Ex No.: 1 **Date:**	APPLICATION DESIGN USING ER, RELATIONAL MODEL - DDL AND DML COMMANDS

COLLEGE MANAGEMENT DATABASE

AIM:

To design ER and relational model to our application and perform DDL and DML commands.

FUNCTIONAL REQUIREMENTS:

- In a college there are various departments which are stored in department entity.

Attributes are Department_ID and name

- Various courses are being offered in the college (i.e B.E CSE) and are given by course entity.

Attributes are Course_ID and name

- Students studing in the college are stored in student entity.

Attributes are Student_ID, name, gender, address, phone_number

- Faculties working in the college are stored in faculty entity.

Attributes are Faculty_ID, name, gender, address, salary, phone_number.

- Books available in the library are given in book entity.

Attributes are Book_ID, name, author, date_issue, date_return.

- Subjects handles by the faculty are given in subjects entity.

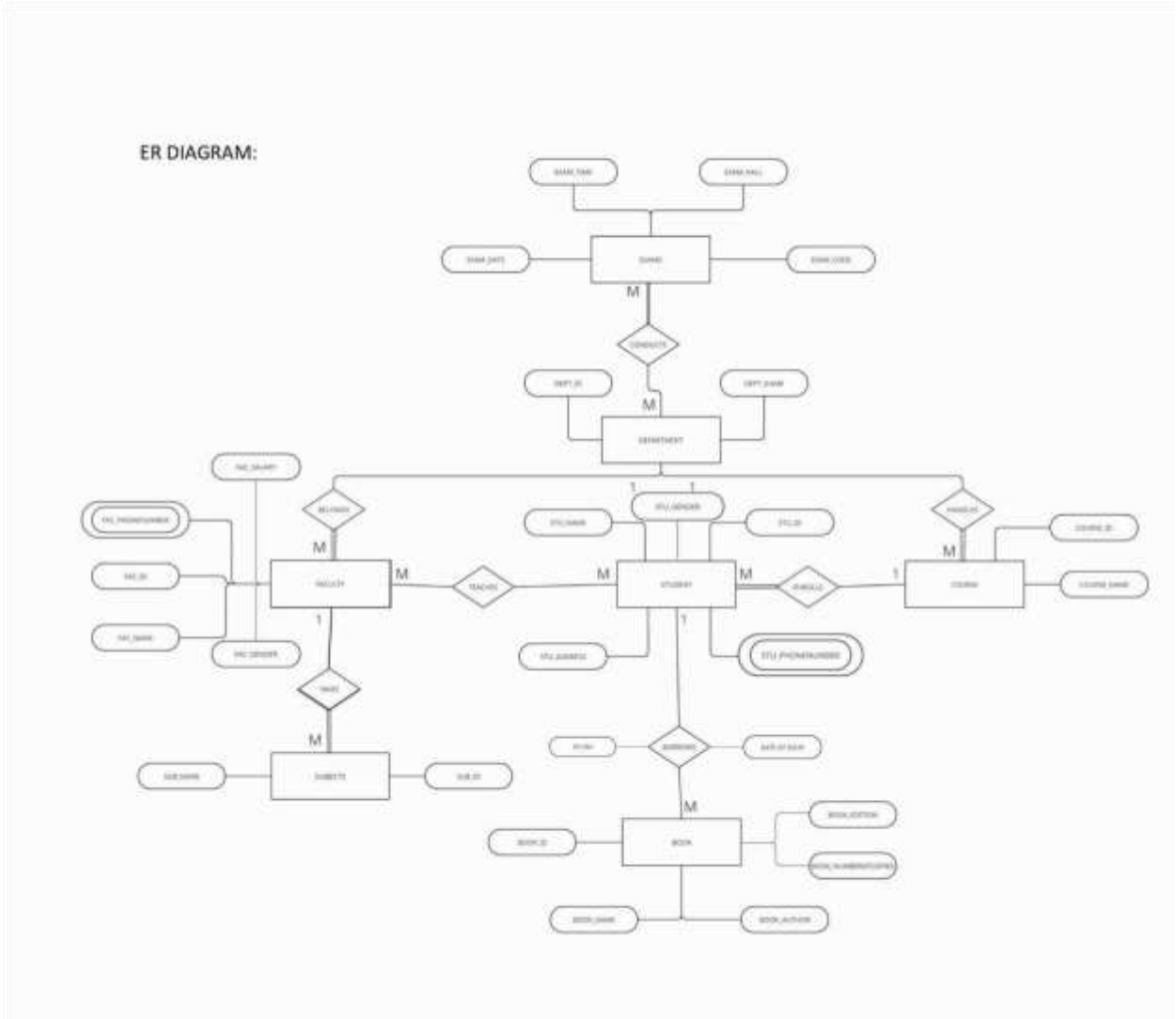
Attributes are Subject_ID and name

- Exams conducted by the department are given in exams entity. Attributes are Exam_ID, name, hall, date and time

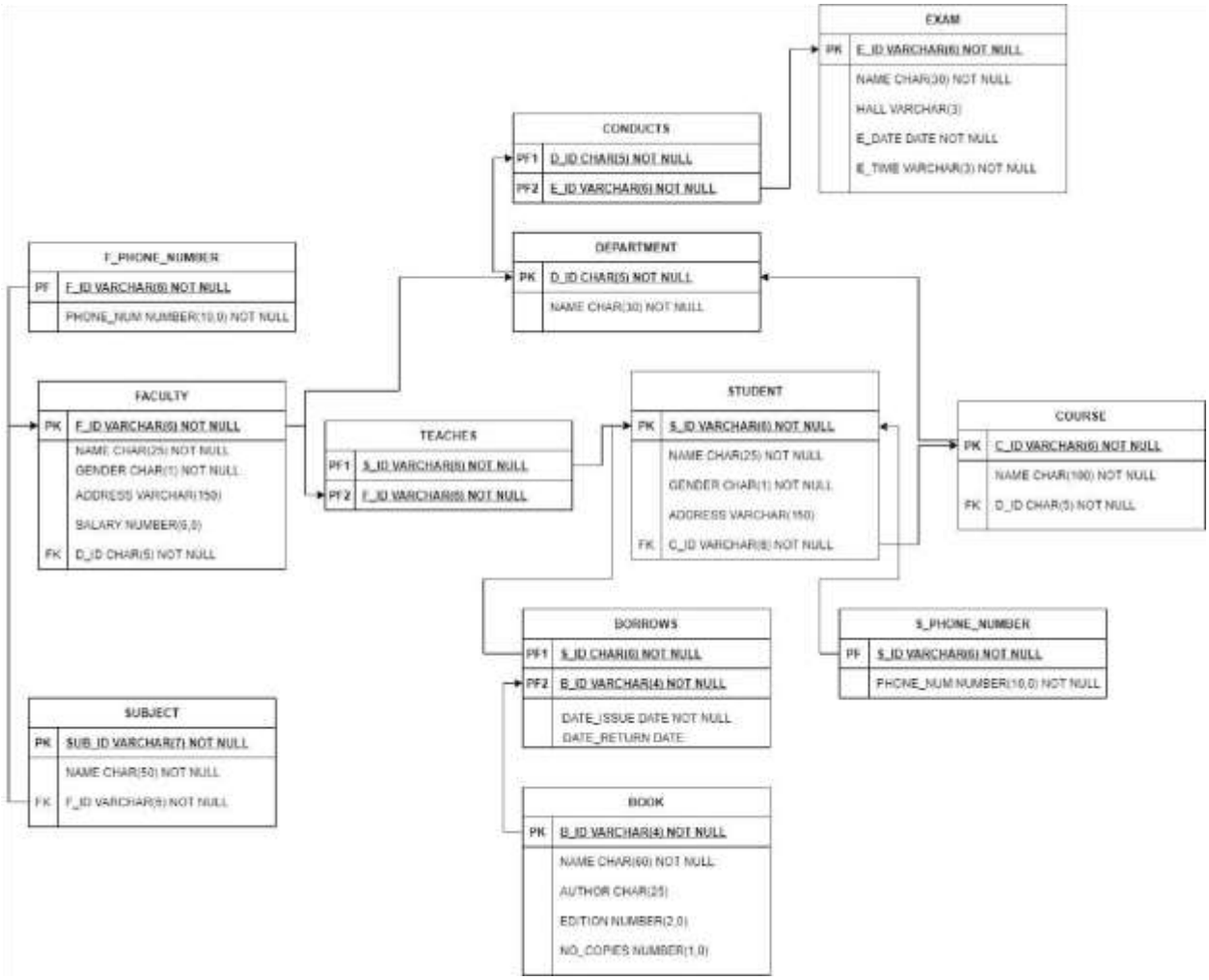
POSSIBLE QUERIES:

- We can find the number of books issued by a particular student.
- Based on E_ID we can find out which department conducts a particular exam.
- We can find the department which handles a specific course.
- We can find the amount of salary a Faculty receives.
- Based on S_ID we can sort the students IDs in ascending or descending order.
- We can find which faculty take a particular subject.

ER-DIAGRAM:



SCHEMA DIAGRAM:



RELATIONAL SCHEMA:

- STUDENT (S_ID, NAME, GENDER, ADDRESS, C_ID)
- S_PH_NUM (S_ID, PHONE_NUM) 1nf cpk
- F_PH_NUM (F_ID, PHONE_NUM)
- BOOK (B_ID, NAME, AUTHOR, EDITION, NO_COPIES)
- BORROWS (S_ID, B_ID, DATE_ISSUE, DATE_RETURN) cpk
- COURSE (C_ID,NAME,D_ID)
- FACULTY (F_ID, NAME, GENDER, ADDRESS, SALARY, D_ID)
- TEACHES (S_ID, F_ID) CPK
- SUBJECT (SUB_ID, NAME, F_ID)

- DEPARTMENT (D_ID, NAME)
- CONDUCTS (D_ID, E_ID) 2NF
- EXAM (E_ID, NAME, HALL, E_DATE, E_TIME)

Tables:

BOOK

B_ID	NAME	AUTHOR	EDITION	NO_COPIES
D001	Dynamics of Structures	Anil Kumar Chopra	3	3
C001	Java: The Complete Reference	Herbert Schildt	11	4
C002	Think Python: An Introduction to Software Design	Allen B. Downey	2	6
F001	Signals and systems	Alan V. Oppenheim	2	3

E_ID	NAME	HALL	E_DATE	E_TIME
E02	DSA	B01	12/25/2021	FN2
E04	THERMO-DYNAMICS	D01	08/25/2021	FN1
E01	DBMS	A01	09/12/2021	FN1
E03	CONSUMER ELECTRONICS	C01	09/01/2021	AF1
E05	OS	A01	09/14/2021	FN2

EXAM

CONDUCTS

D_ID	E_ID
CS	E01
CS	E02
EEE	E03
IT	E05

COURSE

C_ID	NAME	D_ID
C-CSBS	B.E. Computer Science and Business Systems	CS
C-IT	B.Tech. INFORMATION TECHNOLOGY	IT
C-CSE	B.E. Computer Science and Engineering	CS
C-ECE	B.E. Electronics and Communications Engineering	ECE
C-EEE	B.E. Electrical and Electronics Engineering	EEE

DEPARTMENT

D_ID	NAME
CS	COMPUTER SCIENCE
ECE	ELECTRONICS AND COMMUNICATION
CIVIL	CIVIL
IT	INFORMATION TECHNOLOGY
EEE	ELECTRICAL AND ELECTRONICS

S_ID	B_ID	DATE_ISSUE	DATE_RETURN
19B004	D001	01/02/2021	01/05/2021
19C001	C002	06/17/2021	06/24/2021
19D033	C002	03/02/2021	03/21/2021
19C001	C001	04/22/2021	05/22/2021

BORROWS

FACULTY

F_ID	NAME	GENDER	ADDRESS	SALARY	D_ID
FF0004	Mrs. Nirmala A	F	KK Nagar, Madurai	75000	IT
FF0005	Dr. K. Karthik	M	Woraiyur, Trichy	70000	CS
FF0001	Dr.Senthilkumar P	M	Anna Nagar, Chennai	50000	CS
FF0002	DR. Ramesh Babu	M	Srirangam, Trichy	60000	ECE
FF0003	Dr. Rajan Kumar	M	Omalur, Salem	65000	EEE

FACULTY PHONE NUMBER

F_ID	PHONE_NUM
FF0001	9847745884
FF0002	9487350987
FF0003	9547754993
FF0004	8765787986
FF0005	8966446788

STUDENT

S_ID	NAME	GENDER	ADDRESS	C_ID
19B004	Jen	F	Anna nagar, Chennai	C-CSBS
19B054	Kelly	F	Scranton, kolkata	C-CSBS
19C001	AJAY	M	ALAGAR KOVIL, MADURAI	C-CSE
19I023	Archana	F	Manmarpuram, Trichy	C-IT
19D033	Krishnan	M	K.K.Nagar, Trichy	C-ECE

STUDENT PHONE NUMBER

S_ID	PHONE_NUM
19B004	7871470008
19B054	9674566441
19C001	8679536647
19C100	9176998864
19D001	9868564543

SUBJECT

SUB_ID	NAME	F_ID
18CS310	Datastructures and Algorithms	FF0001
18EC420	Comumer Electronics	FF0002
18EE520	Eletrical Appliances	FF0003
18IT350	IOT Lab	FF0004
18IT480	Cloud computing	FF0004
18CS210	Probelm Solving Using Computers	FF0001

TEACHES

S_ID	F_ID
19B054	FF0001
19B054	FF0005
19C001	FF0001
19C001	FF0005
19D033	FF0002
19E005	FF0003
19I023	FF0004

DDL COMMANDS:

1. CREATE

```
CREATE TABLE  s_PH_NUM
   (    s_id VARCHAR2(6),
    "PHONE_NUM" NUMBER(10,0) NOT NULL,
     PRIMARY KEY (s_id,phone_num),
     foreign key(s_id) references student(s_id)
   );

```

```
drop table  s_PH_NUM;
```

2. DROP

Results Explain Describe Saved SQL History

```
Table dropped.
```

3. ALTER

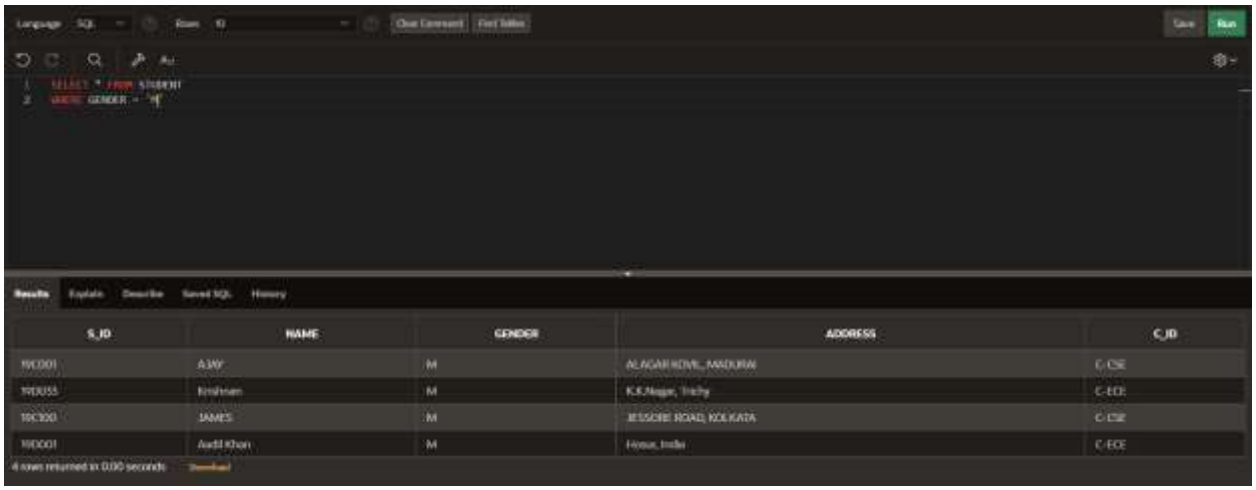
```
ALTER TABLE  "STUDENTS" ADD FOREIGN KEY ("C_ID")
        REFERENCES  "COURSE" ("C_ID")
```

Results Explain Describe Saved SQL History

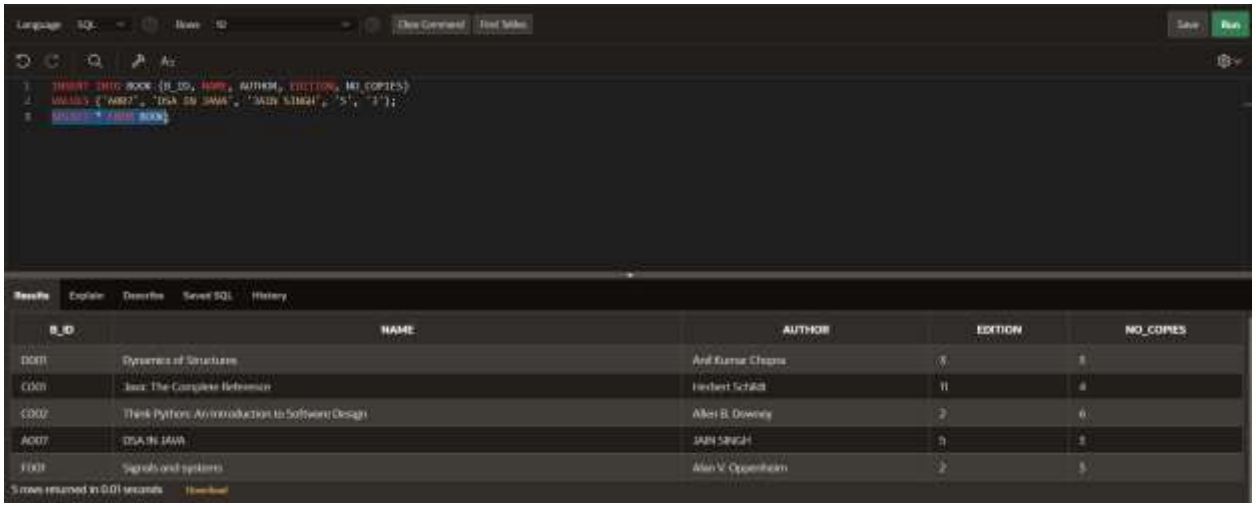
```
Table altered.

0.04 seconds
```

DML COMMANDS:

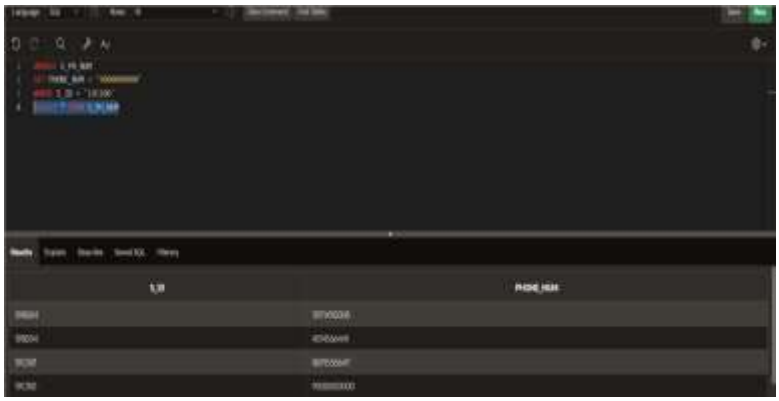
1.SELECT



2.INSERT



3.UPDATE



4.DELETE

RESULT:

Thus ER, relational model DML and DDL commands have been implemented.

Ex No.: 2 **Date:**	APPLICATION DESIGN WITH NORMALIZATION (UPTO 3NF)

AIM:

To normalize all the entities in the database.

NORMALIZATION:

I. STUDENT

S_ID	NAME	GENDER	PHONE NUMBER	ADDRESS	C_ID

The STUDENT table is not in 1 NF because column phone number contain multiple values .Hence the table is decomposed as shown below.

STUDENT

S_ID	NAME	GENDER	ADDRESS	C_ID

STUDENT PHONE NUMBER

S_ID	PHONE NUMBER

The STUDENT table is in 2 NF, there is no partial dependency and also in 1NF

The STUDENT table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF

II. FACULTY

F_ID	NAME	GENDER	PHONE NUMBER	ADDRESS	SALARY	D_ID

The FACULTY table is not in 1 NF because column phone number contain multiple values .Hence the table is decomposed as shown below.

FACULTY

F_ID	NAME	GENDER	PHONE NUMBER	ADDRESS	SALARY	D_ID

FACULTY PHONE NUMBER

S_ID	PHONE NUMBER

The FACULTY table is in 2 NF, there is no partial dependency and also in 1NF

The FACULTY table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF

III. DEPARTMENT

D_ID	NAME

The DEPARTMENT table is in 1 NF because no columns contain multiple values.

The DEPARTMENT table is in 2 NF, there is no partial dependency and also in 1NF

The DEPARTMENT table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF

IV. SUBJECT

SUB_ID	NAME	F_ID

The SUBJECT table is in 1 NF because no columns contain multiple values.

The SUBJECT table is in 2 NF, there is no partial dependency and also in 1NF

The SUBJECT table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF

V. BOOK

B_ID	NAME	AUTHOR	EDITION	NO_COPIES

The BOOK table is in 1 NF because no columns contain multiple values.

The BOOK table is in 2 NF, there is no partial dependency and also in 1NF

The BOOK table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF

VI. COURSE

C_ID	NAME	D_ID

The COURSE table is in 1 NF because no columns contain multiple values.

The COURSE table is in 2 NF, there is no partial dependency and also in 1NF

The COURSE table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF

VII. EXAM

E_ID	NAME	HALL	E_DATE	E_TIME

The EXAM table is in 1 NF because no columns contain multiple values.

The EXAM table is in 2 NF, there is no partial dependency and also in 1NF

The EXAM table is in 3 NF, there is no transitive functional dependency for non- prime attributes and also in 2NF.

RESULT:

Hence all tables have been normalized.

Ex No.: 3 **Date:**	INTEGRITY CONSTRAINTS AND DCL COMMANDS

AIM:

To check the integrity constraints of every tables and implement DCL commands.

Integrity Constraints of table:

1. BOOK

Column Name	Data Type	Nullable	Default	Primary Key
B_ID	VARCHAR2(4)	No	-	1
NAME	VARCHAR2(60)	No	-	-
AUTHOR	CHAR(25)	Yes	-	-
EDITION	NUMBER(2,0)	Yes	1	-
NO_COPIES	NUMBER(1,0)	Yes	1	-

2. BORROWS

Column Name	Data Type	Nullable	Default	Primary Key
S_ID	VARCHAR2(6)	No	-	1
B_ID	VARCHAR2(4)	No	-	2
DATE_ISSUE	DATE	Yes	-	-
DATE_RETURN	DATE	Yes	-	-

3. CONDUCTS

Column Name	Data Type	Nullable	Default	Primary Key
D_ID	CHAR(5)	No	-	1
F_ID	VARCHAR2(6)	No	-	2

4. COURSE

Column Name	Data Type	Nullable	Default	Primary Key
C_ID	VARCHAR2(6)	No	-	1
NAME	CHAR(100)	No	-	-
D_ID	CHAR(5)	No	-	-

5. DEPARTMENT

Column Name	Data Type	Nullable	Default	Primary Key
D_ID	CHAR(5)	No	-	1
NAME	CHAR(50)	No	-	-

6. CONDUCTS

Column Name	Data Type	Nullable	Default	Primary Key
E_ID	VARCHAR2(6)	No	-	1
NAME	VARCHAR2(50)	No	-	-
HALL	VARCHAR2(5)	Yes	-	-
E_DATE	DATE	Yes	-	-
E_TIME	VARCHAR2(5)	Yes	-	-

7. FACULTY

Column Name	Data Type	Nullable	Default	Primary Key
F_ID	VARCHAR2(6)	No	-	1
NAME	CHAR(25)	No	-	-
GENDER	CHAR(1)	Yes	-	-
ADDRESS	VARCHAR2(150)	Yes	-	-
SALARY	NUMBER(6,0)	No	-	-
D_ID	CHAR(5)	Yes	-	-

8. F_PH_NUM

Column Name	Data Type	Nullable	Default	Primary Key
F_ID	VARCHAR2(6)	No	-	1
PHONE_NUM	NUMBER(10,0)	No	-	2

9. STUDENT

Column Name	Data Type	Nullable	Default	Primary Key
S_ID	VARCHAR2(6)	No	-	1
NAME	CHAR(25)	No	-	-
GENDER	CHAR(1)	No	-	-
ADDRESS	VARCHAR2(150)	Yes	-	-
C_ID	VARCHAR2(6)	No	-	-

10. SUBJECT

Column Name	Data Type	Nullable	Default	Primary Key
SUB_ID	VARCHAR2(7)	No	-	1
NAME	VARCHAR2(50)	No	-	-
F_ID	VARCHAR2(6)	Yes	-	-

11. S_PH_NUM

Column Name	Data Type	Nullable	Default	Primary Key
S_ID	VARCHAR2(6)	No	-	1
PHONE_NUM	NUMBER(10,0)	No	-	2

12. TEACHES

Column Name	Data Type	Nullable	Default	Primary Key
S_ID	VARCHAR2(6)	No	-	1
F_ID	VARCHAR2(6)	No	-	2

DCL commands:

1. GRANT

```
grant SELECT,INSERT on "STUDENTS" to NK
```

Results | Explain | Describe | Saved SQL | History

Statement processed.

0.05 seconds

2. REVOKE

```
REVOKE SELECT,INSERT on "STUDENTS" FROM NK
```

Results | Explain | Describe | Saved SQL | History

Statement processed.

0.03 seconds

RESULT:

Hence all integrity constraints and DCL commands have been executed.

Ex No.: 4 **Date:** 6/8/2021	SIMPLE QUERIES

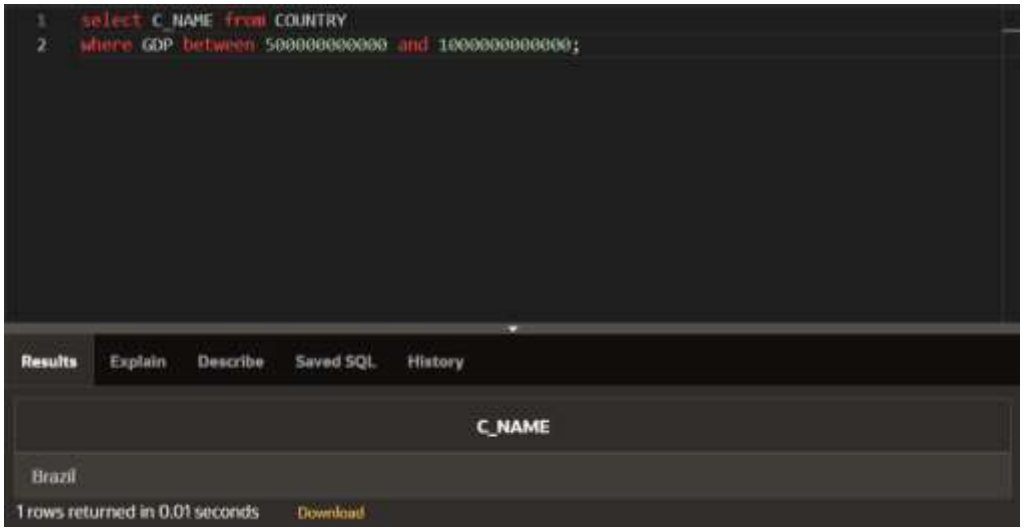
AIM:

To solve the given simple queries.

QUERIES:

1. **COUNTRY**(NAME, CONTINENT, POPULATION, GDP, LIFE_EXPECTANCY)
   RIVER(NAME, ORIGIN, LENGTH)
   CITY(NAME, COUNTRY, POPULATION)

i) Find all countries whose GDP is greater than $500 billion but less than $1 trillion.



ii) List the life expectancy in countries that have river originating in them

Find all cities that are either in South America or whose population is less than 2 million.

```
from CITY
Y in (select C_NAME from COUNTRY where CONTINENT='south america' or POPULATION<2000000);
```

Results | Explain | Describe | Saved SQL | History

CI_NAME
mumbai
Rio

2 rows returned in 0.00 seconds Download

iii) List all cities which are not in South America

```
select CI_NAME from CITY
where CI_COUNTRY not in (select C_NAME from COUNTRY where CONTINENT='south america');
```

Results | Explain | Describe | Saved SQL | History

CI_NAME
mumbai

1 rows returned in 0.01 seconds Download

2. **PERSON** (SS#, NAME, ADDRESS)
   CAR (REGISTRATION_NUMBER, YEAR, MODEL)
   ACCIDENT (DATE, DRIVER, CAR_REG_NO)
   OWNS (SS#, LICENSE)

   (i) Find the names of persons who are involved in an accident.

```
1   SELECT P.NAME FROM PERSON P, ACCIDENT A WHERE P.NAME = A.DRIVER;
```

Results | Explain | Describe | Saved SQL | History

NAME
PRADHIKSHA
RUCHITAA
VARSHA

3 rows returned in 0.04 seconds Download

(ii) Find the registration number of cars which were not involved in any accident.

```
1   SELECT REGISTRATION_NO FROM CAR WHERE REGISTRATION_NO NOT IN (SELECT CAR_REG_NO FROM ACCIDENT);
```

Results | Explain | Describe | Saved SQL | History

REGISTRATION_NO
TN5896

1 rows returned in 0.05 seconds Download

3. **EMPLOYEE**(ENAME, MANAGER_NAME)

   WORKS(ENAME, DEPARTMENT_NAME, SALARY)

   i) Find all employees working under Ragu

```
select ename from EMPLOYEE
where manager_name='ragu';
```

Results | Explain | Describe | Saved SQL | History

ENAME
nitish
ram

2 rows returned in 0.01 seconds Download

ii) Find the department name which is having highest number of Employees.

```
select DEPT_NAME from WORKS
group by DEPT_NAME
having count(*)>=all(select count(*) from WORKS group by DEPT_NAME);
```

Results | Explain | Describe | Saved SQL | History

DEPT_NAME
sales

1 rows returned in 0.01 seconds Download

iii) Find the employee who is getting lowest salary.

```
select ename from WORKS
where SALARY<=all(select SALARY from WORKS);
```

Results | Explain | Describe | Saved SQL | History

ENAME
ram

1 rows returned in 0.04 seconds Download

iv) Find all employees who is getting higher than average salary of sales department

```
select ename from WORKS
where SALARY>(select avg(SALARY) from WORKS where dept_name='sales');
```

Results | Explain | Describe | Saved SQL | History

ENAME
mani
nitish

2 rows returned in 0.01 seconds Download

RESULT:

Thus the given simple queries have been executed successfully.

Ex No.: 5 **Date:** 6/8/2021	COMPLEX QUERIES

AIM:

To solve the given complex queries.

QUERIES:

1. **STOP** (STOPID, NAME)

   TRAIN(TRAINNO,NAME)

   TRAINROUTE(TRAINNO, STOPID,RANK)

i. How many stops are there on Vaigai Express?

```
SELECT MAX(R.RANK) FROM TRAIN T,TRAINROUTE R WHERE T.TRAINNO=R.TRAINNO AND T.NAME='VAIGAI';
```

Results | Explain | Describe | Saved SQL | History

MAX(R.RANK)
2

1 rows returned in 0.00 seconds Download

ii. List the stops on Nellai express in alphabetical order.

```
SELECT S.NAME FROM STOP S,TRAIN T,TRAINROUTE R WHERE S.STOPID=R.STOPID AND T.TRAINNO=R.TRAINNO AND T.NAME='NELLAI'
ORDER BY S.NAME;
```

Results | Explain | Describe | Saved SQL | History

NAME
CHENNAI
POLLACHI

2 rows returned in 0.05 seconds Download

iii. List the second stop on Pearl city express.

```
SELECT S.NAME FROM STOP S,TRAIN T,TRAINROUTE R WHERE S.STOPID=R.STOPID AND T.TRAINNO=R.TRAINNO AND R.RANK=2
AND T.NAME='PEARLCITY';
```

Results | Explain | Describe | Saved SQL | History

NAME
POLLACHI

1 rows returned in 0.04 seconds Download

iv. List the last stop on Pandian express.

```
SELECT S.NAME FROM STOP S, TRAIN T, TRAINROUTE R WHERE S.STOPID = R.STOPID AND T.TRAINNO = R.TRAINNO AND
T.NAME = 'PANDIYAN' AND R.RANK = (SELECT MAX(RANK) FROM TRAIN T1, TRAINROUTE R1 WHERE T1.TRAINNO = R1.TRAINNO
AND T1.NAME = 'PANDIYAN');
```

Results | Explain | Describe | Saved SQL | History

NAME
MADURAI

1 rows returned in 0.05 seconds Download

v. List the train numbers which connect Madurai and Trichy.

```
SELECT T1.TRAINNO FROM TRAINROUTE T1,TRAINROUTE T2,STOP S1,STOP S2
WHERE T1.TRAINNO=T2.TRAINNO AND T1.STOPID=S1.STOPID AND T2.STOPID=S2.STOPID AND S1.NAME='MADURAI' AND S2.NAME='TRICHY';
```

Results | Explain | Describe | Saved SQL | History

TRAINNO
2211

1 rows returned in 0.03 seconds Download

vi. List all the stops that can be reached from Madurai without transferring to different trains.

```
SELECT S2.STOPID,S2.NAME FROM TRAINROUTE T1,TRAINROUTE T2,STOP S1,STOP S2
WHERE T1.TRAINNO=T2.TRAINNO AND T1.STOPID=S1.STOPID AND T2.STOPID=S2.STOPID AND S1.NAME='MADURAI' AND
T1.STOPID<>T2.STOPID AND T1.RANK<T2.RANK;
```

Results | Explain | Describe | Saved SQL | History

STOPID	NAME
3	TRICHY

1 rows returned in 0.03 seconds Download

2. EMPLOYEE(ENAME, ECITY)

COMPANY(CNAME, CCITY)

MANAGES(ENAME,MNAME)

WORKS(ENAME, CNAME, SALARY)

i. Find all employees working in the same city as do their managers.

```sql
Select Manages.ename,Manages.mname from Manages,Employee A,Employee B
Where mname=A.ename and Manages.ename=B.ename and A.ecity=B.ecity;
```

Results | Explain | Describe | Saved SQL | History

ENAME	MNAME
emp5	emp6

ii. Find all employees living in the same city where the company is located.

```sql
Select Works.ename from Works, Company, Employee
Where Works.cname=Company.cname and Employee.ename=Works.ename and ecity=ccity;
```

Results | Explain | Describe | Saved SQL | History

ENAME
emp1
emp7

iii. Find all employees whose salary is higher than the average salary of their company.

```
Select A.ename from Works A
where salary>(select avg(salary) from Works B where A.cname=B.cname)
```

Results | Explain | Describe | Saved SQL | History

ENAME
emp6
emp2
emp4

iv. Find all employees whose salary is higher than average salary of all companies.

```
Select ename from Works
where salary>(select avg(salary) from Works group by cname order by avg(salary) fetch first 1 rows only);
```

Results | Explain | Describe | Saved SQL | History

ENAME
emp6
emp2
emp4
emp7
emp8

5 rows returned in 0.02 seconds Download

v. Find the company name which is having highest average salary.

```
Select cname from Works A
    Group by cname
    Having avg(salary)>=all(select avg(salary) from Works B Group by cname);
```

Results | Explain | Describe | Saved SQL | History

CNAME
com3

1 rows returned in 0.01 seconds Download

vi. Find the total salary of each company.

```
Select cname,sum(salary) from Works
Group by cname;
```

Results | Explain | Describe | Saved SQL | History

CNAME	SUM(SALARY)
com1	70000
com3	150000
com2	150000

3 rows returned in 0.01 seconds Download

vii. Find all employees who earn more than each employee of SBI.

```
Select ename from Works
Where salary>(Select max(salary) from works where cname='com1');
```

Results | Explain | Describe | Saved SQL | History

ENAME
emp4
emp8

2 rows returned in 0.01 seconds Download

viii. Find the company that has the most employees.

```
Select cname from Works
Group by cname
Having count(*)>=all(select count(*) from Works group by cname);
```

Results | Explain | Describe | Saved SQL | History

CNAME
com2

1 rows returned in 0.01 seconds Download

ix. Assume that the companies may be located in several cities. Find all companies located in every city in which SBI is located.

```
select unique A.cname from company A
where not exists((select ccity from company where cname='SBI') minus (select ccity from company B where A.cname=B.cname and B.cname<>'SBI'));
```

Results | Explain | Describe | Saved SQL | History

CNAME
Canara Bank

1 rows returned in 0.02 seconds Download

x. Find the names and cities of residence of all employees who work for Canara Bank and earn more than 50,000.

```
/ from Employee, Works, Company
/ee.ename and Works.cname=Company.cname) and (Works.cname='com2' and salary>50000));
```

Results | Explain | Describe | Saved SQL | History

ENAME	ECITY
emp6	cityc
emp4	citya

2 rows returned in 0.03 seconds Download

RESULT:

Thus the given complex queries have been executed successfully.

Ex No.: 6 **Date:**	DATABASE OBJECTS

AIM:

To create database objects.

1. View
    a. Simple View
    b. Complex View
2. Sequence
3. Index
4. Synonym

VIEW:

SYNTAX:

CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW view_name

[(alias[, alias]...)] AS subquery

[WITH CHECK OPTION [CONSTRAINT constraint]]

[WITH READ ONLY [CONSTRAINT constraint]];

a. Simple View

```
CREATE VIEW ST_ID AS
SELECT NAME,S_ID
FROM STUDENT
```

Results | Explain | Describe | Saved SQL | Hist

View created.

```
SELECT * FROM ST_ID
```

Results | Explain | Describe | Saved SQL | History

NAME	S_ID
Jan	19B004
Kelly	19B054
AJAY	19CD01
Archana	19K023
Krishnan	19D055

b. Complex View

```
CREATE VIEW ST_CONTACT(STUDENT_ID,STUDENT_NAME, CONTACT) AS
SELECT S.S_ID, S.NAME, C.PHONE_NUM FROM STUDENT S, S_PH_NUM C WHERE S.S_ID = C.S_ID;
```

Results | Explain | Describe | Saved SQL | History

```
View created.
```

```
SELECT * FROM ST_CONTACT;
```

STUDENT_ID	STUDENT_NAME	CONTACT
[illegible]	Jim	[illegible]
[illegible]	Kelly	[illegible]
[illegible]	AJAY	[illegible]
[illegible]	Archana	[illegible]

SEQUENCE:

SYNTAX:

CREATE SEQUENCE sequence_name

[INCREMENT BY n]

[START WITH n]

[MAXVALUE n | NOMAXVALUE]

[MINVALUE n | NOMINVALUE]

[CYCLE | NOCYCLE]

[CACHE n | NOCACHE];

```
CREATE SEQUENCE STUDENTS_SEQ START WITH 100 MAXVALUE 300
```

Results | Explain | Describe | Saved SQL | History

```
Sequence created.
```

```
INSERT INTO STUDENTS VALUES(STUDENTS_SEQ.NEXTVAL, 'AADHI');
```

Results | Explain | Describe | Saved SQL | History

```
1 row(s) inserted.
```

```
SELECT * FROM STUDENTS
```

Results | Explain | Describe | Saved SQL | History

S_ID	NAME
101	AADHI
100	HARISH

INDEX:

SYNTAX:

CREATE INDEX index_name ON table (column [, column]...);

```
CREATE INDEX ST_INDEX ON STUDENT(NAME);
```

Results | Explain | Describe | Saved SQL | History

Index created.

```
SELECT NAME FROM STUDENT
```

Results | Explain | Describe | Saved SQL | History

NAME
AJAY
Aadil Khan
Archana
JAMES

SYNONYM:

SYNTAX:

CREATE [PUBLIC] SYNONYM synonym_name FOR object;

```
CREATE SYNONYM STUD FOR STUDENT
```

Results | Explain | Describe | Saved SQL | History

```
Synonym created.
```

```
SELECT * FROM STUD
```

Results | Explain | Describe | Saved SQL | History

S_ID	NAME	GENDER	ADDRESS	C_ID
19B004	Jen	F	Anna nagar, Chennai	C-CSBS
19B054	Kelly	F	Scranton, kolkata	C-CSBS
19C001	AJAY	M	ALAGAR KOVIL, MADURAI	C-CSE
19I023	Archana	F	Mannarpuram, Trichy	C-IT

RESULT:

Thus database objects have been created successful

Ex No.: 7 **Date:**	PL/SQL(FUNCTIONS & PROCEDURES)

FUNCTIONS:

Language SQL Rows 10 Clear Command Find Tables

```
create or replace function total_records
return number is
    total number(2):=0;
begin
    select count(*) into total
    from students;

    return total;
end;
/
```

Results Explain Describe Saved SQL History

```
Function created.
```

0.00 seconds

Language SQL Rows 10 Clear Command Find Tables

```
DECLARE
    c number(2);
BEGIN
    c := total_records();
    dbms_output.put_line('Total no. of student records: ' || c);
END;
/
```

Results Explain Describe Saved SQL History

```
Total no. of student records: 2

Statement processed.
```

0.05 seconds

PROCEDURE :

```
DECLARE
    c_fid faculty.F_ID%type;
    c_sid student.S_ID%type;
    c_fname faculty.name%type;
    c_sname student.name%type;
    message varchar(100);
    CURSOR c_teaches IS
    select F_ID, S_ID from teaches;

FUNCTION f_faculty(x in faculty.F_ID%type)
RETURN char is
    fname faculty.name%type;
BEGIN
    select name into fname
    from faculty where x=faculty.F_ID;

    RETURN fname;
```

```
    END;

    FUNCTION f_student(x in student.S_ID%type) RETURN
    char is
        sname student.name%type;
    BEGIN
        select name into sname      from student
    where x=student.S_ID;

        RETURN sname;
    END;

    PROCEDURE summary IS
    BEGIN
        OPEN c_teaches;
        LOOP
        FETCH c_teaches into c_fid, c_sid;        EXIT
    WHEN                                c_teaches%notfound;
    c_fname:=f_faculty(c_fid);
    c_sname:=f_student(c_sid);          message:=(c_fname
    || ' teaches ' || c_sname);
            dbms_output.put_line(message);
        END LOOP;
        CLOSE c_teaches;
    END;
    BEGIN
    summary;
    END;
    /
```

Language SQL Rows 10 Clear Command Find Tables

```sql
DECLARE
    c_fid faculty.F_ID%type;
    c_sid student.S_ID%type;
    c_fname faculty.name%type;
    c_sname student.name%type;
    message varchar(100);
    CURSOR c_teaches IS
    select F_ID, S_ID from teaches;

FUNCTION f_faculty(x in faculty.F_ID%type)
RETURN char is
```

Results Explain Describe Saved SQL History

```
Dr.Senthilkumar P         teaches Kelly
Dr. K. Karthik            teaches Kelly
Dr.Senthilkumar P         teaches AJAY
Dr. K. Karthik            teaches AJAY
DR. Ramesh Babu           teaches Krishnan
Dr. Rajan Kumar           teaches ROSE
Mrs. Nirmala A            teaches Archana

Statement processed.

0.10 seconds
```

RESULT:

Functions & Procedures from PL/SQL have been successfully implemented into our application.

Ex No.: 8 **Date:**	PL/SQL TRIGGERS

AIM:

To implement PL/SQL Triggers.

SYNTAX:

CREATE [OR REPLACE] TRIGGER trigger_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col_name]

ON table_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)
DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;

TRIGGERS:

1. To create an audit table to record the changes made in the table

```
CREATE OR REPLACE TRIGGER AUDIT_STUDENT
    AFTER DELETE OR INSERT OR UPDATE ON STUDENT
    FOR EACH ROW
    BEGIN
        INSERT INTO AUDIT_STUDENT(USER_NAME,TIME_STAMP,OLDID,NEWID,OLDNAME,NEWNAME,OLDG,NEWG,OLDADD,NEWADD,OLDCOURSE,NEWCOURSE)
        VALUES(USER,SYSDATE,:OLD.S_ID,:NEW.S_ID,:OLD.NAME,:NEW.NAME,:OLD.GENDER,:NEW.GENDER,:OLD.ADDRESS,:NEW.ADDRESS,:OLD.C_ID,:NEW.C_ID);
    END;
```

2. To check the salary constraint of the faculty

```
CREATE TRIGGER SALARYCHECK BEFORE INSERT OR UPDATE ON FACULTY
FOR EACH ROW
WHEN(NEW.SALARY>90000)
BEGIN
RAISE_APPLICATION_ERROR(-20001,'SALARY CANNOT EXCEED 90,000');
DBMS_OUTPUT.PUT_LINE('FACULTY SALARY EXCEEDED LIMIT');
END;
```

Results | Explain | Describe | Saved SQL | History

```
Trigger created.
```

```
INSERT INTO FACULTY VALUES('FF0006', 'Dr. Chokkalingam', 'M', 'Egmore, Chennai', 95000, 'EEE');
```

Results | Explain | Describe | Saved SQL | History

```
ORA-20001: SALARY CANNOT EXCEED 90,000
ORA-06512: at "WKSP_COLLEGEMANAGEMENT.SALARYCHECK", line 2
ORA-04088: error during execution of trigger 'WKSP_COLLEGEMANAGEMENT.SALARYCHECK'
ORA-06512: at "SYS.DBMS_SQL", line 1721


1. INSERT INTO FACULTY VALUES('FF0006', 'Dr. Chokkalingam', 'M', 'Egmore, Chennai',
95000, 'EEE')
```

RESULT:

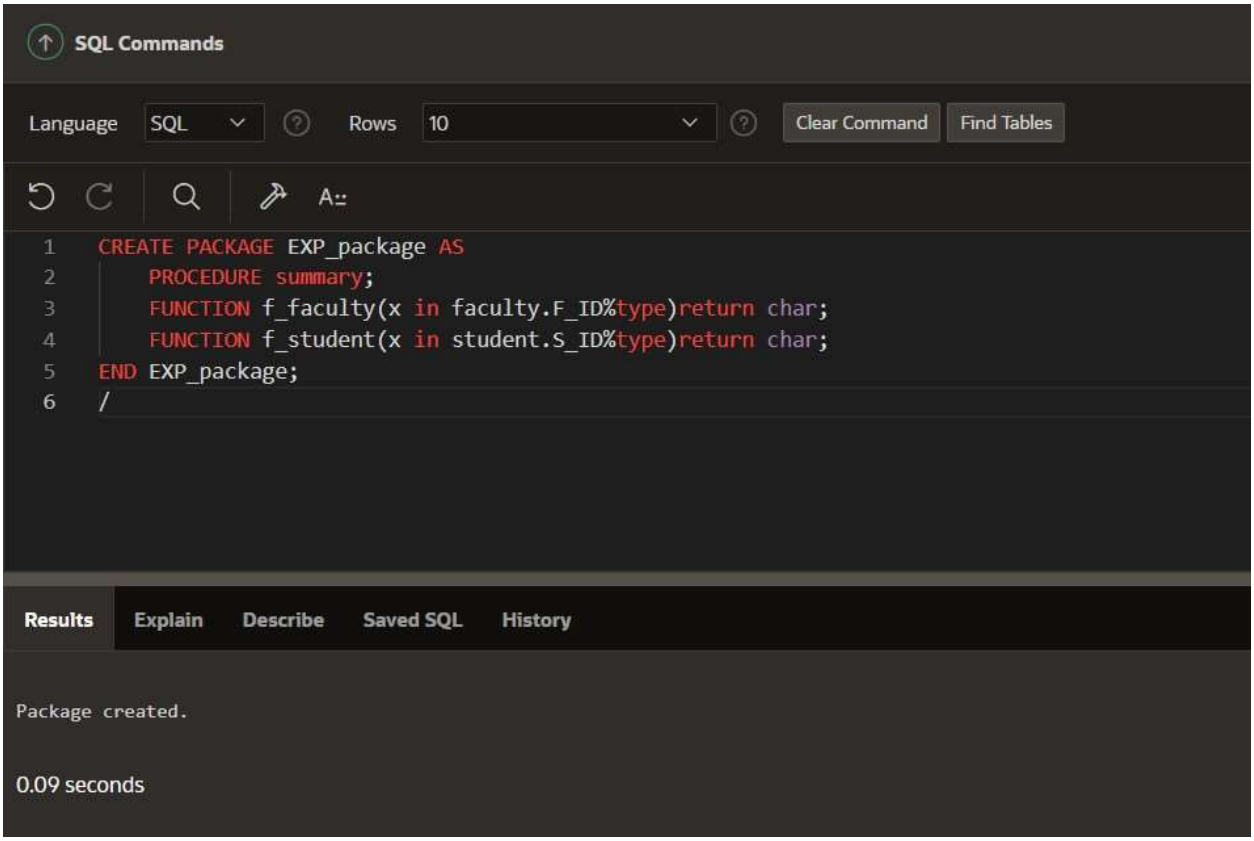
Thus PL/SQL triggers has been successfully implemented.

Ex No.:9 Date:	PL/SQL PACKAGES

Aim:

to implement pl/sql packages

1. Creating package specification



2. Creating package body

```
CREATE PACKAGE body EXP_package AS
FUNCTION f_faculty(x in faculty.F_ID%type) RETURN
char is
    fname faculty.name%type; BEGIN
select name into fname     from
faculty where x=faculty.F_ID;

    RETURN fname;
END;

FUNCTION f_student(x in student.S_ID%type) RETURN
char is
    sname student.name%type; BEGIN
select name into sname     from
student where x=student.S_ID;

    RETURN sname;
END;

PROCEDURE summary IS     CURSOR
c_teaches IS     select F_ID, S_ID
from teaches;     c_fid
faculty.F_ID%type;     c_sid
student.S_ID%type;     c_fname
faculty.name%type;     c_sname
student.name%type;     message
varchar(100);
BEGIN
    OPEN c_teaches;
    LOOP
    FETCH c_teaches into c_fid, c_sid;
```

SQL Commands

Language SQL Rows 10 Clear Command Find Tables

```sql
CREATE PACKAGE body EXP_package AS

FUNCTION f_faculty(x in faculty.F_ID%type)
RETURN char is
    fname faculty.name%type;
BEGIN
    select name into fname
    from faculty where x=faculty.F_ID;

    RETURN fname;
END;

FUNCTION f_student(x in student.S_ID%type)
RETURN char is
```

Results Explain Describe Saved SQL History

```
Package Body created.
```

0.15 seconds

Language SQL Rows 10 Clear Command Find Tables

```sql
BEGIN
    EXP_package.summary;
END;
/
```

Results Explain Describe Saved SQL History

```
Dr.Senthilkumar P        teaches Kelly
Dr. K. Karthik           teaches Kelly
Dr.Senthilkumar P        teaches AJAY
Dr. K. Karthik           teaches AJAY
DR. Ramesh Babu          teaches Krishnan
Dr. Rajan Kumar          teaches ROSE
Mrs. Nirmala A           teaches Archana

Statement processed.
```

0.07 seconds

RESULT:

Packages from PL/SQL have been successfully implemented into our application.