STUDENT MANAGEMENT SYSTEM

DONE BY

SREE POOJA K (RA1811030010054)

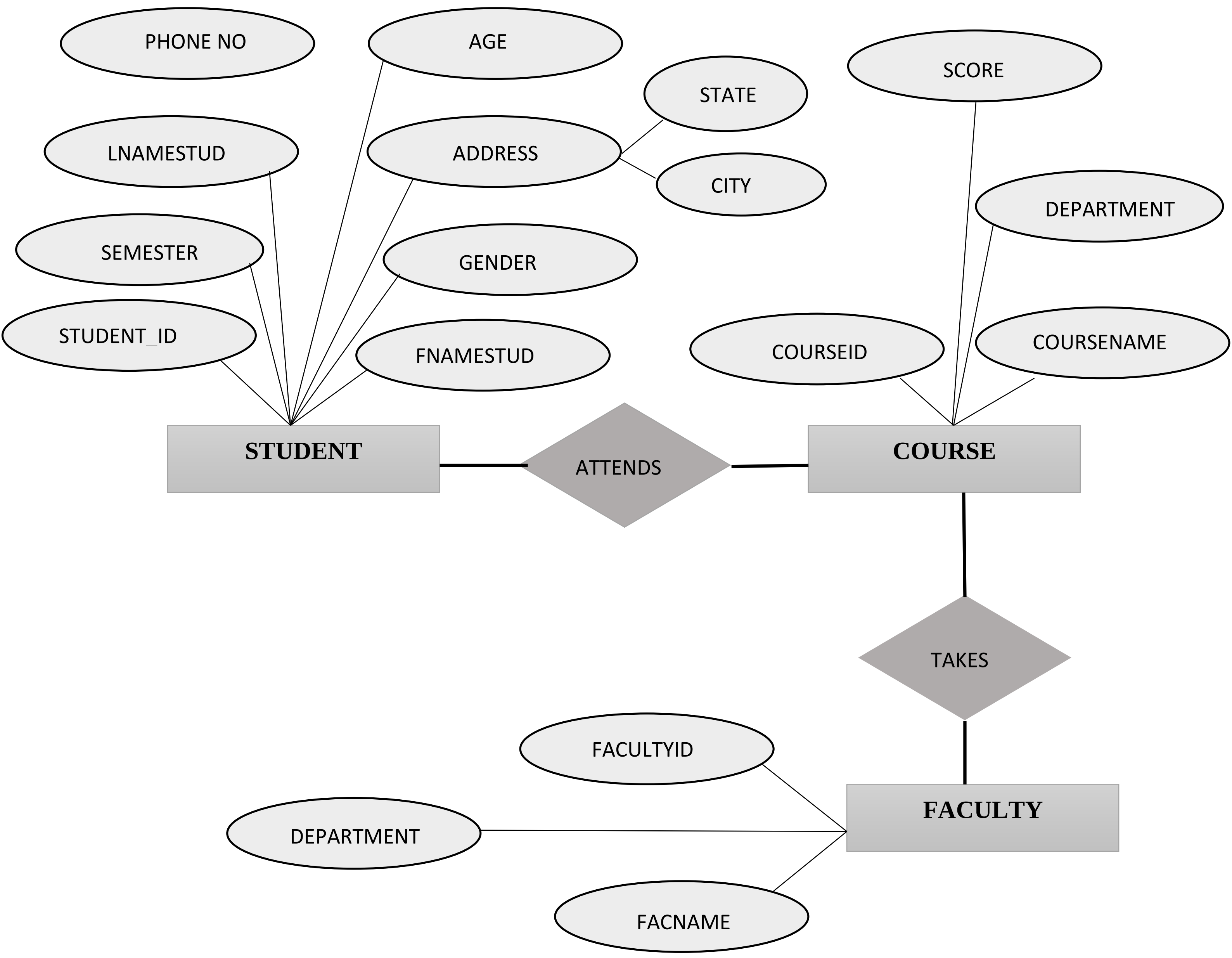
SWETHA CHEPURI (RA1811030010059)

RAJATH KIRAN (RA1811030010044)

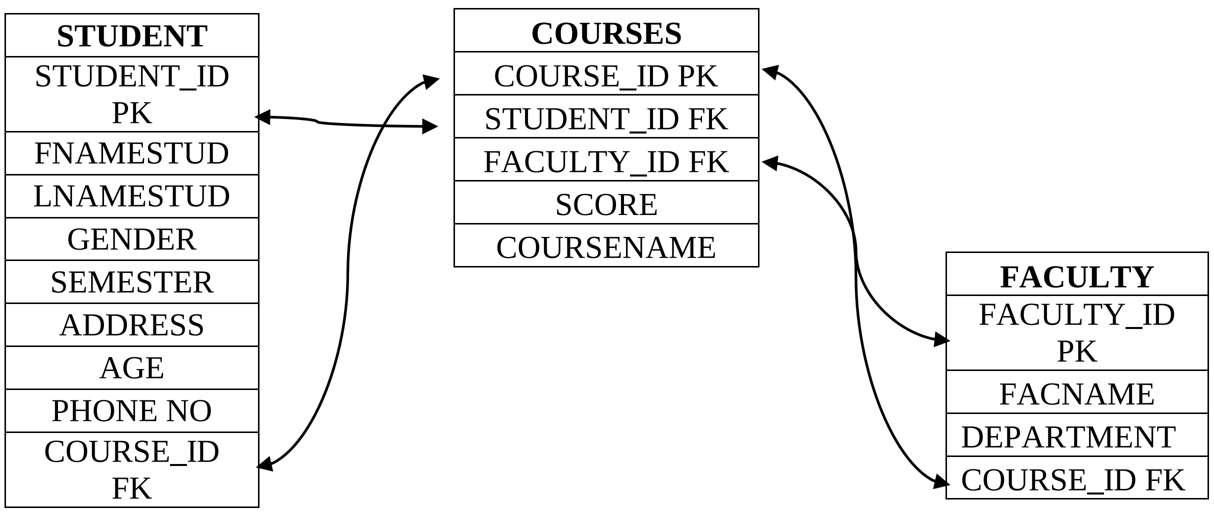
SUSHANTH RAVIPALLI (RA1811030010043)

EXP NO	EXPERIMENT	DONE BY
1	E-R DIAGRAM	RA1811030010044
2	RELATIONAL SCHEMA	RA1811030010044
3	DDLWITH CONSTRAINT	RA1811030010054
4	DML COMMANDS AND SQL FUNCTIONS	RA1811030010054
5	SQL OPERATIONS	RA1811030010044
6	AGGREGATION	RA1811030010043
7	JOINS	RA1811030010059
8	SUBQUERIES	RA1811030010059
9	PLSQL	RA1811030010043
10	CURSORS AND TRIGGERS	RA1811030010043

STUDENT MANAGEMENT SYSTEM E-R DIAGRAM



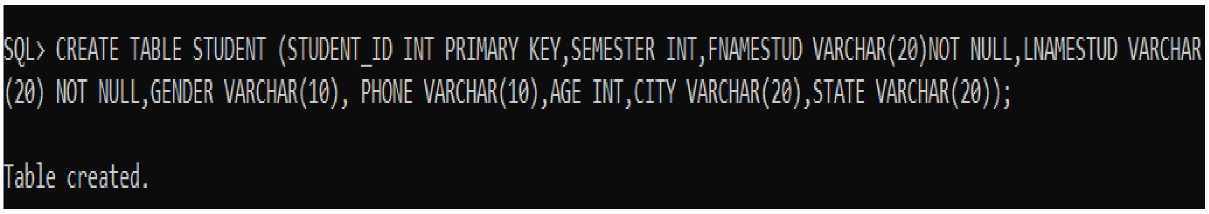
STUDENT MANAGEMENT SYSTEM RELATIONAL SCHEMA



DDL COMMANDS WITH CONSTRAINTS

CREATING STUDENT TABLE

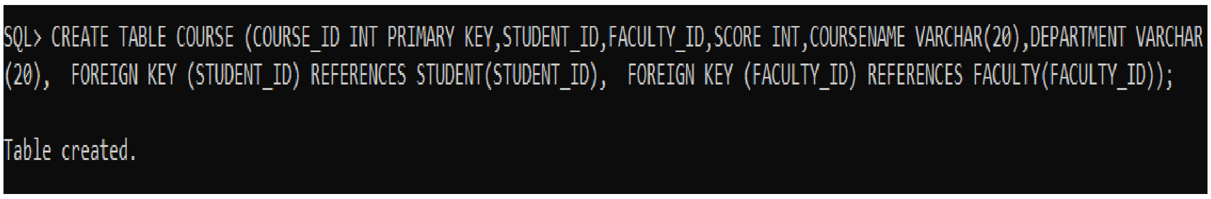
CREATE TABLE STUDENT (STUDENT_ID INT PRIMARY KEY,SEMESTER INT,FNAMESTUD VARCHAR(20)NOT NULL,LNAMESTUD VARCHAR(20) NOT NULL,GENDER VARCHAR(10), PHONE VARCHAR(10),AGE INT,CITY VARCHAR(20),STATE VARCHAR(20));



```
SQL> CREATE TABLE STUDENT (STUDENT_ID INT PRIMARY KEY,SEMESTER INT,FNAMESTUD VARCHAR(20)NOT NULL,LNAMESTUD VARCHAR
(20) NOT NULL,GENDER VARCHAR(10), PHONE VARCHAR(10),AGE INT,CITY VARCHAR(20),STATE VARCHAR(20));

Table created.
```

CREATING COURSE TABLE

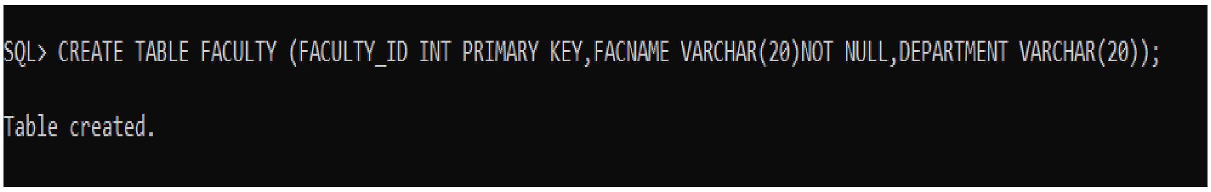
CREATE TABLE COURSE (COURSE_ID INT PRIMARY KEY,STUDENT_ID,FACULTY_ID,SCORE INT,COURSENAME VARCHAR(20),DEPARTMENT VARCHAR(20), FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT(STUDENT_ID), FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY(FACULTY_ID));



```
SQL> CREATE TABLE COURSE (COURSE_ID INT PRIMARY KEY,STUDENT_ID,FACULTY_ID,SCORE INT,COURSENAME VARCHAR(20),DEPARTMENT VARCHAR
(20),  FOREIGN KEY (STUDENT_ID) REFERENCES STUDENT(STUDENT_ID),  FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY(FACULTY_ID));

Table created.
```

CREATING FACULTY TABLE

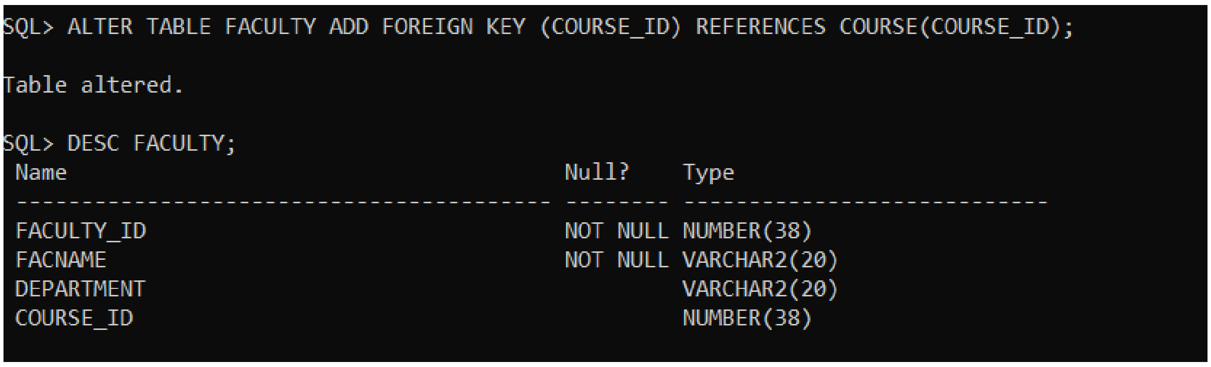
CREATE TABLE FACULTY (FACULTY_ID INT PRIMARY KEY,FACNAME VARCHAR(20)NOT NULL,DEPARTMENT VARCHAR(20));



```
SQL> CREATE TABLE FACULTY (FACULTY_ID INT PRIMARY KEY,FACNAME VARCHAR(20)NOT NULL,DEPARTMENT VARCHAR(20));

Table created.
```

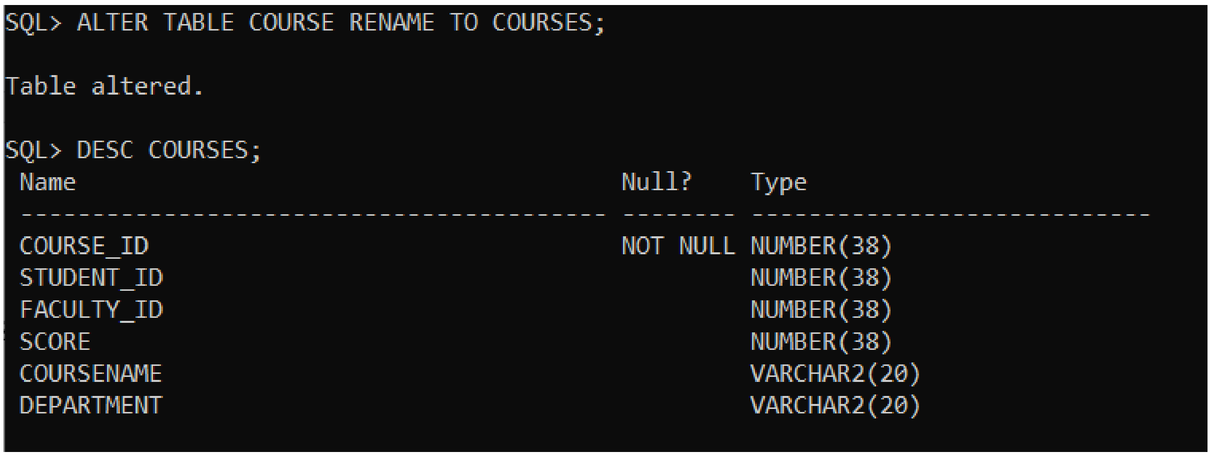
ALTERING FACULTY TABLE TO ADD FOREIGN KEY

ALTER TABLE FACULTY ADD FOREIGN KEY (COURSE_ID) REFERENCES COURSE(COURSE_ID);


```
SQL> ALTER TABLE FACULTY ADD FOREIGN KEY (COURSE_ID) REFERENCES COURSE(COURSE_ID);

Table altered.

SQL> DESC FACULTY;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FACULTY_ID                                NOT NULL NUMBER(38)
 FACNAME                                   NOT NULL VARCHAR2(20)
 DEPARTMENT                                         VARCHAR2(20)
 COURSE_ID                                          NUMBER(38)
```

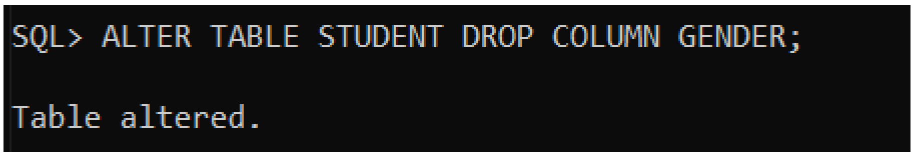
RENAMING COURSE TO COURSES

ALTER TABLE COURSE RENAME TO COURSES;


```
SQL> ALTER TABLE COURSE RENAME TO COURSES;

Table altered.

SQL> DESC COURSES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_ID                                 NOT NULL NUMBER(38)
 STUDENT_ID                                         NUMBER(38)
 FACULTY_ID                                         NUMBER(38)
 SCORE                                              NUMBER(38)
 COURSENAME                                         VARCHAR2(20)
 DEPARTMENT                                         VARCHAR2(20)
```

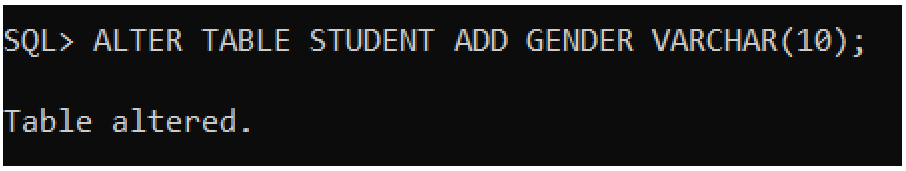
DROP GENDER FROM STUDENT

ALTER TABLE STUDENT DROP COLUMN GENDER;

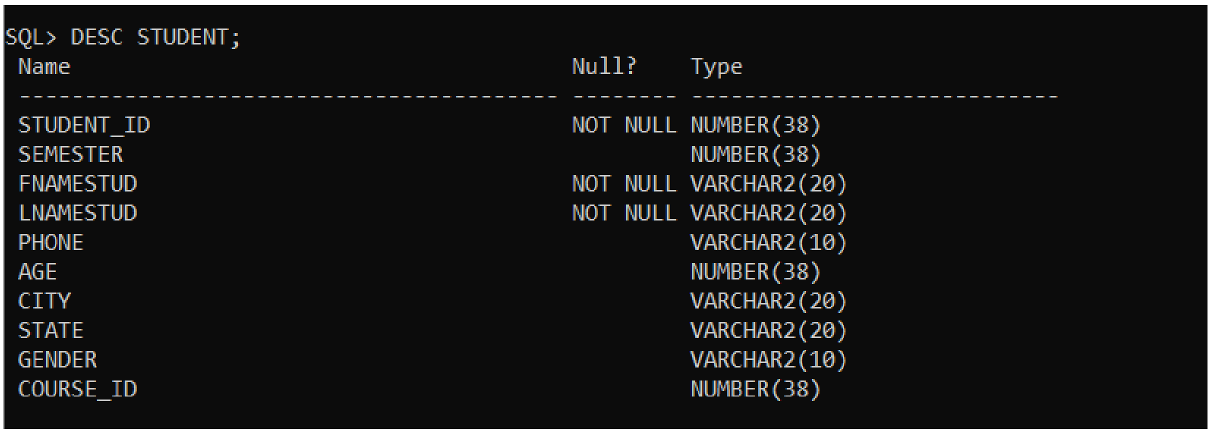

```
SQL> ALTER TABLE STUDENT DROP COLUMN GENDER;

Table altered.
```

TO ADD GENDER TO STUDENT

ALTER TABLE STUDENT ADD GENDER VARCHAR(10);

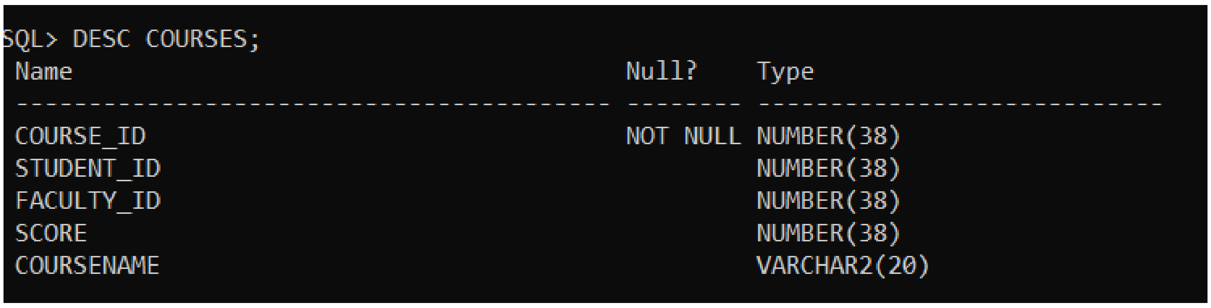

```
SQL> ALTER TABLE STUDENT ADD GENDER VARCHAR(10);

Table altered.
```

OBJECTS OF STUDENT TABLE



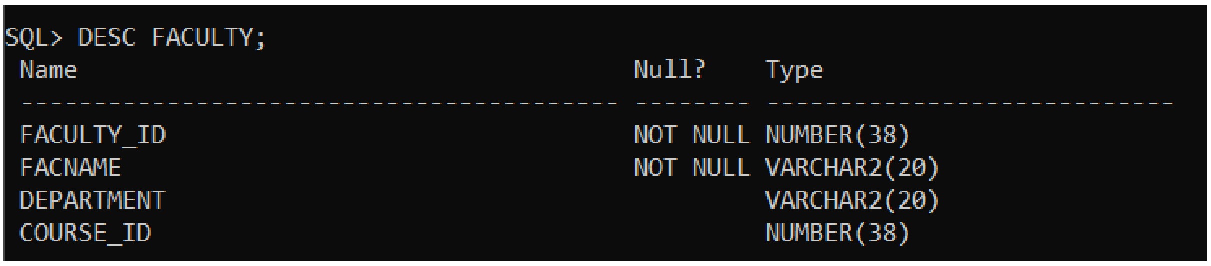
```
SQL> DESC STUDENT;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                NOT NULL NUMBER(38)
 SEMESTER                                           NUMBER(38)
 FNAMESTUD                                 NOT NULL VARCHAR2(20)
 LNAMESTUD                                 NOT NULL VARCHAR2(20)
 PHONE                                              VARCHAR2(10)
 AGE                                                NUMBER(38)
 CITY                                               VARCHAR2(20)
 STATE                                              VARCHAR2(20)
 GENDER                                             VARCHAR2(10)
 COURSE_ID                                          NUMBER(38)
```

OBJECTS OF COURSES TABLE



```
SQL> DESC COURSES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_ID                                 NOT NULL NUMBER(38)
 STUDENT_ID                                         NUMBER(38)
 FACULTY_ID                                         NUMBER(38)
 SCORE                                              NUMBER(38)
 COURSENAME                                         VARCHAR2(20)
```

OBJECTS OF COURSES TABLE

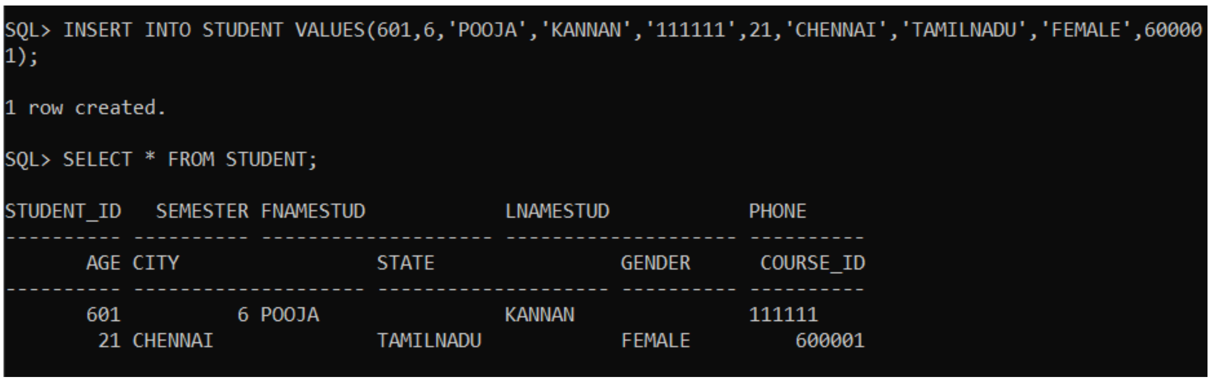


```
SQL> DESC FACULTY;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FACULTY_ID                                NOT NULL NUMBER(38)
 FACNAME                                   NOT NULL VARCHAR2(20)
 DEPARTMENT                                         VARCHAR2(20)
 COURSE_ID                                          NUMBER(38)
```

DML COMMANDS AND SQL FUNCTIONS

INSERT A NEW ROW TO STUDENT

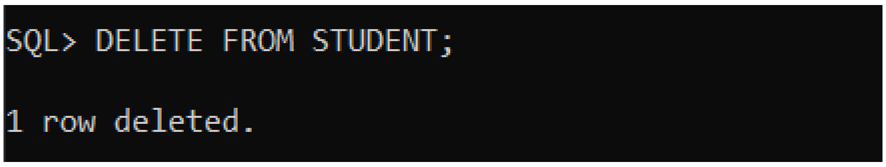
INSERT INTO STUDENT VALUES(601,6,'POOJA','KANNAN','111111',21,'CHENNAI','TAMILNADU','FEMALE',600001);



```
SQL> INSERT INTO STUDENT VALUES(601,6,'POOJA','KANNAN','111111',21,'CHENNAI','TAMILNADU','FEMALE',60000
1);

1 row created.

SQL> SELECT * FROM STUDENT;

STUDENT_ID   SEMESTER FNAMESTUD            LNAMESTUD            PHONE
---------- ---------- -------------------- -------------------- ----------
       AGE CITY                STATE                GENDER      COURSE_ID
---------- ------------------- ------------------- ---------- ----------
       601          6 POOJA                KANNAN               111111
        21 CHENNAI             TAMILNADU            FEMALE         600001
```

DELETE DATA FROM STUDENT

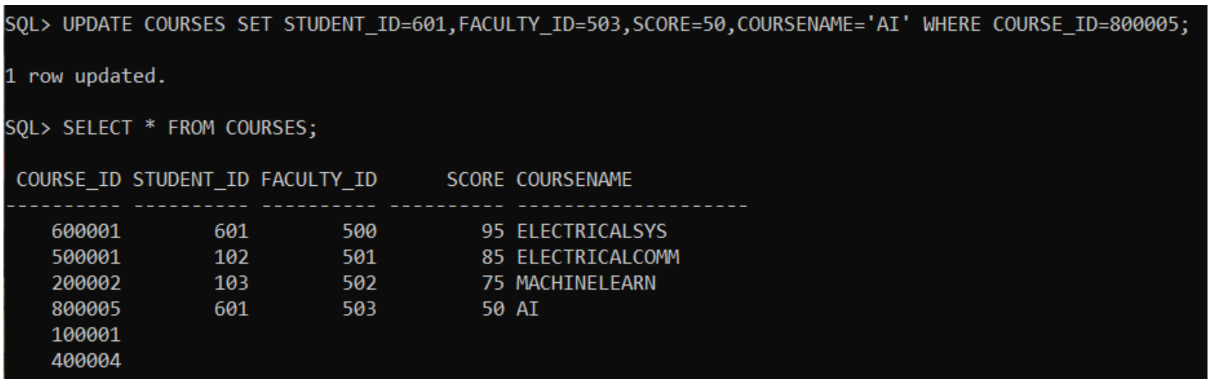
DELETE FROM STUDENT;



```
SQL> DELETE FROM STUDENT;

1 row deleted.
```

UPDATE VALUES TO COURSES

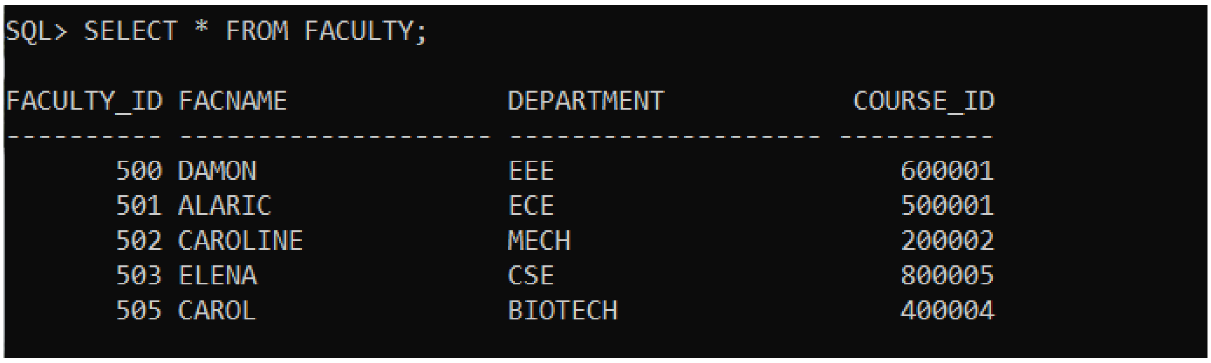
UPDATE COURSES SET STUDENT_ID=601,FACULTY_ID=503,SCORE=50,COURSENAME='AI' WHERE COURSE_ID=800005;



```
SQL> UPDATE COURSES SET STUDENT_ID=601,FACULTY_ID=503,SCORE=50,COURSENAME='AI' WHERE COURSE_ID=800005;

1 row updated.

SQL> SELECT * FROM COURSES;

 COURSE_ID STUDENT_ID FACULTY_ID      SCORE COURSENAME
---------- ---------- ---------- ---------- --------------------
    600001        601        500         95 ELECTRICALSYS
    500001        102        501         85 ELECTRICALCOMM
    200002        103        502         75 MACHINELEARN
    800005        601        503         50 AI
    100001
    400004
```

SELECT ALL DETAILS OF FACULTY;

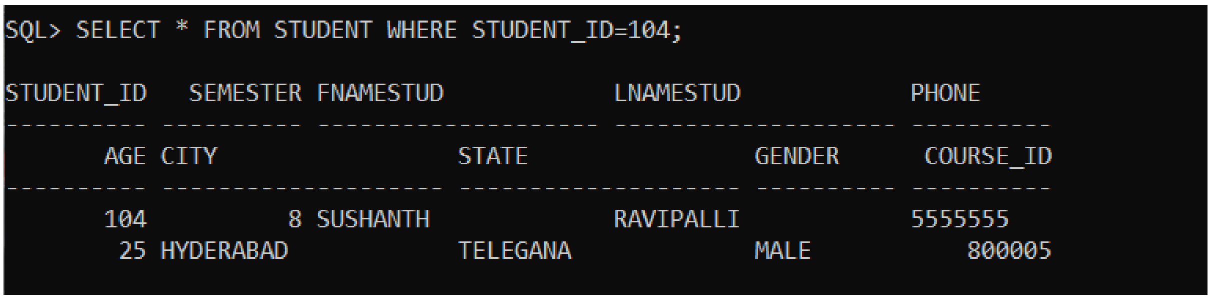
SELECT * FROM FACULTY;



```
SQL> SELECT * FROM FACULTY;

FACULTY_ID FACNAME              DEPARTMENT           COURSE_ID
---------- -------------------- -------------------- ----------
       500 DAMON                EEE                      600001
       501 ALARIC               ECE                      500001
       502 CAROLINE             MECH                     200002
       503 ELENA                CSE                      800005
       505 CAROL                BIOTECH                  400004
```

FIND DETAILS OF STUDENT WHOSE ID IS 104

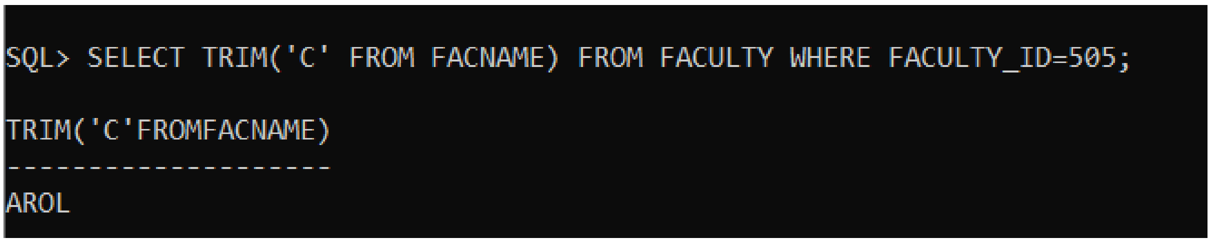
SELECT * FROM STUDENT WHERE STUDENT_ID=104;



```
SQL> SELECT * FROM STUDENT WHERE STUDENT_ID=104;

STUDENT_ID   SEMESTER FNAMESTUD           LNAMESTUD            PHONE
---------- ---------- ------------------- -------------------- ----------
       AGE CITY                 STATE                GENDER      COURSE_ID
---------- -------------------- -------------------- ---------- ----------
       104          8 SUSHANTH            RAVIPALLI            5555555
        25 HYDERABAD            TELEGANA             MALE           800005
```

TRIM C FROM FACULTY NAME WHOSE ID IS 505

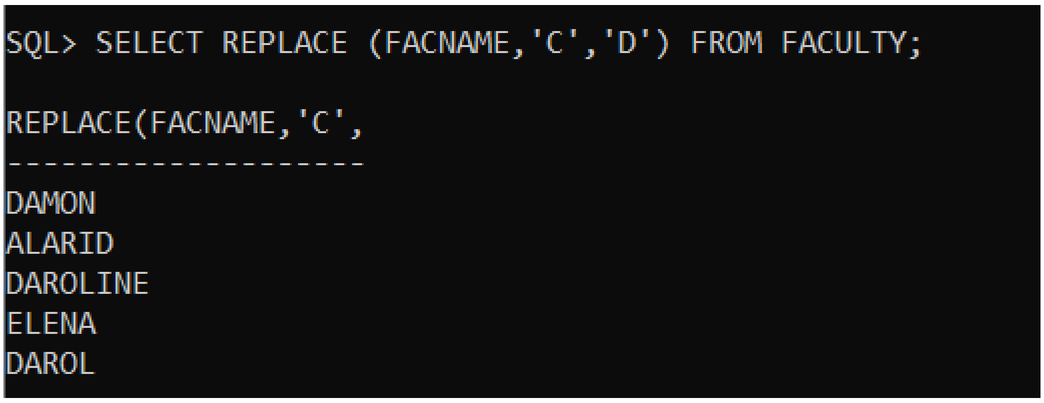
SELECT TRIM('C' FROM FACNAME) FROM FACULTY WHERE FACULTY_ID=505;



```
SQL> SELECT TRIM('C' FROM FACNAME) FROM FACULTY WHERE FACULTY_ID=505;

TRIM('C'FROMFACNAME)
--------------------
AROL
```

REPLACE C WITH D FROM FACULTY NAME

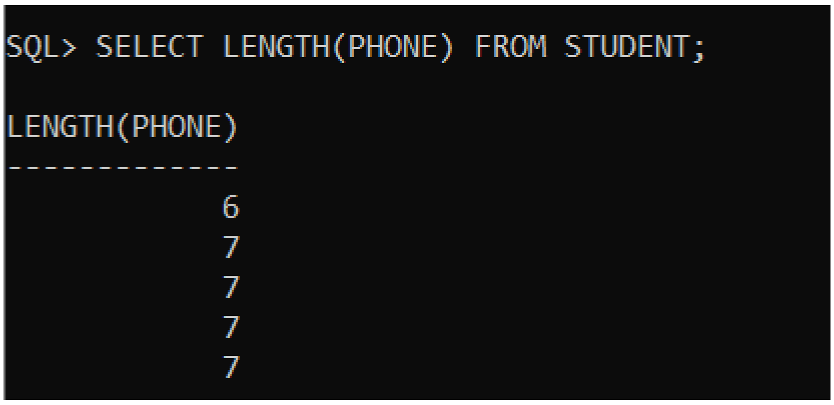
SQL> SELECT REPLACE (FACNAME,'C','D') FROM FACULTY;



```
SQL> SELECT REPLACE (FACNAME,'C','D') FROM FACULTY;

REPLACE(FACNAME,'C',
--------------------
DAMON
ALARID
DAROLINE
ELENA
DAROL
```

FIND THE LEGTH OF PHONE NO FROM STUDENT

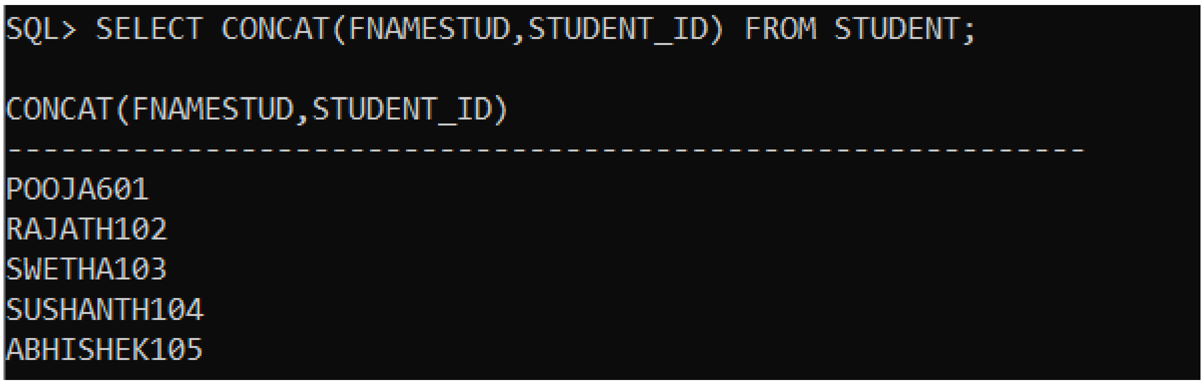
SELECT LENGTH(PHONE) FROM STUDENT;



```
SQL> SELECT LENGTH(PHONE) FROM STUDENT;

LENGTH(PHONE)
-------------
            6
            7
            7
            7
            7
```

CONCAT FIRSTNAME AND STUDENT ID OF STUDENT FROM STUDENT TABLE

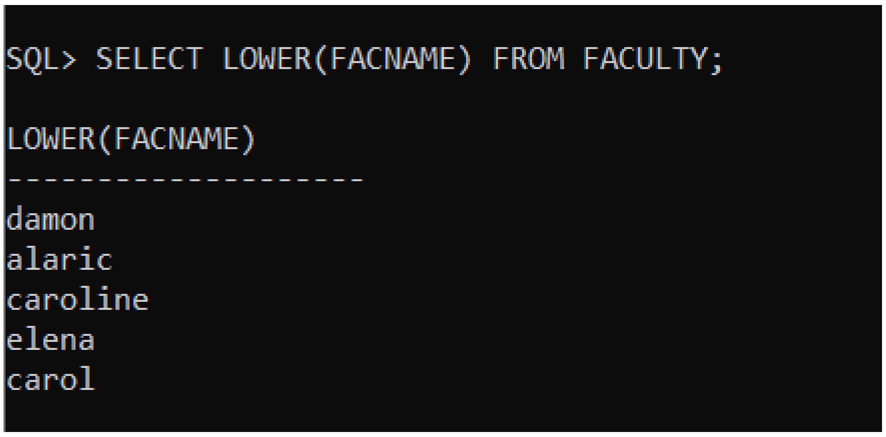
SELECT CONCAT(FNAMESTUD,STUDENT_ID) FROM STUDENT;



```
SQL> SELECT CONCAT(FNAMESTUD,STUDENT_ID) FROM STUDENT;

CONCAT(FNAMESTUD,STUDENT_ID)
------------------------------------------------------------
POOJA601
RAJATH102
SWETHA103
SUSHANTH104
ABHISHEK105
```

TO LOWER FACULTY NAME FROM FACULTY TABLE

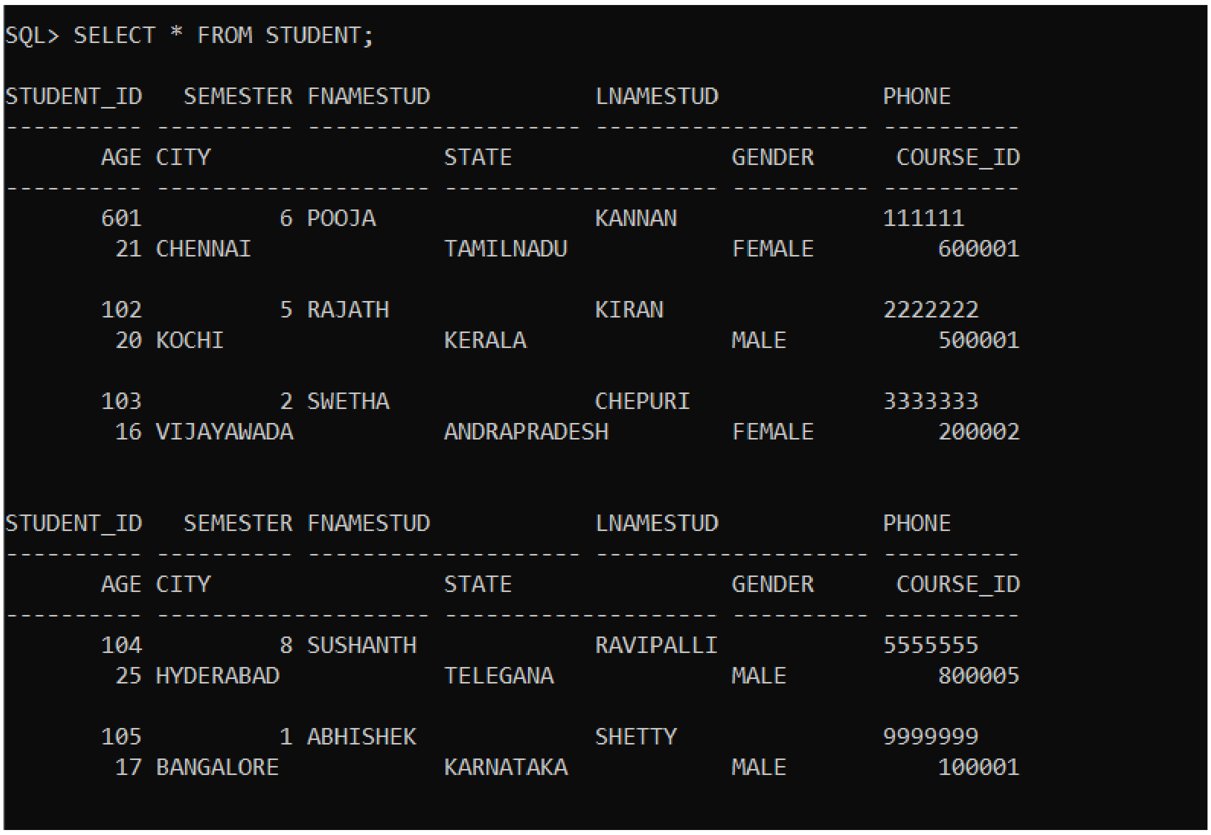
SELECT LOWER(FACNAME) FROM FACULTY;



```
SQL> SELECT LOWER(FACNAME) FROM FACULTY;

LOWER(FACNAME)
--------------------
damon
alaric
caroline
elena
carol
```

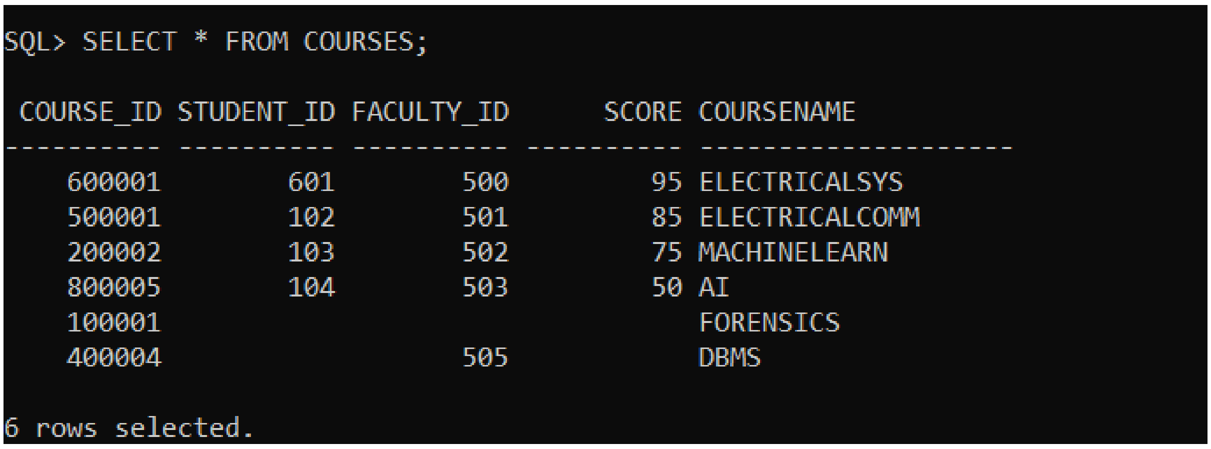
SHOW ALL DETAILS OF STUDENTS

SELECT * FROM STUDENT;



```
SQL> SELECT * FROM STUDENT;

STUDENT_ID   SEMESTER FNAMESTUD            LNAMESTUD            PHONE
---------- ---------- -------------------- -------------------- ----------
       AGE CITY                 STATE                GENDER      COURSE_ID
---------- -------------------- -------------------- ---------- ----------
       601          6 POOJA                KANNAN               111111
        21 CHENNAI              TAMILNADU            FEMALE         600001

       102          5 RAJATH               KIRAN                2222222
        20 KOCHI                KERALA               MALE           500001

       103          2 SWETHA               CHEPURI              3333333
        16 VIJAYAWADA           ANDRAPRADESH         FEMALE         200002


STUDENT_ID   SEMESTER FNAMESTUD            LNAMESTUD            PHONE
---------- ---------- -------------------- -------------------- ----------
       AGE CITY                 STATE                GENDER      COURSE_ID
---------- -------------------- -------------------- ---------- ----------
       104          8 SUSHANTH             RAVIPALLI            5555555
        25 HYDERABAD            TELEGANA             MALE           800005

       105          1 ABHISHEK             SHETTY               9999999
        17 BANGALORE            KARNATAKA            MALE           100001
```

SHOW ALL DETAILS OF COURSES

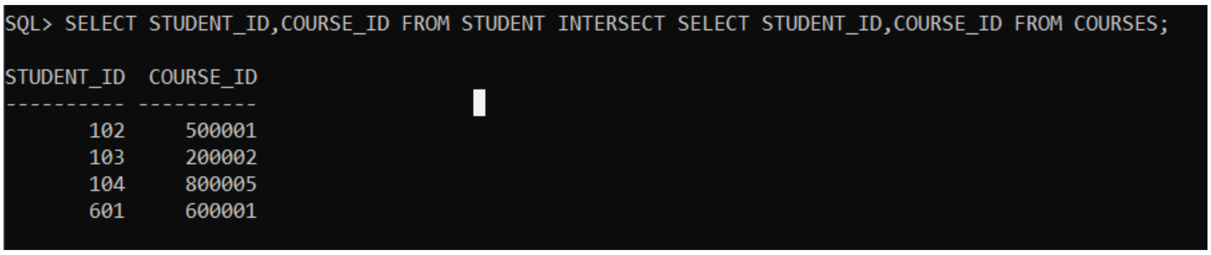


```
SQL> SELECT * FROM COURSES;

 COURSE_ID STUDENT_ID FACULTY_ID      SCORE COURSENAME
---------- ---------- ---------- ---------- --------------------
    600001        601        500         95 ELECTRICALSYS
    500001        102        501         85 ELECTRICALCOMM
    200002        103        502         75 MACHINELEARN
    800005        104        503         50 AI
    100001                                  FORENSICS
    400004                   505            DBMS

6 rows selected.
```

SQL OPERATIONS

FIND DETAILS OF STUDENT WHO HAS ENTROLLED IN A COURSE

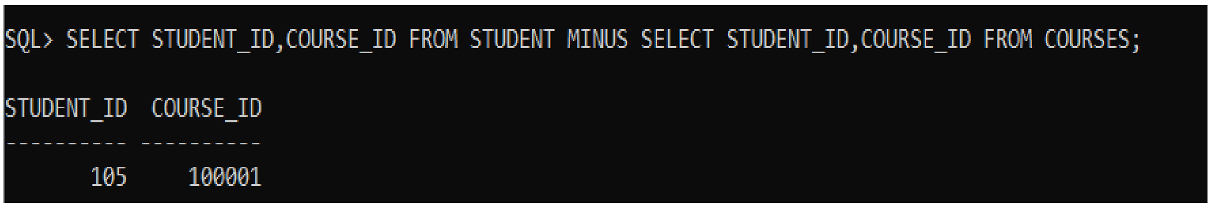
SELECT STUDENT_ID,COURSE_ID FROM STUDENT INTERSECT SELECT STUDENT_ID,COURSE_ID FROM COURSES;



```
SQL> SELECT STUDENT_ID,COURSE_ID FROM STUDENT INTERSECT SELECT STUDENT_ID,COURSE_ID FROM COURSES;

STUDENT_ID  COURSE_ID
---------- ----------
       102     500001
       103     200002
       104     800005
       601     600001
```

FIND STUDENT ID OF STUDENTS WHO HAS NOT ENROLLED IN A COURSE

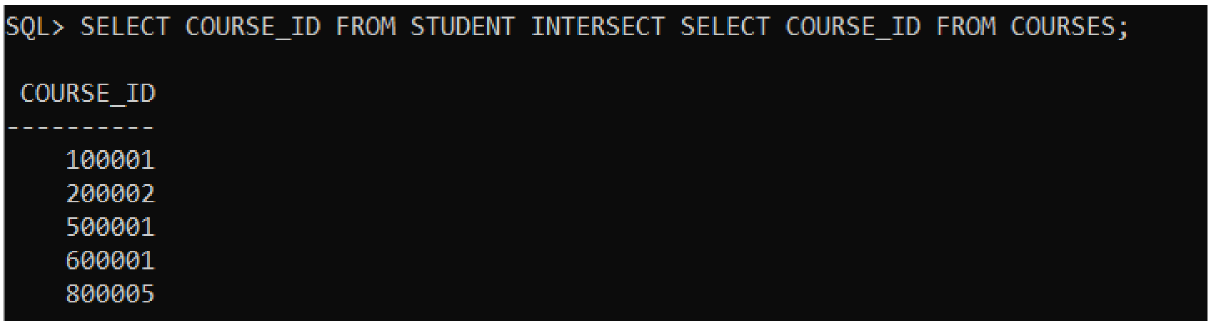
SELECT STUDENT_ID,COURSE_ID FROM STUDENT MINUS SELECT STUDENT_ID,COURSE_ID FROM COURSES;



```
SQL> SELECT STUDENT_ID,COURSE_ID FROM STUDENT MINUS SELECT STUDENT_ID,COURSE_ID FROM COURSES;

STUDENT_ID  COURSE_ID
---------- ----------
       105     100001
```

FIND COURSE ID OF STUDENTS WHO HAVE REGISTED FOR A COURSE

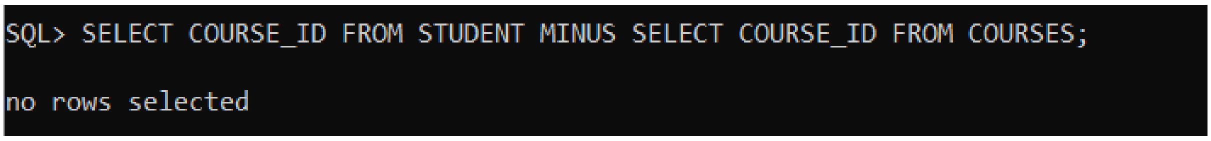
SELECT COURSE_ID FROM STUDENT INTERSECT SELECT COURSE_ID FROM COURSES;



```
SQL> SELECT COURSE_ID FROM STUDENT INTERSECT SELECT COURSE_ID FROM COURSES;

 COURSE_ID
----------
    100001
    200002
    500001
    600001
    800005
```

FIND STUDENTS WHO HAVE NOT REGISTERED FOR A COURSE

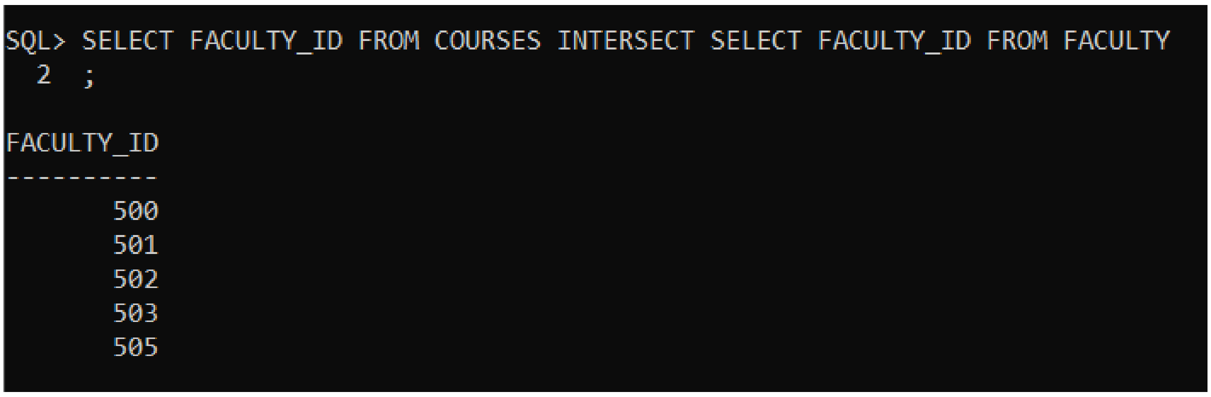
SELECT COURSE_ID FROM STUDENT MINUS SELECT COURSE_ID FROM COURSES;



```
SQL> SELECT COURSE_ID FROM STUDENT MINUS SELECT COURSE_ID FROM COURSES;

no rows selected
```

FIND FACULTY ID WHO HAS ASSIGNED FOR A COURSE

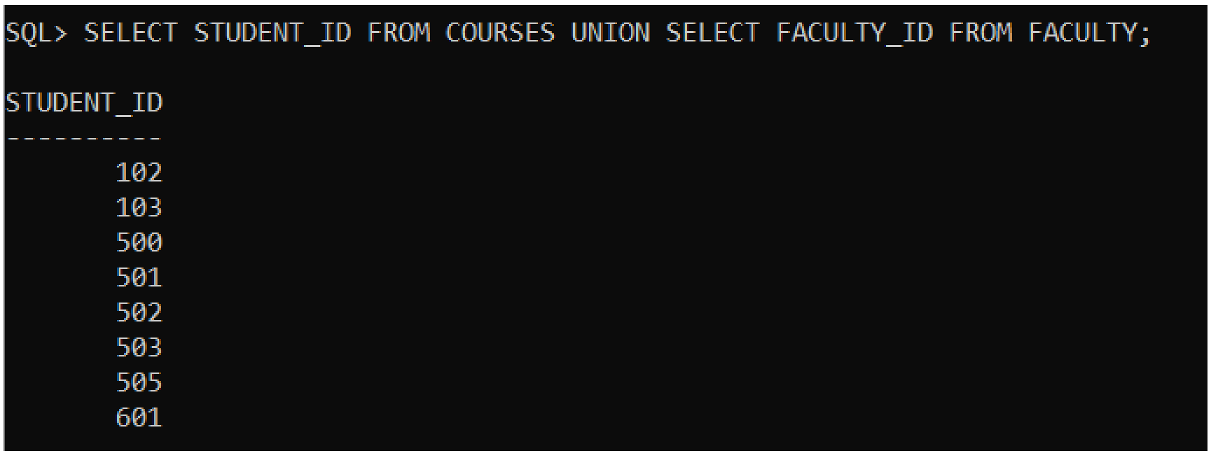
SELECT FACULTY_ID FROM COURSES INTERSECT SELECT FACULTY_ID FROM FACULTY;



```
SQL> SELECT FACULTY_ID FROM COURSES INTERSECT SELECT FACULTY_ID FROM FACULTY
  2  ;

FACULTY_ID
----------
       500
       501
       502
       503
       505
```

GET IDS OF STUDENT AND FACULTY

SELECT STUDENT_ID FROM COURSES UNION SELECT FACULTY_ID FROM FACULTY;

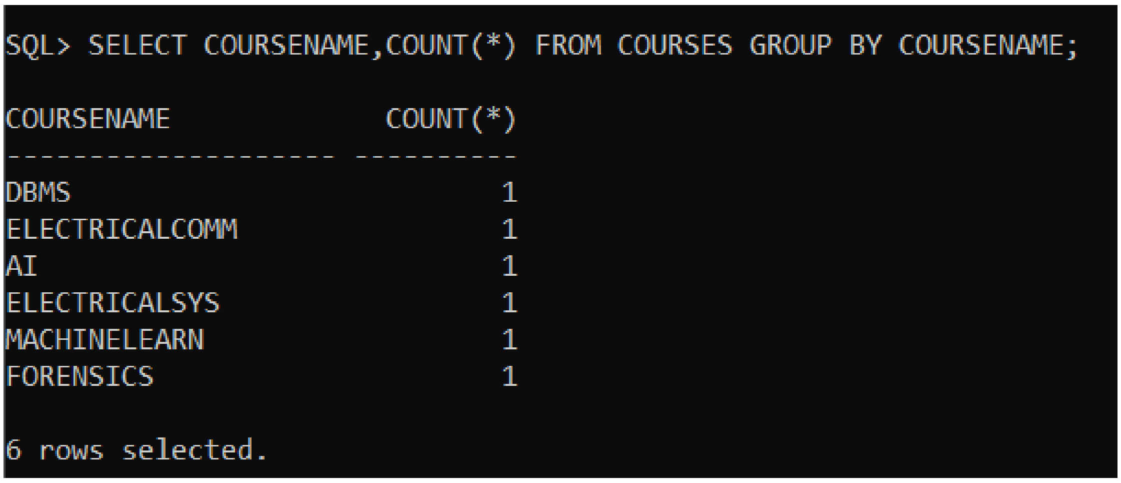


```
SQL> SELECT STUDENT_ID FROM COURSES UNION SELECT FACULTY_ID FROM FACULTY;

STUDENT_ID
----------
       102
       103
       500
       501
       502
       503
       505
       601
```

AGGREGATION

FIND TOTANLE NO OF STUDNETS WHO HAS TAKEN DIFFERENT COURSES

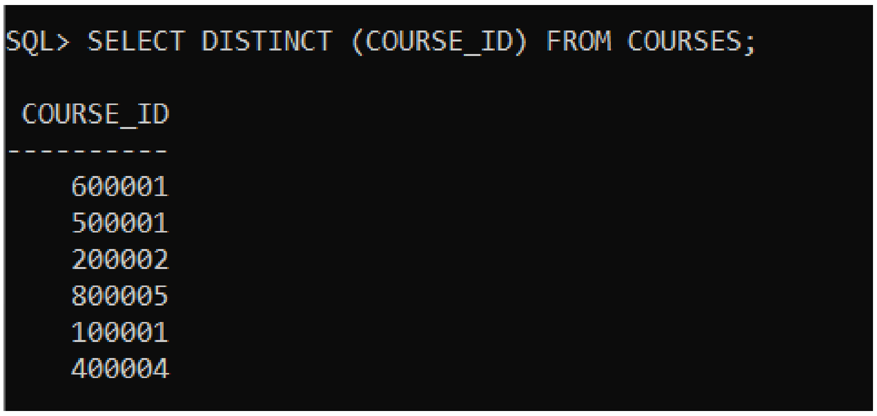
SELECT COURSENAME,COUNT(*) FROM COURSES GROUP BY COURSENAME;



```
SQL> SELECT COURSENAME,COUNT(*) FROM COURSES GROUP BY COURSENAME;

COURSENAME             COUNT(*)
-------------------- ----------
DBMS                          1
ELECTRICALCOMM                1
AI                            1
ELECTRICALSYS                 1
MACHINELEARN                  1
FORENSICS                     1

6 rows selected.
```

FIND DISTINCT COURSE IDS

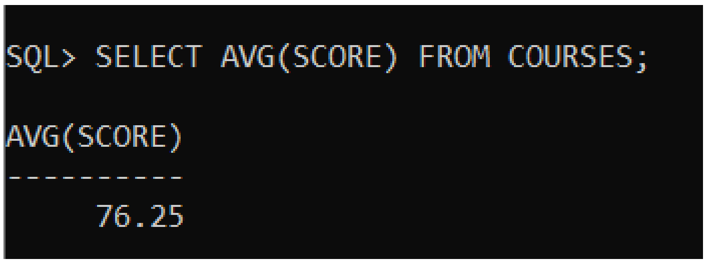
SELECT DISTINCT (COURSE_ID) FROM COURSES;



```
SQL> SELECT DISTINCT (COURSE_ID) FROM COURSES;

 COURSE_ID
----------
    600001
    500001
    200002
    800005
    100001
    400004
```

FIND AVERAGE SCORE SCORED BY STUDENTS

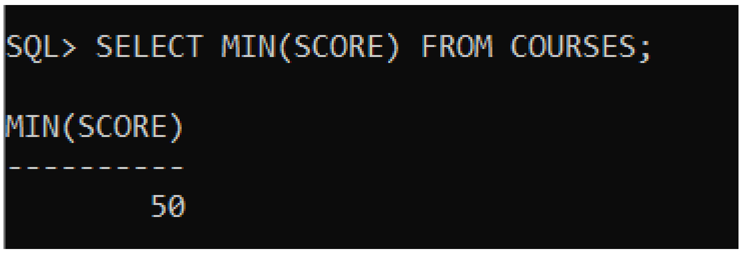
SELECT AVG(SCORE) FROM COURSES;



```
SQL> SELECT AVG(SCORE) FROM COURSES;

AVG(SCORE)
----------
     76.25
```

FIND MINIMUM SCORE SCORED BY STUDENTS

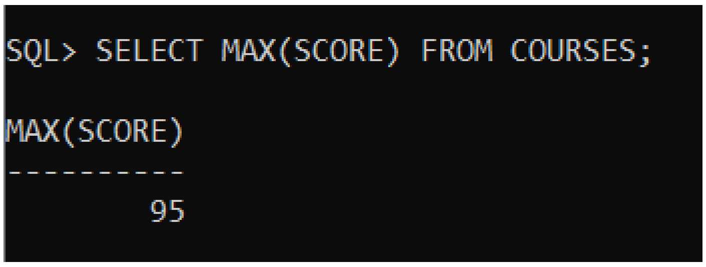
SELECT MIN(SCORE) FROM COURSES;



```
SQL> SELECT MIN(SCORE) FROM COURSES;

MIN(SCORE)
----------
        50
```

FIND MAXIMUM SCORE SCORED BY STUDENTS

SQL> SELECT MAX(SCORE) FROM COURSES;

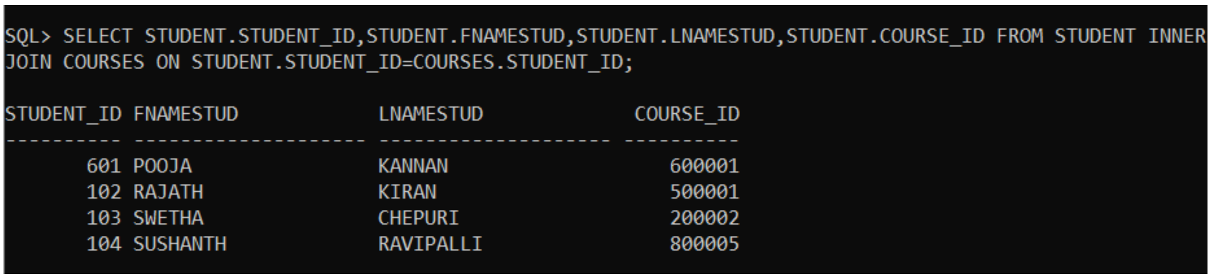


```
SQL> SELECT MAX(SCORE) FROM COURSES;

MAX(SCORE)
----------
        95
```

JOINS

FIND STUDENT ID STUDENT NAME AND COURSE ID OF STUDENTS ENROLLED FOR A COURSE

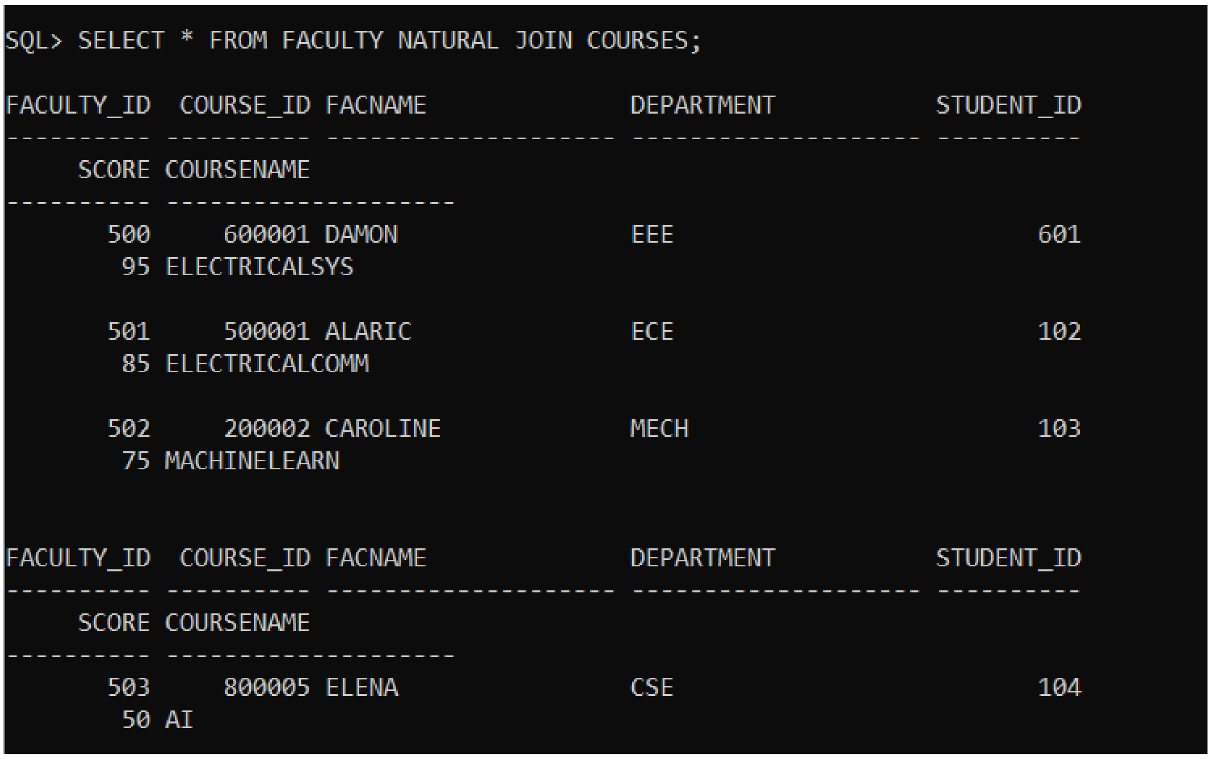
SELECT STUDENT.STUDENT_ID,STUDENT.FNAMESTUD,STUDENT.LNAMESTUD,STUDENT.COURSE_ID FROM STUDENT INNER JOIN COURSES ON STUDENT.STUDENT_ID=COURSES.STUDENT_ID;



```
SQL> SELECT STUDENT.STUDENT_ID,STUDENT.FNAMESTUD,STUDENT.LNAMESTUD,STUDENT.COURSE_ID FROM STUDENT INNER
JOIN COURSES ON STUDENT.STUDENT_ID=COURSES.STUDENT_ID;

STUDENT_ID FNAMESTUD            LNAMESTUD             COURSE_ID
---------- -------------------- -------------------- ----------
       601 POOJA                KANNAN                   600001
       102 RAJATH               KIRAN                    500001
       103 SWETHA               CHEPURI                  200002
       104 SUSHANTH             RAVIPALLI                800005
```

GET DETAILS ABOUT FACULTY WHO IS ASSIGNED TO A COURSE

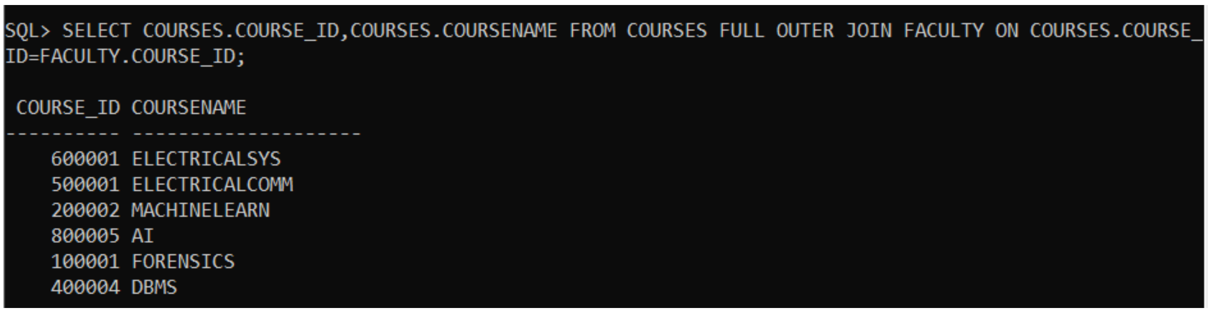
SELECT * FROM FACULTY NATURAL JOIN COURSES;



```
SQL> SELECT * FROM FACULTY NATURAL JOIN COURSES;

FACULTY_ID  COURSE_ID FACNAME              DEPARTMENT           STUDENT_ID
---------- ---------- -------------------- -------------------- ----------
     SCORE COURSENAME
---------- --------------------
       500     600001 DAMON                EEE                         601
        95 ELECTRICALSYS

       501     500001 ALARIC               ECE                         102
        85 ELECTRICALCOMM

       502     200002 CAROLINE             MECH                        103
        75 MACHINELEARN


FACULTY_ID  COURSE_ID FACNAME              DEPARTMENT           STUDENT_ID
---------- ---------- -------------------- -------------------- ----------
     SCORE COURSENAME
---------- --------------------
       503     800005 ELENA                CSE                         104
        50 AI
```

LIST COURSE ID AND COURSE NAME OF COURSES WITH FACULTY

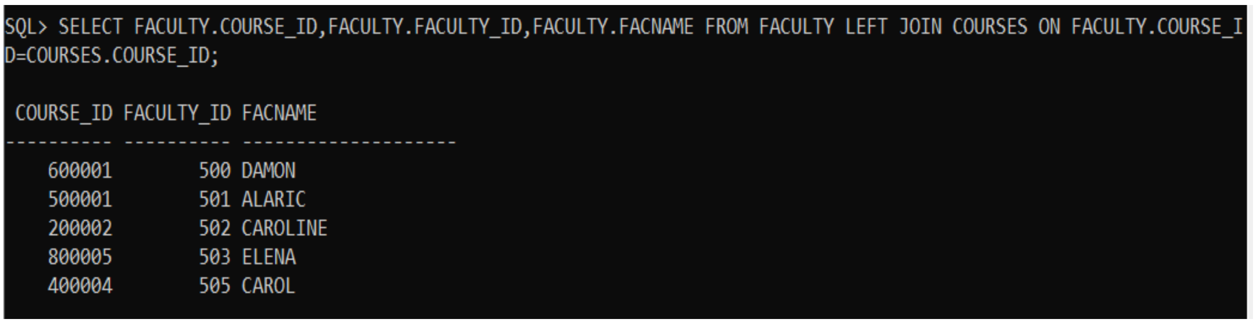
SELECT COURSES.COURSE_ID,COURSES.COURSENAME FROM COURSES FULL OUTER JOIN FACULTY ON COURSES.COURSE_ID=FACULTY.COURSE_ID;



```
SQL> SELECT COURSES.COURSE_ID,COURSES.COURSENAME FROM COURSES FULL OUTER JOIN FACULTY ON COURSES.COURSE_
ID=FACULTY.COURSE_ID;

 COURSE_ID COURSENAME
---------- --------------------
    600001 ELECTRICALSYS
    500001 ELECTRICALCOMM
    200002 MACHINELEARN
    800005 AI
    100001 FORENSICS
    400004 DBMS
```

LIST COURSE ID, FACULTY ID AND FACULTY NAME OF FACULTY WHO IS ASSIGNED TO A COURSE

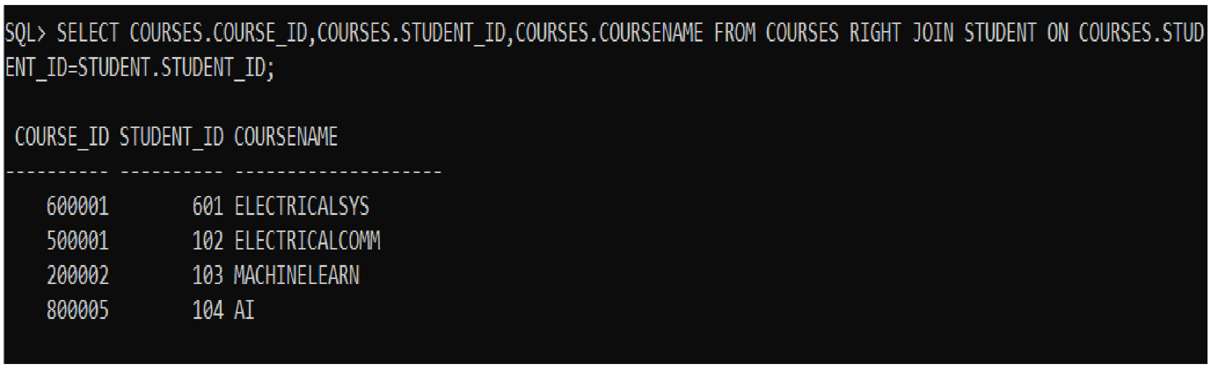
SELECT FACULTY.COURSE_ID,FACULTY.FACULTY_ID,FACULTY.FACNAME FROM FACULTY LEFT JOIN COURSES ON FACULTY.COURSE_ID=COURSES.COURSE_ID;



```
SQL> SELECT FACULTY.COURSE_ID,FACULTY.FACULTY_ID,FACULTY.FACNAME FROM FACULTY LEFT JOIN COURSES ON FACULTY.COURSE_I
D=COURSES.COURSE_ID;

 COURSE_ID FACULTY_ID FACNAME
---------- ---------- -------------------
    600001        500 DAMON
    500001        501 ALARIC
    200002        502 CAROLINE
    800005        503 ELENA
    400004        505 CAROL
```

LIST COURSE ID, STUDENT ID AND COURSENAME OF STUDENTS WHO HAS ENROLLED FOR A COURSE

SELECT COURSES.COURSE_ID, COURSES.STUDENT_ID, COURSES.COURSENAME FROM COURSES RIGHT JOIN STUDENT ON COURSES.STUDENT_ID=STUDENT.STUDENT_ID;

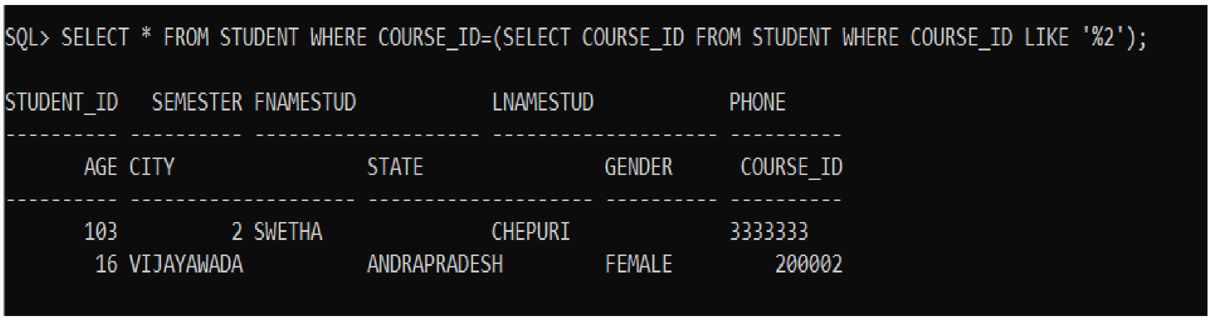


```
SQL> SELECT COURSES.COURSE_ID,COURSES.STUDENT_ID,COURSES.COURSENAME FROM COURSES RIGHT JOIN STUDENT ON COURSES.STUD
ENT_ID=STUDENT.STUDENT_ID;

 COURSE_ID STUDENT_ID COURSENAME
---------- ---------- -------------------
    600001        601 ELECTRICALSYS
    500001        102 ELECTRICALCOMM
    200002        103 MACHINELEARN
    800005        104 AI
```

SUBQUERIES

GET DETAILS ABOUT STUDENTS WHOSE COURSE ID ENDS WITH 2

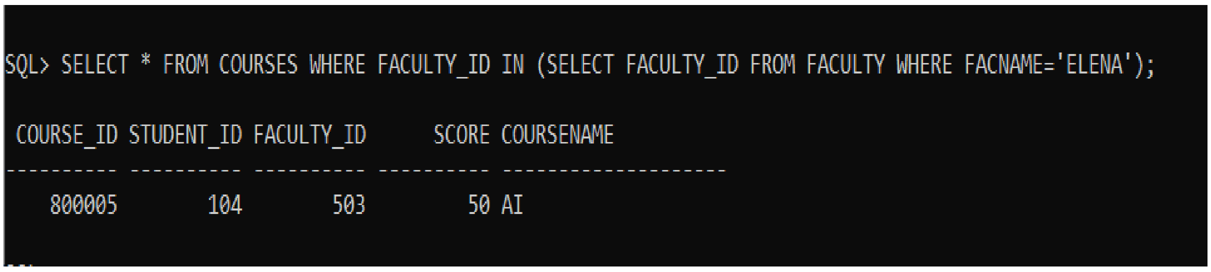
SELECT * FROM STUDENT WHERE COURSE_ID=(SELECT COURSE_ID FROM STUDENT WHERE COURSE_ID LIKE '%2');



```
SQL> SELECT * FROM STUDENT WHERE COURSE_ID=(SELECT COURSE_ID FROM STUDENT WHERE COURSE_ID LIKE '%2');

STUDENT_ID   SEMESTER FNAMESTUD            LNAMESTUD            PHONE
---------- ---------- -------------------- -------------------- ----------
       AGE CITY                 STATE                GENDER      COURSE_ID
---------- -------------------- -------------------- ---------- ----------
       103          2 SWETHA               CHEPURI              3333333
        16 VIJAYAWADA           ANDRAPRADESH         FEMALE         200002
```

FIND ALL DETAILS WHOSE FACULTY IS ELENA

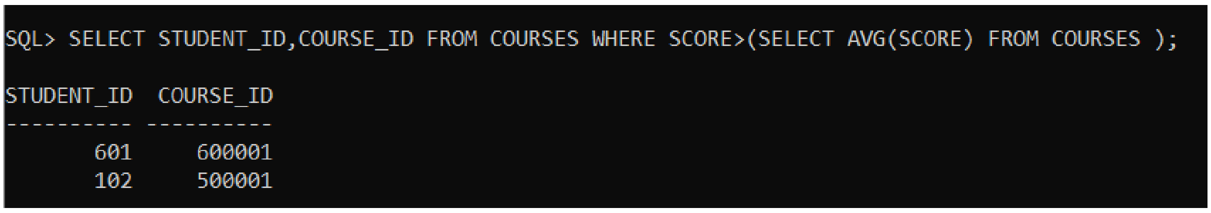
SELECT * FROM COURSES WHERE FACULTY_ID IN (SELECT FACULTY_ID FROM FACULTY WHERE FACNAME='ELENA');



```
SQL> SELECT * FROM COURSES WHERE FACULTY_ID IN (SELECT FACULTY_ID FROM FACULTY WHERE FACNAME='ELENA');

 COURSE_ID STUDENT_ID FACULTY_ID      SCORE COURSENAME
---------- ---------- ---------- ---------- --------------------
    800005        104        503         50 AI
```

GET LIST OF STUDENTS WHO HAS SCORED ABOVE AVERAGE SCORE

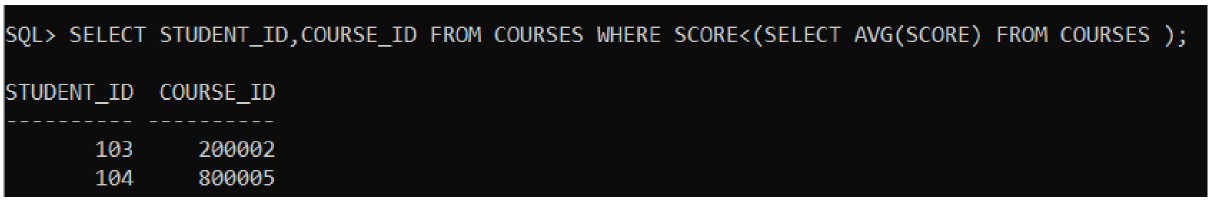
SELECT STUDENT_ID,COURSE_ID FROM COURSES WHERE SCORE>(SELECT AVG(SCORE) FROM COURSES);



```
SQL> SELECT STUDENT_ID,COURSE_ID FROM COURSES WHERE SCORE>(SELECT AVG(SCORE) FROM COURSES );

STUDENT_ID  COURSE_ID
---------- ----------
       601     600001
       102     500001
```

GET LIST OF STUDENTS WHO HAS SCORED BELOW AVERAGE SCORE

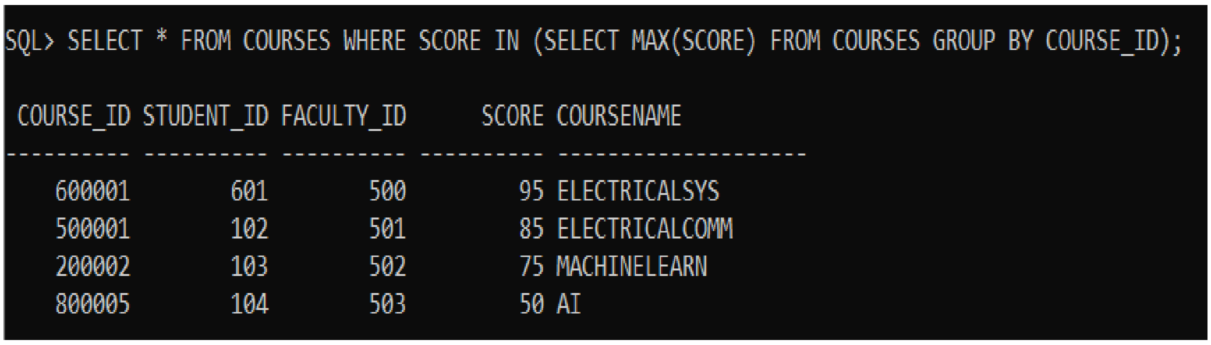
SELECT STUDENT_ID,COURSE_ID FROM COURSES WHERE SCORE<(SELECT AVG(SCORE) FROM COURSES);



```
SQL> SELECT STUDENT_ID,COURSE_ID FROM COURSES WHERE SCORE<(SELECT AVG(SCORE) FROM COURSES );

STUDENT_ID  COURSE_ID
---------- ----------
       103     200002
       104     800005
```

FIND HIGHEST SCORED STUDENT FROM EACH COURSES

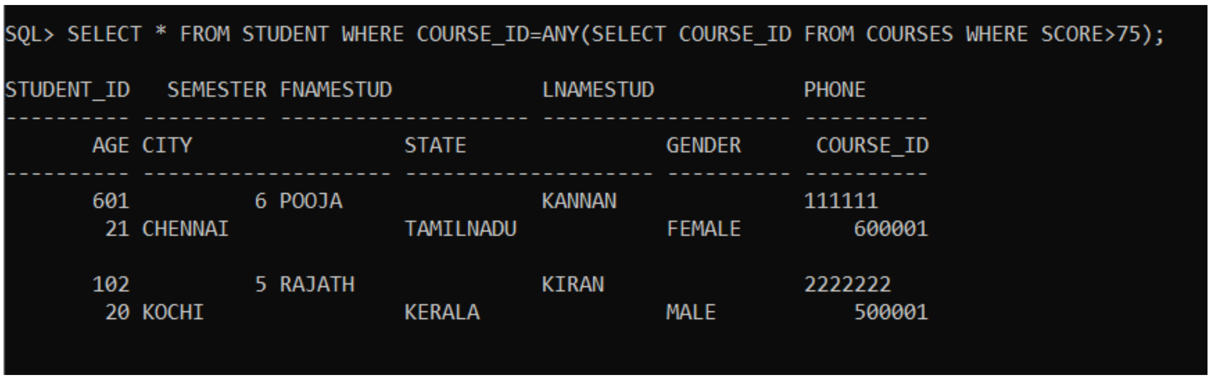
SELECT * FROM COURSES WHERE SCORE IN (SELECT MAX(SCORE) FROM COURSES GROUP BY COURSE_ID);



```
SQL> SELECT * FROM COURSES WHERE SCORE IN (SELECT MAX(SCORE) FROM COURSES GROUP BY COURSE_ID);

 COURSE_ID STUDENT_ID FACULTY_ID      SCORE COURSENAME
---------- ---------- ---------- ---------- --------------------
    600001        601        500         95 ELECTRICALSYS
    500001        102        501         85 ELECTRICALCOMM
    200002        103        502         75 MACHINELEARN
    800005        104        503         50 AI
```

FIND DETAILS OF STUDENTS WHO HAS SCORE ABOVE 75

SELECT * FROM STUDENT WHERE COURSE_ID=ANY(SELECT COURSE_ID FROM COURSES WHERE SCORE>75);

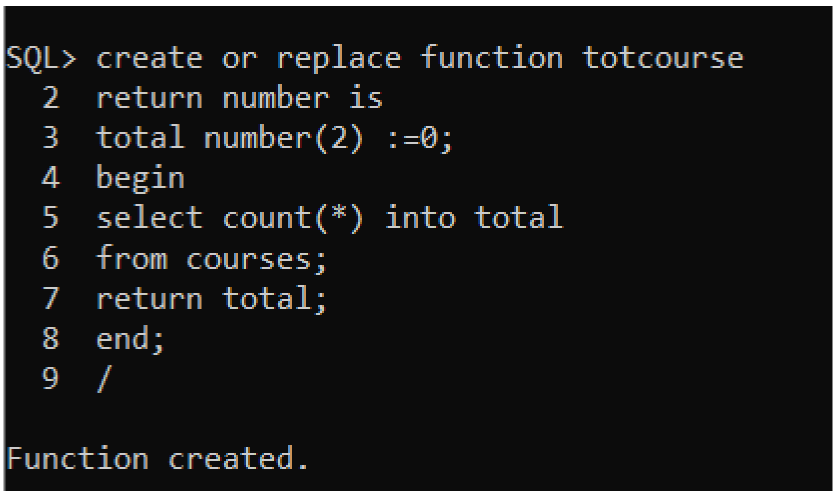


```
SQL> SELECT * FROM STUDENT WHERE COURSE_ID=ANY(SELECT COURSE_ID FROM COURSES WHERE SCORE>75);

STUDENT_ID   SEMESTER FNAMESTUD            LNAMESTUD            PHONE
---------- ---------- -------------------- -------------------- ----------
       AGE CITY                 STATE                GENDER      COURSE_ID
---------- -------------------- -------------------- ---------- ----------
       601          6 POOJA                KANNAN               111111
        21 CHENNAI              TAMILNADU            FEMALE         600001

       102          5 RAJATH               KIRAN                2222222
        20 KOCHI                KERALA               MALE           500001
```

PLSQL

FUNCTION TO CALCULATE THE TOTAL NUMBER OF COURSES

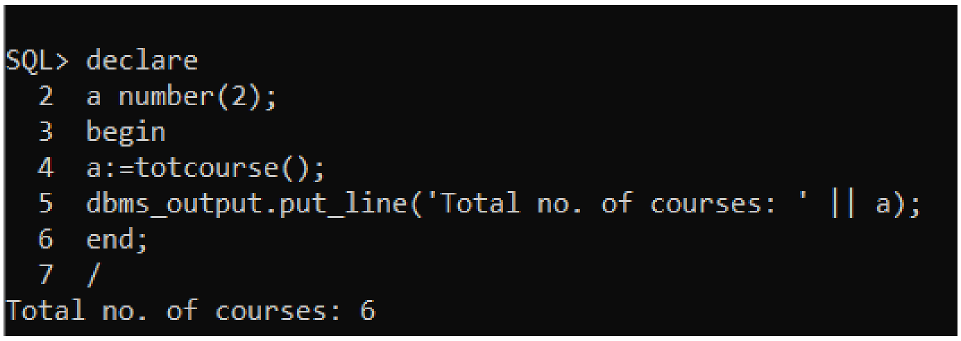
```
CREATE OR REPLACE FUNCTION TOTCOURSE
RETURN NUMBER IS
TOTAL NUMBER(2) :=0;
BEGIN
SELECT COUNT(*) INTO TOTAL
FROM COURSES;
RETURN TOTAL;
END;
/
```



```
SQL> create or replace function totcourse
return number is
total number(2) :=0;
begin
select count(*) into total
from courses;
return total;
end;
/

Function created.
```

CALLING THE FUCTION

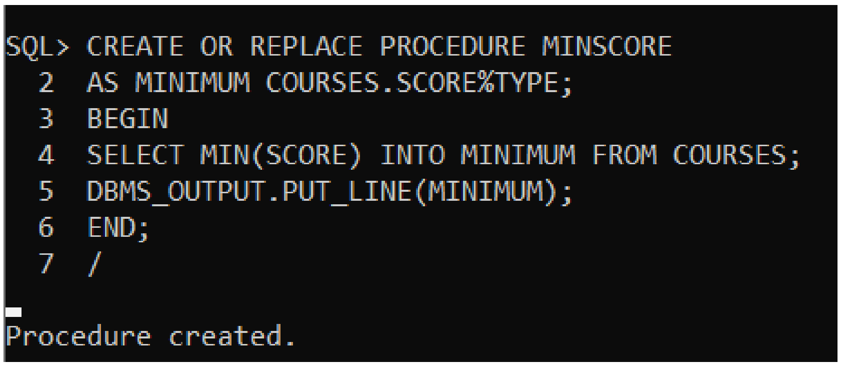
```
DECLARE
A NUMBER(2);
BEGIN
A:=TOTCOURSE();
DBMS_OUTPUT.PUT_LINE('TOTAL NO. OF COURSES: ' || A);
END;
/
```



```
SQL> declare
  2  a number(2);
  3  begin
  4  a:=totcourse();
  5  dbms_output.put_line('Total no. of courses: ' || a);
  6  end;
  7  /
Total no. of courses: 6
```

FUCTION TO FIND THE LOWEST SCORE

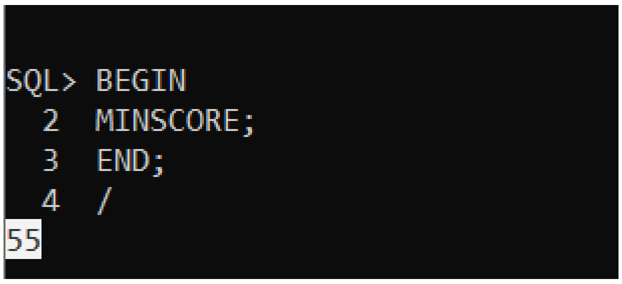
```
CREATE OR REPLACE PROCEDURE MINSCORE
  2  AS MINIMUM COURSES.SCORE%TYPE;
  3  BEGIN
  4  SELECT MIN(SCORE) INTO MINIMUM FROM COURSES;
  5  DBMS_OUTPUT.PUT_LINE(MINIMUM);
  6  END;
  7  /
```



```
SQL> CREATE OR REPLACE PROCEDURE MINSCORE
  2  AS MINIMUM COURSES.SCORE%TYPE;
  3  BEGIN
  4  SELECT MIN(SCORE) INTO MINIMUM FROM COURSES;
  5  DBMS_OUTPUT.PUT_LINE(MINIMUM);
  6  END;
  7  /

Procedure created.
```

CALLING THE FUCTION

```
BEGIN
  2  MIMSCORE;
  3  END;
  4  /
```

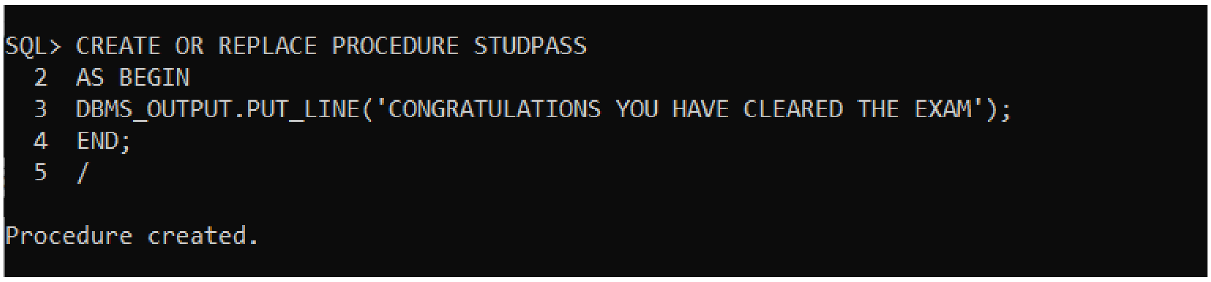


```
SQL> BEGIN
  2  MINSCORE;
  3  END;
  4  /
55
```

CHECK IF STUDENT WITH STUDENT ID 601 HAS CLEARED THE EXAM

CREATING PROCEDURE

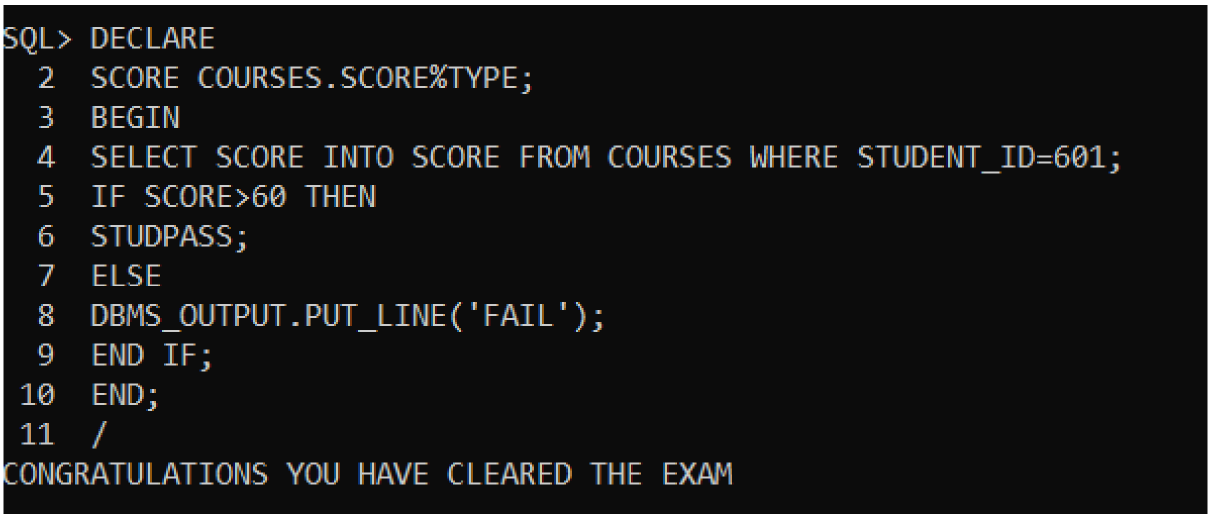
```
CREATE OR REPLACE PROCEDURE STUDPASS
  2  AS BEGIN
  3  DBMS_OUTPUT.PUT_LINE('CONGRATULATIONS YOU HAVE CLEARED THE EXAM');
  4  END;
  5  /
```



```
SQL> CREATE OR REPLACE PROCEDURE STUDPASS
  2  AS BEGIN
  3  DBMS_OUTPUT.PUT_LINE('CONGRATULATIONS YOU HAVE CLEARED THE EXAM');
  4  END;
  5  /

Procedure created.
```

DECLARING THE PROCEDURE

```
DECLARE
  2  SCORE COURSES.SCORE%TYPE;
  3  BEGIN
  4  SELECT SCORE INTO SCORE FROM COURSES WHERE STUDENT_ID=601;
  5  IF SCORE>60 THEN
  6  STUDPASS;
  7  ELSE
  8  DBMS_OUTPUT.PUT_LINE('FAIL');
  9  END IF;
 10  END;
 11  /
```

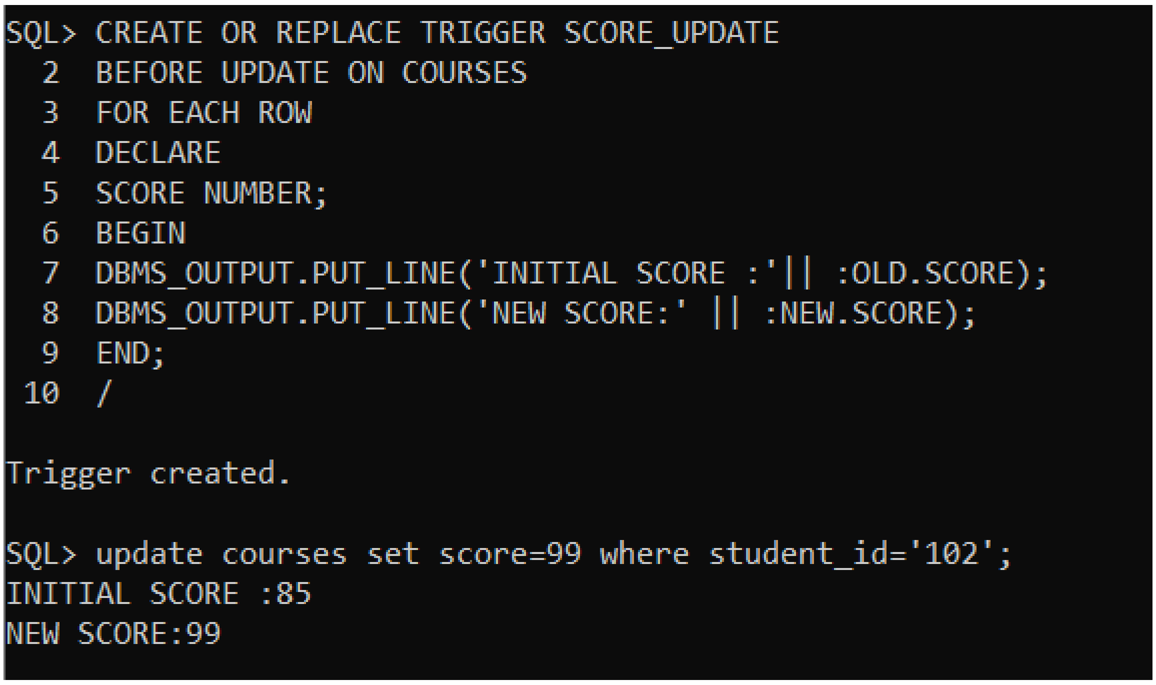


```
SQL> DECLARE
SCORE COURSES.SCORE%TYPE;
BEGIN
SELECT SCORE INTO SCORE FROM COURSES WHERE STUDENT_ID=601;
IF SCORE>60 THEN
STUDPASS;
ELSE
DBMS_OUTPUT.PUT_LINE('FAIL');
END IF;
END;
/
CONGRATULATIONS YOU HAVE CLEARED THE EXAM
```

TRIGGERS

CREATE TRIGGER TO UPDATE SCORE OF STUDENT WITH ID 102 TO 99

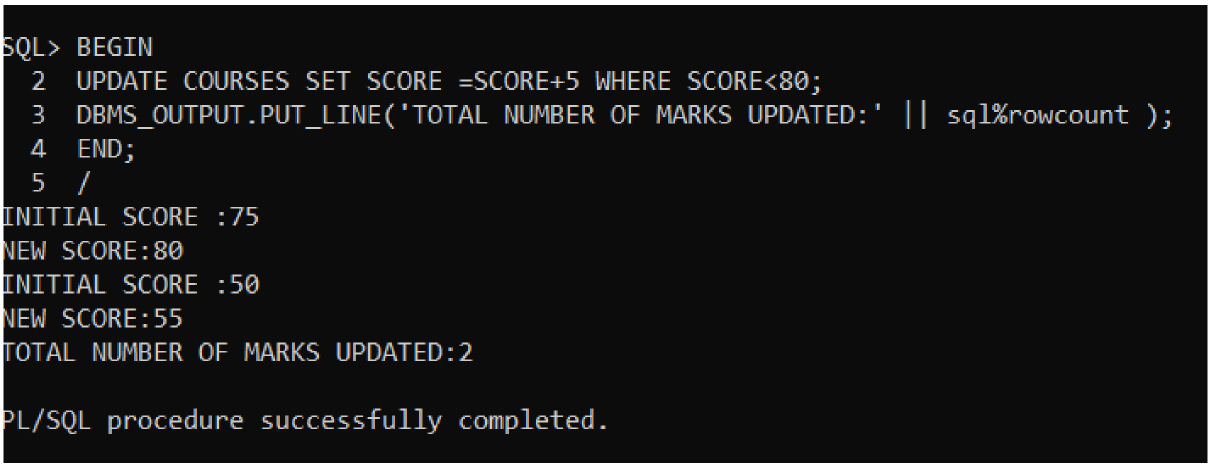
```
SQL> CREATE OR REPLACE TRIGGER SCORE_UPDATE
BEFORE UPDATE ON COURSES
FOR EACH ROW
DECLARE
SCORE NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('INITIAL SCORE :'|| :OLD.SCORE);
DBMS_OUTPUT.PUT_LINE('NEW SCORE:' || :NEW.SCORE);
END;
/
```



```
SQL> CREATE OR REPLACE TRIGGER SCORE_UPDATE
BEFORE UPDATE ON COURSES
FOR EACH ROW
DECLARE
SCORE NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('INITIAL SCORE :'|| :OLD.SCORE);
DBMS_OUTPUT.PUT_LINE('NEW SCORE:' || :NEW.SCORE);
END;
/

Trigger created.

SQL> update courses set score=99 where student_id='102';
INITIAL SCORE :85
NEW SCORE:99
```

INCREASE MARKS BY 5 TO THOSE STUDENTS WHOSE SCORE IS LESS THAN 80

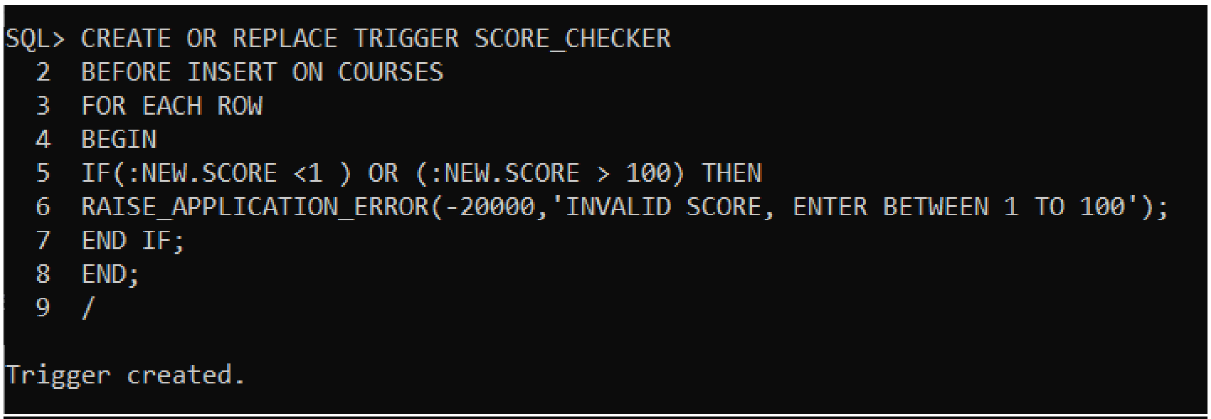
```
SQL> BEGIN
  2  UPDATE COURSES SET SCORE =SCORE+5 WHERE SCORE<80;
  3  DBMS_OUTPUT.PUT_LINE('TOTAL NUMBER OF MARKS UPDATED:' || sql%rowcount );
  4  END;
  5  /
```



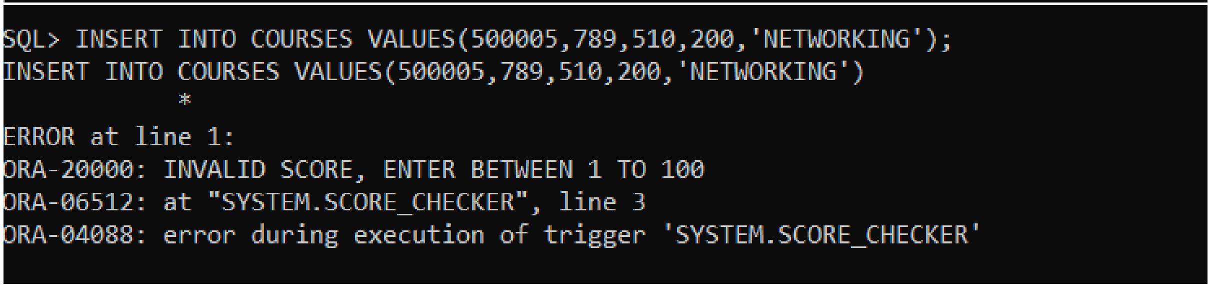
```
SQL> BEGIN
  2  UPDATE COURSES SET SCORE =SCORE+5 WHERE SCORE<80;
  3  DBMS_OUTPUT.PUT_LINE('TOTAL NUMBER OF MARKS UPDATED:' || sql%rowcount );
  4  END;
  5  /
INITIAL SCORE :75
NEW SCORE:80
INITIAL SCORE :50
NEW SCORE:55
TOTAL NUMBER OF MARKS UPDATED:2

PL/SQL procedure successfully completed.
```

CHECK IF THE SCORE IS ENTERED BETWEEN 1 AND 100. ELSE SHOW ERROR

```
CREATE OR REPLACE TRIGGER SCORE_CHECKER
  2  BEFORE INSERT ON COURSES
  3  FOR EACH ROW
  4  BEGIN
  5  IF(:NEW.SCORE <1 ) OR (:NEW.SCORE > 100) THEN
  6  RAISE_APPLICATION_ERROR(-20000,'INVALID SCORE, ENTER BETWEEN 1 TO 100');
  7  END IF;
  8  END;
  9  /
```



```
SQL> CREATE OR REPLACE TRIGGER SCORE_CHECKER
  2  BEFORE INSERT ON COURSES
  3  FOR EACH ROW
  4  BEGIN
  5  IF(:NEW.SCORE <1 ) OR (:NEW.SCORE > 100) THEN
  6  RAISE_APPLICATION_ERROR(-20000,'INVALID SCORE, ENTER BETWEEN 1 TO 100');
  7  END IF;
  8  END;
  9  /

Trigger created.
```



```
SQL> INSERT INTO COURSES VALUES(500005,789,510,200,'NETWORKING');
INSERT INTO COURSES VALUES(500005,789,510,200,'NETWORKING')
            *
ERROR at line 1:
ORA-20000: INVALID SCORE, ENTER BETWEEN 1 TO 100
ORA-06512: at "SYSTEM.SCORE_CHECKER", line 3
ORA-04088: error during execution of trigger 'SYSTEM.SCORE_CHECKER'
```