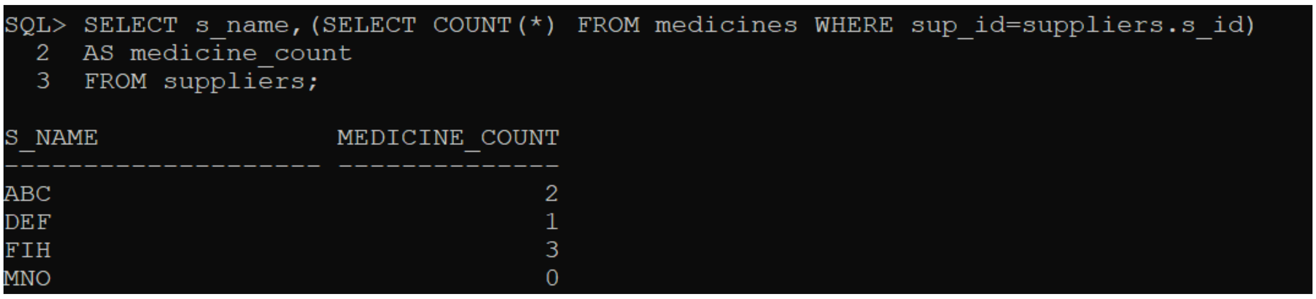
```
SQL> SELECT s_name,s_pnum
  2  FROM suppliers
  3  WHERE s_id NOT IN (SELECT sup_id FROM medicines);

S_NAME               S_PNUM
-------------------- ---------------
MNO                  9843076433
```

SELECT m_id,m_name,m_stockleft
FROM medicines
WHERE sup_id=(SELECT s_id FROM suppliers WHERE s_name='ABC');



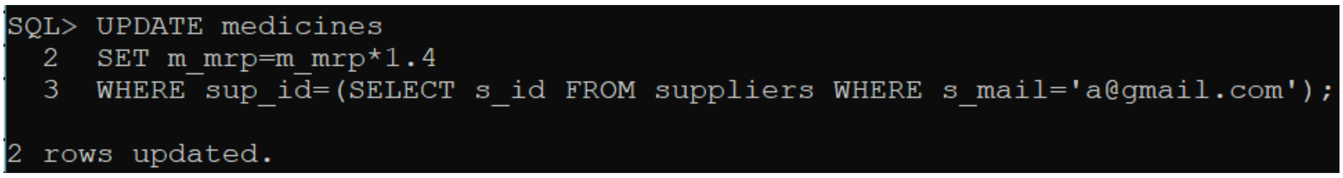
```
SQL> SELECT m_id,m_name,m_stockleft
  2  FROM medicines
  3  WHERE sup_id=(SELECT s_id FROM suppliers WHERE s_name='ABC');

      M_ID M_NAME               M_STOCKLEFT
---------- -------------------- -----------
       101 PainAway                     120
       103 thorcare                      25
```

SELECT s_name,(SELECT COUNT(*) FROM medicines WHERE sup_id=suppliers.s_id)
AS medicine_count
FROM suppliers;



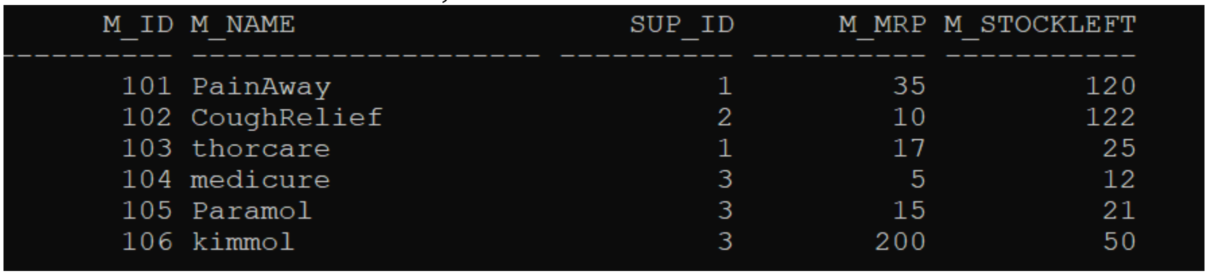
```
SQL> SELECT s_name,(SELECT COUNT(*) FROM medicines WHERE sup_id=suppliers.s_id)
  2  AS medicine_count
  3  FROM suppliers;

S_NAME               MEDICINE_COUNT
-------------------- --------------
ABC                               2
DEF                               1
FIH                               3
MNO                               0
```

UPDATE medicines
SET m_mrp=m_mrp*1.4
WHERE sup_id=(SELECT s_id FROM suppliers WHERE s_mail='a@gmail.com');



```
SQL> UPDATE medicines
  2  SET m_mrp=m_mrp*1.4
  3  WHERE sup_id=(SELECT s_id FROM suppliers WHERE s_mail='a@gmail.com');

2 rows updated.
```

SELECT * FROM medicines;

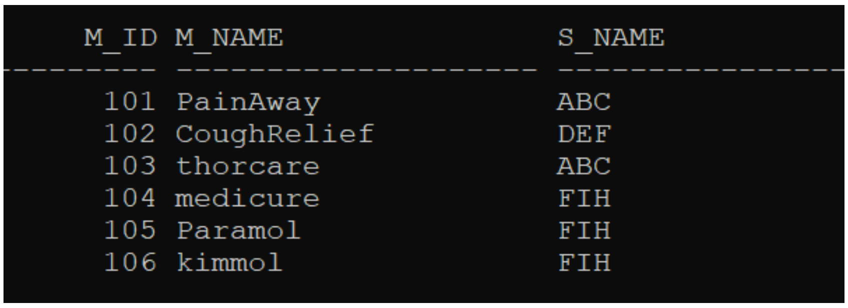


```
      M_ID M_NAME                   SUP_ID      M_MRP M_STOCKLEFT
---------- -------------------- ---------- ---------- -----------
       101 PainAway                      1         35         120
       102 CoughRelief                   2         10         122
       103 thorcare                      1         17          25
       104 medicure                      3          5          12
       105 Paramol                       3         15          21
       106 kimmol                        3        200          50
```

SIMPLE JOIN OPERATIONS ARE USED IN CREATED TABLES:

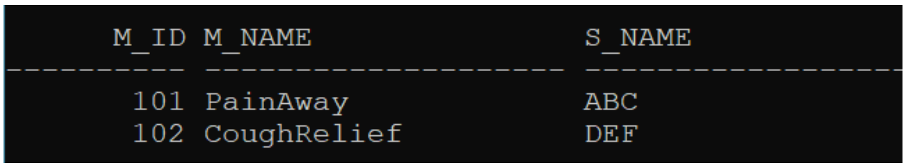
INNER JOIN:

SELECT m_id,m_name,s_name FROM medicines
INNER JOIN suppliers ON sup_id=s_id;



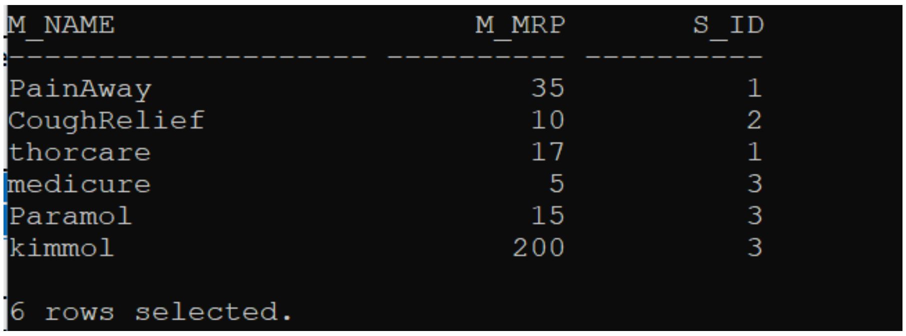
```
     M_ID M_NAME               S_NAME
--------- -------------------- ----------------
      101 PainAway             ABC
      102 CoughRelief          DEF
      103 thorcare             ABC
      104 medicure             FIH
      105 Paramol              FIH
      106 kimmol               FIH
```

SELECT m_id,m_name,s_name FROM medicines
INNER JOIN suppliers ON sup_id=s_id AND m_stockleft>50;



```
      M_ID M_NAME               S_NAME
---------- -------------------- ------------------
       101 PainAway             ABC
       102 CoughRelief          DEF
```

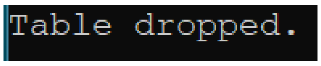
SELECT m_name,m_mrp,s_id FROM suppliers
INNER JOIN medicines ON sup_id=s_id;



```
M_NAME                    M_MRP       S_ID
-------------------- ---------- ----------
PainAway                     35          1
CoughRelief                  10          2
thorcare                     17          1
medicure                      5          3
Paramol                      15          3
kimmol                      200          3

6 rows selected.
```

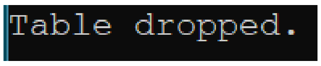
DROP TABLE:

DROP TABLE medicines;



```
Table dropped.
```

DROP TABLE suppliers;



```
Table dropped.
```

RESULT:

Hence database table for suppliers and medicines have been created successfully and explored subqueries, simple join operations.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	06 & 07
Date	10/10/2023

INTRODUCTION TO DATABASE DESIGN NATURAL, EQUI, CROSS JOINS, SET OPERATORS AND TCL

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO2	Write simple and complex SQL queries using DML and DCL commands.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

To create a database table for suppliers and medicine to explore natural, equi, outer, cross joins and set operators among this TCL commands also inserted.

SQL COMMANDS:

```
SET LINESIZE 100;
SET AUTOCOMMIT ON;
```

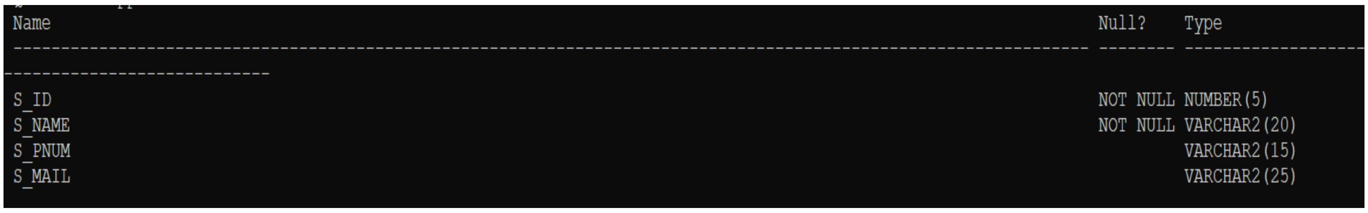
SUPPLIERS AND MEDICINES ENTITY

CREATING SUPPLIERS TABLE :

```
CREATE TABLE suppliers (
   s_id NUMBER(5) PRIMARY KEY,
   s_name VARCHAR2(20) NOT NULL,
   s_pnum VARCHAR2(15),
   s_mail VARCHAR2(25) UNIQUE);
```

```
Table created.
```

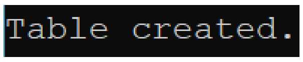
DESC suppliers;



```
Name                                                Null?    Type
--------------------------------------------------- -------- ------------------
S_ID                                                NOT NULL NUMBER(5)
S_NAME                                              NOT NULL VARCHAR2(20)
S_PNUM                                                       VARCHAR2(15)
S_MAIL                                                       VARCHAR2(25)
```

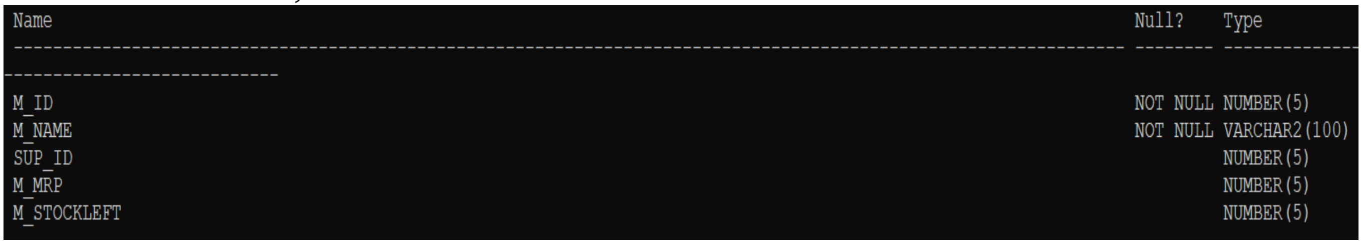
CREATING MEDICINES TABLE :

```
CREATE TABLE medicines (
    m_id NUMBER(5) PRIMARY KEY,
    m_name VARCHAR2(25) NOT NULL,
    sup_id NUMBER(5),
    m_mrp NUMBER(5),
    m_stockleft NUMBER(5));
```



```
Table created.
```

DESC medicines;



```
Name                                                Null?    Type
--------------------------------------------------- -------- -------------
M_ID                                                NOT NULL NUMBER(5)
M_NAME                                              NOT NULL VARCHAR2(100)
SUP_ID                                                       NUMBER(5)
M_MRP                                                        NUMBER(5)
M_STOCKLEFT                                                  NUMBER(5)
```

INSERTING DATA INTO CREATED TABLE :

SUPPLIERS TABLE:

INSERT INTO suppliers VALUES(1,'ABC','9344002774','a@gmail.com');

```
1 row created.
```

INSERT INTO suppliers VALUES(2,'DEF','8344005793','b@gmail.com');

```
1 row created.
```

INSERT INTO suppliers VALUES(3,'FIH','5342027774','c@gmail.com');

```
1 row created.
```

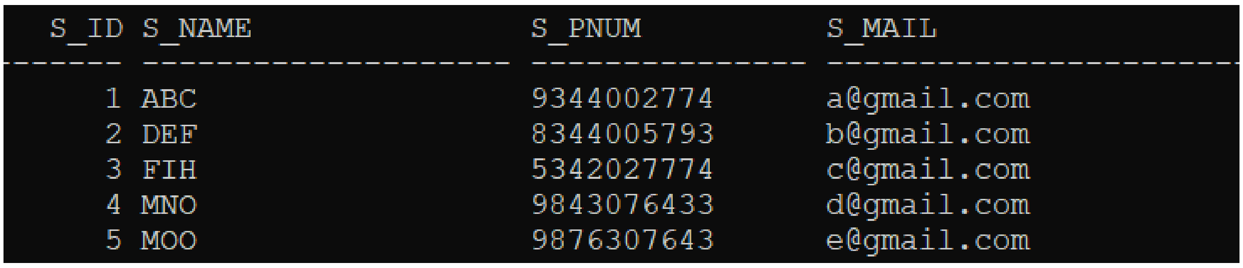
INSERT INTO suppliers VALUES(4,'MNO','9843076433','d@gmail.com');

```
1 row created.
```

INSERT INTO suppliers VALUES(5,'MOO','9876307643','e@gmail.com');

```
1 row created.
```

SELECT * FROM suppliers;



```
  S_ID S_NAME                S_PNUM          S_MAIL
------ -------------------- --------------- ---------------------
     1 ABC                  9344002774      a@gmail.com
     2 DEF                  8344005793      b@gmail.com
     3 FIH                  5342027774      c@gmail.com
     4 MNO                  9843076433      d@gmail.com
     5 MOO                  9876307643      e@gmail.com
```

MEDICINES TABLE:

INSERT INTO medicines VALUES(101,'PainAway',1,25,120);

```
1 row created.
```

INSERT INTO medicines VALUES(102,'CoughRelief',2,10,122);

```
1 row created.
```

INSERT INTO medicines VALUES(103,'thorcare',1,12,25);

```
1 row created.
```

INSERT INTO medicines VALUES(104,'medicure',3,5,12);

```
1 row created.
```

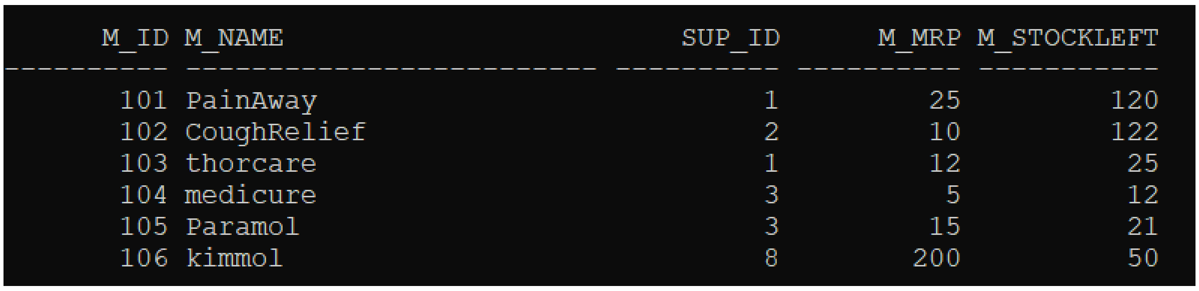
INSERT INTO medicines VALUES(105,'Paramol',3,15,21);

```
1 row created.
```

INSERT INTO medicines VALUES(106,'kimmol',8,200,50);

```
1 row created.
```

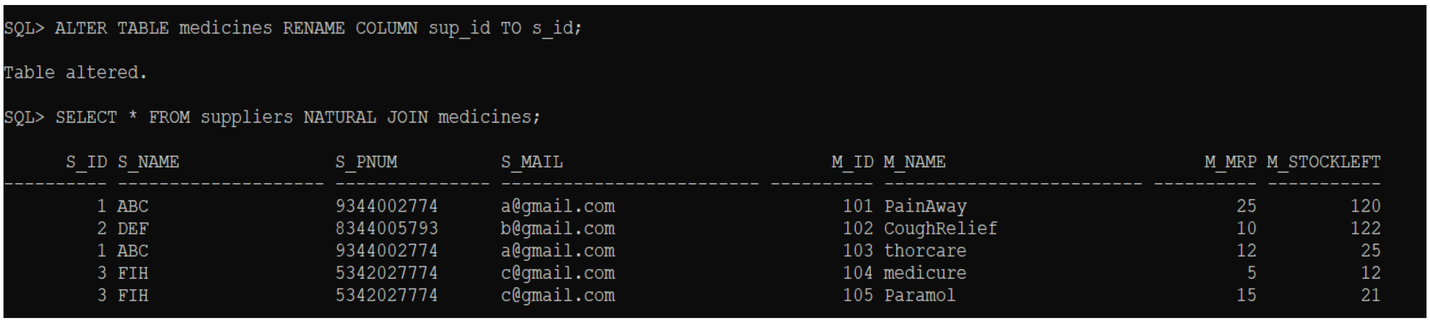
SELECT * FROM medicines;



```
      M_ID M_NAME                        SUP_ID      M_MRP M_STOCKLEFT
---------- ------------------------- ---------- ---------- -----------
       101 PainAway                           1         25         120
       102 CoughRelief                        2         10         122
       103 thorcare                           1         12          25
       104 medicure                           3          5          12
       105 Paramol                            3         15          21
       106 kimmol                             8        200          50
```

NATURAL JOIN:

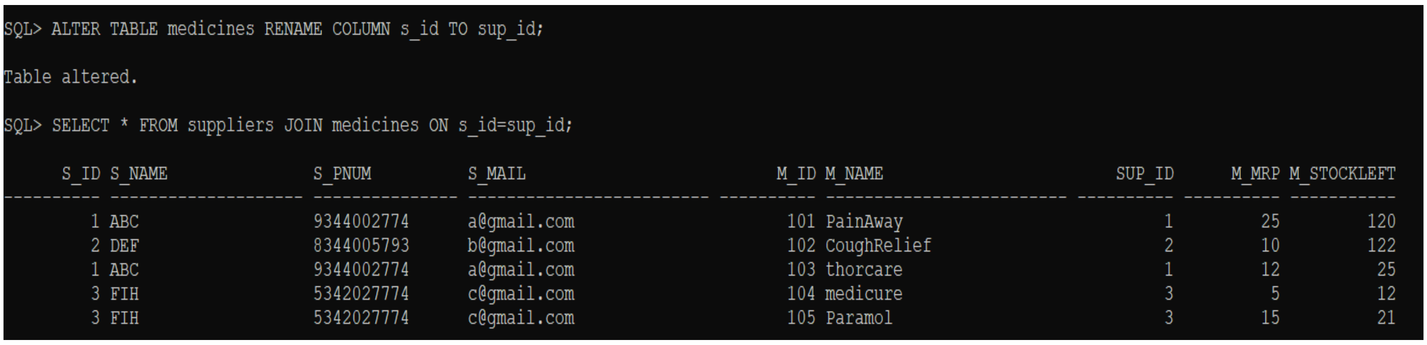
ALTER TABLE medicines RENAME COLUMN sup_id TO s_id;
SELECT * FROM suppliers NATURAL JOIN medicines;



```
SQL> ALTER TABLE medicines RENAME COLUMN sup_id TO s_id;

Table altered.

SQL> SELECT * FROM suppliers NATURAL JOIN medicines;

      S_ID S_NAME               S_PNUM          S_MAIL                          M_ID M_NAME                        M_MRP M_STOCKLEFT
---------- -------------------- --------------- ------------------------- ---------- ------------------------- ---------- -----------
         1 ABC                  9344002774      a@gmail.com                      101 PainAway                          25         120
         2 DEF                  8344005793      b@gmail.com                      102 CoughRelief                       10         122
         1 ABC                  9344002774      a@gmail.com                      103 thorcare                          12          25
         3 FIH                  5342027774      c@gmail.com                      104 medicure                           5          12
         3 FIH                  5342027774      c@gmail.com                      105 Paramol                           15          21
```

EQUI JOIN:

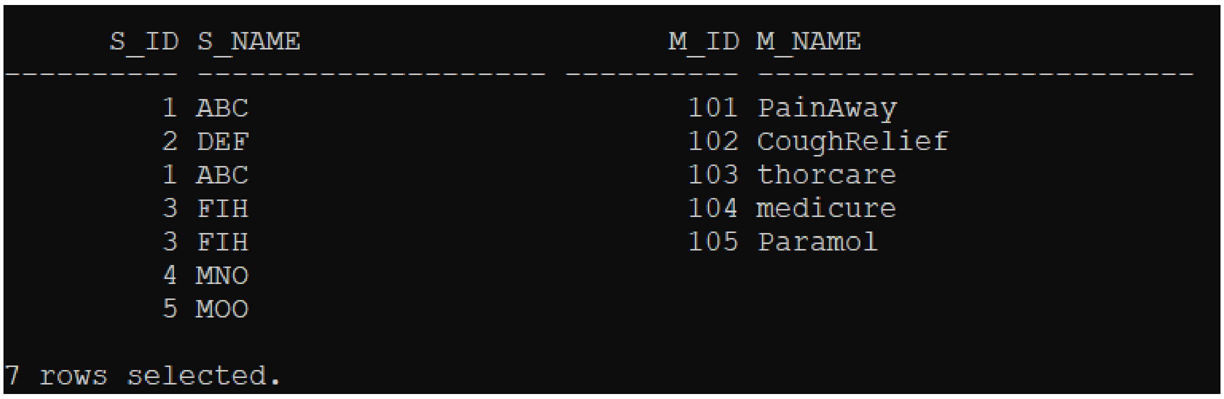
ALTER TABLE medicines RENAME COLUMN s_id TO sup_id;
SELECT * FROM suppliers JOIN medicines ON s_id=sup_id;



```
SQL> ALTER TABLE medicines RENAME COLUMN s_id TO sup_id;

Table altered.

SQL> SELECT * FROM suppliers JOIN medicines ON s_id=sup_id;

      S_ID S_NAME               S_PNUM          S_MAIL                          M_ID M_NAME                        SUP_ID      M_MRP M_STOCKLEFT
---------- -------------------- --------------- ------------------------- ---------- ------------------------- ---------- ---------- -----------
         1 ABC                  9344002774      a@gmail.com                      101 PainAway                           1         25         120
         2 DEF                  8344005793      b@gmail.com                      102 CoughRelief                        2         10         122
         1 ABC                  9344002774      a@gmail.com                      103 thorcare                           1         12          25
         3 FIH                  5342027774      c@gmail.com                      104 medicure                           3          5          12
         3 FIH                  5342027774      c@gmail.com                      105 Paramol                            3         15          21
```

LEFT (OUTER) JOIN:

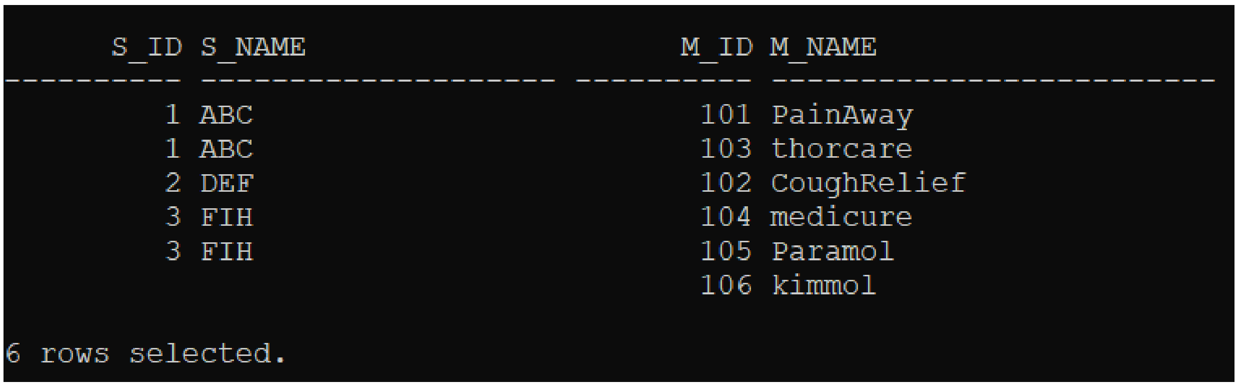
SELECT s_id,s_name,m_ID,m_name FROM suppliers LEFT JOIN medicines ON s_id=sup_id;



```
      S_ID S_NAME                     M_ID M_NAME
---------- -------------------- ---------- -------------------------
         1 ABC                         101 PainAway
         2 DEF                         102 CoughRelief
         1 ABC                         103 thorcare
         3 FIH                         104 medicure
         3 FIH                         105 Paramol
         4 MNO
         5 MOO

7 rows selected.
```

RIGHT (OUTER) JOIN:

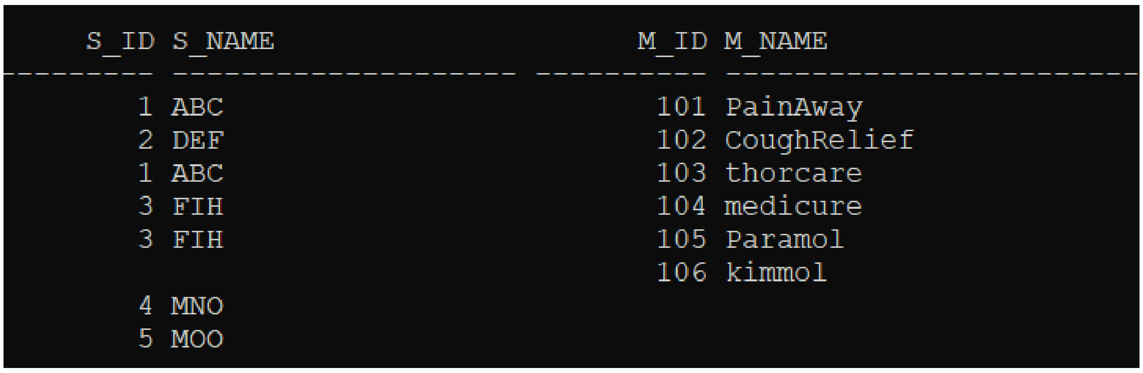
SELECT s_id,s_name,m_ID,m_name FROM suppliers RIGHT JOIN medicines ON s_id=sup_id;



```
      S_ID S_NAME                                M_ID M_NAME
---------- -------------------- ---------- -------------------------
         1 ABC                                    101 PainAway
         1 ABC                                    103 thorcare
         2 DEF                                    102 CoughRelief
         3 FIH                                    104 medicure
         3 FIH                                    105 Paramol
                                                  106 kimmol

6 rows selected.
```

FULL (OUTER) JOIN:

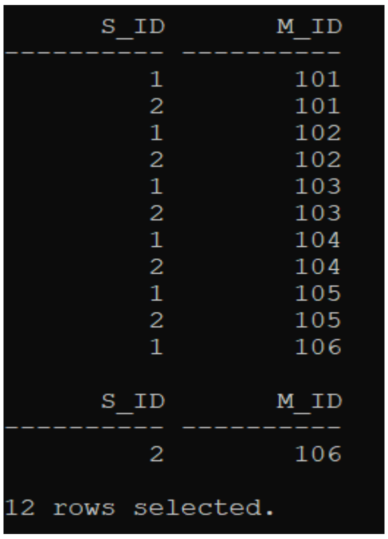
SELECT s_id,s_name,m_ID,m_name FROM suppliers FULL OUTER JOIN medicines ON s_id=sup_id;



```
      S_ID S_NAME                                M_ID M_NAME
---------- -------------------- ---------- -------------------------
         1 ABC                                    101 PainAway
         2 DEF                                    102 CoughRelief
         1 ABC                                    103 thorcare
         3 FIH                                    104 medicure
         3 FIH                                    105 Paramol
                                                  106 kimmol
         4 MNO
         5 MOO
```

CROSS JOIN:

SELECT s_id,m_id FROM suppliers CROSS JOIN medicines where s_id<3;



```
      S_ID       M_ID
---------- ----------
         1        101
         2        101
         1        102
         2        102
         1        103
         2        103
         1        104
         2        104
         1        105
         2        105
         1        106

      S_ID       M_ID
---------- ----------
         2        106

12 rows selected.
```

CREATING TABLE :

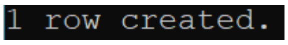
```
CREATE TABLE pharmacy
(
branch_id VARCHAR2(5),
branch_name VARCHAR2(20),
branch_phoneno NUMBER(10),
branch_mail VARCHAR2(30),
branch_location VARCHAR2(50)
);
```

```
Table created.
```

INSERTING DATA INTO CREATED TABLE:

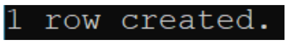
PHARMACY TABLE:

```
INSERT INTO pharmacy VALUES (‘101’, ‘Palvorn’, 9632587412, ‘palvorn@gmail.com’,
‘pallavaram’);
```



```
1 row created.
```

```
INSERT INTO pharmacy VALUES (‘102’, ‘Calvorn’, 9632587512, ‘calvorn@gmail.com’,
‘chrompet’);
```



```
1 row created.
```

ALTER TABLE SUPPLIERS

```
ALTER TABLE suppliers ADD b_id VARCHAR2(4);
```

```
Table altered.
```

UPDATING SUPPLIERS

```
UPDATE suppliers SET b_id=101 WHERE s_id=1 OR s_id=2 OR s_id=5;
```

```
3 rows updated.
```

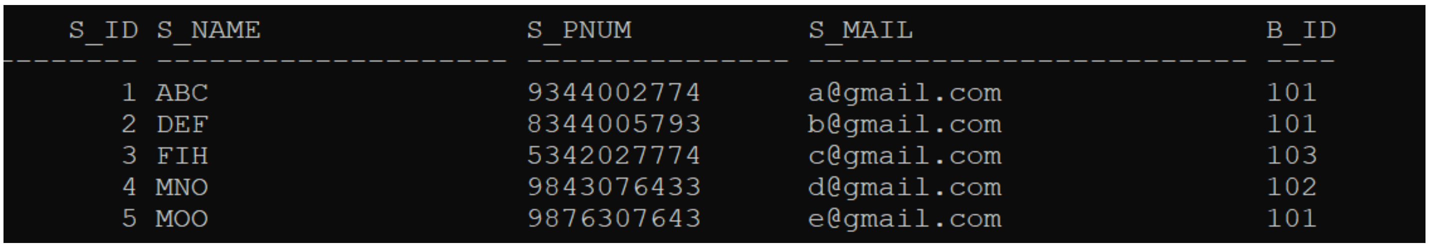
```
UPDATE suppliers SET b_id=103 WHERE s_id=3;
```

```
1 row updated.
```

```
UPDATE suppliers SET b_id=102 WHERE s_id=4;
```

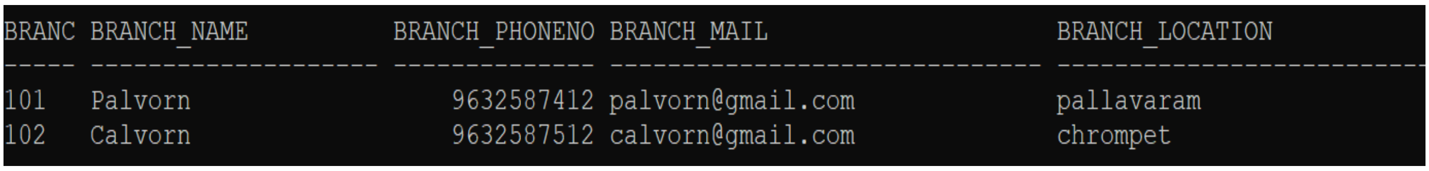
```
1 row updated.
```

SELECT * FROM suppliers;



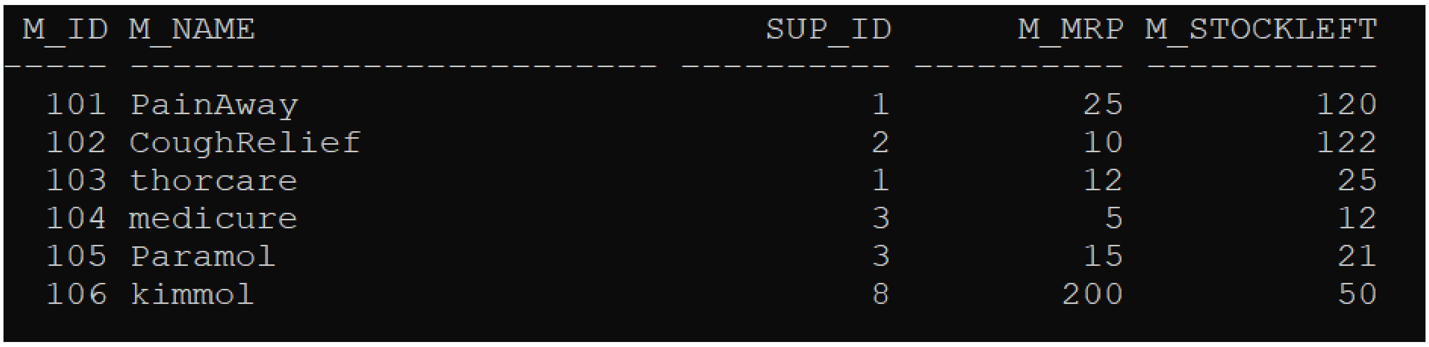
```
    S_ID S_NAME                    S_PNUM          S_MAIL                     B_ID
-------- -------------------- --------------- ------------------------- ----
       1 ABC                  9344002774      a@gmail.com               101
       2 DEF                  8344005793      b@gmail.com               101
       3 FIH                  5342027774      c@gmail.com               103
       4 MNO                  9843076433      d@gmail.com               102
       5 MOO                  9876307643      e@gmail.com               101
```

SELECT * FROM pharmacy;



```
BRANC BRANCH_NAME          BRANCH_PHONENO BRANCH_MAIL                    BRANCH_LOCATION
----- -------------------- -------------- ------------------------------ -------------------------
101   Palvorn                  9632587412 palvorn@gmail.com              pallavaram
102   Calvorn                  9632587512 calvorn@gmail.com              chrompet
```

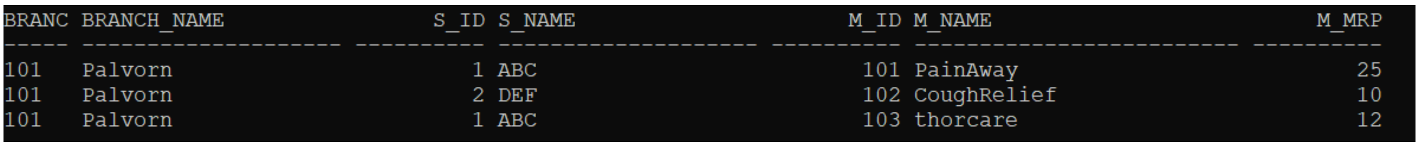
SELECT * FROM medicines;



```
 M_ID M_NAME                          SUP_ID      M_MRP M_STOCKLEFT
----- ------------------------- ---------- ---------- -----------
  101 PainAway                           1         25         120
  102 CoughRelief                        2         10         122
  103 thorcare                           1         12          25
  104 medicure                           3          5          12
  105 Paramol                            3         15          21
  106 kimmol                             8        200          50
```

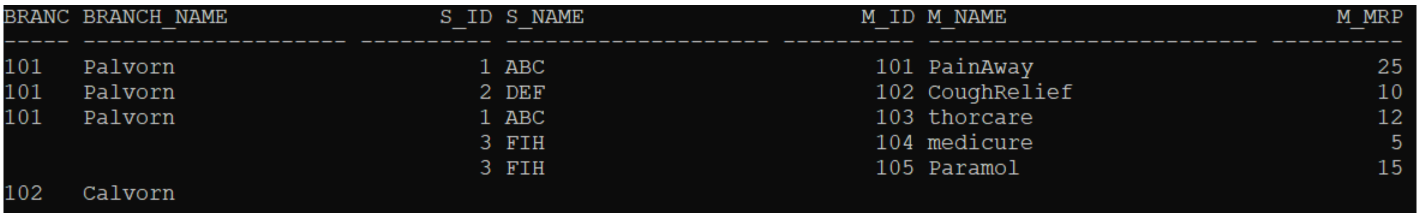
<u>JOIN OPERATIONS ON THREE TABLES:</u>

SELECT branch_id,branch_name,s_id,s_name,m_id,m_name,m_mrp FROM medicines INNER JOIN suppliers ON sup_id = s_id INNER JOIN pharmacy ON b_id = branch_id;



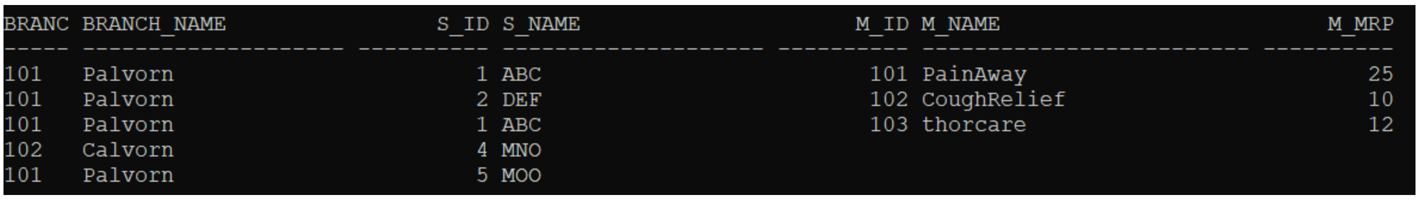
```
BRANC BRANCH_NAME                S_ID S_NAME                     M_ID M_NAME                         M_MRP
----- -------------------- ---------- -------------------- ---------- ------------------------- ----------
101   Palvorn                       1 ABC                         101 PainAway                          25
101   Palvorn                       2 DEF                         102 CoughRelief                       10
101   Palvorn                       1 ABC                         103 thorcare                          12
```

SELECT branch_id,branch_name,s_id,s_name,m_id,m_name,m_mrp FROM medicines INNER JOIN suppliers ON sup_id = s_id FULL OUTER JOIN pharmacy ON b_id = branch_id;



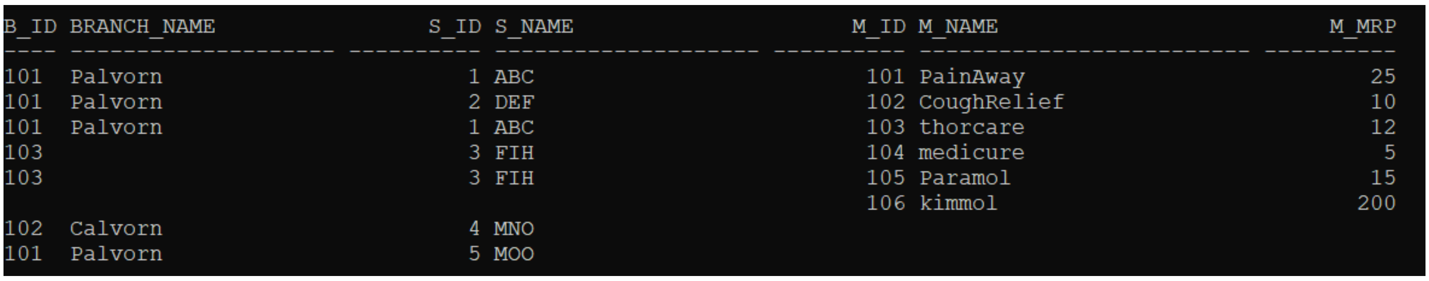
```
BRANC BRANCH_NAME                S_ID S_NAME                     M_ID M_NAME                         M_MRP
----- -------------------- ---------- -------------------- ---------- ------------------------- ----------
101   Palvorn                       1 ABC                         101 PainAway                          25
101   Palvorn                       2 DEF                         102 CoughRelief                       10
101   Palvorn                       1 ABC                         103 thorcare                          12
                                    3 FIH                         104 medicure                           5
                                    3 FIH                         105 Paramol                           15
102   Calvorn
```

SELECT branch_id,branch_name,s_id,s_name,m_id,m_name,m_mrp FROM medicines FULL OUTER JOIN suppliers ON sup_id = s_id INNER JOIN pharmacy ON b_id = branch_id;



```
BRANC BRANCH_NAME                S_ID S_NAME                     M_ID M_NAME                         M_MRP
----- -------------------- ---------- -------------------- ---------- ------------------------- ----------
101   Palvorn                       1 ABC                         101 PainAway                          25
101   Palvorn                       2 DEF                         102 CoughRelief                       10
101   Palvorn                       1 ABC                         103 thorcare                          12
102   Calvorn                       4 MNO
101   Palvorn                       5 MOO
```

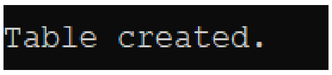
SELECT b_id,branch_name,s_id,s_name,m_id,m_name,m_mrp FROM medicines FULL OUTER JOIN suppliers ON sup_id = s_id FULL OUTER JOIN pharmacy ON b_id = branch_id;



```
B_ID BRANCH_NAME                S_ID S_NAME                     M_ID M_NAME                         M_MRP
---- -------------------- ---------- -------------------- ---------- ------------------------- ----------
101  Palvorn                       1 ABC                         101 PainAway                          25
101  Palvorn                       2 DEF                         102 CoughRelief                       10
101  Palvorn                       1 ABC                         103 thorcare                          12
103                                3 FIH                         104 medicure                           5
103                                3 FIH                         105 Paramol                           15
                                                                 106 kimmol                           200
102  Calvorn                       4 MNO
101  Palvorn                       5 MOO
```

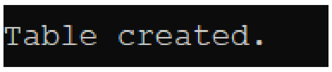
CREATE SUB TABLES:

CREATE TABLE r1 AS (SELECT s_id,s_name FROM suppliers);



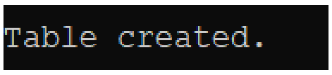
```
Table created.
```

CREATE TABLE r2 AS (SELECT s_id,s_pnum,s_mail FROM suppliers);



```
Table created.
```

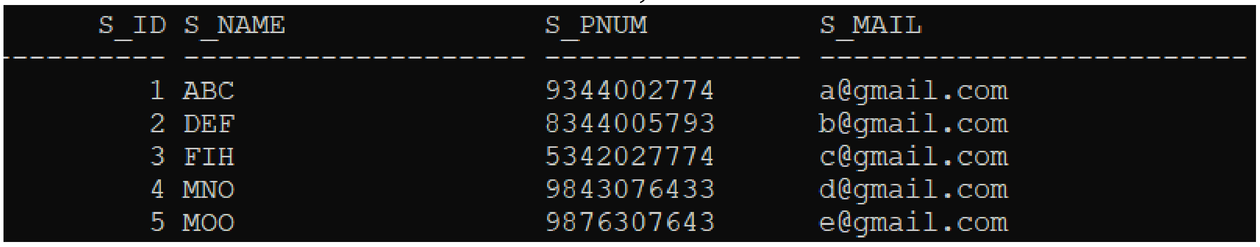
CREATE TABLE r3 AS (SELECT s_id,s_pnum FROM suppliers WHERE s_id<3);



```
Table created.
```

LOSSLESS DECOMPOSITION:

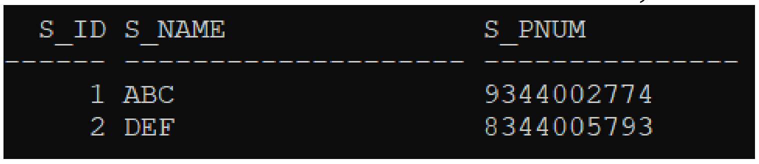
SELECT * FROM r1 NATURAL JOIN r2;



```
      S_ID S_NAME               S_PNUM          S_MAIL
---------- -------------------- --------------- -------------------------
         1 ABC                  9344002774      a@gmail.com
         2 DEF                  8344005793      b@gmail.com
         3 FIH                  5342027774      c@gmail.com
         4 MNO                  9843076433      d@gmail.com
         5 MOO                  9876307643      e@gmail.com
```

LOSSY DECOMPOSITION:

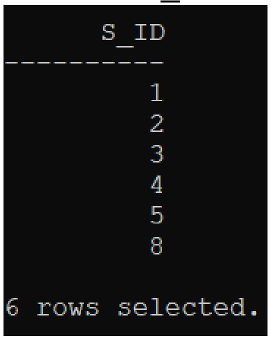
SELECT * FROM r1 NATURAL JOIN r3;



```
  S_ID S_NAME               S_PNUM
------ -------------------- ---------------
     1 ABC                  9344002774
     2 DEF                  8344005793
```

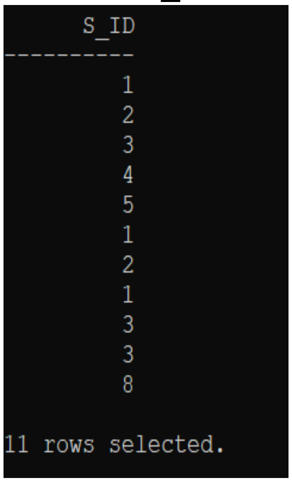
SET OPERATORS:

UNION:
SELECT s_id FROM suppliers UNION SELECT sup_id FROM medicines;



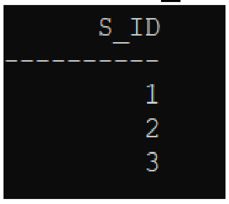
```
      S_ID
----------
         1
         2
         3
         4
         5
         8

6 rows selected.
```

UNION ALL:
SELECT s_id FROM suppliers UNION ALL SELECT sup_id FROM medicines;



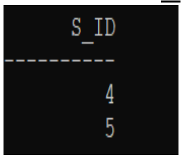
```
      S_ID
----------
         1
         2
         3
         4
         5
         1
         2
         1
         3
         3
         8

11 rows selected.
```

INTERSECTION:
SELECT s_id FROM suppliers INTERSECT SELECT sup_id FROM medicines;

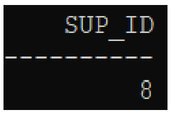


```
      S_ID
----------
         1
         2
         3
```

MINUS:
SELECT s_id FROM suppliers MINUS SELECT sup_id FROM medicines;


```
      S_ID
----------
         4
         5
```

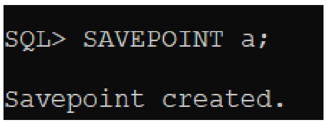
SELECT sup_id FROM medicines MINUS SELECT s_id FROM suppliers;


```
    SUP_ID
----------
         8
```

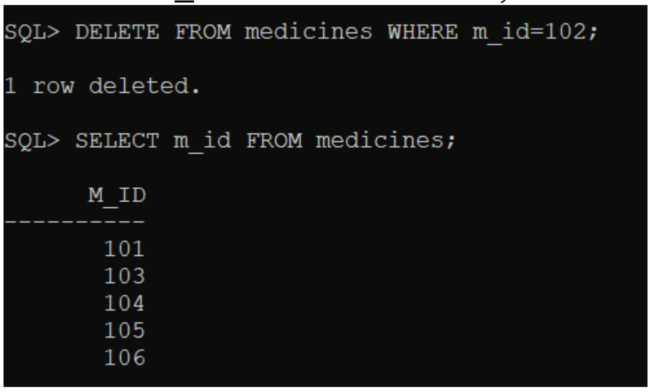
TCL COMMANDS:

SAVEPOINT:

SAVEPOINT a;

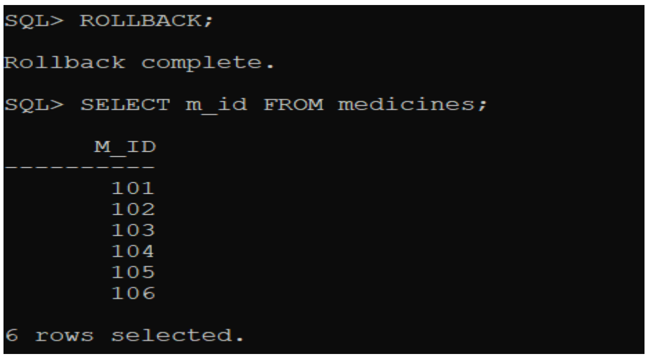

```
SQL> SAVEPOINT a;

Savepoint created.
```

DELETE FROM medicines WHERE m_id=102;
SELECT m_id FROM medicines;


```
SQL> DELETE FROM medicines WHERE m_id=102;

1 row deleted.

SQL> SELECT m_id FROM medicines;

      M_ID
----------
       101
       103
       104
       105
       106
```

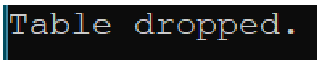
ROLLBACK:

ROLLBACK;
SELECT m_id FROM medicines;


```
SQL> ROLLBACK;

Rollback complete.

SQL> SELECT m_id FROM medicines;

      M_ID
----------
       101
       102
       103
       104
       105
       106

6 rows selected.
```

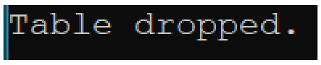
DROP TABLE:

DROP TABLE medicines;



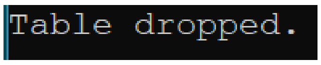
```
Table dropped.
```

DROP TABLE suppliers;



```
Table dropped.
```

DROP TABLE pharmacy;



```
Table dropped.
```

RESULT:

Hence database table for suppliers and medicines have been created successfully and explored outer, cross, natural, equi , set operators and TCL commands.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	08 - A
Date	17/10/2023

WORKING WITH PL/SQL BASICS

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO4	Use Advanced features such as stored procedures and triggers and incorporate in GUI based application development.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

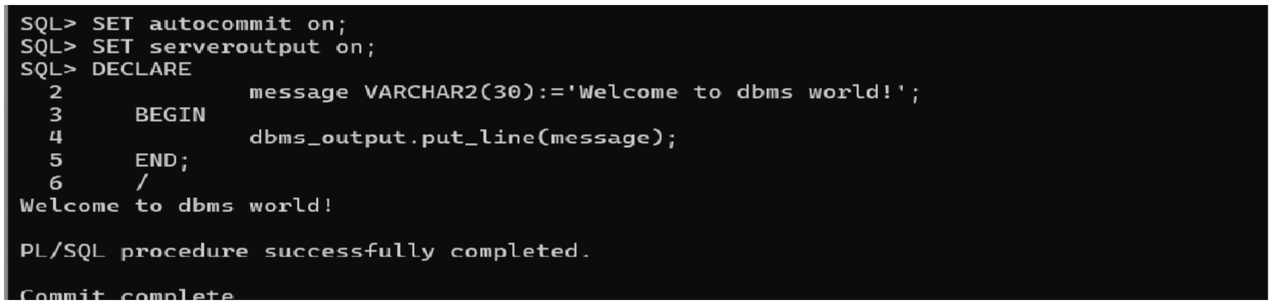
To explore the basics of PL/SQL by solving basic problems.

PL/SQL CODE:

```
SET SERVEROUTPUT ON;
SET AUTOCOMMIT ON;
```

SIMPLE HELLO WORLD PROGRAM

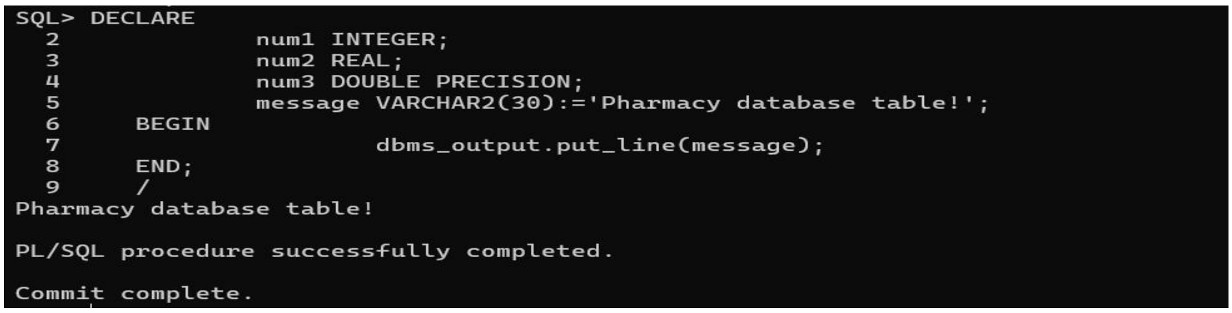
```
DECLARE
message VARCHAR2(30):='Welcome to dbms world!';
BEGIN
dbms_output.put_line(message);
END;
/
```



```
SQL> SET autocommit on;
SQL> SET serveroutput on;
SQL> DECLARE
                message VARCHAR2(30):='Welcome to dbms world!';
        BEGIN
                dbms_output.put_line(message);
        END;
        /
Welcome to dbms world!

PL/SQL procedure successfully completed.

Commit complete
```

DECLARING VARIOUS DATA TYPES

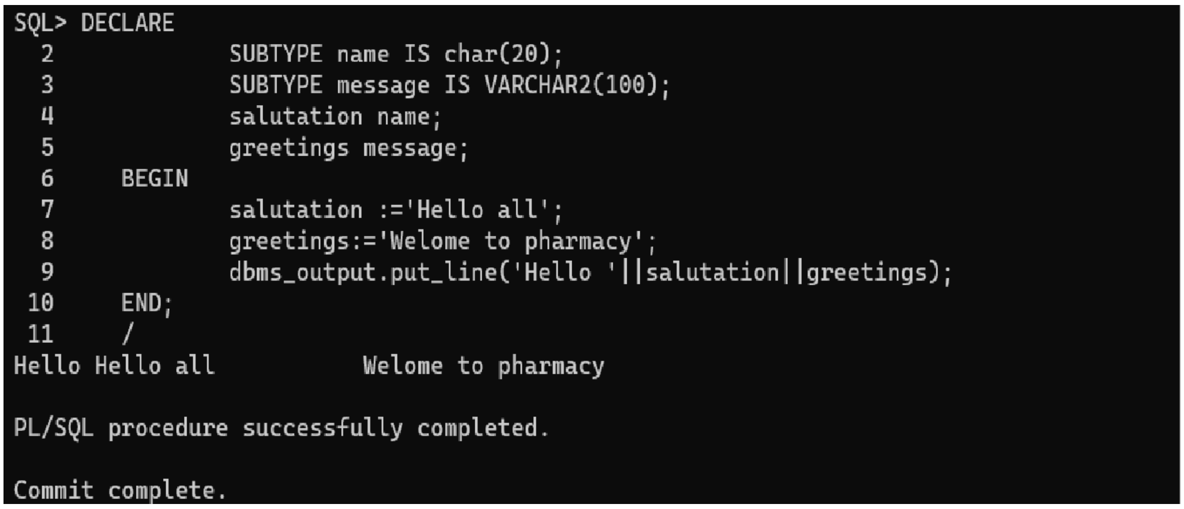
```
DECLARE
        num1 INTEGER;
        num2 REAL;
        num3 DOUBLE PRECISION;
        message VARCHAR2(30):='Pharmacy database table!';
BEGIN
        dbms_output.put_line(message);
END;
/
```



```
SQL> DECLARE
                num1 INTEGER;
                num2 REAL;
                num3 DOUBLE PRECISION;
                message VARCHAR2(30):='Pharmacy database table!';
        BEGIN
                        dbms_output.put_line(message);
        END;
        /
Pharmacy database table!

PL/SQL procedure successfully completed.

Commit complete.
```

USER DEFINED SUBTYPES

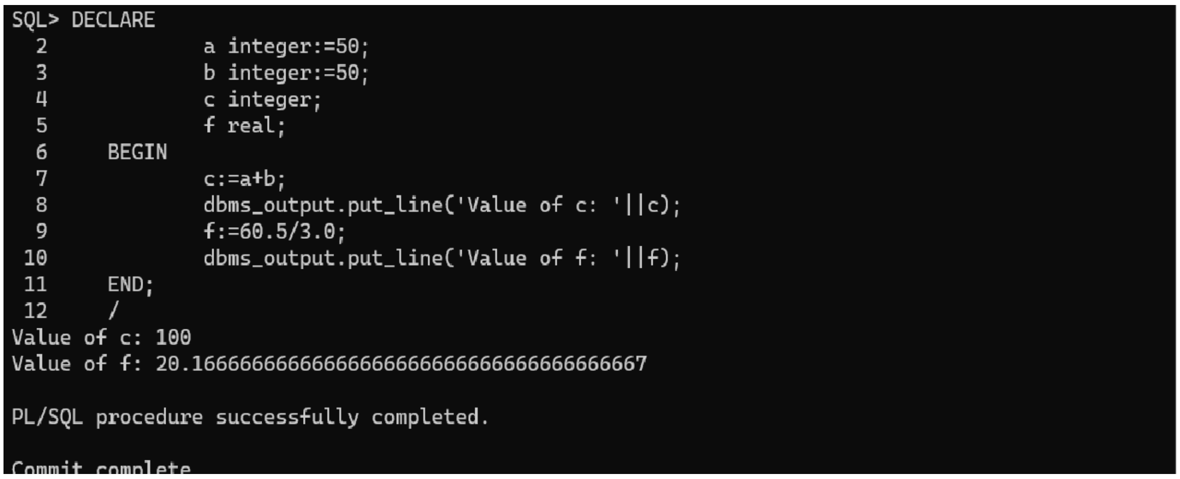
```
DECLARE
        SUBTYPE name IS char(20);
        SUBTYPE message IS VARCHAR2(100);
        salutation name;
        greetings message;
BEGIN
        salutation :='Hello all';
        greetings:='Welome to pharmacy';
        dbms_output.put_line('Hello '||salutation||greetings);
END;
/
```



```
SQL> DECLARE
             SUBTYPE name IS char(20);
             SUBTYPE message IS VARCHAR2(100);
             salutation name;
             greetings message;
      BEGIN
             salutation :='Hello all';
             greetings:='Welome to pharmacy';
             dbms_output.put_line('Hello '||salutation||greetings);
      END;
      /
Hello Hello all           Welome to pharmacy

PL/SQL procedure successfully completed.

Commit complete.
```

PLSQL FOR VARIABLE DECLARATION AND INITIALISATION

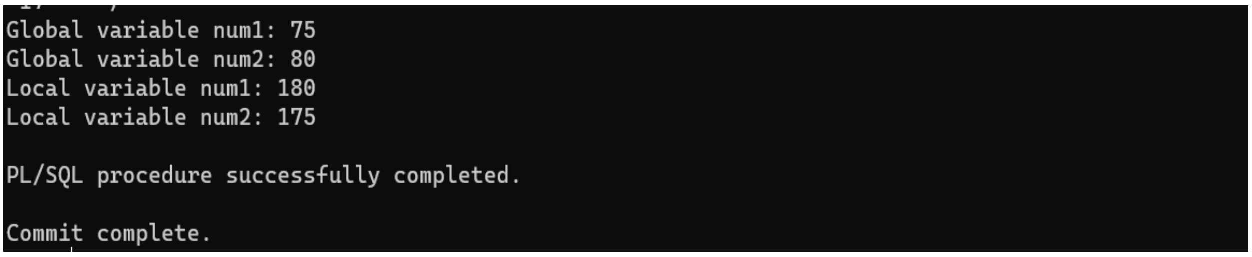
```
DECLARE
        a integer:=50;
        b integer:=50;
        c integer;
        f real;
BEGIN
        c:=a+b;
        dbms_output.put_line('Value of c: '||c);
        f:=60.5/3.0;
        dbms_output.put_line('Value of f: '||f);
END;
/
```



```
SQL> DECLARE
             a integer:=50;
             b integer:=50;
             c integer;
             f real;
      BEGIN
             c:=a+b;
             dbms_output.put_line('Value of c: '||c);
             f:=60.5/3.0;
             dbms_output.put_line('Value of f: '||f);
      END;
      /
Value of c: 100
Value of f: 20.16666666666666666666666666666666666667

PL/SQL procedure successfully completed.

Commit complete.
```

LOCAL AND GLOBAL VARIABLES

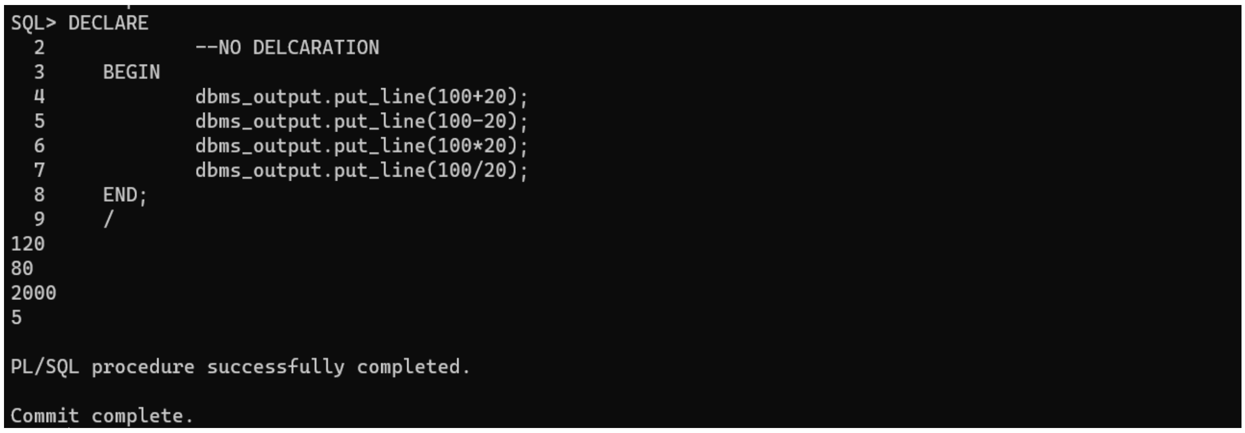
```
DECLARE
        ---Global Variables
        num1 number:=75;
        num2 number:=80;
BEGIN
        dbms_output.put_line('Global variable num1: '||num1);
        dbms_output.put_line('Global variable num2: '||num2);
        DECLARE
                ---Local Variables
                num1 number:=180;
                num2 number:=175;
        BEGIN
                dbms_output.put_line('Local variable num1: '||num1);
                dbms_output.put_line('Local variable num2: '||num2);
        END;
END;
/
```



```
Global variable num1: 75
Global variable num2: 80
Local variable num1: 180
Local variable num2: 175

PL/SQL procedure successfully completed.

Commit complete.
```

OPERATIONS

```
DECLARE
        --NO DELCARATION
        BEGIN
        dbms_output.put_line(100+20);
        dbms_output.put_line(100-20);
        dbms_output.put_line(100*20);
        dbms_output.put_line(100/20);
END;
/
```

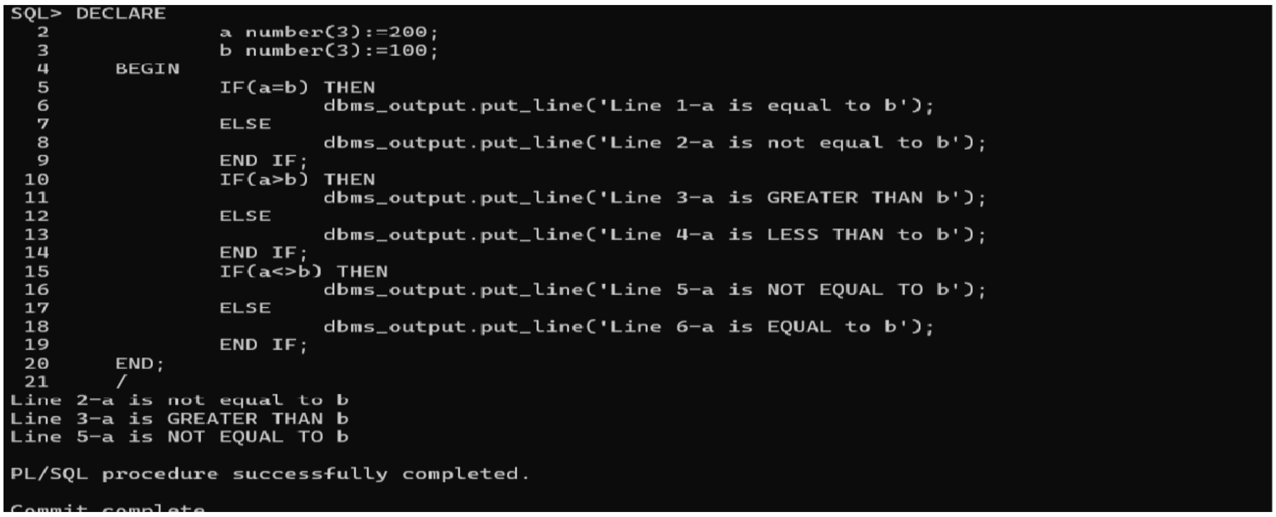


```
SQL> DECLARE
  2            --NO DELCARATION
  3     BEGIN
  4            dbms_output.put_line(100+20);
  5            dbms_output.put_line(100-20);
  6            dbms_output.put_line(100*20);
  7            dbms_output.put_line(100/20);
  8     END;
  9     /
120
80
2000
5

PL/SQL procedure successfully completed.

Commit complete.
```

IF ELSE

```
DECLARE
        a number(3):=200;
        b number(3):=100;
BEGIN
        IF(a=b) THEN
                dbms_output.put_line('Line 1-a is equal to b');
        ELSE
                dbms_output.put_line('Line 2-a is not equal to b');
        END IF;
        IF(a>b)
        THEN
               dbms_output.put_line('Line 3-a is GREATER THAN b');
        ELSE
                 dbms_output.put_line('Line 4-a is LESS THAN to b');
        END IF;
        IF(a<>b)
        THEN
```

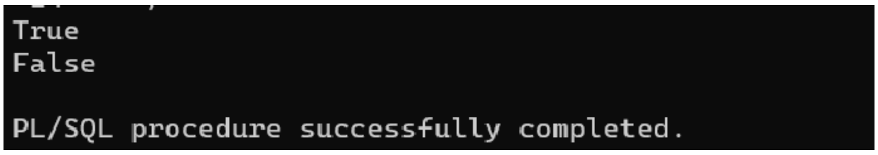
```
            dbms_output.put_line('Line 5-a is NOT EQUAL TO b');
        ELSE
            dbms_output.put_line('Line 6-a is EQUAL to b');
        END IF;
END;
/
```



```
SQL> DECLARE
                a number(3):=200;
                b number(3):=100;
        BEGIN
                IF(a=b) THEN
                        dbms_output.put_line('Line 1-a is equal to b');
                ELSE
                        dbms_output.put_line('Line 2-a is not equal to b');
                END IF;
                IF(a>b) THEN
                        dbms_output.put_line('Line 3-a is GREATER THAN b');
                ELSE
                        dbms_output.put_line('Line 4-a is LESS THAN to b');
                END IF;
                IF(a<>b) THEN
                        dbms_output.put_line('Line 5-a is NOT EQUAL TO b');
                ELSE
                        dbms_output.put_line('Line 6-a is EQUAL to b');
                END IF;
        END;
        /
Line 2-a is not equal to b
Line 3-a is GREATER THAN b
Line 5-a is NOT EQUAL TO b

PL/SQL procedure successfully completed.

Commit complete.
```

PROCEDURE

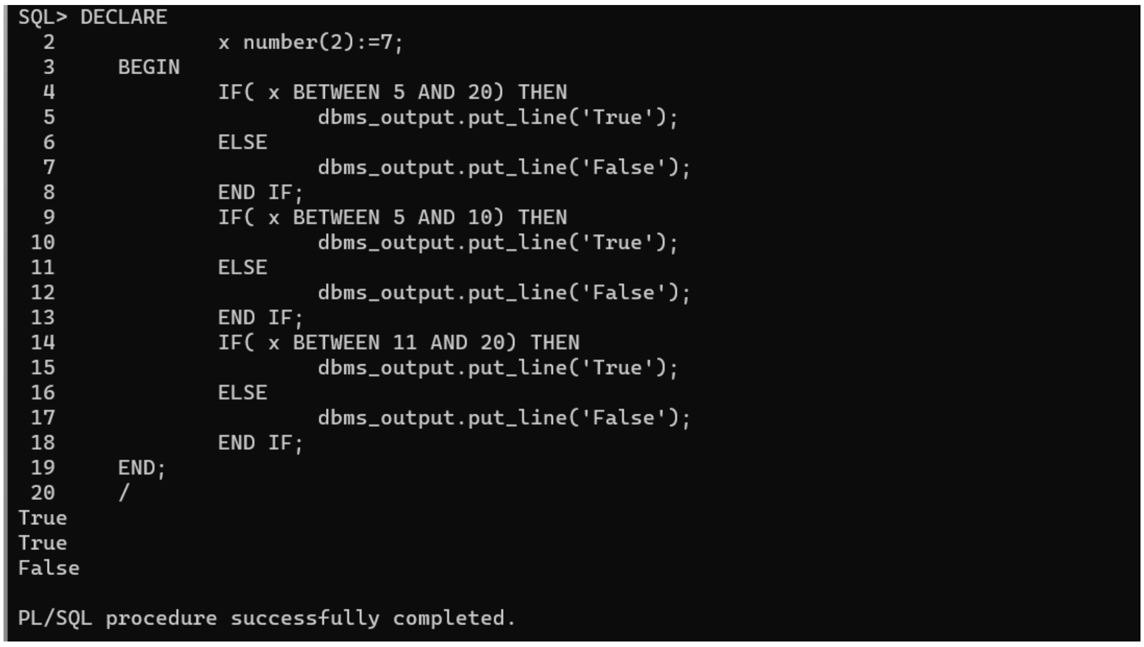
```
DECLARE
        PROCEDURE compare( value varchar2, pattern varchar2) IS
        BEGIN
                IF value LIKE pattern THEN
                        dbms_output.put_line('True');
                ELSE
                        dbms_output.put_line('False');
                END IF;
        END;
        BEGIN
                compare('Gowtham','A%a');
                compare('Aadharsh','A%h');
        END;
        /
```



```
True
False

PL/SQL procedure successfully completed.
```

BETWEEN OPERATOR

```
DECLARE
        x number(2):=7;
BEGIN
        IF( x BETWEEN 5 AND 20) THEN
                dbms_output.put_line('True');
        ELSE
                dbms_output.put_line('False');
        END IF;
        IF( x BETWEEN 5 AND 10) THEN
                dbms_output.put_line('True');
        ELSE
                dbms_output.put_line('False');
        END IF;
        IF( x BETWEEN 11 AND 20) THEN
                dbms_output.put_line('True');
        ELSE
                dbms_output.put_line('False');
        END IF;
END;
/
```



```
SQL> DECLARE
                x number(2):=7;
        BEGIN
                IF( x BETWEEN 5 AND 20) THEN
                        dbms_output.put_line('True');
                ELSE
                        dbms_output.put_line('False');
                END IF;
                IF( x BETWEEN 5 AND 10) THEN
                        dbms_output.put_line('True');
                ELSE
                        dbms_output.put_line('False');
                END IF;
                IF( x BETWEEN 11 AND 20) THEN
                        dbms_output.put_line('True');
                ELSE
                        dbms_output.put_line('False');
                END IF;
        END;
        /
True
True
False

PL/SQL procedure successfully completed.
```

RESULT:

Hence the PL/SQL Basics has been explored and implemented practically.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	08 -B
Date	31/10/2023

LOOPS, PROCEDURE, FUNCTIONS IN PL/SQL

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO4	Use Advanced features such as stored procedures and triggers and incorporate in GUI based application development.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

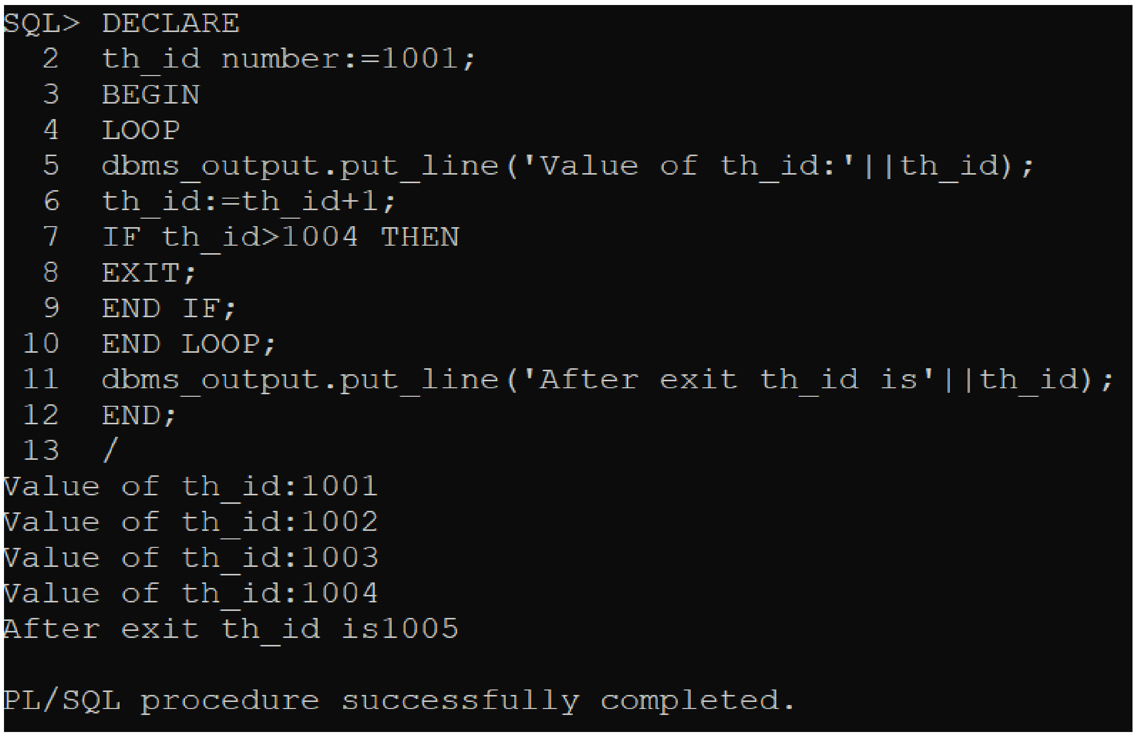
To work with the concept of Looping, creating Procedures, Functions in PL/SQL.

PL/SQL CODE :

```
SET SERVEROUTPUT ON;
SET AUTOCOMMIT ON;
```

loop condition

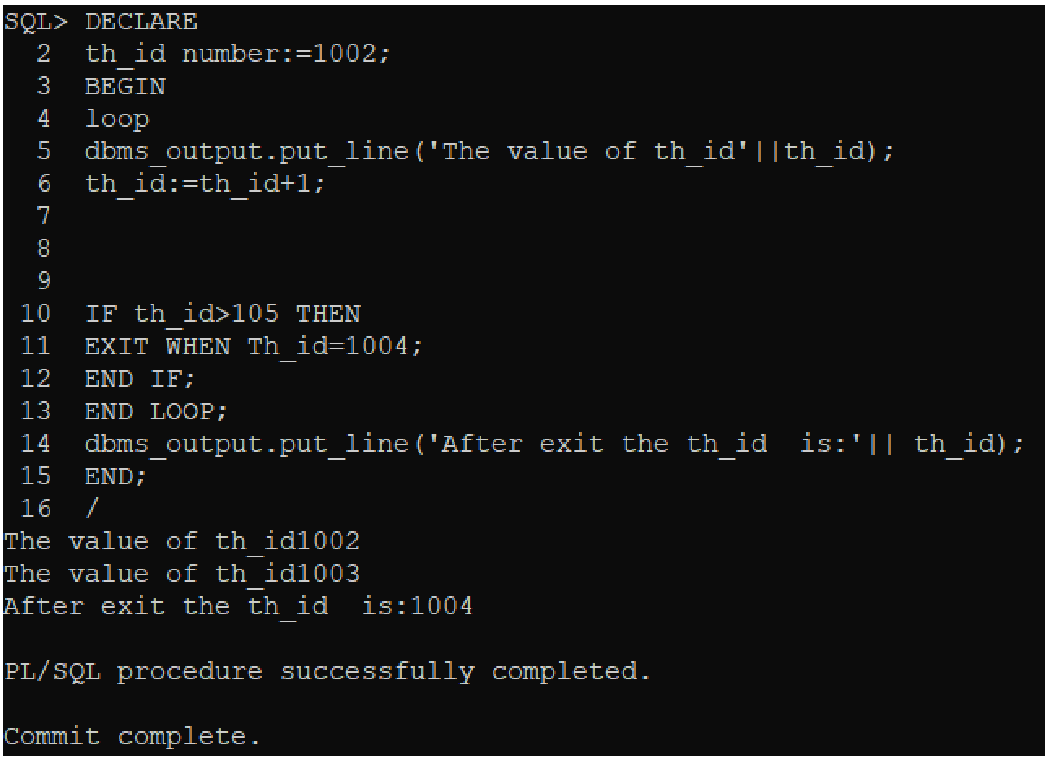
```
DECLARE
        th_id number:=1001;
BEGIN
        LOOP
             dbms_output.put_line('Value of th_id:'||th_id);
             th_id:=th_id+1;
             IF th_id>1004 THEN
             EXIT;
             END IF;
        END LOOP;
        dbms_output.put_line('After exit th_id is'||th_id);
END;
/
```



```
SQL> DECLARE
th_id number:=1001;
BEGIN
LOOP
dbms_output.put_line('Value of th_id:'||th_id);
th_id:=th_id+1;
IF th_id>1004 THEN
EXIT;
END IF;
END LOOP;
dbms_output.put_line('After exit th_id is'||th_id);
END;
/
Value of th_id:1001
Value of th_id:1002
Value of th_id:1003
Value of th_id:1004
After exit th_id is1005

PL/SQL procedure successfully completed.
```

loop with when

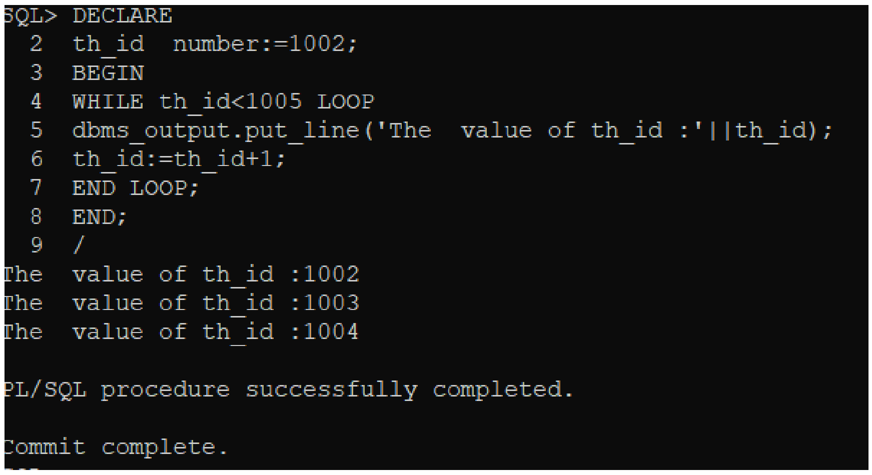
```
DECLARE
		th_id number:=1002;
BEGIN
		loop
				dbms_output.put_line('The value of th_id'||th_id);
				th_id:=th_id+1;
				IF th_id>105 THEN
				EXIT WHEN Th_id=1004;
				END IF;
		END LOOP;
		dbms_output.put_line('After exit the th_id  is:'|| th_id);
END;
/
```



```
SQL> DECLARE
th_id number:=1002;
BEGIN
loop
dbms_output.put_line('The value of th_id'||th_id);
th_id:=th_id+1;



IF th_id>105 THEN
EXIT WHEN Th_id=1004;
END IF;
END LOOP;
dbms_output.put_line('After exit the th_id  is:'|| th_id);
END;
/
The value of th_id1002
The value of th_id1003
After exit the th_id  is:1004

PL/SQL procedure successfully completed.

Commit complete.
```

while loop

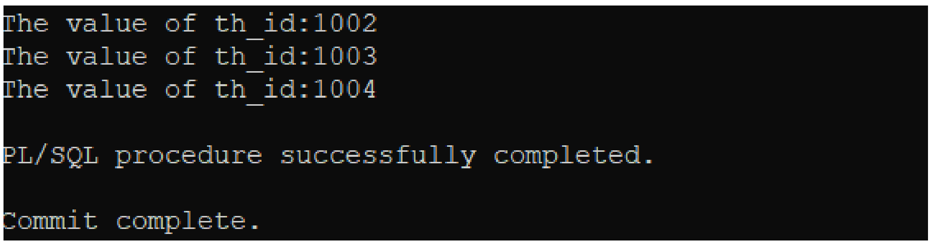
```
DECLARE
        th_id  number:=1002;
BEGIN
        WHILE th_id<1005 LOOP
                dbms_output.put_line('The  value of th_id :'||th_id);
                th_id:=th_id+1;
        END LOOP;
END;
/
```



```
SQL> DECLARE
  2  th_id  number:=1002;
  3  BEGIN
  4  WHILE th_id<1005 LOOP
  5  dbms_output.put_line('The  value of th_id :'||th_id);
  6  th_id:=th_id+1;
  7  END LOOP;
  8  END;
  9  /
The  value of th_id :1002
The  value of th_id :1003
The  value of th_id :1004

PL/SQL procedure successfully completed.

Commit complete.
```

for loop

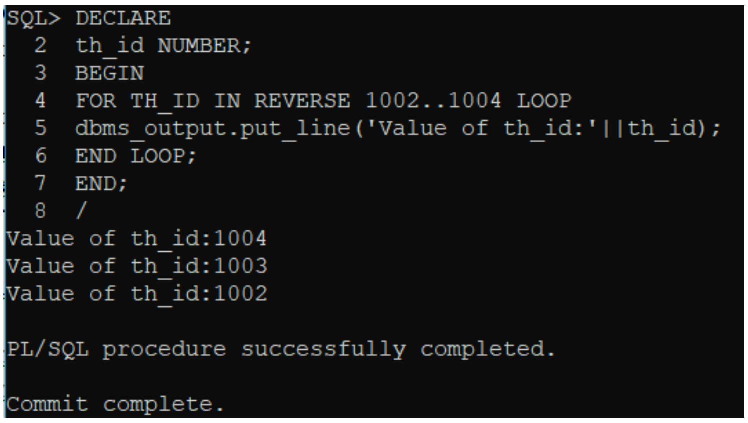
```
DECLARE
        th_id  NUMBER;
BEGIN
        FOR th_id IN 1002..1004 LOOP
                dbms_output.put_line('The value of th_id:'||th_id);
        END LOOP;
END;
/
```



```
The value of th_id:1002
The value of th_id:1003
The value of th_id:1004

PL/SQL procedure successfully completed.

Commit complete.
```

for loop reverse

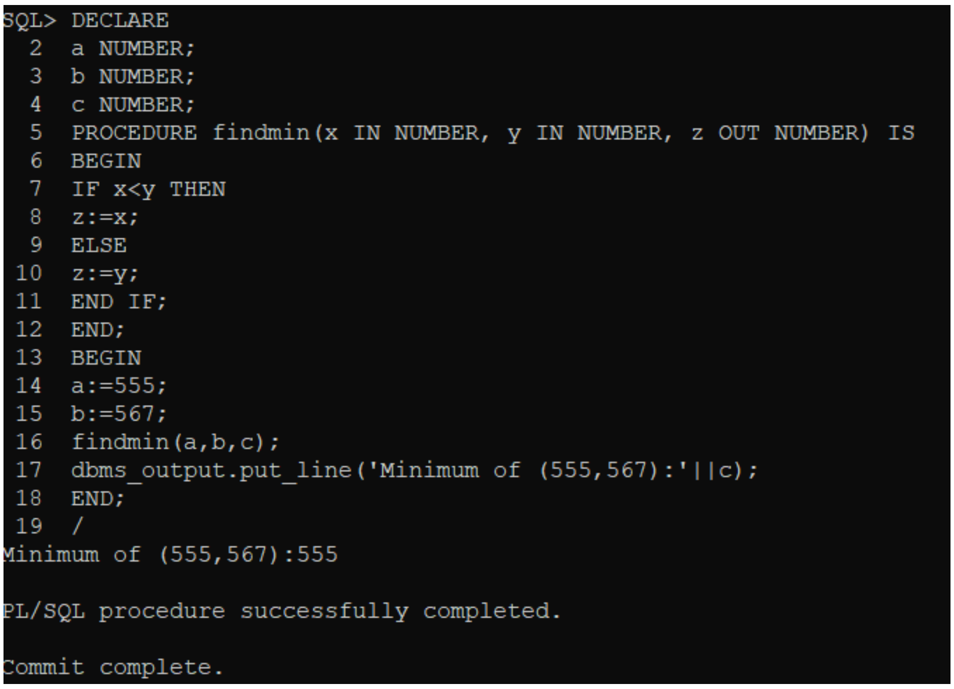
```
DECLARE
		th_id NUMBER;
BEGIN
		FOR TH_ID IN REVERSE 1002..1004 LOOP
				dbms_output.put_line('Value of th_id:'||th_id);
		END LOOP;
END;
/
```



```
SQL> DECLARE
  2  th_id NUMBER;
  3  BEGIN
  4  FOR TH_ID IN REVERSE 1002..1004 LOOP
  5  dbms_output.put_line('Value of th_id:'||th_id);
  6  END LOOP;
  7  END;
  8  /
Value of th_id:1004
Value of th_id:1003
Value of th_id:1002

PL/SQL procedure successfully completed.

Commit complete.
```

procedure finding minimum number

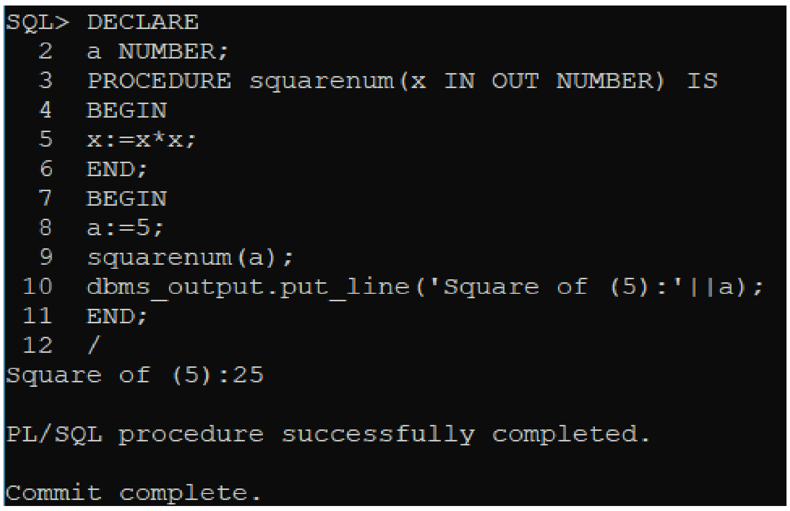
```
DECLARE
		a NUMBER;
		b NUMBER;
		c NUMBER;
	   PROCEDURE findmin(x IN NUMBER, y IN NUMBER, z OUT NUMBER) IS
		BEGIN
				IF x<y THEN
				z:=x;
				ELSE
				z:=y;
				END IF;
		END;
BEGIN
		a:=555;
		b:=567;
		findmin(a,b,c);
		dbms_output.put_line('Minimum of (555,567):'||c);
END;
/
```



```
SQL> DECLARE
a NUMBER;
b NUMBER;
c NUMBER;
PROCEDURE findmin(x IN NUMBER, y IN NUMBER, z OUT NUMBER) IS
BEGIN
IF x<y THEN
z:=x;
ELSE
z:=y;
END IF;
END;
BEGIN
a:=555;
b:=567;
findmin(a,b,c);
dbms_output.put_line('Minimum of (555,567):'||c);
END;
/
Minimum of (555,567):555

PL/SQL procedure successfully completed.

Commit complete.
```

procedure to find square of a number

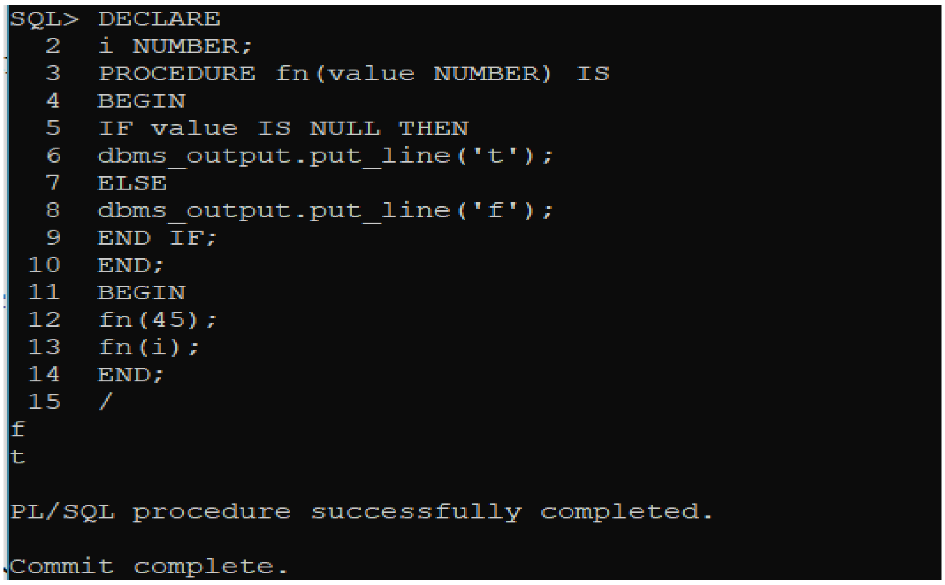
```
DECLARE
        a NUMBER;
        PROCEDURE squarenum(x IN OUT NUMBER) IS
        BEGIN
                x:=x*x;
        END;
BEGIN
        a:=5;
        squarenum(a);
        dbms_output.put_line('Square of (5):'||a);
END;
/
```



```
SQL> DECLARE
a NUMBER;
PROCEDURE squarenum(x IN OUT NUMBER) IS
BEGIN
x:=x*x;
END;
BEGIN
a:=5;
squarenum(a);
dbms_output.put_line('Square of (5):'||a);
END;
/
Square of (5):25

PL/SQL procedure successfully completed.

Commit complete.
```

procedure to find null

```
DECLARE
        i NUMBER;
        PROCEDURE fn(value NUMBER) IS
        BEGIN
                IF value IS NULL THEN
                dbms_output.put_line('t');
                ELSE
                dbms_output.put_line('f');
                END IF;
        END;
BEGIN
        fn(45);
        fn(i);
END;
/
```



```
SQL> DECLARE
  2  i NUMBER;
  3  PROCEDURE fn(value NUMBER) IS
  4  BEGIN
  5  IF value IS NULL THEN
  6  dbms_output.put_line('t');
  7  ELSE
  8  dbms_output.put_line('f');
  9  END IF;
 10  END;
 11  BEGIN
 12  fn(45);
 13  fn(i);
 14  END;
 15  /
f
t

PL/SQL procedure successfully completed.

Commit complete.
```

function used in medicines table

```
CREATE TABLE medicines (
   m_id NUMBER(5) PRIMARY KEY,
   m_name VARCHAR2(20) NOT NULL,
   sup_id NUMBER(5),
   m_mrp NUMBER(5),
   m_stockleft NUMBER(5)
);
```

```
Table created.
```

```
INSERT INTO medicines VALUES(101, 'PainAway', 1, 25, 120);
```

```
1 row created.
```

```
INSERT INTO medicines VALUES(102, 'CoughRelief', 2, 10, 122);
```

```
1 row created.
```

```
INSERT INTO medicines VALUES(103, 'thorcare', 1, 12, 25);
```

```
1 row created.
```

```
INSERT INTO medicines VALUES(104, 'medicure', 3, 5, 12);
```

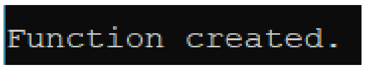
```
1 row created.
```

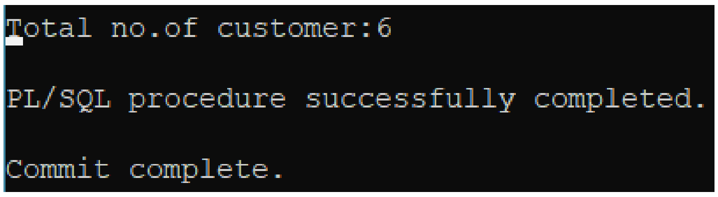
```
INSERT INTO medicines VALUES(105, 'Paramol', 3, 15, 21);
```

```
1 row created.
```

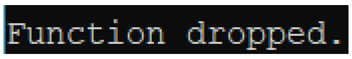
```
INSERT INTO medicines VALUES(106, 'kimmol', 3, 200, 50);
```

```
1 row created.
```

```
CREATE FUNCTION total
RETURN NUMBER IS tot NUMBER:=0;
BEGIN
        SELECT count(*) INTO tot FROM medicines;
        RETURN tot;
END;
/
```

```
Function created.
```

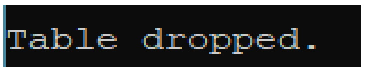
```
DECLARE
        c NUMBER;
BEGIN
        c:=total();
        dbms_output.put_line('Total no.of customer:'||c);
END;
/
```

```
Total no.of customer:6

PL/SQL procedure successfully completed.

Commit complete.
```

DROP function total;



```
Function dropped.
```

DROP TABLE medicines;



```
Table dropped.
```

RESULT:

The Looping, Creating Procedures, Functions and Linking a Table with PL/SQL have been done successfully.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	08 - C
Date	07/11/2023

WORKING WITH EXCEPTIONS, CURSORS AND TRIGGERS IN PL/SQL

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO4	Use Advanced features such as stored procedures and triggers and incorporate in GUI based application development.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

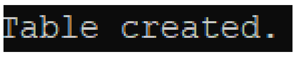
To Work with Exceptions, Cursors, Triggers in PL/SQL.

PL/SQL CODE:

```
SET SERVEROUTPUT ON;
SET AUTOCOMMIT ON;
```

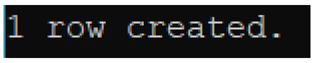
CREATING AND INSERTING VALUES TO MEDICINES TABLE:

```
CREATE TABLE medicines (
    m_id NUMBER(5) PRIMARY KEY,
    m_name VARCHAR2(20) NOT NULL,
    sup_id NUMBER(5),
    m_mrp NUMBER(5),
    m_stockleft NUMBER(5)
);
```



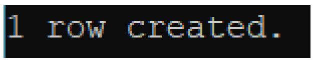
```
Table created.
```

```
INSERT INTO medicines VALUES(101, 'PainAway', 1, 25, 120);
```



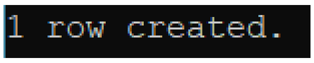
```
1 row created.
```

```
INSERT INTO medicines VALUES(102, 'CoughRelief', 2, 10, 122);
```



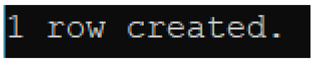
```
1 row created.
```

```
INSERT INTO medicines VALUES(103, 'thorcare', 1, 12, 25);
```



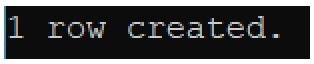
```
1 row created.
```

```
INSERT INTO medicines VALUES(104, 'medicure', 3, 5, 12);
```



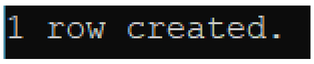
```
1 row created.
```

```
INSERT INTO medicines VALUES(105, 'Paramol', 3, 15, 21);
```



```
1 row created.
```

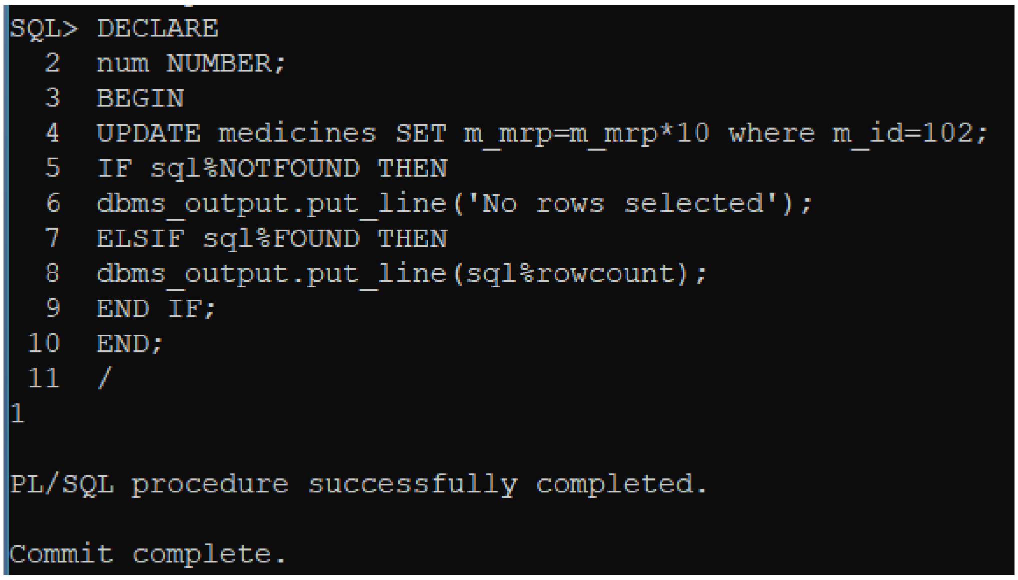
```
INSERT INTO medicines VALUES(106, 'kimmol', 3, 200, 50);
```



```
1 row created.
```

--IMPLICIT CURSORS:

```
DECLARE
        num NUMBER;
BEGIN
        UPDATE medicines SET m_mrp=m_mrp*10 where m_id=102;
        IF sql%NOTFOUND THEN
        dbms_output.put_line('No rows selected');
        ELSIF sql%FOUND THEN
        dbms_output.put_line(sql%rowcount);
        END IF;
END;
/
```

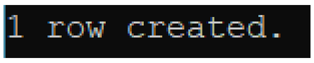


```
SQL> DECLARE
  2  num NUMBER;
  3  BEGIN
  4  UPDATE medicines SET m_mrp=m_mrp*10 where m_id=102;
  5  IF sql%NOTFOUND THEN
  6  dbms_output.put_line('No rows selected');
  7  ELSIF sql%FOUND THEN
  8  dbms_output.put_line(sql%rowcount);
  9  END IF;
 10  END;
 11  /
1

PL/SQL procedure successfully completed.

Commit complete.
```

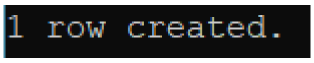
CREATING AND INSERTING VALUES TO SUPPLIERS TABLE:

```
CREATE TABLE suppliers (
   s_id NUMBER(5) PRIMARY KEY,
   s_name VARCHAR2(20) NOT NULL,
   s_pnum VARCHAR2(15),
   s_mail VARCHAR2(20)
);
```

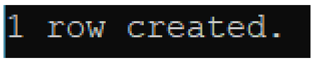
```
Table created.
```

```
INSERT INTO suppliers VALUES(1, 'ABC', '9344002774', 'a@gmail.com');
```


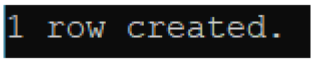
```
1 row created.
```

```
INSERT INTO suppliers VALUES(2, 'DEF', '8344005793', 'b@gmail.com');
```


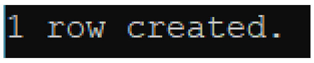
```
1 row created.
```

```
INSERT INTO suppliers VALUES(3, 'FIH', '5342027774', 'c@gmail.com');
```


```
1 row created.
```

```
INSERT INTO suppliers VALUES(4, 'MNO', '9843076433', 'd@gmail.com');
```


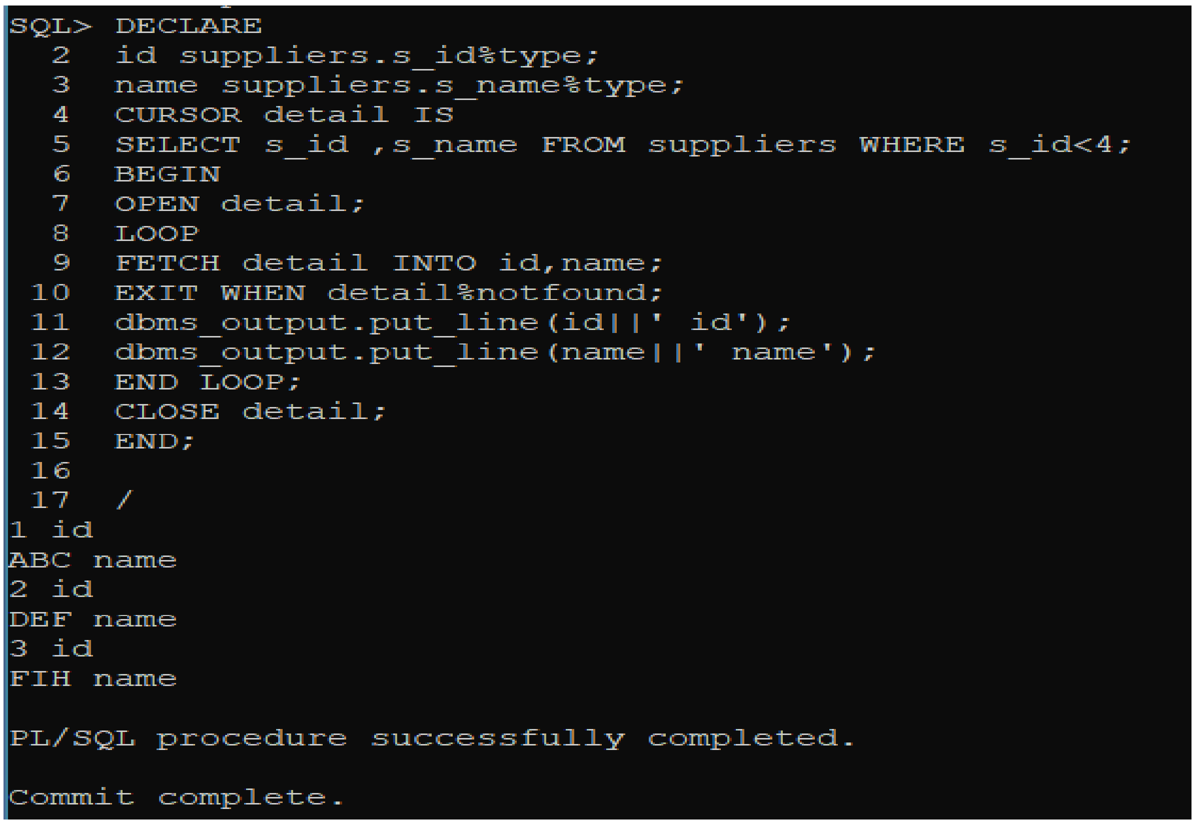
```
1 row created.
```

```
INSERT INTO suppliers VALUES(5, 'MOO', '9876307643', 'e@gmail.com');
```


```
1 row created.
```

--EXPLICIT CURSORS:

```
DECLARE
        id suppliers.s_id%type;
        name suppliers.s_name%type;
        CURSOR detail IS
                SELECT s_id ,s_name FROM suppliers WHERE s_id<4;
BEGIN
        OPEN detail;
        LOOP
                FETCH detail INTO id,name;
                EXIT WHEN detail%notfound;
                dbms_output.put_line(id||' id');
                dbms_output.put_line(name||' name');
        END LOOP;
        CLOSE detail;
END;
/
```

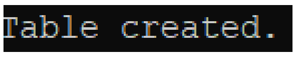


```
SQL> DECLARE
  2  id suppliers.s_id%type;
  3  name suppliers.s_name%type;
  4  CURSOR detail IS
  5  SELECT s_id ,s_name FROM suppliers WHERE s_id<4;
  6  BEGIN
  7  OPEN detail;
  8  LOOP
  9  FETCH detail INTO id,name;
 10  EXIT WHEN detail%notfound;
 11  dbms_output.put_line(id||' id');
 12  dbms_output.put_line(name||' name');
 13  END LOOP;
 14  CLOSE detail;
 15  END;
 16
 17  /
1 id
ABC name
2 id
DEF name
3 id
FIH name

PL/SQL procedure successfully completed.

Commit complete.
```

.

CREATING BACKUP TABLE FOR MEDICINES:

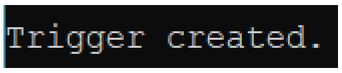
```
CREATE TABLE medicines_backup (
    m_id NUMBER(5),
    m_name VARCHAR2(20),
    sup_id NUMBER(5),
    m_mrp NUMBER(5),
    m_stockleft NUMBER(5)
);
```



```
Table created.
```

--TRIGGER AFTER:

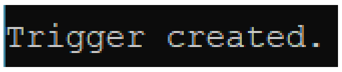
```
CREATE OR REPLACE TRIGGER upmed AFTER
UPDATE ON medicines FOR EACH ROW
BEGIN
   INSERT INTO medicines_backup (m_id, m_name, sup_id, m_mrp, m_stockleft)
   VALUES(:OLD.m_id,:OLD.m_name,:OLD.sup_id,:OLD.m_mrp,:OLD.m_stockleft);
END;
/
```



```
Trigger created.
```

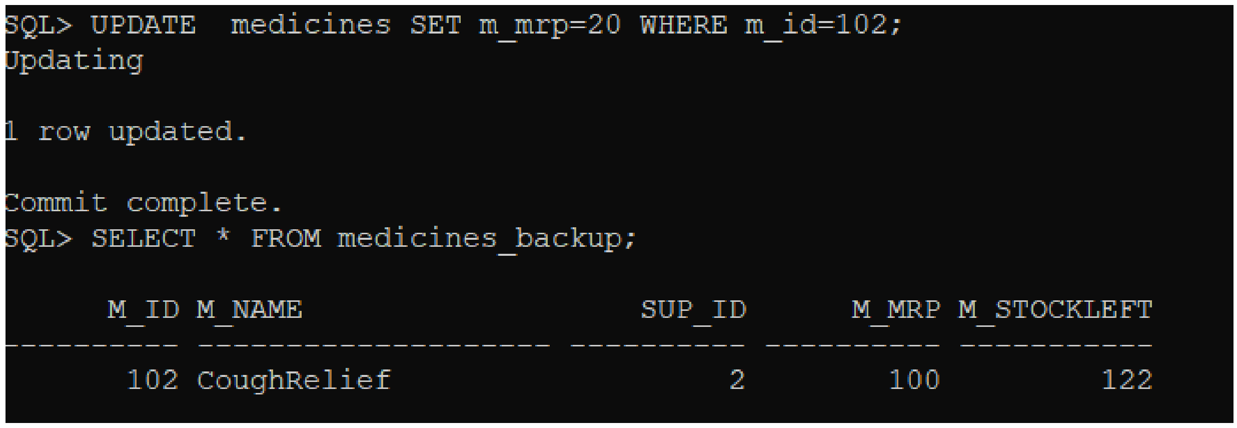
--TRIGGER BEFORE:

```
CREATE OR REPLACE TRIGGER tracking BEFORE
INSERT OR UPDATE OR DELETE ON medicines
BEGIN
        CASE
        WHEN INSERTING THEN
        DBMS_OUTPUT.PUT_LINE('Inserting');
        WHEN DELETING THEN
        DBMS_OUTPUT.PUT_LINE('Deleting');
        WHEN UPDATING THEN
        DBMS_OUTPUT.PUT_LINE('Updating');
        END CASE;
END;
/
```



```
Trigger created.
```

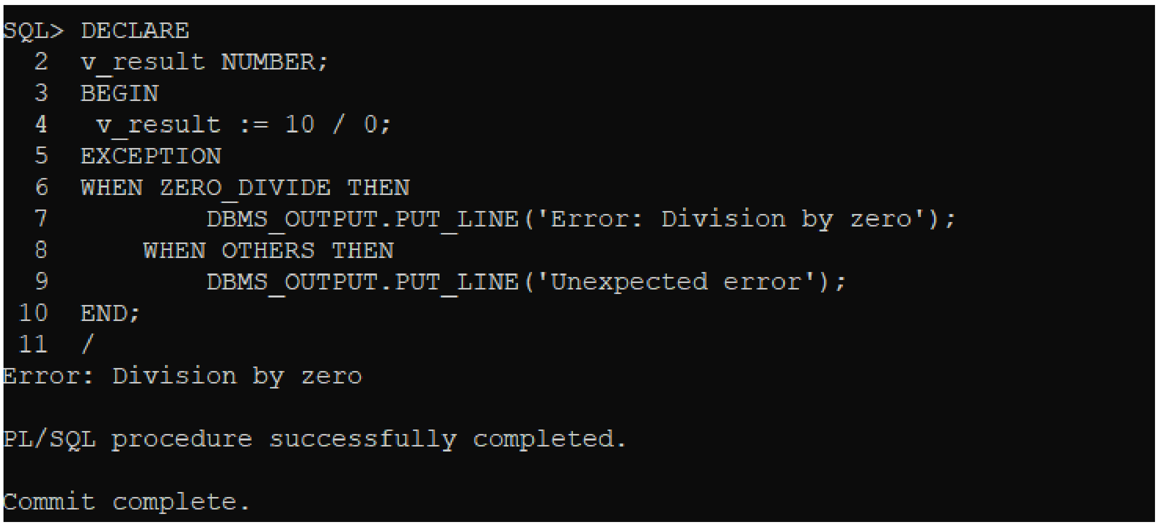
```
UPDATE  medicines SET m_mrp=20 WHERE m_id=102;

SELECT * FROM medicines_backup;
```



```
SQL> UPDATE  medicines SET m_mrp=20 WHERE m_id=102;
Updating

1 row updated.

Commit complete.
SQL> SELECT * FROM medicines_backup;

      M_ID M_NAME                   SUP_ID      M_MRP M_STOCKLEFT
---------- -------------------- ---------- ---------- -----------
       102 CoughRelief                   2        100         122
```

--INBUILT EXCEPTION:

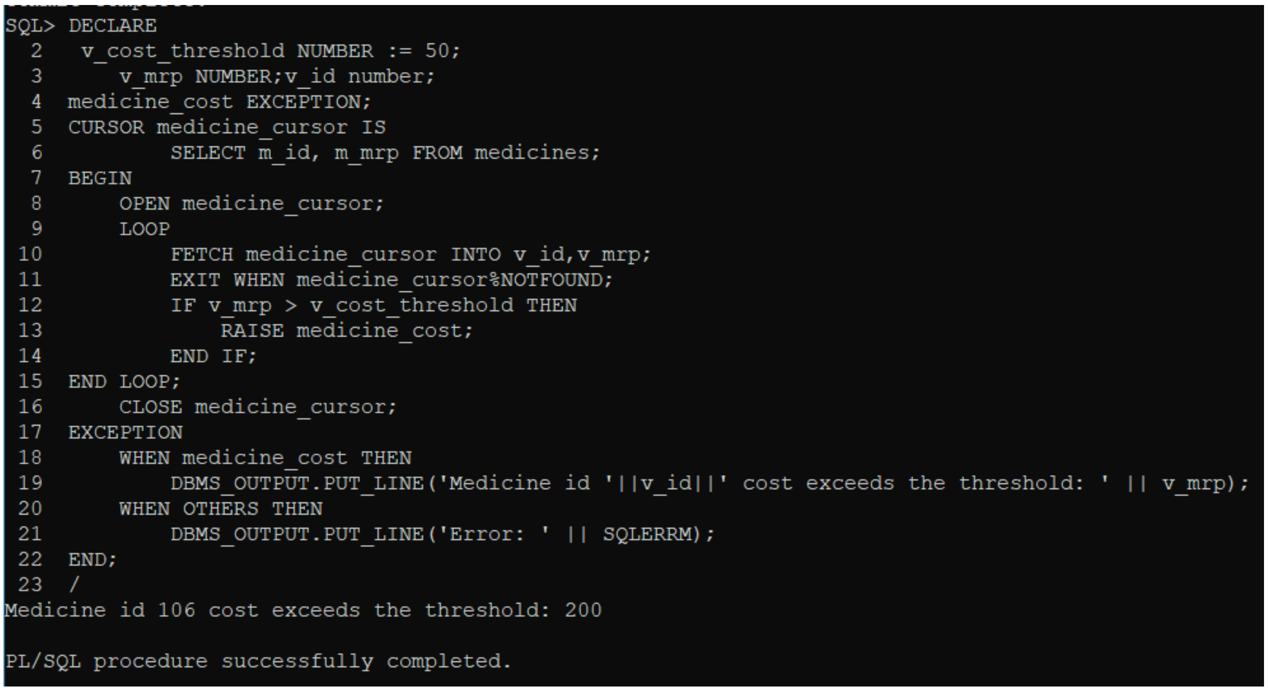
```
DECLARE
        v_result NUMBER;
BEGIN
         v_result := 10 / 0;
EXCEPTION
        WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Error: Division by zero');
        WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE('Unexpected error');
END;
/
```



```
SQL> DECLARE
  2  v_result NUMBER;
  3  BEGIN
  4   v_result := 10 / 0;
  5  EXCEPTION
  6  WHEN ZERO_DIVIDE THEN
  7          DBMS_OUTPUT.PUT_LINE('Error: Division by zero');
  8      WHEN OTHERS THEN
  9          DBMS_OUTPUT.PUT_LINE('Unexpected error');
 10  END;
 11  /
Error: Division by zero

PL/SQL procedure successfully completed.

Commit complete.
```

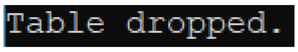
--USERDEFINED EXCEPTION:

```
DECLARE
        v_cost_threshold NUMBER := 50;
        v_mrp NUMBER;v_id number;
        medicine_cost EXCEPTION;
CURSOR medicine_cursor IS
       SELECT m_id, m_mrp FROM medicines;
BEGIN
        OPEN medicine_cursor;
        LOOP
                FETCH medicine_cursor INTO v_id,v_mrp;
                EXIT WHEN medicine_cursor%NOTFOUND;
                IF v_mrp > v_cost_threshold THEN
                        RAISE medicine_cost;
                END IF;
                END LOOP;
         CLOSE medicine_cursor;
        EXCEPTION
        WHEN medicine_cost THEN
        DBMS_OUTPUT.PUT_LINE('Medicine id '||v_id||' cost exceeds the threshold: ' ||
v_mrp);
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



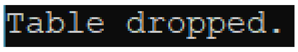
```
SQL> DECLARE
  v_cost_threshold NUMBER := 50;
     v_mrp NUMBER;v_id number;
 medicine_cost EXCEPTION;
 CURSOR medicine_cursor IS
         SELECT m_id, m_mrp FROM medicines;
 BEGIN
     OPEN medicine_cursor;
     LOOP
         FETCH medicine_cursor INTO v_id,v_mrp;
         EXIT WHEN medicine_cursor%NOTFOUND;
         IF v_mrp > v_cost_threshold THEN
             RAISE medicine_cost;
         END IF;
 END LOOP;
     CLOSE medicine_cursor;
 EXCEPTION
     WHEN medicine_cost THEN
         DBMS_OUTPUT.PUT_LINE('Medicine id '||v_id||' cost exceeds the threshold: ' || v_mrp);
     WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
 END;
 /
Medicine id 106 cost exceeds the threshold: 200

PL/SQL procedure successfully completed.
```

DROP TABLE medicines;



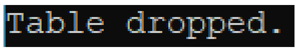
```
Table dropped.
```

DROP TABLE suppliers;



```
Table dropped.
```

DROP TABLE medicines_backup;



```
Table dropped.
```

RESULT:

The Triggers, Cursors, Exception Handling was successfully done in PL/SQL.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	09
Date	14/11/2023

CREATE INDEX DATABASE TABLE WITH LARGE NUMBER OF RECORDS

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO4	Use Advanced features such as stored procedures and triggers andincorporate in GUI based application development.
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

To create an Index in Oracle and work with the same.

QUERIES:

Relation: Student

CREATING INDEX :

Syntax:

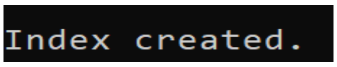
CREATE INDEX index_name

ON table_name (column1 [, column2, ...]);

Query:

CREATE INDEX stu2_ind ON student2 (name);

Output:



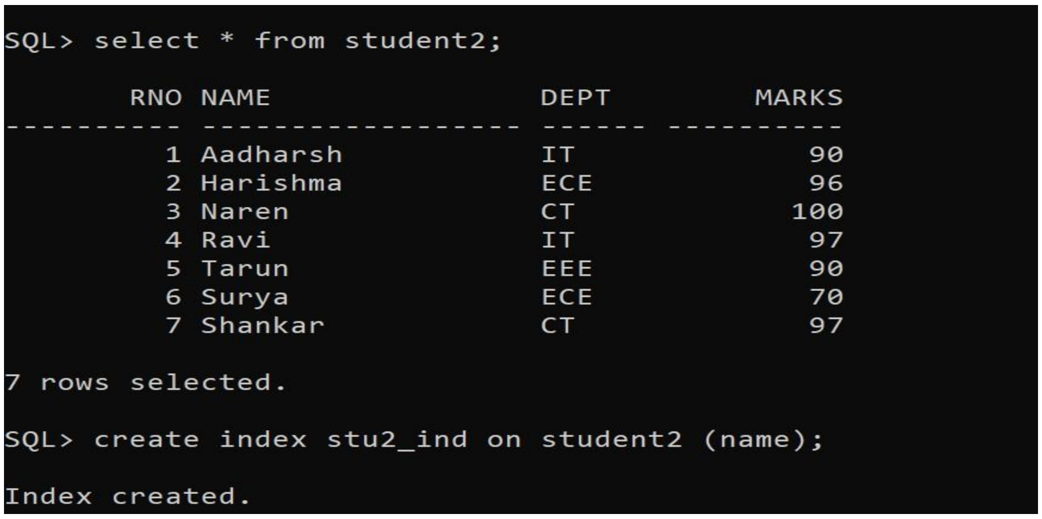
```
Index created.
```

EXECUTING SELECT QUERY :

Query:

SELECT * FROM student2;

Output:



```
SQL> select * from student2;

       RNO NAME                 DEPT        MARKS
---------- -------------------- ------ ----------
         1 Aadharsh             IT             90
         2 Harishma             ECE            96
         3 Naren                CT            100
         4 Ravi                 IT             97
         5 Tarun                EEE            90
         6 Surya                ECE            70
         7 Shankar              CT             97

7 rows selected.

SQL> create index stu2_ind on student2 (name);

Index created.
```

CHECKING THE AVAILABLE INDEXES AND ITS STATUS ON A TABLE :

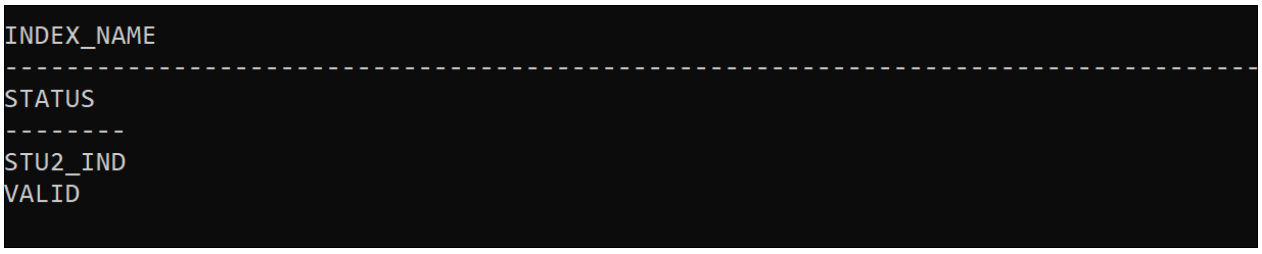
Syntax:

SELECT index_name,status FROM all_indexes WHERE table_name = ‘<Table_Name>’;

Query:

SELECT index_name,status FROM all_indexes WHERE table_name = ‘Student2’;

Output:



```
INDEX_NAME
--------------------------------------------------------------------------------
STATUS
--------
STU2_IND
VALID
```

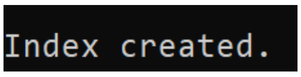
Relation: Employees

CREATING INDEX :

Query:

CREATE INDEX emp_ind ON employees (emp_salary);

Output:



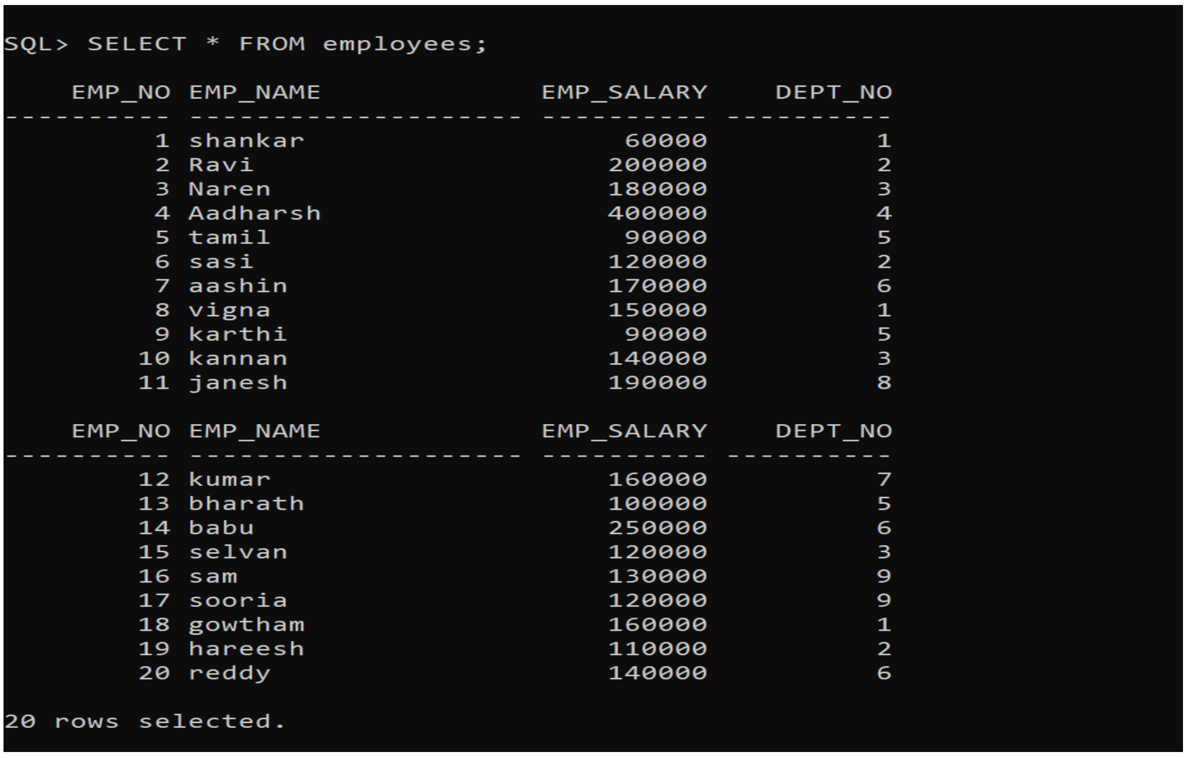
```
Index created.
```

EXECUTING SELECT QUERY :

Query:

SELECT * FROM employees;

Output:



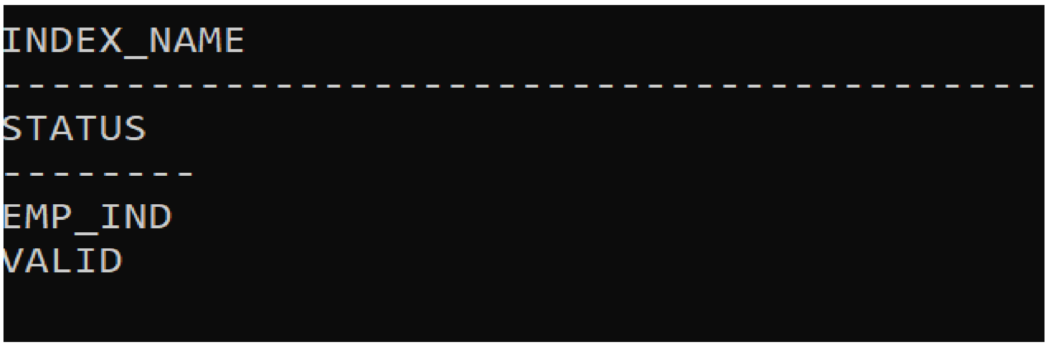
```
SQL> SELECT * FROM employees;

    EMP_NO EMP_NAME             EMP_SALARY    DEPT_NO
---------- -------------------- ---------- ----------
         1 shankar                   60000          1
         2 Ravi                     200000          2
         3 Naren                    180000          3
         4 Aadharsh                 400000          4
         5 tamil                     90000          5
         6 sasi                     120000          2
         7 aashin                   170000          6
         8 vigna                    150000          1
         9 karthi                    90000          5
        10 kannan                   140000          3
        11 janesh                   190000          8

    EMP_NO EMP_NAME             EMP_SALARY    DEPT_NO
---------- -------------------- ---------- ----------
        12 kumar                    160000          7
        13 bharath                  100000          5
        14 babu                     250000          6
        15 selvan                   120000          3
        16 sam                      130000          9
        17 sooria                   120000          9
        18 gowtham                  160000          1
        19 hareesh                  110000          2
        20 reddy                    140000          6

20 rows selected.
```

CHECKING THE AVAILABLE INDEXES AND ITS STATUS ON A TABLE

QUERY:

SELECT index_name,status FROM all_indexes WHERE table_name = ‘employees’;

OUTPUT:



```
INDEX_NAME
--------------------------------------------
STATUS
--------
EMP_IND
VALID
```

RESULT:

The Index in Oracle has been created successfully.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**

Exp No	10
Date	21/11/2023

CREATE A XML DATABASE AND VALIDATE IT USING XML SCHEMA

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO5	Create XML database and validate with meta-data (XML schema)
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

To create and validate xml code for student and employee tables.

QUERIES:

Relation: Employee

XML Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<employee>
<emp>
<name>Aadharsh</name>
<age>19</age>
<department>Executive</department>
```

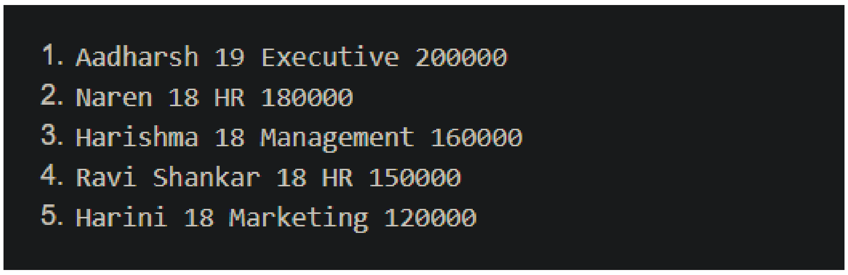
```
<salary>200000</salary>
</emp>
<emp>
<name>Naren</name>
<age>18</age>
<department>HR</department>
<salary>180000</salary>
</emp>
<emp>
<name>Harishma</name>
<age>18</age>
<department>Management</department>
<salary>160000</salary>
</emp>
<emp>
<name>Ravi Shankar</name>
<age>18</age>
<department>HR</department>
<salary>150000</salary>
</emp>
<emp>
<name>Harini</name>
<age>18</age>
<department>Marketing</department>
<salary>120000</salary>
</emp>
</employee>
```

ABSOLUTE XPATH QUERY :

QUERY:

/employee/emp

OUTPUT:

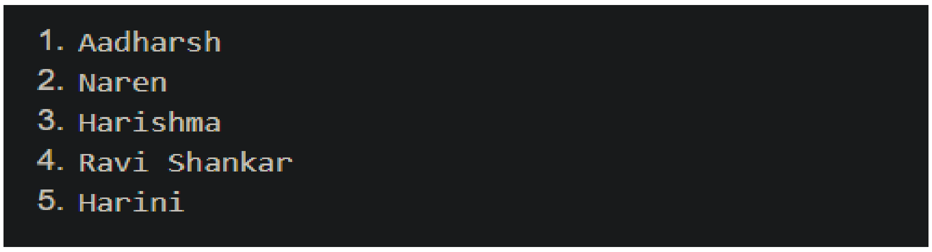


```
1. Aadharsh 19 Executive 200000
2. Naren 18 HR 180000
3. Harishma 18 Management 160000
4. Ravi Shankar 18 HR 150000
5. Harini 18 Marketing 120000
```

QUERY:

/employee/emp/name

OUTPUT:

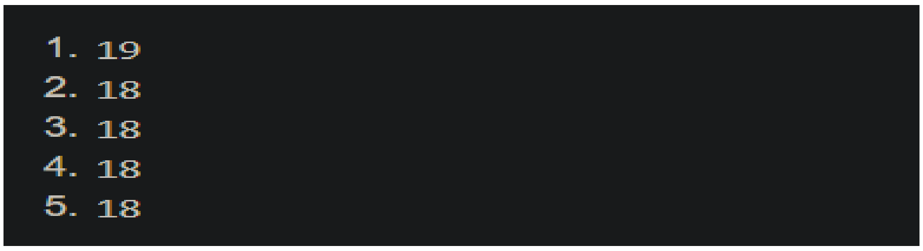


```
1. Aadharsh
2. Naren
3. Harishma
4. Ravi Shankar
5. Harini
```

QUERY:

employee/emp/age

OUTPUT:



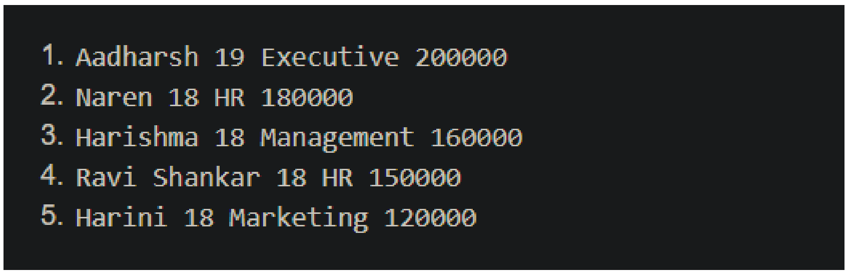
```
1. 19
2. 18
3. 18
4. 18
5. 18
```

RELATIVE XPATH QUERY :

QUERY:

//emp

OUTPUT:

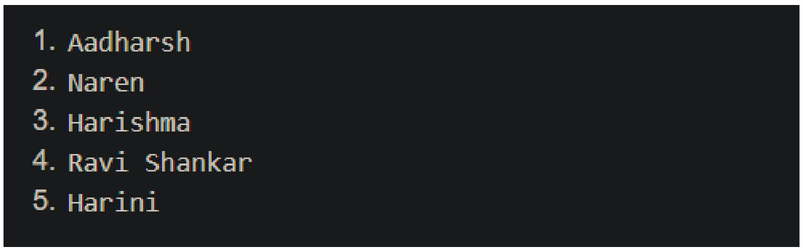


```
1. Aadharsh 19 Executive 200000
2. Naren 18 HR 180000
3. Harishma 18 Management 160000
4. Ravi Shankar 18 HR 150000
5. Harini 18 Marketing 120000
```

QUERY:

//name

OUTPUT:

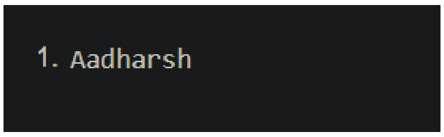


```
1. Aadharsh
2. Naren
3. Harishma
4. Ravi Shankar
5. Harini
```

QUERY:

//name[contains(text(),'Aadharsh')]

OUTPUT:



```
1. Aadharsh
```

EMPLOYEE XSD:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="employee">
<xs:complexType>
<xs:sequence>
<xs:element name="emp" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="name"/>
<xs:element type="xs:byte" name="age"/>
<xs:element type="xs:string" name="department"/>
<xs:element type="xs:int" name="salary"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Relation: Student

<u>XML Code</u>:

```
<?xml version="1.0" encoding="UTF-8"?>
<college>
<Department>
<IT>
<student>
<name>S.Aadharsh</name>
<age>19</age>
<phone>9876543321</phone>
<marks>90</marks>
</student>
<student>
<name>Naren</name>
<age>18</age>
<phone>9876523421</phone>
<marks>100</marks>
</student>
<student>
<name>Harishma</name>
<age>18</age>
<phone>9876349816</phone>
<marks>98</marks>
</student>
</IT>
<CT>
<student>
<name>Janani</name>
```

```
<age>19</age>
<phone>9876542130</phone>
<marks>100</marks>
</student>
<student>
<name>Mithun</name>
<age>18</age>
<phone>9874563321</phone>
<marks>95</marks>
</student>
<student>
<name>Sudharshan</name>
<age>19</age>
<phone>8076543321</phone>
<marks>80</marks>
</student>
</CT>
<ECE>
<student>
<name>Surya</name>
<age>19</age>
<phone>7010255275</phone>
<marks>90</marks>
</student>
<student>
<name>Tarun</name>
<age>19</age>
<phone>9629147042</phone>
<marks>99</marks>
</student>
```

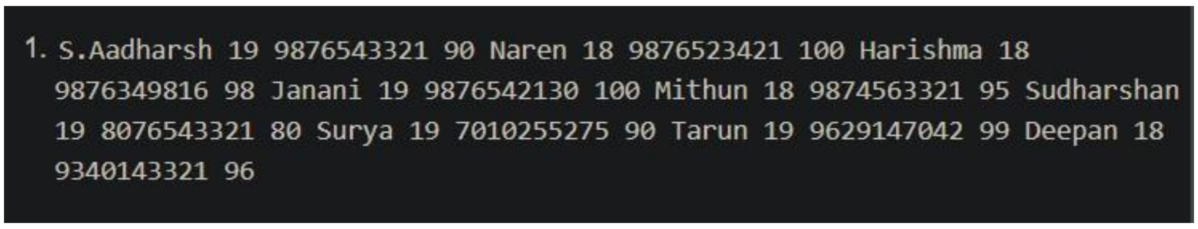
```
<student>
<name>Deepan</name>
<age>18</age>
<phone>9340143321</phone>
<marks>96</marks>
</student>
</ECE>
</Department>
</college>
```

<u>**ABSOLUTE XPATH QUERY :**</u>

Query:

/college

OUTPUT:

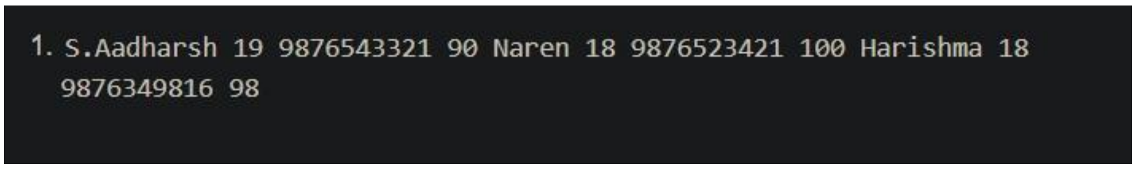


```
1. S.Aadharsh 19 9876543321 90 Naren 18 9876523421 100 Harishma 18
   9876349816 98 Janani 19 9876542130 100 Mithun 18 9874563321 95 Sudharshan
   19 8076543321 80 Surya 19 7010255275 90 Tarun 19 9629147042 99 Deepan 18
   9340143321 96
```

QUERY:

/college/Department/IT

OUTPUT:

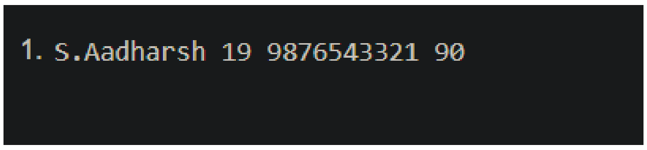


```
1. S.Aadharsh 19 9876543321 90 Naren 18 9876523421 100 Harishma 18
   9876349816 98
```

QUERY:

/college/Department/IT/student[name = 'S.Aadharsh']

OUTPUT:



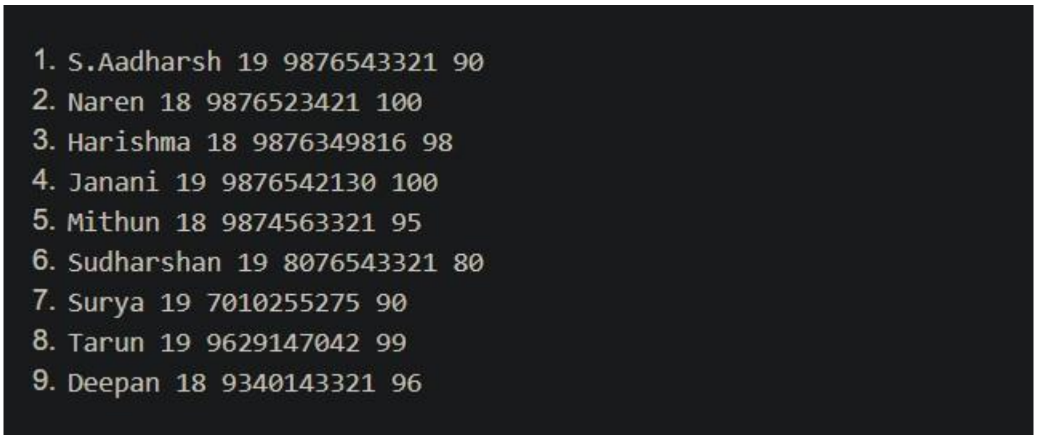
```
1. S.Aadharsh 19 9876543321 90
```

RELATIVE XPATH QUERY:

QUERY:

//student

OUTPUT:

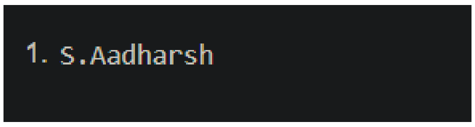


```
1. S.Aadharsh 19 9876543321 90
2. Naren 18 9876523421 100
3. Harishma 18 9876349816 98
4. Janani 19 9876542130 100
5. Mithun 18 9874563321 95
6. Sudharshan 19 8076543321 80
7. Surya 19 7010255275 90
8. Tarun 19 9629147042 99
9. Deepan 18 9340143321 96
```

QUERY:

//student/name[contains(text(),'S.Aadharsh')]

OUTPUT:

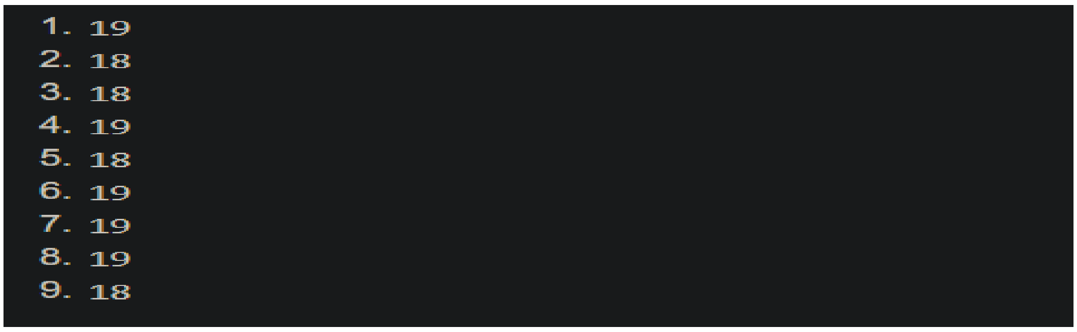


```
1. S.Aadharsh
```

QUERY:

//age

OUTPUT:



```
1. 19
2. 18
3. 18
4. 19
5. 18
6. 19
7. 19
8. 19
9. 18
```

<u>Employee XSD:</u>

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="college">
<xs:complexType>
<xs:sequence>
<xs:element name="Department">
<xs:complexType>
<xs:sequence>
<xs:element name="IT">
<xs:complexType>
<xs:sequence>
<xs:element name="student" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="name"/>
<xs:element type="xs:byte" name="age"/>
<xs:element type="xs:long" name="phone"/>
```

```
<xs:element type="xs:byte" name="marks"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CT">
<xs:complexType>
<xs:sequence>
<xs:element name="student" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="name"/>
<xs:element type="xs:byte" name="age"/>
<xs:element type="xs:long" name="phone"/>
<xs:element type="xs:byte" name="marks"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ECE">
<xs:complexType>
<xs:sequence>
```

```
<xs:element name="student" maxOccurs="unbounded" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="name"/>
<xs:element type="xs:byte" name="age"/>
<xs:element type="xs:long" name="phone"/>
<xs:element type="xs:byte" name="marks"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

RESULT:

Thus xml code for student and employee tables were created and validated.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement andValidate	**…../10**	**…../10**

Exp No	11
Date	28/11/2023

CREATING A DATABASE USING NOSQL DATABASE TOOL

This experiment maps to the following CO and PO.

CO1	Create databases with different types of key constraints.
CO6	Create and manipulate data using NOSQL database
PO1	Engineering Knowledge.
PO3	Design And Development Of Solutions.
PO4	Conduct Experiments / Collect Analysis.
PO5	Modern Tool Usage.
PO9	Individual and Team Work.
PO10	Communication.

AIM:

To implement CRUD operations on student collection in MongoDB.

QUERIES:

CREATING THE DATABASE :

SYNTAX:

```
use mydatabase
```

QUERY:

```
use student
```

OUTPUT:

```
switched to db student
```

Creating A Collection:

SYNTAX:

```
db.createCollection("Collection_name")
```

QUERY:

```
db.createCollection("student_collection")
```

OUTPUT:

```
{ ok: 1 }
```

INSERTION:

Insertion Using InsertOne:

SYNTAX:

```
db.collectionName.insertOne(
{
field1: value1,
field2: value2,
// Additional fields as needed
}
)
```

QUERY:

```
db.studentcollection.insertOne({ regno: 1, name: "Aadharsh", branch: "IT", address:
"Chromepet", phoneno: 9876543210, total_marks: 90 })
```

OUTPUT:

```
{
acknowledged: true,
insertedId: ObjectId("656afde271e5ff331ee0ae9d")
}
```

<u>Insertion Using InsertMany:</u>

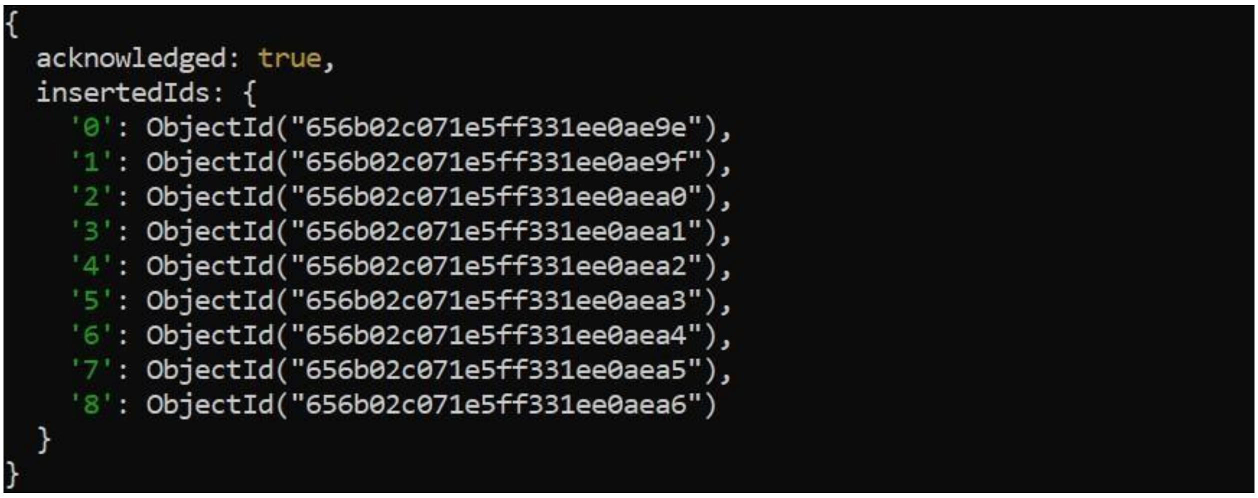
SYNTAX:

```
db.collectionName.insertMany(
[
{ field1: value1, field2: value2, ... },
{ field1: value1, field2: value2, ... },
// Additional documents as needed
]
)
```

QUERY:

```
db.studentcollection.insertMany([{ regno: 2, name: "Harishma", branch: "ECE", address: "Velachery", phoneno: 9876543201, total_marks: 98},
{regno:3,name:"Naren",branch:"CT",address:"Pallavaram",phoneno:9876543102,total_marks:100},
{regno:3,name:"Ravi",branch:"IT",address:"Tirusulam",phoneno:9876534210,total_marks:95},
{regno:5,name:"Shankar",branch:"PT",address:"Meenambakkam",phoneno:8976534210,total_marks:97},
{regno:6,name:"Anisha",branch:"EI",address:"Kallakurichi",phoneno:8967534210,total_marks:94},
{regno:6,name:"Harini",branch:"ECE",address:"Koyambedu",phoneno:8967543210,total_marks:98},
{regno:8,name:"Tamil",branch:"AERO",address:"Guindy",phoneno:8976543210,total_marks:93},
{regno:9,name:"Anu",branch:"AERO",address:"Guindy",phoneno:6776543210,total_marks:92},
{regno:10,name:"Yuvashri",branch:"Auto",address:"Saidapet",phoneno:6767543210,total_marks:99} ])
```

OUTPUT:



```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("656b02c071e5ff331ee0ae9e"),
    '1': ObjectId("656b02c071e5ff331ee0ae9f"),
    '2': ObjectId("656b02c071e5ff331ee0aea0"),
    '3': ObjectId("656b02c071e5ff331ee0aea1"),
    '4': ObjectId("656b02c071e5ff331ee0aea2"),
    '5': ObjectId("656b02c071e5ff331ee0aea3"),
    '6': ObjectId("656b02c071e5ff331ee0aea4"),
    '7': ObjectId("656b02c071e5ff331ee0aea5"),
    '8': ObjectId("656b02c071e5ff331ee0aea6")
  }
}
```

RETRIEVAL :

Retrieval Using find:

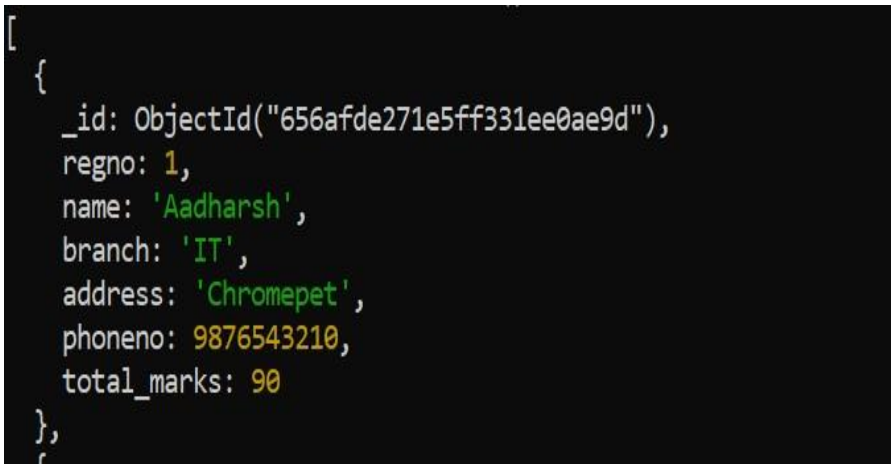
SYNTAX:

db.collectionname.find({condition},{projection})

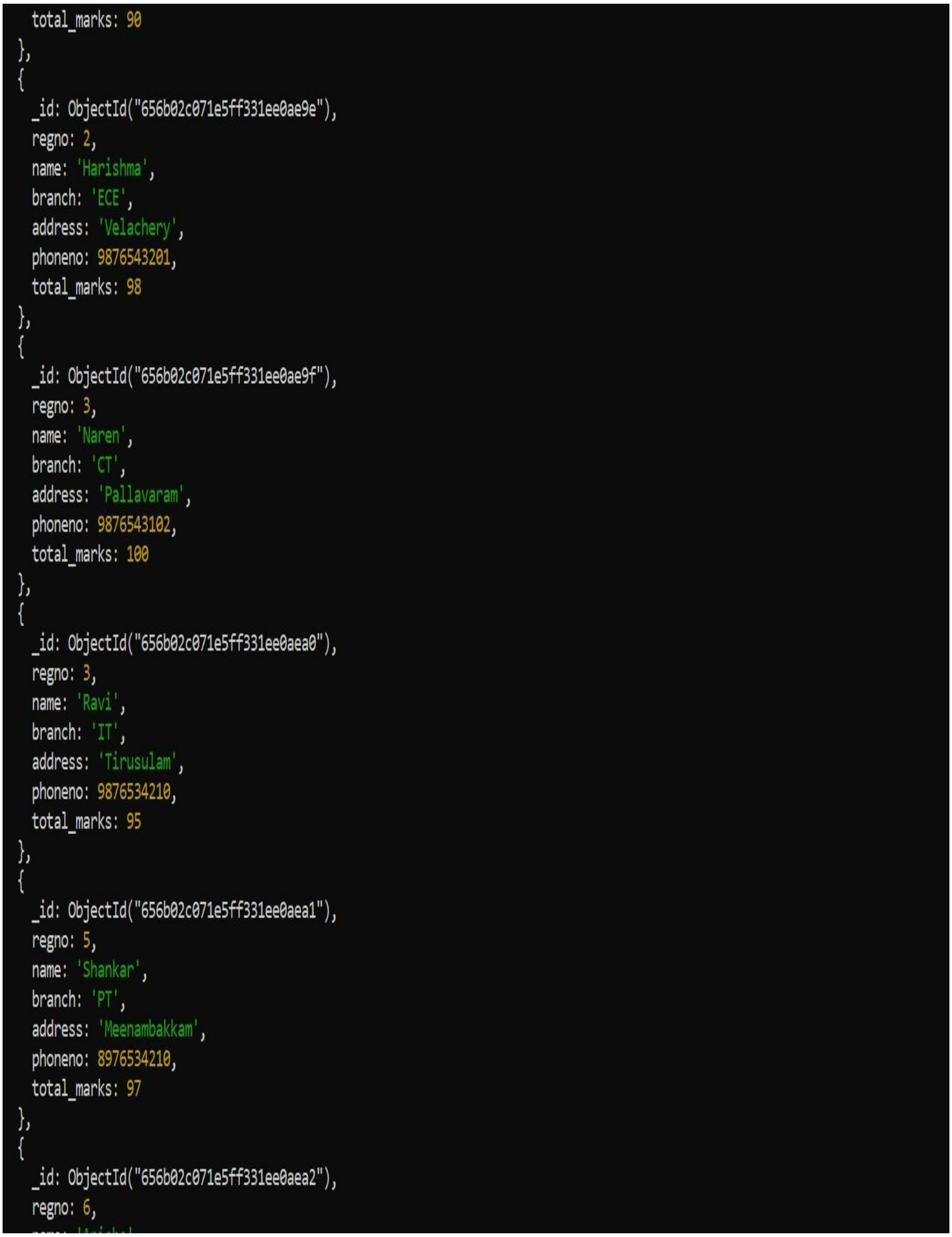
QUERY:

db.studentcollection.find()

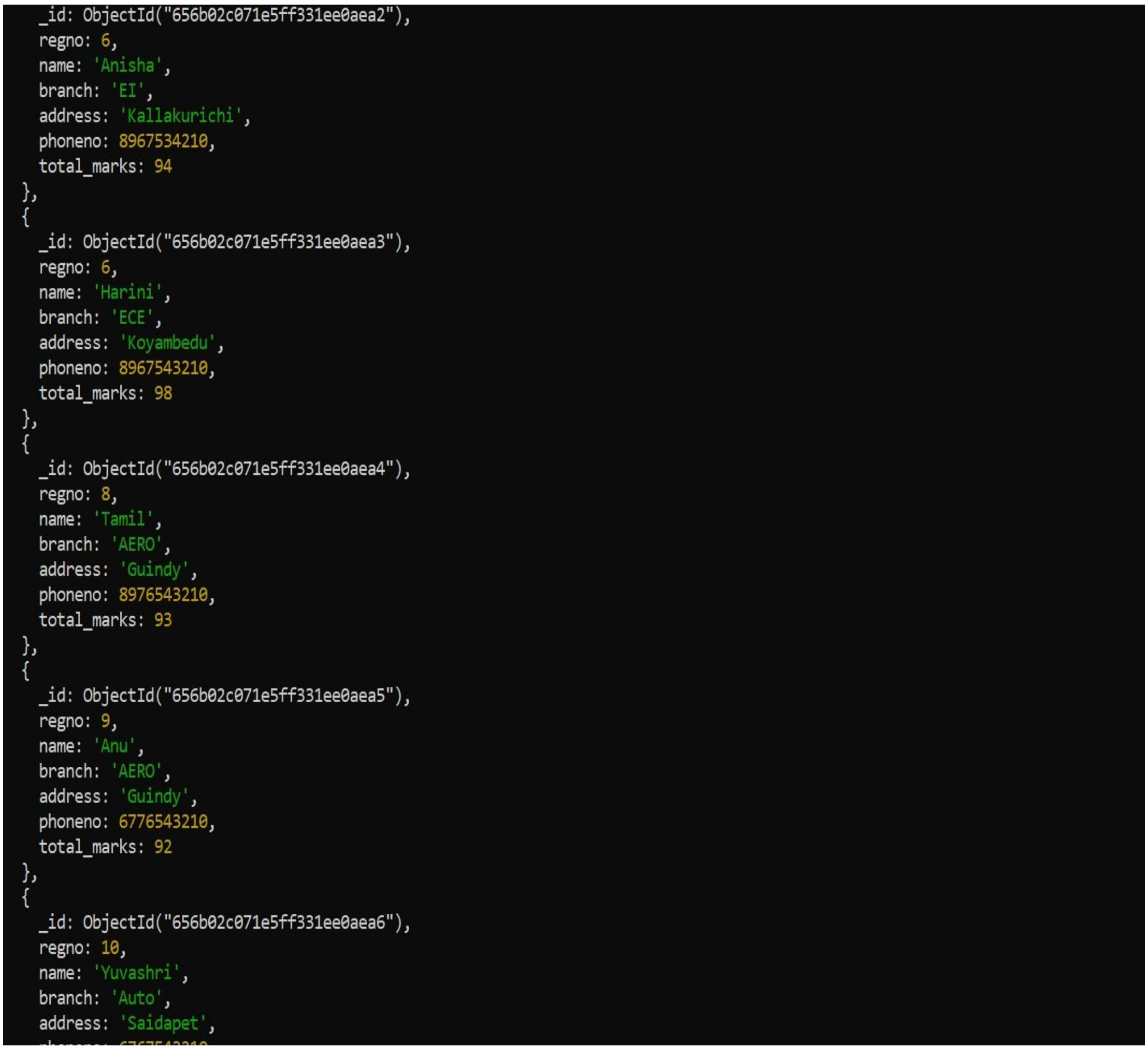
OUTPUT:



```
[
  {
    _id: ObjectId("656afde271e5ff331ee0ae9d"),
    regno: 1,
    name: 'Aadharsh',
    branch: 'IT',
    address: 'Chromepet',
    phoneno: 9876543210,
    total_marks: 90
  },
```



```
  total_marks: 90
},
{
  _id: ObjectId("656b02c071e5ff331ee0ae9e"),
  regno: 2,
  name: 'Harishma',
  branch: 'ECE',
  address: 'Velachery',
  phoneno: 9876543201,
  total_marks: 98
},
{
  _id: ObjectId("656b02c071e5ff331ee0ae9f"),
  regno: 3,
  name: 'Naren',
  branch: 'CT',
  address: 'Pallavaram',
  phoneno: 9876543102,
  total_marks: 100
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea0"),
  regno: 3,
  name: 'Ravi',
  branch: 'IT',
  address: 'Tirusulam',
  phoneno: 9876534210,
  total_marks: 95
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea1"),
  regno: 5,
  name: 'Shankar',
  branch: 'PT',
  address: 'Meenambakkam',
  phoneno: 8976534210,
  total_marks: 97
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea2"),
  regno: 6,
```



```
  _id: ObjectId("656b02c071e5ff331ee0aea2"),
  regno: 6,
  name: 'Anisha',
  branch: 'EI',
  address: 'Kallakurichi',
  phoneno: 8967534210,
  total_marks: 94
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea3"),
  regno: 6,
  name: 'Harini',
  branch: 'ECE',
  address: 'Koyambedu',
  phoneno: 8967543210,
  total_marks: 98
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea4"),
  regno: 8,
  name: 'Tamil',
  branch: 'AERO',
  address: 'Guindy',
  phoneno: 8976543210,
  total_marks: 93
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea5"),
  regno: 9,
  name: 'Anu',
  branch: 'AERO',
  address: 'Guindy',
  phoneno: 6776543210,
  total_marks: 92
},
{
  _id: ObjectId("656b02c071e5ff331ee0aea6"),
  regno: 10,
  name: 'Yuvashri',
  branch: 'Auto',
  address: 'Saidapet',
```

<u>Retrieval Using</u>

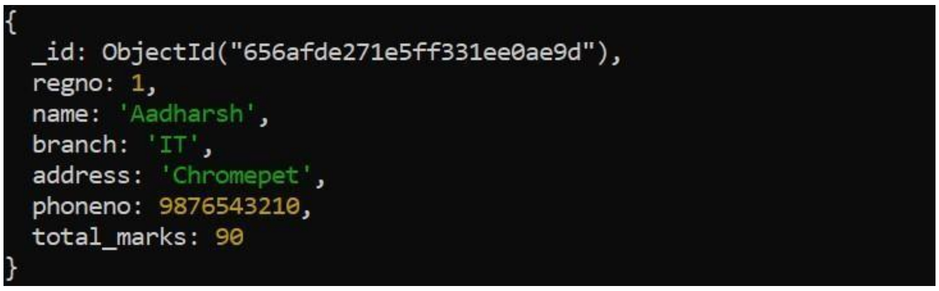
<u>findOne:</u>SYNTAX:

db.collectionName.findOne(query, projection)

QUERY:

db.studentcollection.findOne()

OUTPUT:



```
{
  _id: ObjectId("656afde271e5ff331ee0ae9d"),
  regno: 1,
  name: 'Aadharsh',
  branch: 'IT',
  address: 'Chromepet',
  phoneno: 9876543210,
  total_marks: 90
}
```

<u>UPDATION:</u>

<u>Updation Using UpdateOne:</u>

SYNTAX:

db.collectionName.updateOne(

filter,

update,

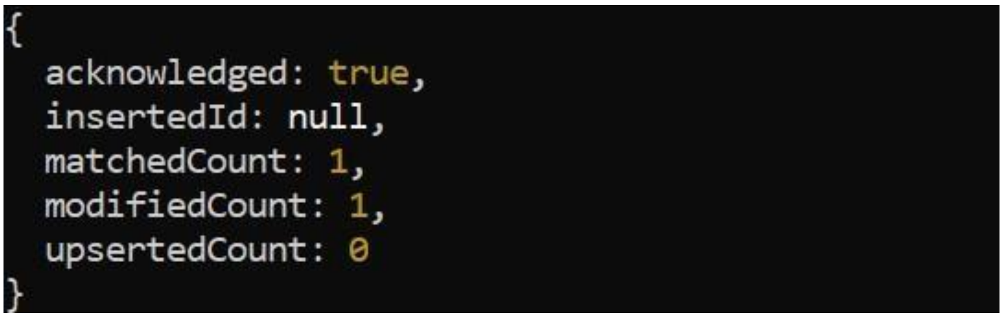
options

)

QUERY:

db.studentcollection.updateOne({regno:3,name:"Ravi"},{$set:{regno:4}})

OUTPUT:



```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

<u>Updation Using UpdateMany:</u>

SYNTAX:

db.collectionName.updateMany(

filter,

update,

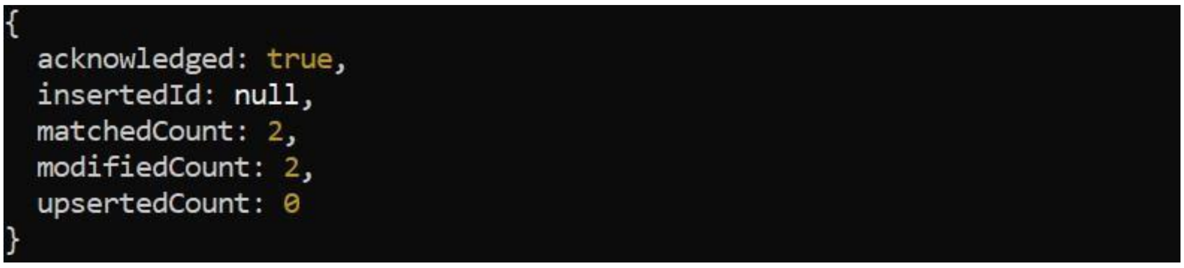
options

)

QUERY:

db.studentcollection.updateMany({branch:"IT"},{$inc:{total_marks:1}})

OUTPUT:



```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

<u>Updation With Upsert:</u>

SYNTAX:

db.collectionName.update(

<query>,

<update>,

{

upsert: true,

// Other options if needed

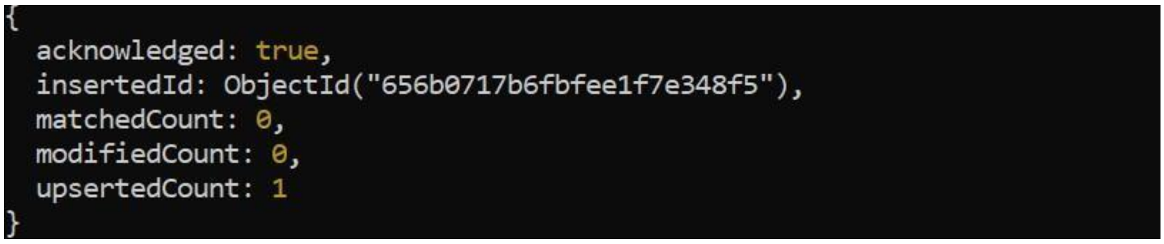
}

)

QUERY:

db.studentcollection.updateOne({regno:11},{$set:{regno:11,name:"Aashin",branch:"CT", address:"UK",phoneno:612345789,total_marks:100}},{upsert:true})

OUTPUT:



```
{
  acknowledged: true,
  insertedId: ObjectId("656b0717b6fbfee1f7e348f5"),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

<u>DELETION:</u>

<u>Deletion Using DeleteOne:</u>

db.collectionName.deleteOne(

filter,

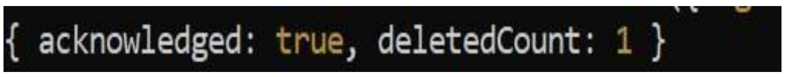
options

)

QUERY:

db.studentcollection.deleteOne({regno:11})

OUTPUT:



```
{ acknowledged: true, deletedCount: 1 }
```

Deletion Using

DeleteMany:SYNTAX:

db.collectionName.deleteMany

(filter,

options

)

QUERY:

1. **Adding 3 entries:**

db.studentcollection.insertMany([{regno:11,name:"Aashin",branch:"Mech",address:"kanya kumari",phoneno:8765432109,total_marks:100},

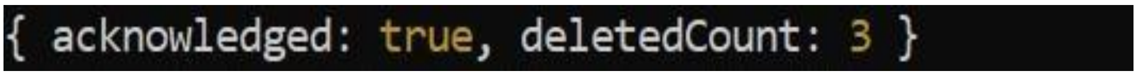
{regno:12,name:"Karthi",branch:"Mech",address:"sriperumbudur",phoneno:8675432109,total_marks:94},

{regno:13,name:"Kannan",branch:"Mech",address:"adyar",phoneno:8657432109,total_marks:92}])

2. **DELETION:**

db.studentcollection.deleteMany({branch:"Mech"})

OUTPUT:



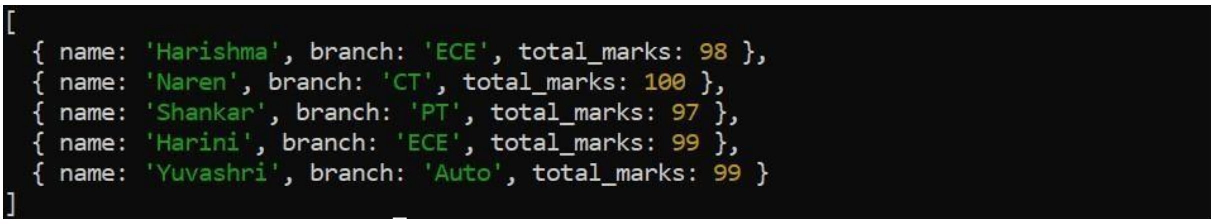
```
{ acknowledged: true, deletedCount: 3 }
```

Logical Operations In Find:

QUERY:

db.studentcollection.find({total_marks:{$gt:95}},{_id:0,name:1,branch:1,total_marks:1})

OUTPUT:

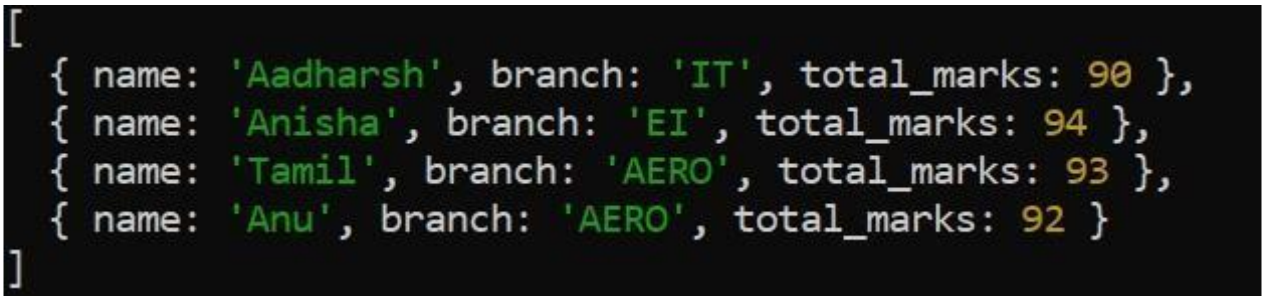


```
[
  { name: 'Harishma', branch: 'ECE', total_marks: 98 },
  { name: 'Naren', branch: 'CT', total_marks: 100 },
  { name: 'Shankar', branch: 'PT', total_marks: 97 },
  { name: 'Harini', branch: 'ECE', total_marks: 99 },
  { name: 'Yuvashri', branch: 'Auto', total_marks: 99 }
]
```

QUERY:

db.studentcollection.find({total_marks:{$lt:95}},{_id:0,name:1,branch:1,total_marks:1})

OUTPUT:

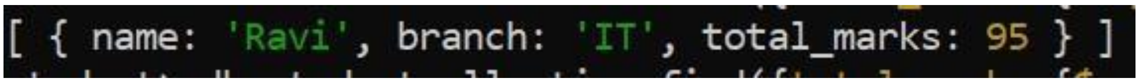


```
[
  { name: 'Aadharsh', branch: 'IT', total_marks: 90 },
  { name: 'Anisha', branch: 'EI', total_marks: 94 },
  { name: 'Tamil', branch: 'AERO', total_marks: 93 },
  { name: 'Anu', branch: 'AERO', total_marks: 92 }
]
```

QUERY:

db.studentcollection.find({total_marks:{$eq:95}},{_id:0,name:1,branch:1,total_marks:1})

OUTPUT:

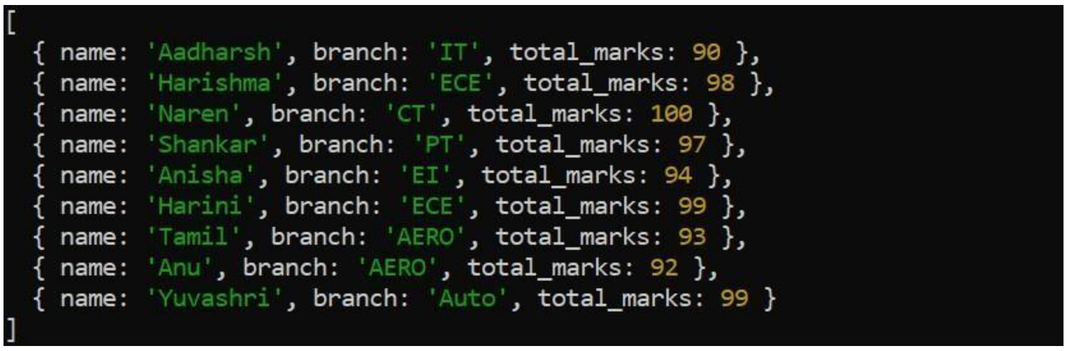


```
[ { name: 'Ravi', branch: 'IT', total_marks: 95 } ]
```

QUERY:

db.studentcollection.find({total_marks:{$ne:95}},{_id:0,name:1,branch:1,total_marks:1})

OUTPUT:



```
[
  { name: 'Aadharsh', branch: 'IT', total_marks: 90 },
  { name: 'Harishma', branch: 'ECE', total_marks: 98 },
  { name: 'Naren', branch: 'CT', total_marks: 100 },
  { name: 'Shankar', branch: 'PT', total_marks: 97 },
  { name: 'Anisha', branch: 'EI', total_marks: 94 },
  { name: 'Harini', branch: 'ECE', total_marks: 99 },
  { name: 'Tamil', branch: 'AERO', total_marks: 93 },
  { name: 'Anu', branch: 'AERO', total_marks: 92 },
  { name: 'Yuvashri', branch: 'Auto', total_marks: 99 }
]
```

RESULT:

The database has been created using NOSQL and executed many queries.

Evaluation Criteria	**Observation**	**Record**
Ability for problem definition and realization	**…../10**	**…../10**
Ability to design and analysis	**…../10**	**…../10**
Ability to implement and Validate	**…../10**	**…../10**