NAME : KAMRAN ANSARI

REG NO : 22MCA0223

Consider the following relational database schema for teaching-learning process in a university.

PROFESSOR(Prof_id, Prof_name, Email, Mobile, Specialty, Dept_id)

SCHOOL(SCode, Scl_name, Prof_id, Location)

DEPARTMENT(Dept_id, Dname, SCode, Prof_id)

COURSE(Crs_code, Crs_name, Description, Credits, Hours)

CLASS(Cls_code, Slot, Stime, Etime, Crs_code, Prof_id, Room_no, Sem_code, Day_of_week)

SEMESTER(Sem_code, Term, Year, Sdate, Edate)

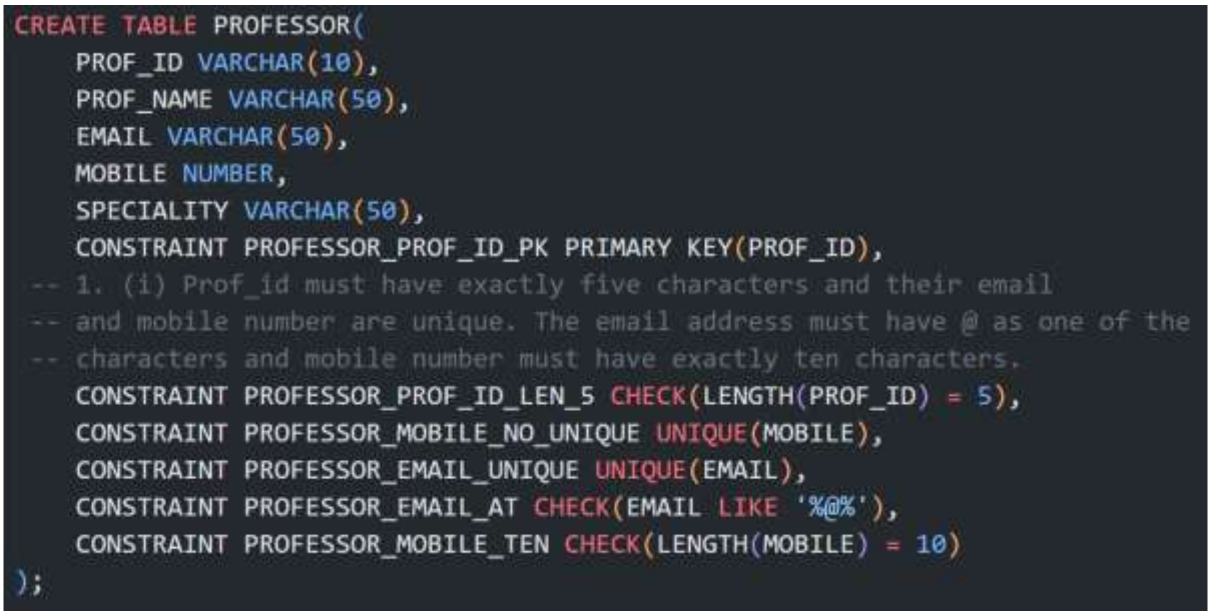
STUDENT(Reg_no, Sname, Address, DoB, Email, Mobile, Dept_id, Prof_id)

ENROLL(Cls_code, Reg_no, Enroll_time, Grade)

STUDENT_VISA(Reg_no, Visa_status)

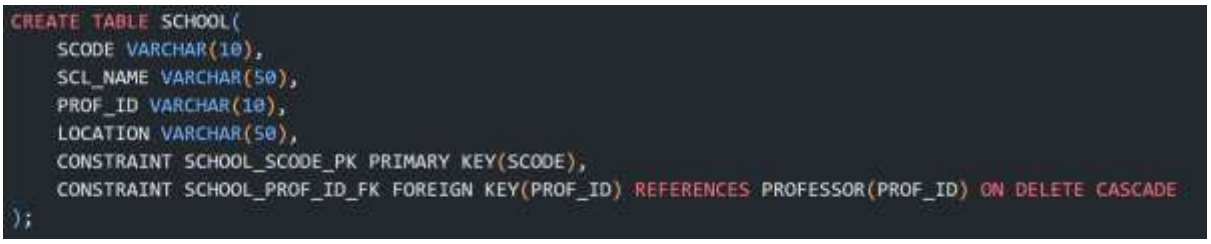
PROGRAMME(Prog_code, Prog_name, Prog_preamble, Scode, Dept_id)

<u>PROFESSOR</u>



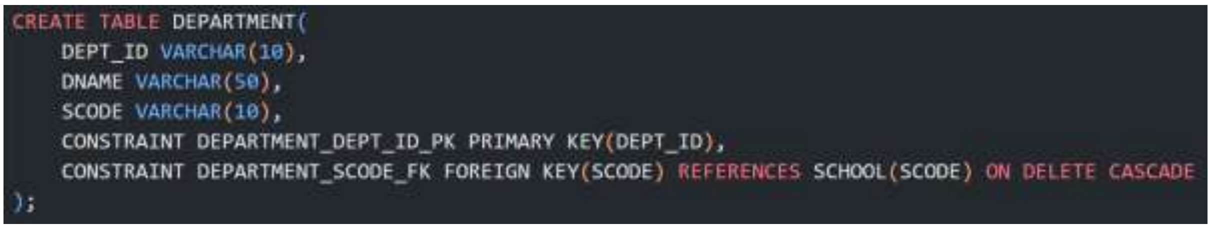
```
CREATE TABLE PROFESSOR(
    PROF_ID VARCHAR(10),
    PROF_NAME VARCHAR(50),
    EMAIL VARCHAR(50),
    MOBILE NUMBER,
    SPECIALITY VARCHAR(50),
    CONSTRAINT PROFESSOR_PROF_ID_PK PRIMARY KEY(PROF_ID),
 -- 1. (i) Prof_id must have exactly five characters and their email
 -- and mobile number are unique. The email address must have @ as one of the
 -- characters and mobile number must have exactly ten characters.
    CONSTRAINT PROFESSOR_PROF_ID_LEN_5 CHECK(LENGTH(PROF_ID) = 5),
    CONSTRAINT PROFESSOR_MOBILE_NO_UNIQUE UNIQUE(MOBILE),
    CONSTRAINT PROFESSOR_EMAIL_UNIQUE UNIQUE(EMAIL),
    CONSTRAINT PROFESSOR_EMAIL_AT CHECK(EMAIL LIKE '%@%'),
    CONSTRAINT PROFESSOR_MOBILE_TEN CHECK(LENGTH(MOBILE) = 10)
);
```

SCHOOL



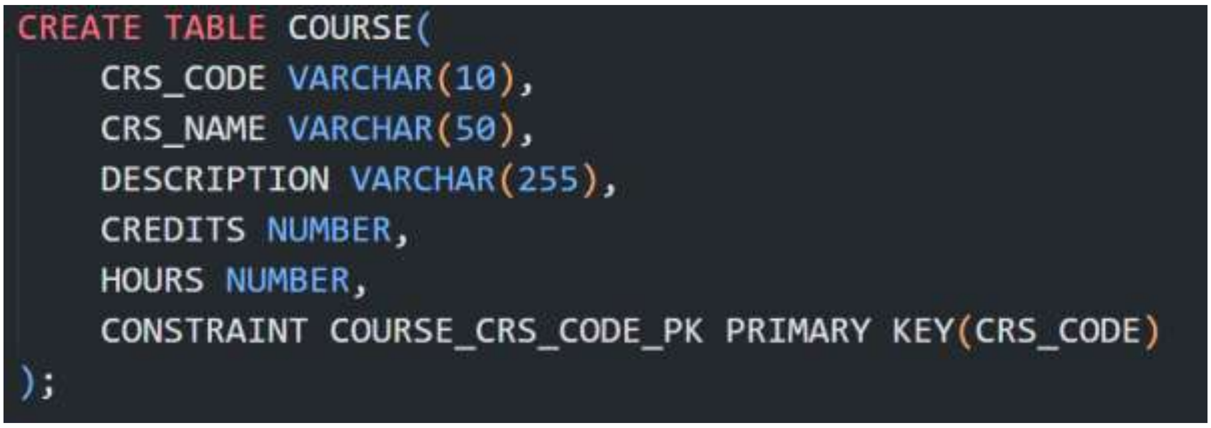
```sql
CREATE TABLE SCHOOL(
    SCODE VARCHAR(10),
    SCL_NAME VARCHAR(50),
    PROF_ID VARCHAR(10),
    LOCATION VARCHAR(50),
    CONSTRAINT SCHOOL_SCODE_PK PRIMARY KEY(SCODE),
    CONSTRAINT SCHOOL_PROF_ID_FK FOREIGN KEY(PROF_ID) REFERENCES PROFESSOR(PROF_ID) ON DELETE CASCADE
);
```

DEPARTMENT



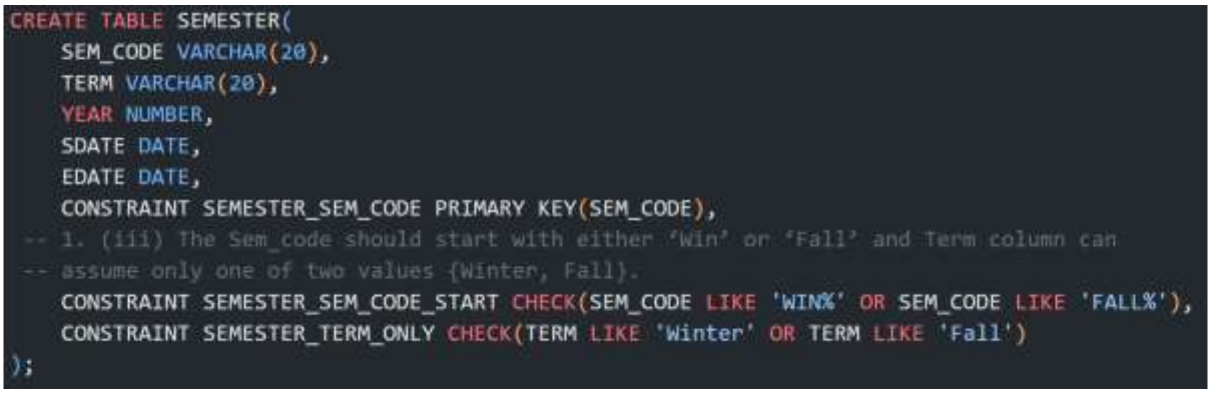
```sql
CREATE TABLE DEPARTMENT(
    DEPT_ID VARCHAR(10),
    DNAME VARCHAR(50),
    SCODE VARCHAR(10),
    CONSTRAINT DEPARTMENT_DEPT_ID_PK PRIMARY KEY(DEPT_ID),
    CONSTRAINT DEPARTMENT_SCODE_FK FOREIGN KEY(SCODE) REFERENCES SCHOOL(SCODE) ON DELETE CASCADE
);
```

COURSE



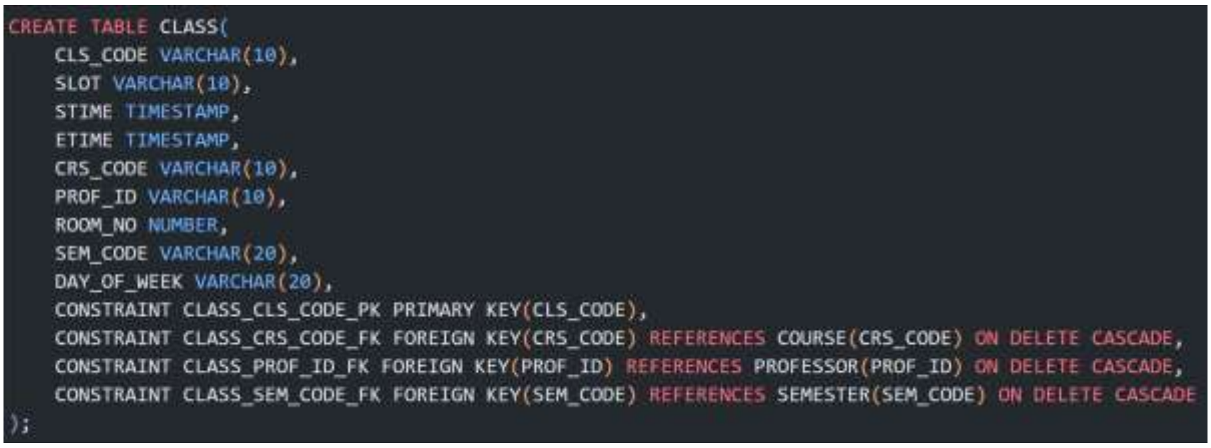
```sql
CREATE TABLE COURSE(
    CRS_CODE VARCHAR(10),
    CRS_NAME VARCHAR(50),
    DESCRIPTION VARCHAR(255),
    CREDITS NUMBER,
    HOURS NUMBER,
    CONSTRAINT COURSE_CRS_CODE_PK PRIMARY KEY(CRS_CODE)
);
```

SEMESTER



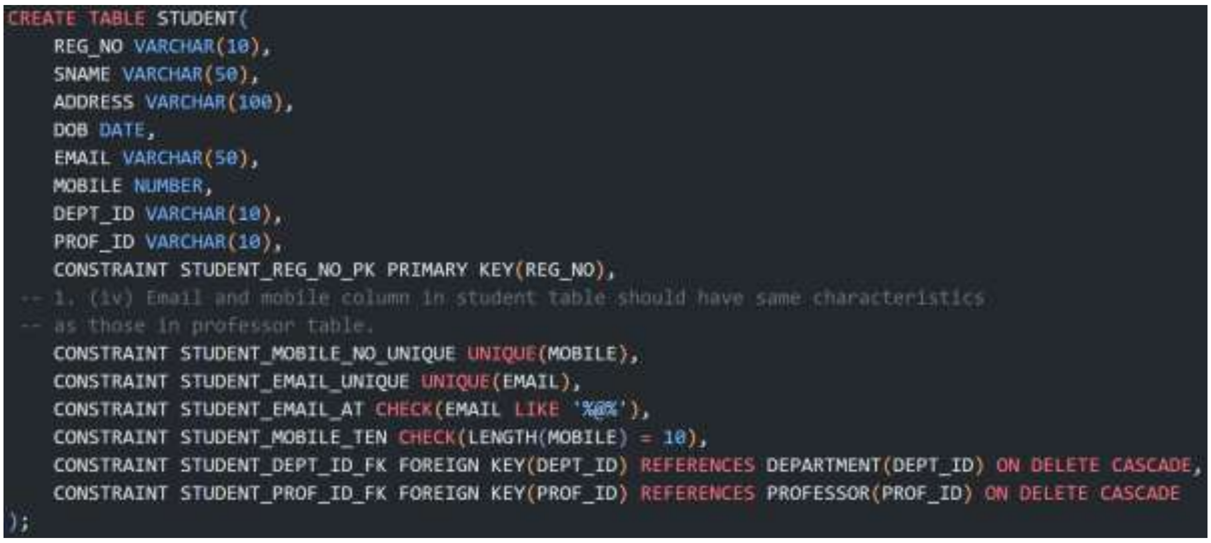
```sql
CREATE TABLE SEMESTER(
    SEM_CODE VARCHAR(20),
    TERM VARCHAR(20),
    YEAR NUMBER,
    SDATE DATE,
    EDATE DATE,
    CONSTRAINT SEMESTER_SEM_CODE PRIMARY KEY(SEM_CODE),
-- 1. (iii) The Sem_code should start with either ‘Win’ or ‘Fall’ and Term column can
-- assume only one of two values {Winter, Fall}.
    CONSTRAINT SEMESTER_SEM_CODE_START CHECK(SEM_CODE LIKE 'WIN%' OR SEM_CODE LIKE 'FALL%'),
    CONSTRAINT SEMESTER_TERM_ONLY CHECK(TERM LIKE 'Winter' OR TERM LIKE 'Fall')
);
```

CLASS



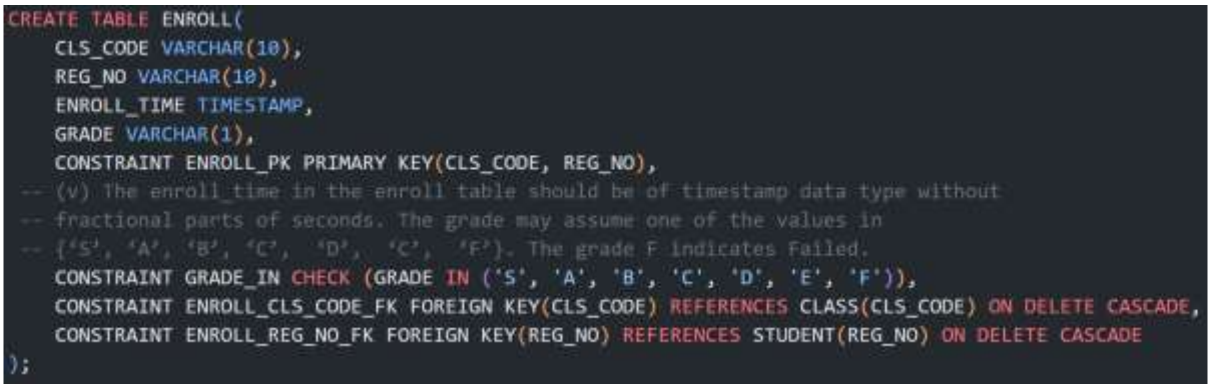
```sql
CREATE TABLE CLASS(
    CLS_CODE VARCHAR(10),
    SLOT VARCHAR(10),
    STIME TIMESTAMP,
    ETIME TIMESTAMP,
    CRS_CODE VARCHAR(10),
    PROF_ID VARCHAR(10),
    ROOM_NO NUMBER,
    SEM_CODE VARCHAR(20),
    DAY_OF_WEEK VARCHAR(20),
    CONSTRAINT CLASS_CLS_CODE_PK PRIMARY KEY(CLS_CODE),
    CONSTRAINT CLASS_CRS_CODE_FK FOREIGN KEY(CRS_CODE) REFERENCES COURSE(CRS_CODE) ON DELETE CASCADE,
    CONSTRAINT CLASS_PROF_ID_FK FOREIGN KEY(PROF_ID) REFERENCES PROFESSOR(PROF_ID) ON DELETE CASCADE,
    CONSTRAINT CLASS_SEM_CODE_FK FOREIGN KEY(SEM_CODE) REFERENCES SEMESTER(SEM_CODE) ON DELETE CASCADE
);
```

STUDENT



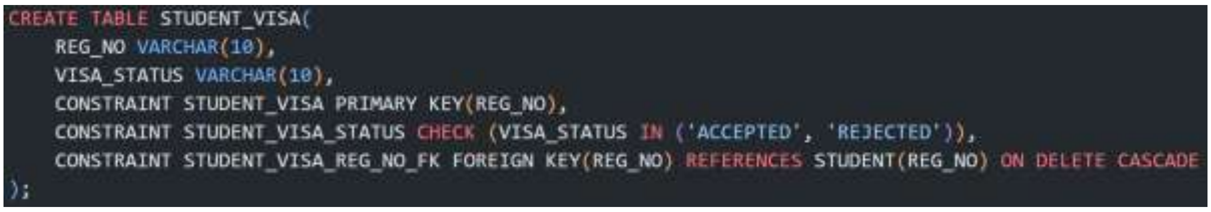
```sql
CREATE TABLE STUDENT(
    REG_NO VARCHAR(10),
    SNAME VARCHAR(50),
    ADDRESS VARCHAR(100),
    DOB DATE,
    EMAIL VARCHAR(50),
    MOBILE NUMBER,
    DEPT_ID VARCHAR(10),
    PROF_ID VARCHAR(10),
    CONSTRAINT STUDENT_REG_NO_PK PRIMARY KEY(REG_NO),
 -- 1. (iv) Email and mobile column in student table should have same characteristics
 -- as those in professor table.
    CONSTRAINT STUDENT_MOBILE_NO_UNIQUE UNIQUE(MOBILE),
    CONSTRAINT STUDENT_EMAIL_UNIQUE UNIQUE(EMAIL),
    CONSTRAINT STUDENT_EMAIL_AT CHECK(EMAIL LIKE '%@%'),
    CONSTRAINT STUDENT_MOBILE_TEN CHECK(LENGTH(MOBILE) = 10),
    CONSTRAINT STUDENT_DEPT_ID_FK FOREIGN KEY(DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID) ON DELETE CASCADE,
    CONSTRAINT STUDENT_PROF_ID_FK FOREIGN KEY(PROF_ID) REFERENCES PROFESSOR(PROF_ID) ON DELETE CASCADE
);
```

ENROLL



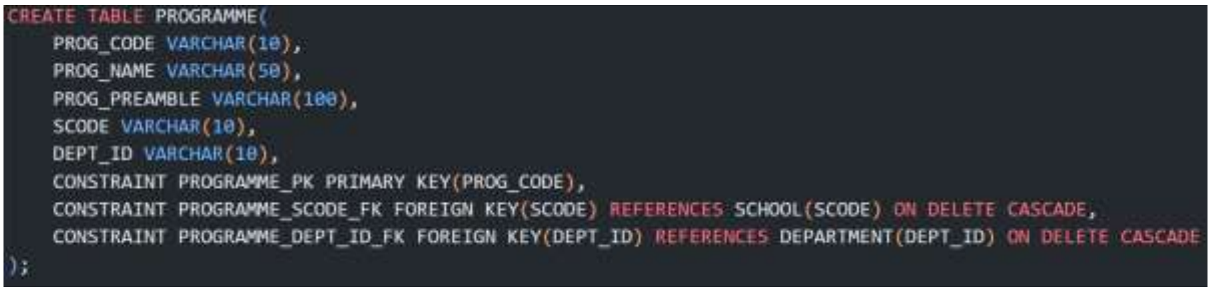
```sql
CREATE TABLE ENROLL(
    CLS_CODE VARCHAR(10),
    REG_NO VARCHAR(10),
    ENROLL_TIME TIMESTAMP,
    GRADE VARCHAR(1),
    CONSTRAINT ENROLL_PK PRIMARY KEY(CLS_CODE, REG_NO),
 -- (v) The enroll_time in the enroll table should be of timestamp data type without
 -- fractional parts of seconds. The grade may assume one of the values in
 -- {'S', 'A', 'B', 'C',  'D',  'C',  'F'}. The grade F indicates Failed.
    CONSTRAINT GRADE_IN CHECK (GRADE IN ('S', 'A', 'B', 'C', 'D', 'E', 'F')),
    CONSTRAINT ENROLL_CLS_CODE_FK FOREIGN KEY(CLS_CODE) REFERENCES CLASS(CLS_CODE) ON DELETE CASCADE,
    CONSTRAINT ENROLL_REG_NO_FK FOREIGN KEY(REG_NO) REFERENCES STUDENT(REG_NO) ON DELETE CASCADE
);
```

STUDENT_VISA



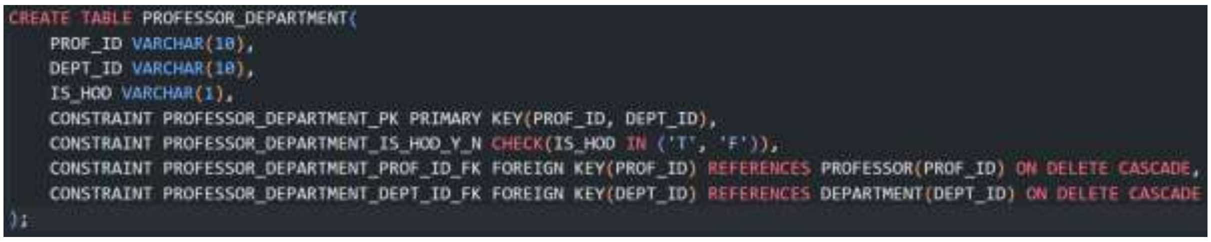
```
CREATE TABLE STUDENT_VISA(
    REG_NO VARCHAR(10),
    VISA_STATUS VARCHAR(10),
    CONSTRAINT STUDENT_VISA PRIMARY KEY(REG_NO),
    CONSTRAINT STUDENT_VISA_STATUS CHECK (VISA_STATUS IN ('ACCEPTED', 'REJECTED')),
    CONSTRAINT STUDENT_VISA_REG_NO_FK FOREIGN KEY(REG_NO) REFERENCES STUDENT(REG_NO) ON DELETE CASCADE
);
```

PROGRAMME



```
CREATE TABLE PROGRAMME(
    PROG_CODE VARCHAR(10),
    PROG_NAME VARCHAR(50),
    PROG_PREAMBLE VARCHAR(100),
    SCODE VARCHAR(10),
    DEPT_ID VARCHAR(10),
    CONSTRAINT PROGRAMME_PK PRIMARY KEY(PROG_CODE),
    CONSTRAINT PROGRAMME_SCODE_FK FOREIGN KEY(SCODE) REFERENCES SCHOOL(SCODE) ON DELETE CASCADE,
    CONSTRAINT PROGRAMME_DEPT_ID_FK FOREIGN KEY(DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID) ON DELETE CASCADE
);
```

PROFESSOR_DEPARTMENT



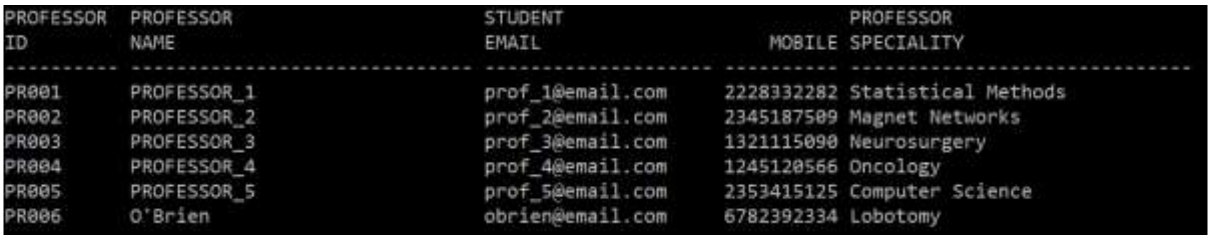
```
CREATE TABLE PROFESSOR_DEPARTMENT(
    PROF_ID VARCHAR(10),
    DEPT_ID VARCHAR(10),
    IS_HOD VARCHAR(1),
    CONSTRAINT PROFESSOR_DEPARTMENT_PK PRIMARY KEY(PROF_ID, DEPT_ID),
    CONSTRAINT PROFESSOR_DEPARTMENT_IS_HOD_Y_N CHECK(IS_HOD IN ('T', 'F')),
    CONSTRAINT PROFESSOR_DEPARTMENT_PROF_ID_FK FOREIGN KEY(PROF_ID) REFERENCES PROFESSOR(PROF_ID) ON DELETE CASCADE,
    CONSTRAINT PROFESSOR_DEPARTMENT_DEPT_ID_FK FOREIGN KEY(DEPT_ID) REFERENCES DEPARTMENT(DEPT_ID) ON DELETE CASCADE
);
```

2. Enter data into the above tables. (Learn also how to enter data interactively.). Display the content of each table. Use column formatting while displaying data.

COLUMN FORMATTING

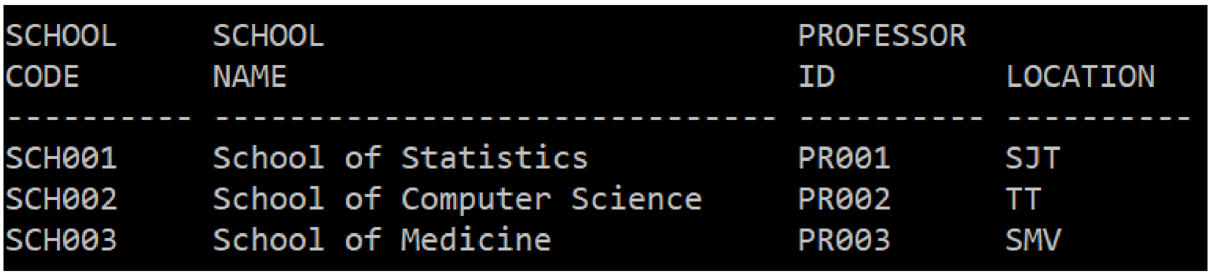
```
SET LINESIZE 1000

COLUMN PROF_ID HEADING 'PROFESSOR|ID' FORMAT A10

COLUMN DEPT_ID HEADING 'DEPARTMENT|ID' FORMAT A10

COLUMN IS_HOD HEADING 'IS HOD' FORMAT A15

COLUMN PROG_CODE HEADING 'PROGRAMME|CODE' FORMAT A10

COLUMN PROG_NAME HEADING 'PROGRAMME|NAME' FORMAT A20

COLUMN PROG_PREAMBLE HEADING 'PROGRAMME|PREAMBLE' FORMAT A50

COLUMN SCODE HEADING 'SCHOOL|CODE' FORMAT A10
COLUMN SNAME HEADING 'STUDENT|NAME' FORMAT A20

COLUMN EMAIL HEADING 'STUDENT|EMAIL' FORMAT A20

COLUMN ADDRESS HEADING 'STUDENT|ADDRESS' FORMAT A30

COLUMN REG_NO HEADING 'STUDENT|REG NO' FORMAT A10

COLUMN PROF_NAME HEADING 'PROFESSOR|NAME' FORMAT A20

COLUMN SPECIALITY HEADING 'PROFESSOR|SPECIALITY' FORMAT A30

COLUMN CLS_CODE HEADING 'CLASS|CODE' FORMAT A10

COLUMN ROOM_NO HEADING 'ROOM NO' FORMAT 9999

COLUMN SLOT FORMAT A10

COLUMN DURATION FORMAT 99

COLUMN STIME HEADING 'STARTING|TIME' FORMAT A10

COLUMN ETIME HEADING 'END|TIME' FORMAT A10

COLUMN CRS_CODE HEADING 'COURSE|CODE' FORMAT A10

COLUMN CRS_NAME HEADING 'COURSE|NAME' FORMAT A20

COLUMN DESCRIPTION HEADING 'COURSE|DESCRIPTION' FORMAT A30

COLUMN PROF_NAME HEADING 'PROFESSOR|NAME' FORMAT A30

COLUMN DNAME HEADING 'DEPARTMENT|NAME' FORMAT A30

COLUMN GRADE FORMAT A10

COLUMN ENROLL_TIME HEADING 'ENROLL|TIME' FORMAT A40

COLUMN SDATE HEADING 'START|DATE'

COLUMN EDATE HEADING 'END|DATE'
COLUMN SEMESTERDURATION HEADING 'SEMESTER|DURATION'

COLUMN CUS_CODE HEADING 'CUSTOMER|CODE' FORMAT 9999

COLUMN CUS_NAME HEADING 'CUSTOMER|NAME' FORMAT A20

COLUMN CUS_ADDRESS HEADING 'CUSTOMER|ADDRESS' FORMAT A50

COLUMN CUS_MOBILE HEADING 'CUSTOMER|MOBILE'
```

<u>PROFESSOR</u>



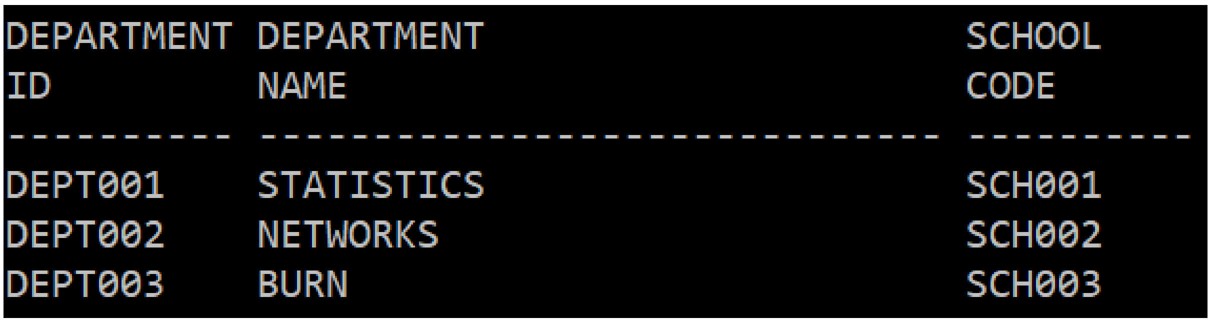
```
PROFESSOR  PROFESSOR                     STUDENT                          PROFESSOR
ID         NAME                          EMAIL                     MOBILE SPECIALITY
---------- ----------------------------- ------------------- ---------- ------------------------------
PR001      PROFESSOR_1                   prof_1@email.com    2228332282 Statistical Methods
PR002      PROFESSOR_2                   prof_2@email.com    2345187509 Magnet Networks
PR003      PROFESSOR_3                   prof_3@email.com    1321115090 Neurosurgery
PR004      PROFESSOR_4                   prof_4@email.com    1245120566 Oncology
PR005      PROFESSOR_5                   prof_5@email.com    2353415125 Computer Science
PR006      O'Brien                       obrien@email.com    6782392334 Lobotomy
```

<u>SCHOOL</u>



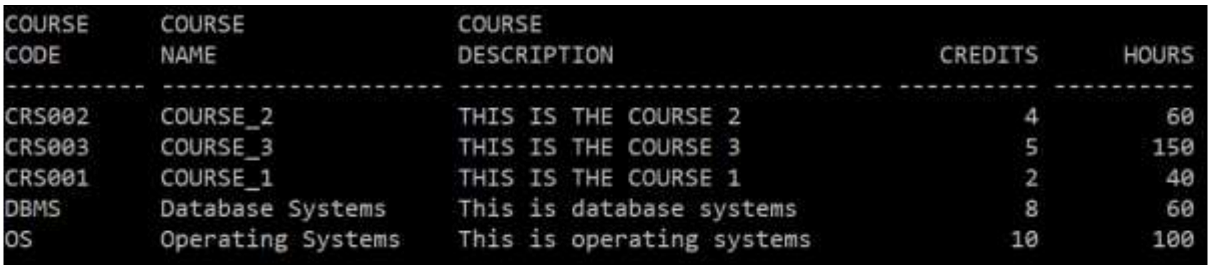
```
SCHOOL     SCHOOL                         PROFESSOR
CODE       NAME                           ID         LOCATION
---------- ------------------------------ ---------- ----------
SCH001     School of Statistics           PR001      SJT
SCH002     School of Computer Science     PR002      TT
SCH003     School of Medicine             PR003      SMV
```

<u>DEPARTMENT</u>



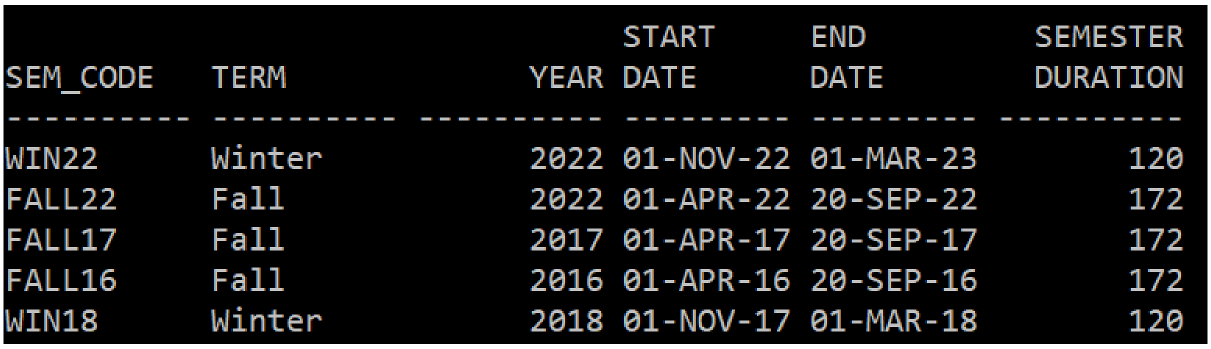
```
DEPARTMENT DEPARTMENT                     SCHOOL
ID         NAME                           CODE
---------- ------------------------------ ----------
DEPT001    STATISTICS                     SCH001
DEPT002    NETWORKS                       SCH002
DEPT003    BURN                           SCH003
```

COURSE



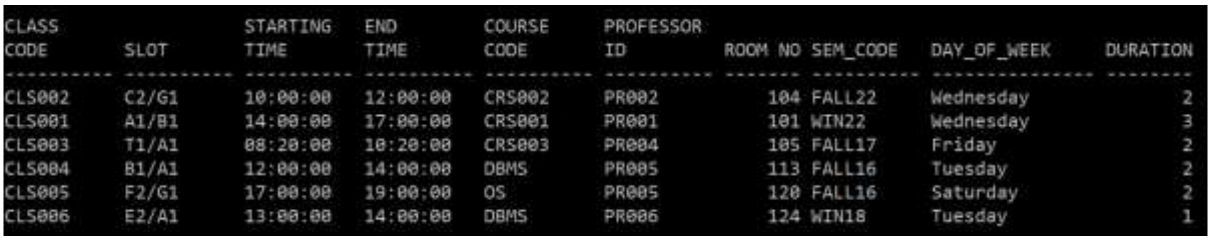
```
COURSE     COURSE               COURSE
CODE       NAME                 DESCRIPTION                     CREDITS      HOURS
---------- -------------------- ------------------------------ ---------- ----------
CRS002     COURSE_2             THIS IS THE COURSE 2                    4         60
CRS003     COURSE_3             THIS IS THE COURSE 3                    5        150
CRS001     COURSE_1             THIS IS THE COURSE 1                    2         40
DBMS       Database Systems     This is database systems                8         60
OS         Operating Systems    This is operating systems              10        100
```

SEMESTER



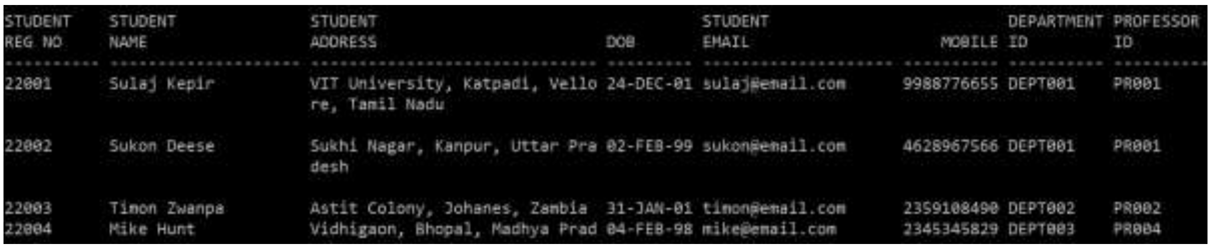
```
                                      START     END          SEMESTER
SEM_CODE   TERM                  YEAR DATE      DATE         DURATION
---------- ---------- ---------- --------- --------- ----------
WIN22      Winter                2022 01-NOV-22 01-MAR-23         120
FALL22     Fall                  2022 01-APR-22 20-SEP-22         172
FALL17     Fall                  2017 01-APR-17 20-SEP-17         172
FALL16     Fall                  2016 01-APR-16 20-SEP-16         172
WIN18      Winter                2018 01-NOV-17 01-MAR-18         120
```

CLASS



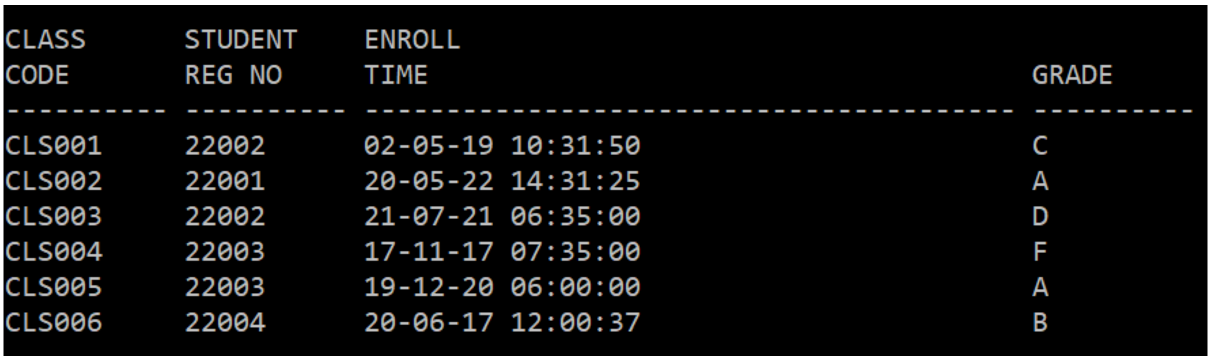
```
CLASS                STARTING   END        COURSE     PROFESSOR
CODE       SLOT      TIME       TIME       CODE       ID         ROOM NO SEM_CODE   DAY_OF_WEEK    DURATION
---------- ---------- ---------- ---------- ---------- ---------- ------- ---------- -------------- --------
CLS002     C2/G1      10:00:00   12:00:00   CRS002     PR002          104 FALL22     Wednesday             2
CLS001     A1/B1      14:00:00   17:00:00   CRS001     PR001          101 WIN22      Wednesday             3
CLS003     T1/A1      08:20:00   10:20:00   CRS003     PR004          105 FALL17     Friday                2
CLS004     B1/A1      12:00:00   14:00:00   DBMS       PR005          113 FALL16     Tuesday               2
CLS005     F2/G1      17:00:00   19:00:00   OS         PR005          120 FALL16     Saturday              2
CLS006     E2/A1      13:00:00   14:00:00   DBMS       PR006          124 WIN18      Tuesday               1
```

STUDENT



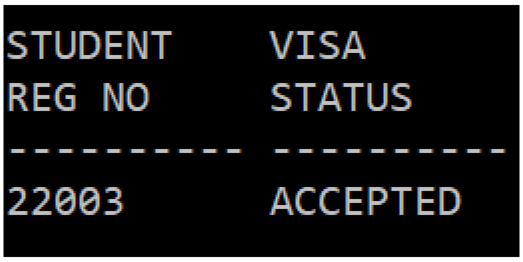
```
STUDENT    STUDENT              STUDENT                                  STUDENT                                DEPARTMENT PROFESSOR
REG NO     NAME                 ADDRESS                        DOB       EMAIL                      MOBILE ID         ID
---------- -------------------- ------------------------------ --------- -------------------- ---------- ---------- ----------
22001      Sulaj Kepir          VIT University, Katpadi, Vello 24-DEC-01 sulaj@email.com      9988776655 DEPT001    PR001
                                re, Tamil Nadu

22002      Sukon Deese          Sukhi Nagar, Kanpur, Uttar Pra 02-FEB-99 sukon@email.com      4628967566 DEPT001    PR001
                                desh

22003      Timon Zwanpa         Astit Colony, Johanes, Zambia  31-JAN-01 timon@email.com      2359108490 DEPT002    PR002
22004      Mike Hunt            Vidhigaon, Bhopal, Madhya Prad 04-FEB-98 mike@email.com       2345345829 DEPT003    PR004
```

ENROLL



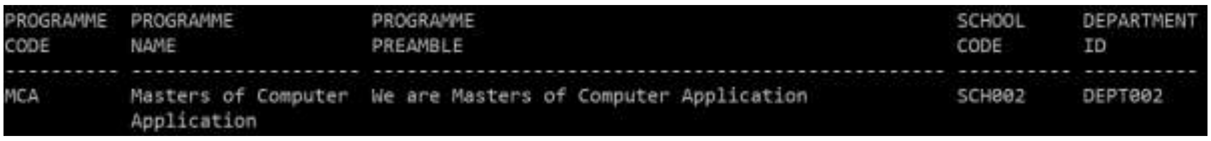
```
CLASS      STUDENT    ENROLL
CODE       REG NO     TIME                                       GRADE
---------- ---------- ------------------------------------------ ----------
CLS001     22002      02-05-19 10:31:50                          C
CLS002     22001      20-05-22 14:31:25                          A
CLS003     22002      21-07-21 06:35:00                          D
CLS004     22003      17-11-17 07:35:00                          F
CLS005     22003      19-12-20 06:00:00                          A
CLS006     22004      20-06-17 12:00:37                          B
```

STUDENT_VISA



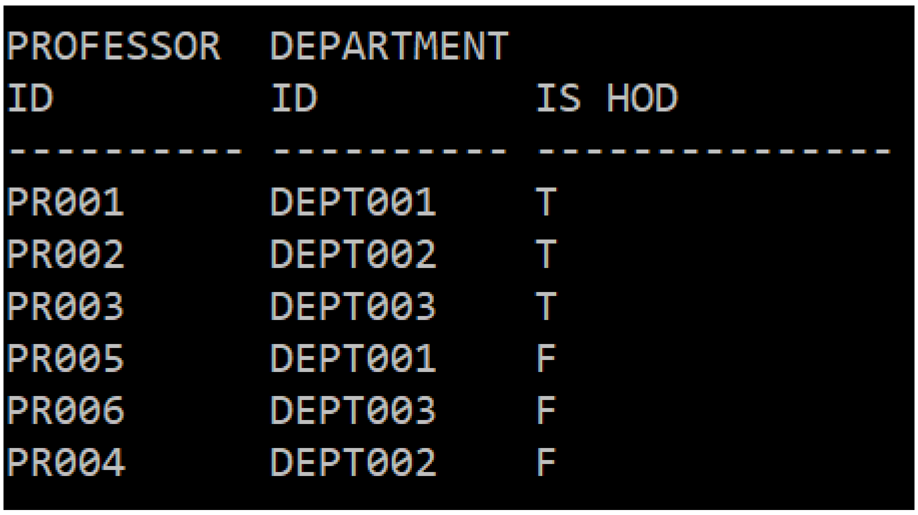
```
STUDENT    VISA
REG NO     STATUS
---------- ----------
22003      ACCEPTED
```

PROGRAMME



```
PROGRAMME  PROGRAMME            PROGRAMME                                           SCHOOL     DEPARTMENT
CODE       NAME                 PREAMBLE                                            CODE       ID
---------- -------------------- --------------------------------------------------- ---------- ----------
MCA        Masters of Computer  We are Masters of Computer Application              SCH002     DEPT002
           Application
```

PROFESSOR_DEPARTMENT



```
PROFESSOR  DEPARTMENT
ID         ID         IS HOD
---------- ---------- ---------------
PR001      DEPT001    T
PR002      DEPT002    T
PR003      DEPT003    T
PR005      DEPT001    F
PR006      DEPT003    F
PR004      DEPT002    F
```

3. Queries

```
-- 3. (i) Display name, email address and
-- address for those students who live in
-- Katpadi area and whose name has an l as
-- the third character
SELECT
    SNAME,
    EMAIL,
    ADDRESS
FROM
    STUDENT
WHERE
    ADDRESS LIKE '%Katpadi%'
    AND SNAME LIKE '__l%';
```

```
STUDENT              STUDENT              STUDENT
NAME                 EMAIL                ADDRESS
==================== ==================== ==============================
Sulaj Kepir          sulaj@email.com      VIT University, Katpadi, Vello
                                          re, Tamil Nadu
```

```sql
-- 3. (ii) Display name, email address and
-- address for those students who are not
-- from Tamil Nadu.
SELECT
    SNAME,
    EMAIL,
    ADDRESS
FROM
    STUDENT
WHERE
    ADDRESS NOT LIKE '%Tamil Nadu%';
```

```
STUDENT              STUDENT              STUDENT
NAME                 EMAIL                ADDRESS
==================== ==================== ==============================
Sukon Deese          sukon@email.com      Sukhi Nagar, Kanpur, Uttar Pra
                                          desh

Timon Zwanpa         timon@email.com      Astit Colony, Johanes, Zambia
Mike Hunt            mike@email.com       Vidhigaon, Bhopal, Madhya Prad
                                          esh
```

```sql
-- 3. (iii) Display name, email address
-- and address of foreign students only.
SELECT
    STUDENT.SNAME,
    STUDENT.EMAIL,
    STUDENT.ADDRESS
FROM
    STUDENT
    INNER JOIN STUDENT_VISA
    ON STUDENT.REG_NO = STUDENT_VISA.REG_NO;
```

```
STUDENT              STUDENT              STUDENT
NAME                 EMAIL                ADDRESS
==================== ==================== ==============================
Timon Zwanpa         timon@email.com      Astit Colony, Johanes, Zambia
```

```sql
-- 3. (iv) List the name of professors
-- along with their specialty who belong
-- to School of Medicine.

SELECT
    PROFESSOR.PROF_NAME,
    PROFESSOR.SPECIALITY
FROM
    PROFESSOR
    INNER JOIN SCHOOL
    ON SCHOOL.SCL_NAME = 'School of Medicine'
    AND SCHOOL.PROF_ID = PROFESSOR.PROF_ID;
```

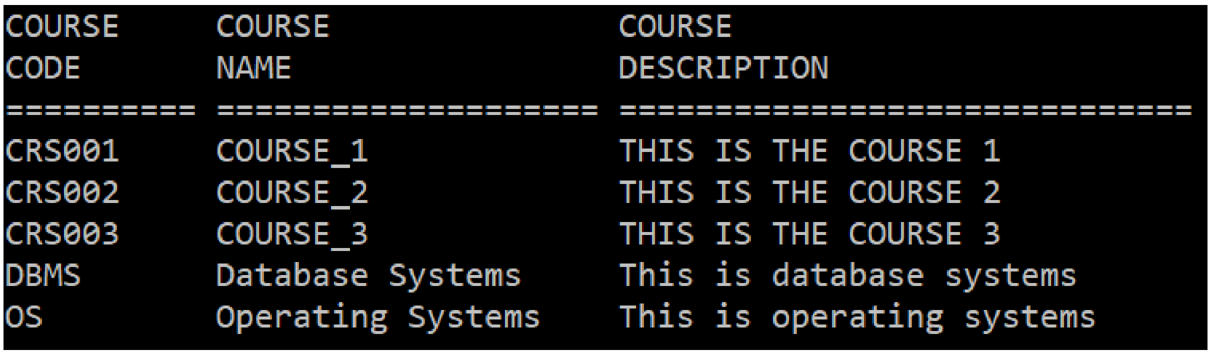
```
PROFESSOR                         PROFESSOR
NAME                              SPECIALITY
================================= =================================
PROFESSOR_3                       Neurosurgery
```

```sql
-- 3. (v) Display name of the school and
-- name of professor who chairs the school.

SELECT
    SCHOOL.SCL_NAME,
    PROFESSOR.PROF_NAME
FROM
    PROFESSOR
    INNER JOIN SCHOOL
    ON SCHOOL.PROF_ID = PROFESSOR.PROF_ID;
```

```
SCHOOL                            PROFESSOR
NAME                              NAME
================================= =================================
School of Statistics              PROFESSOR_1
School of Computer Science        PROFESSOR_2
School of Medicine                PROFESSOR_3
```

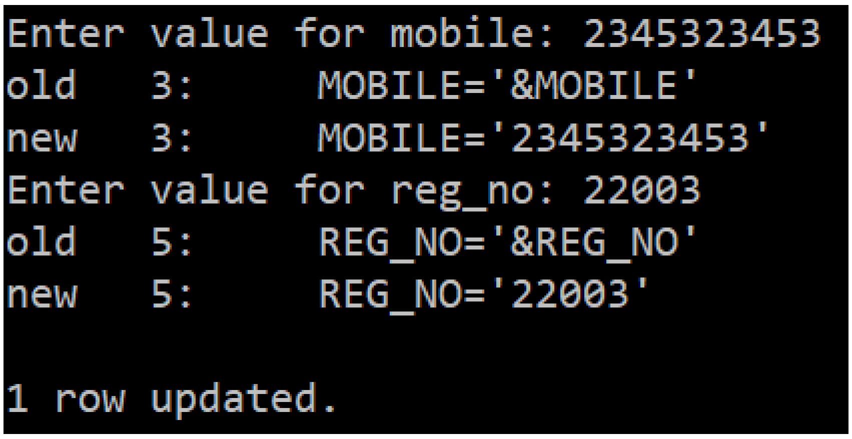
```sql
-- 3. (vi) List course code, course name and
-- course description in alphabetic order of
-- course code.

SELECT
```

```sql
    CRS_CODE,
    CRS_NAME,
    DESCRIPTION
FROM
    COURSE
ORDER BY
    CRS_CODE;
```



```
COURSE     COURSE               COURSE
CODE       NAME                 DESCRIPTION
========== ==================== ==============================
CRS001     COURSE_1             THIS IS THE COURSE 1
CRS002     COURSE_2             THIS IS THE COURSE 2
CRS003     COURSE_3             THIS IS THE COURSE 3
DBMS       Database Systems     This is database systems
OS         Operating Systems    This is operating systems
```

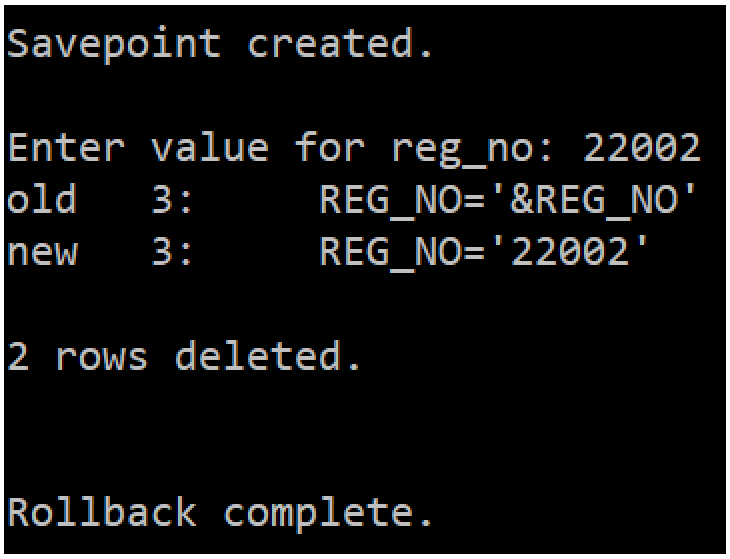
```sql
-- 3. (vii) Change the mobile number of a student
-- interactively.
UPDATE STUDENT
SET
    MOBILE='&MOBILE'
WHERE
    REG_NO='&REG_NO';
```



```
Enter value for mobile: 2345323453
old   3:      MOBILE='&MOBILE'
new   3:      MOBILE='2345323453'
Enter value for reg_no: 22003
old   5:      REG_NO='&REG_NO'
new   5:      REG_NO='22003'

1 row updated.
```

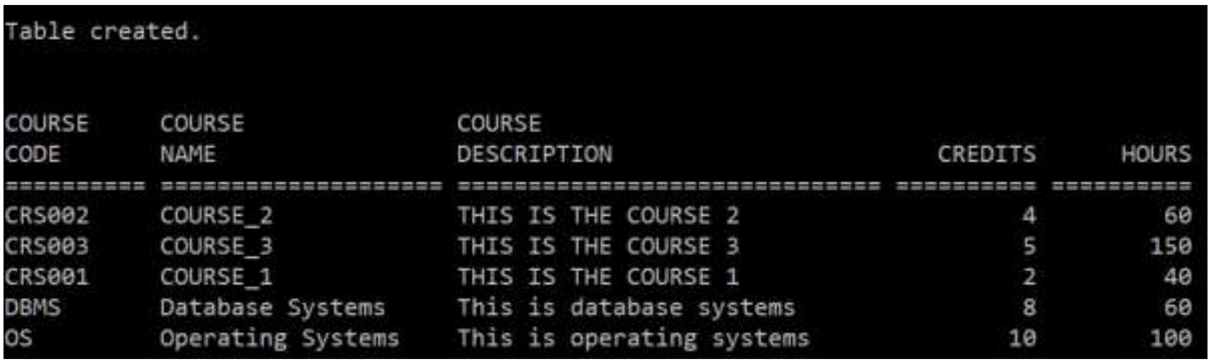
```sql
-- 3. (viii) Remove enrollment information of a
-- student from a particular course interactively.
-- How would you recover the data?
-- By creating a savepoint and rollbacking to it.
SAVEPOINT BEFORE_VIII;
```

```sql
DELETE FROM ENROLL
WHERE
    REG_NO='&REG_NO';

ROLLBACK TO BEFORE_VIII;
```



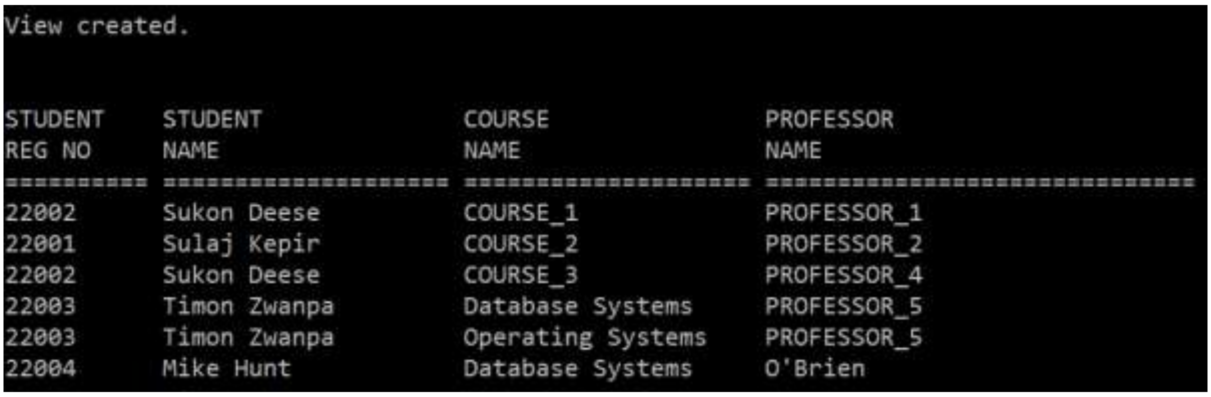
```
Savepoint created.

Enter value for reg_no: 22002
old   3:      REG_NO='&REG_NO'
new   3:      REG_NO='22002'

2 rows deleted.


Rollback complete.
```

```sql
-- (ix) Create a duplicate of course table

CREATE TABLE COURSE_DUPLICATE AS
    SELECT
        *
    FROM
        COURSE;

SELECT
    *
FROM
    COURSE_DUPLICATE;
```



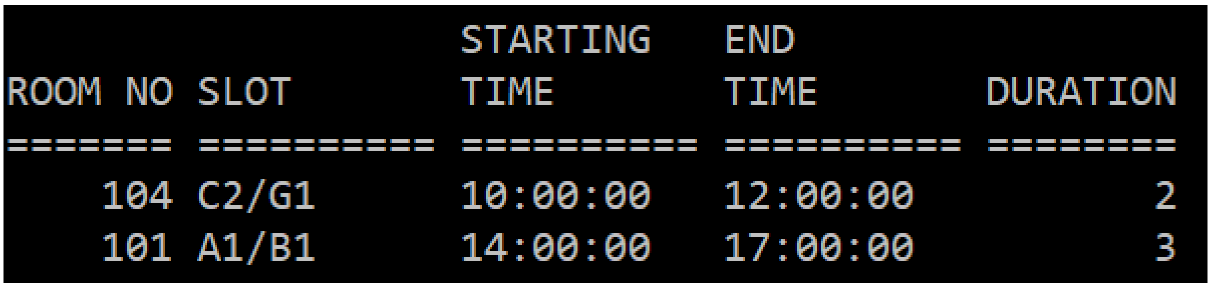
```
Table created.


COURSE     COURSE               COURSE
CODE       NAME                 DESCRIPTION                       CREDITS      HOURS
========== ==================== ============================== ========== ==========
CRS002     COURSE_2             THIS IS THE COURSE 2                    4         60
CRS003     COURSE_3             THIS IS THE COURSE 3                    5        150
CRS001     COURSE_1             THIS IS THE COURSE 1                    2         40
DBMS       Database Systems     This is database systems                8         60
OS         Operating Systems    This is operating systems              10        100
```

```sql
-- (x)  Create a view for list of students
-- (Reg_no, Sname) and the courses they
-- have registered along with name of
-- professors teaching the course
CREATE VIEW STUDENT_COURSE_VIEW AS
    SELECT
        STUDENT.REG_NO,
        STUDENT.SNAME,
        COURSE.CRS_NAME,
        PROFESSOR.PROF_NAME
    FROM
        STUDENT
        INNER JOIN ENROLL
        ON ENROLL.REG_NO = STUDENT.REG_NO
        INNER JOIN CLASS
        ON ENROLL.CLS_CODE = CLASS.CLS_CODE
        INNER JOIN COURSE
        ON COURSE.CRS_CODE = CLASS.CRS_CODE
        INNER JOIN PROFESSOR
        ON PROFESSOR.PROF_ID = CLASS.PROF_ID;

SELECT
    *
FROM
    STUDENT_COURSE_VIEW;
```



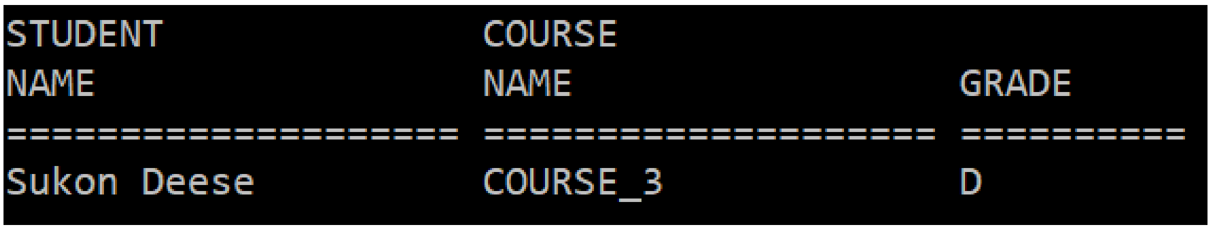
```
View created.


STUDENT    STUDENT                COURSE                 PROFESSOR
REG NO     NAME                   NAME                   NAME
========== ====================== ====================== ================================
22002      Sukon Deese            COURSE_1               PROFESSOR_1
22001      Sulaj Kepir            COURSE_2               PROFESSOR_2
22002      Sukon Deese            COURSE_3               PROFESSOR_4
22003      Timon Zwanpa           Database Systems       PROFESSOR_5
22003      Timon Zwanpa           Operating Systems      PROFESSOR_5
22004      Mike Hunt              Database Systems       O'Brien
```

```sql
-- (xi) List the room number, slot, start time,
-- end time and duration of every class held on
-- Wednesdays in descending order of room number
SELECT
    ROOM_NO,
    SLOT,
    STIME,
    ETIME,
    EXTRACT (HOUR
FROM
    ETIME - STIME) AS "DURATION"
FROM
    CLASS
WHERE
    DAY_OF_WEEK = 'Wednesday'
ORDER BY
    ROOM_NO DESC;
```



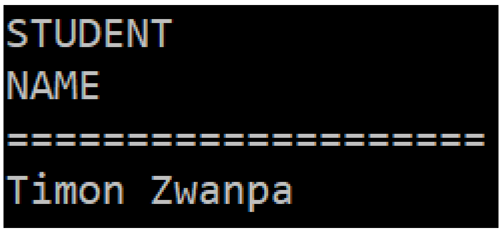
```
                          STARTING    END
ROOM NO SLOT              TIME        TIME        DURATION
======= =========== =========== =========== ========
    104 C2/G1       10:00:00    12:00:00           2
    101 A1/B1       14:00:00    17:00:00           3
```

```
--3. (xii) Display the name and grade of a
-- student in different courses underwent
-- in fall semester 2017 - 18 (Fall 2017)
SELECT
    STUDENT.SNAME,
    COURSE.CRS_NAME,
    ENROLL.GRADE
FROM
    STUDENT
    INNER JOIN ENROLL
    ON ENROLL.REG_NO = STUDENT.REG_NO
    INNER JOIN CLASS
    ON CLASS.CLS_CODE = ENROLL.CLS_CODE
    INNER JOIN COURSE
    ON COURSE.CRS_CODE = CLASS.CRS_CODE
    INNER JOIN SEMESTER
    ON SEMESTER.SEM_CODE = CLASS.SEM_CODE
    AND SEMESTER.TERM = 'Fall'
    AND SEMESTER.YEAR = 2017;
```



```
STUDENT               COURSE
NAME                  NAME                  GRADE
===================== ===================== ==========
Sukon Deese           COURSE_3              D
```

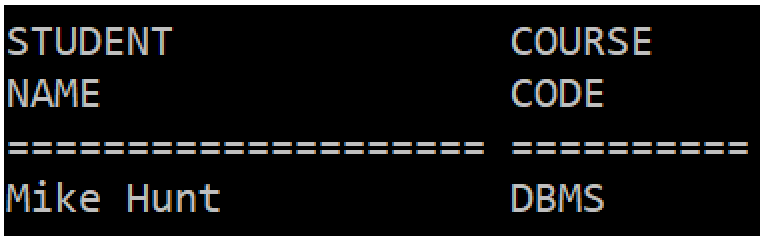
```sql
-- 3. (xiii) Find out name of students who have
-- taken Database Systems course as well as
-- Operating Systems course in fall semester
-- 2016 - 17 (Fall 2016)
SELECT
    STUDENT.SNAME
FROM
    STUDENT
    INNER JOIN ENROLL
    ON ENROLL.REG_NO = STUDENT.REG_NO
    INNER JOIN CLASS
    ON CLASS.CLS_CODE = ENROLL.CLS_CODE
    INNER JOIN COURSE
    ON CLASS.CRS_CODE = COURSE.CRS_CODE
    AND COURSE.CRS_CODE IN ('OS',
    'DBMS')
    INNER JOIN SEMESTER
    ON SEMESTER.SEM_CODE = CLASS.SEM_CODE
    AND SEMESTER.TERM = 'Fall'
    AND SEMESTER.YEAR = 2016
GROUP BY
    STUDENT.SNAME
HAVING
    COUNT(DISTINCT COURSE.CRS_CODE) = 2;
```



```
STUDENT
NAME
=====================
Timon Zwanpa
```

```sql
-- 3. (xiv) Find out name of students who have
-- taken Database Systems course but have not
-- taken Operating Systems course in winter
-- semester 2017 - 18 (Winter 2018).
SELECT
```

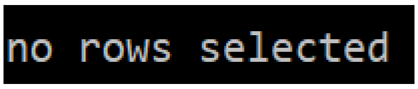
```
    STUDENT.SNAME,
    COURSE.CRS_CODE
FROM
    STUDENT
    INNER JOIN ENROLL
    ON ENROLL.REG_NO = STUDENT.REG_NO
    INNER JOIN CLASS
    ON ENROLL.CLS_CODE = CLASS.CLS_CODE
    INNER JOIN COURSE
    ON COURSE.CRS_CODE = CLASS.CRS_CODE
    AND COURSE.CRS_CODE IN ('DBMS',
    'OS')
    INNER JOIN SEMESTER
    ON SEMESTER.SEM_CODE = CLASS.SEM_CODE
    AND SEMESTER.YEAR = 2018
    AND SEMESTER.TERM = 'Winter' MINUS
    SELECT
        STUDENT.SNAME,
        COURSE.CRS_CODE
    FROM
        STUDENT
        INNER JOIN ENROLL
        ON ENROLL.REG_NO = STUDENT.REG_NO
        INNER JOIN CLASS
        ON ENROLL.CLS_CODE = CLASS.CLS_CODE
        INNER JOIN COURSE
        ON COURSE.CRS_CODE = CLASS.CRS_CODE
        AND COURSE.CRS_CODE = 'OS'
        INNER JOIN SEMESTER
        ON SEMESTER.SEM_CODE = CLASS.SEM_CODE
        AND SEMESTER.YEAR = 2018
        AND SEMESTER.TERM = 'Winter';
```



```
STUDENT                      COURSE
NAME                         CODE
============================ ==========
Mike Hunt                    DBMS
```

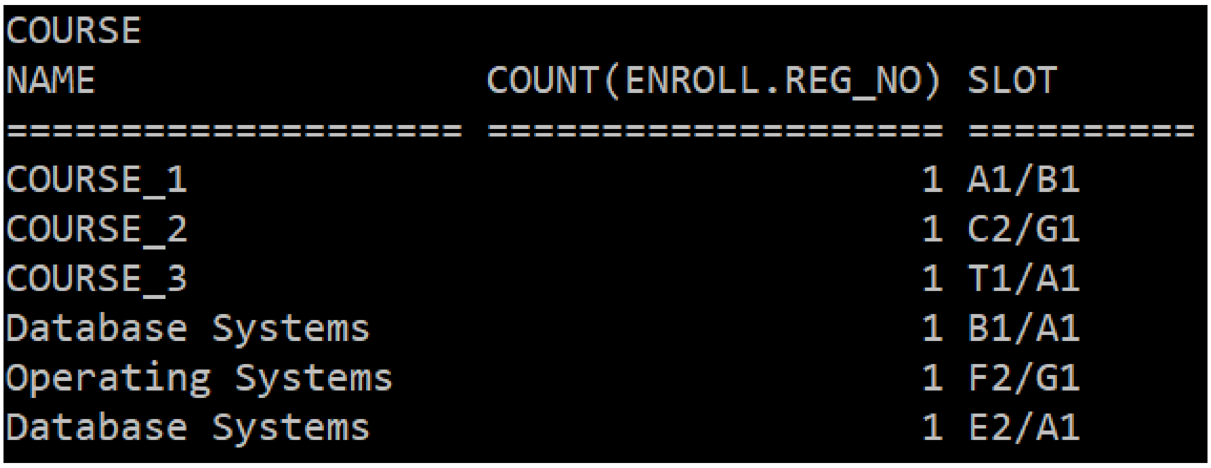
```
-- 3. (xv) List the registration number and name of
-- the students who have registered for maximum
-- number of credits in Winter 17-18 (Winter 2018)
```

```sql
-- semester. (Assume that the maximum number of
-- credits = 26)
SELECT
    STUDENT.REG_NO,
    STUDENT.SNAME
FROM
    STUDENT,
    ENROLL,
    CLASS,
    COURSE
WHERE
    STUDENT.REG_NO = ENROLL.REG_NO
    AND CLASS.CLS_CODE = ENROLL.CLS_CODE
    AND CLASS.CRS_CODE = COURSE.CRS_CODE
GROUP BY
    STUDENT.REG_NO,
    STUDENT.SNAME
HAVING
    SUM(COURSE.CREDITS) = 26;
```



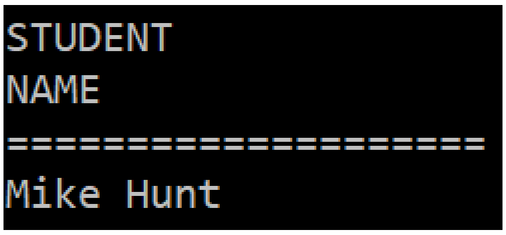
```
no rows selected
```

```sql
-- 3. (xvi) List the name of the course and the number
-- of students registered in each slot for course under
-- different faculty members.
SELECT
    COURSE.CRS_NAME,
    COUNT(ENROLL.REG_NO),
    SLOT
FROM
    COURSE,
    ENROLL,
    CLASS
WHERE
    ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND CLASS.CRS_CODE = COURSE.CRS_CODE
GROUP BY
    ENROLL.REG_NO,
    COURSE.CRS_NAME,
    SLOT;
```



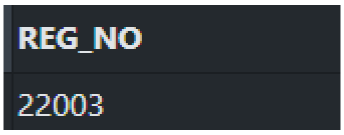
```
COURSE
NAME                  COUNT(ENROLL.REG_NO) SLOT
===================== ==================== ==========
COURSE_1                                 1 A1/B1
COURSE_2                                 1 C2/G1
COURSE_3                                 1 T1/A1
Database Systems                         1 B1/A1
Operating Systems                        1 F2/G1
Database Systems                         1 E2/A1
```

```sql
-- 3. (xvii) Find out the name of the students who have
-- registered in all the courses being taught by
-- Prof. O'Brien in Winter 17-18 (Winter 2018).
SELECT
    STUDENT.SNAME
FROM
    STUDENT,
    PROFESSOR,
    CLASS,
    ENROLL
WHERE
    ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND ENROLL.REG_NO = STUDENT.REG_NO
    AND CLASS.PROF_ID = PROFESSOR.PROF_ID
    AND PROFESSOR.PROF_NAME = 'O''Brien';
```



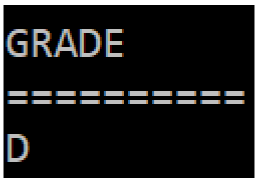
```
STUDENT
NAME
=====================
Mike Hunt
```

```sql
-- only works on vscode
-- 3. (xviii) List the registration number of the students
-- who registered in Database Systems course on
-- November 17, 2017
SELECT
    STUDENT.REG_NO
FROM
    STUDENT,
    ENROLL,
    CLASS
WHERE
    ENROLL.REG_NO = STUDENT.REG_NO
    AND ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND ENROLL.ENROLL_TIME >= '17-NOV-2017'
    AND ENROLL.ENROLL_TIME < '18-NOV-2017'
    AND CLASS.CRS_CODE = 'DBMS';
```



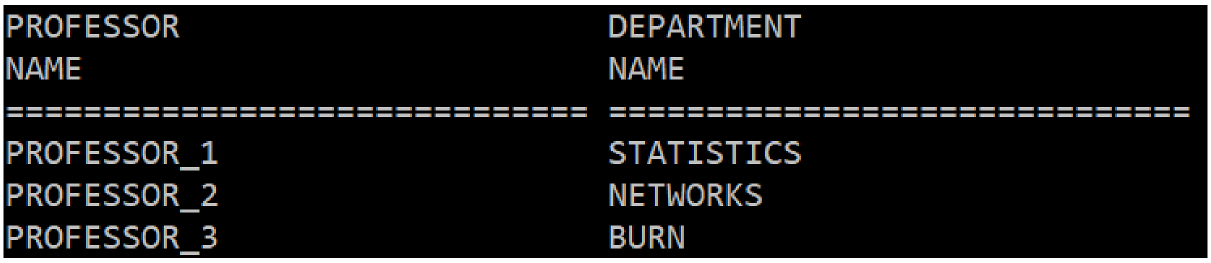
REG_NO
22003

```sql
-- 3. (xix) Write a query to display the grade of a student
-- given his/her registration number and the course name
-- for Fall semester 17-18 (Fall 2017).
SELECT
    ENROLL.GRADE
FROM
    ENROLL,
    CLASS
WHERE
    ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND CLASS.SEM_CODE = 'FALL17'
    AND ENROLL.REG_NO = '22002'
    AND CLASS.CRS_CODE = 'CRS003';
```



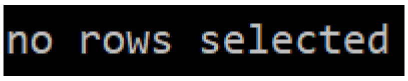
```
GRADE
==========
D
```

```sql
-- 3. (xx) List the name of departments and the name
-- professors who is in charge of the department.
SELECT
    PROFESSOR.PROF_NAME,
    DEPARTMENT.DNAME
FROM
    PROFESSOR,
    DEPARTMENT,
    PROFESSOR_DEPARTMENT
WHERE
    PROFESSOR_DEPARTMENT.PROF_ID = PROFESSOR.PROF_ID
    AND PROFESSOR_DEPARTMENT.DEPT_ID = DEPARTMENT.DEPT_ID
    AND PROFESSOR_DEPARTMENT.IS_HOD = 'T';
```



```
PROFESSOR                      DEPARTMENT
NAME                           NAME
============================== ==============================
PROFESSOR_1                    STATISTICS
PROFESSOR_2                    NETWORKS
PROFESSOR_3                    BURN
```

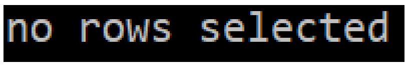
```sql
-- 3. (xxi) List the name of schools with students
-- strength higher than 7000.
SELECT
    SCHOOL.SCL_NAME
FROM
    SCHOOL,
    STUDENT,
    DEPARTMENT
WHERE
    STUDENT.DEPT_ID = DEPARTMENT.DEPT_ID
    AND DEPARTMENT.SCODE = SCHOOL.SCODE
GROUP BY
    SCHOOL.SCL_NAME
HAVING
    COUNT(STUDENT.REG_NO) > 7000;
```



```
no rows selected
```

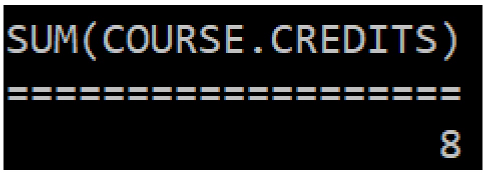
```sql
-- 3. (xxii) List the name of the department(s) under
-- school of medicine with student strength higher than the
-- average students of all the departments in the school.
SELECT
    DEPARTMENT.DNAME
FROM
    DEPARTMENT,
    SCHOOL,
    STUDENT
WHERE
    STUDENT.DEPT_ID = DEPARTMENT.DEPT_ID
    AND DEPARTMENT.SCODE = SCHOOL.SCODE
    AND SCHOOL.SCL_NAME = 'School of Medicine'
GROUP BY
```

```
    DEPARTMENT.DNAME
HAVING
    COUNT(STUDENT.REG_NO) > (
        SELECT
            COUNT(DISTINCT STUDENT.REG_NO) / COUNT(DISTINCT
DEPARTMENT.DEPT_ID)
        FROM
            STUDENT,
            DEPARTMENT
    );
```



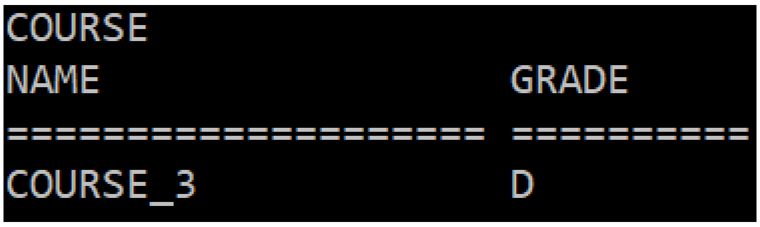
```
no rows selected
```

```
-- 3. (xxiii) Given the registration number of a student,
-- display the total credits registered by him/her in
-- Winter 17-18 (Winter 2018).
SELECT
    SUM(COURSE.CREDITS)
FROM
    ENROLL,
    CLASS,
    COURSE,
    STUDENT
WHERE
    ENROLL.REG_NO = 22004
    AND STUDENT.REG_NO = ENROLL.REG_NO
    AND ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND CLASS.CRS_CODE = COURSE.CRS_CODE
    AND CLASS.SEM_CODE = 'WIN18'
GROUP BY
    STUDENT.REG_NO;
```



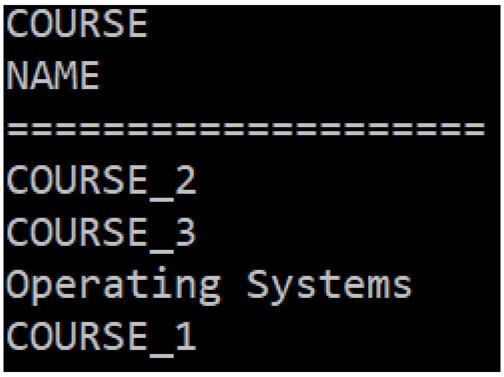
```
SUM(COURSE.CREDITS)
===================
                  8
```

```
-- 3. (xxiv) Given the registration number of a student,
-- display her/his grade in the course she/he registered
-- in Fall 17-18 (Fall 2017).
SELECT
    COURSE.CRS_NAME,
    ENROLL.GRADE
FROM
    ENROLL,
    COURSE,
    STUDENT,
    CLASS
WHERE
    ENROLL.REG_NO = STUDENT.REG_NO
    AND STUDENT.REG_NO = '22002'
    AND ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND CLASS.CRS_CODE = COURSE.CRS_CODE
    AND CLASS.SEM_CODE = 'FALL17';
```



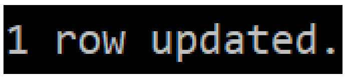
```
COURSE
NAME                        GRADE
=========================== ==========
COURSE_3                    D
```

```
-- 3. (xxv) Display the name of the courses that are not
-- being offered in Winter 17-18 (Winter 2018).
SELECT
    COURSE.CRS_NAME
FROM
    COURSE
WHERE
    COURSE.CRS_CODE NOT IN(
        SELECT
            COURSE.CRS_CODE
        FROM
            CLASS,
            COURSE
        WHERE
            CLASS.SEM_CODE = 'WIN18'
            AND CLASS.CRS_CODE = COURSE.CRS_CODE
    );
```



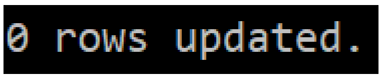
```
COURSE
NAME
====================
COURSE_2
COURSE_3
Operating Systems
COURSE_1
```

```sql
-- 3. (xxvi) Write necessary SQL statement to advance the
-- start time and end time of every class by ten minutes
-- in Fall 18-19 (Fall 2017)

UPDATE CLASS
SET
    STIME = STIME + INTERVAL '10' MINUTE,
    ETIME = ETIME + INTERVAL '10' MINUTE
WHERE
    SEM_CODE = 'FALL17';
```



```
1 row updated.
```

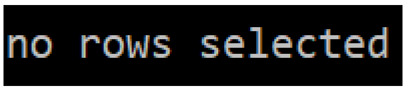
```sql
-- 3. (xxvii) Write necessary SQL statement
-- to advance the start date and end date of
-- Fall 18-19 semester by one week with
-- respect to Fall semester of 17-18(Fall 2017)
UPDATE CLASS
SET
    STIME = STIME + INTERVAL '7' DAY,
    ETIME = ETIME + INTERVAL '7' DAY
WHERE
    SEM_CODE = 'FALL18';
```



```
0 rows updated.
```

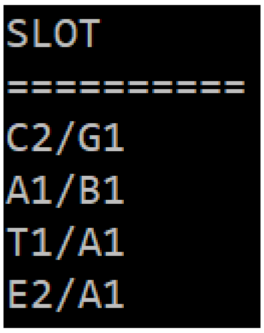
```sql
-- 3. (xxviii) Find out the name list of
-- students who had secured ‘S’ grade in
-- at least 50% of the courses cleared by
-- her/him.
SELECT
    STUDENT.SNAME
```

```sql
FROM
    STUDENT,
    ENROLL
WHERE
    STUDENT.REG_NO = ENROLL.REG_NO
    AND ENROLL.GRADE = 'S'
GROUP BY
    STUDENT.SNAME
HAVING
    COUNT(GRADE) >= 0.5 * (
        SELECT
            COUNT(GRADE)
        FROM
            STUDENT,
            ENROLL
        WHERE
            STUDENT.REG_NO = ENROLL.REG_NO
            AND ENROLL.GRADE IN ('S',
            'A',
            'B',
            'C',
            'D',
            'E')
        GROUP BY
            GRADE
    );
```



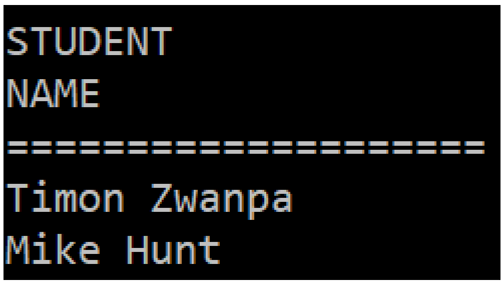
```
no rows selected
```

```sql
-- 3. (xxix) Given the registration number
-- of a student, find out his/her free slots.
SELECT
    CLASS.SLOT
FROM
    ENROLL,
    CLASS
WHERE
    ENROLL.CLS_CODE = CLASS.CLS_CODE
    AND CLASS.SLOT NOT IN (
        SELECT
            CLASS.SLOT
        FROM
            ENROLL,
            CLASS
        WHERE
            ENROLL.CLS_CODE = CLASS.CLS_CODE
            AND ENROLL.REG_NO = '22003'
    );
```



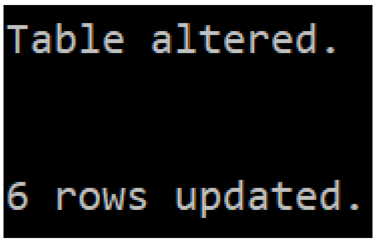
```
SLOT
==========
C2/G1
A1/B1
T1/A1
E2/A1
```

```sql
-- 3. (xxx) Find out the name list of students
-- who have classes in the afternoon session
-- only on a specific day of the week.
SELECT
    STUDENT.SNAME
FROM
    STUDENT,
    ENROLL,
    CLASS
WHERE
    ENROLL.REG_NO = STUDENT.REG_NO
    AND CLASS.CLS_CODE = ENROLL.CLS_CODE
    AND CLASS.DAY_OF_WEEK = 'Tuesday'
    AND EXTRACT(HOUR FROM CLASS.STIME) >= 12;
```



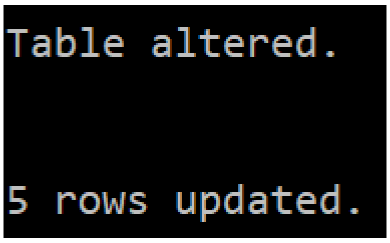
```
STUDENT
NAME
====================
Timon Zwanpa
Mike Hunt
```

```sql
-- 3. (xxxi) Add a column named ‘Duration’
-- (to indicate duration of a class) with
-- appropriate data type to the CLASS
-- table and populate the column from values
-- of start time and end time columns.
ALTER TABLE CLASS ADD DURATION NUMBER;

UPDATE CLASS
SET
    DURATION = EXTRACT(
        HOUR FROM ETIME - STIME
    );
```



```
Table altered.


6 rows updated.
```

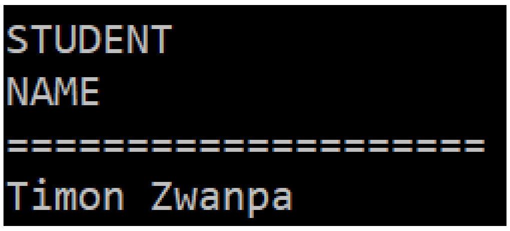
```sql
-- 3. (xxxii) Add a column named
-- 'SemesterDuration' (indicating duration
-- of a semester) with appropriate data type
-- to the SEMESTER table and populate the
-- column from values of start date and end
-- date columns.
ALTER TABLE SEMESTER ADD SEMESTERDURATION NUMBER;

UPDATE SEMESTER
SET
    SEMESTERDURATION = EDATE - SDATE;
```



```
Table altered.


5 rows updated.
```

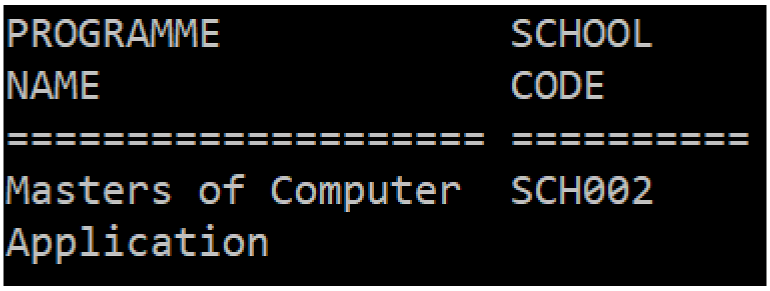
```sql
-- 3. (xxxiii) Find out the list of students who
-- are undergoing MCA program.
SELECT
    STUDENT.SNAME
FROM
    STUDENT,
    PROGRAMME
```

```sql
WHERE
    STUDENT.DEPT_ID = PROGRAMME.DEPT_ID
    AND PROGRAMME.PROG_CODE = 'MCA';
```



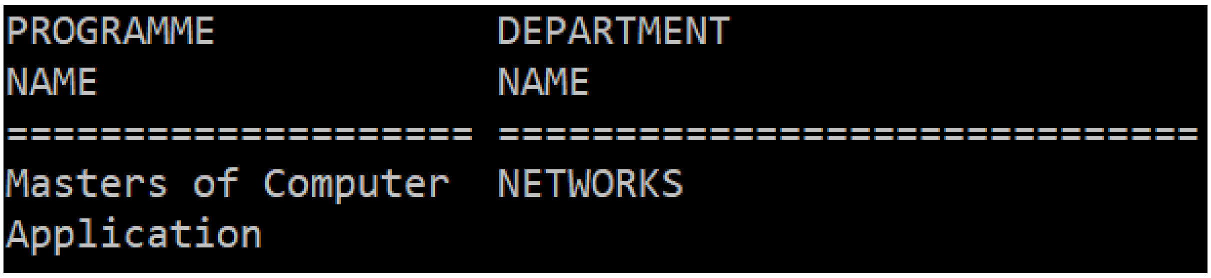
```
STUDENT
NAME
=====================
Timon Zwanpa
```

```sql
-- 3. (xxxiv) Display the name of programs and the
-- name of school offering the program
SELECT
    PROGRAMME.PROG_NAME,
    SCHOOL.SCODE
FROM
    PROGRAMME,
    SCHOOL
WHERE
    PROGRAMME.SCODE = SCHOOL.SCODE;
```



```
PROGRAMME             SCHOOL
NAME                  CODE
===================== ==========
Masters of Computer   SCH002
Application
```

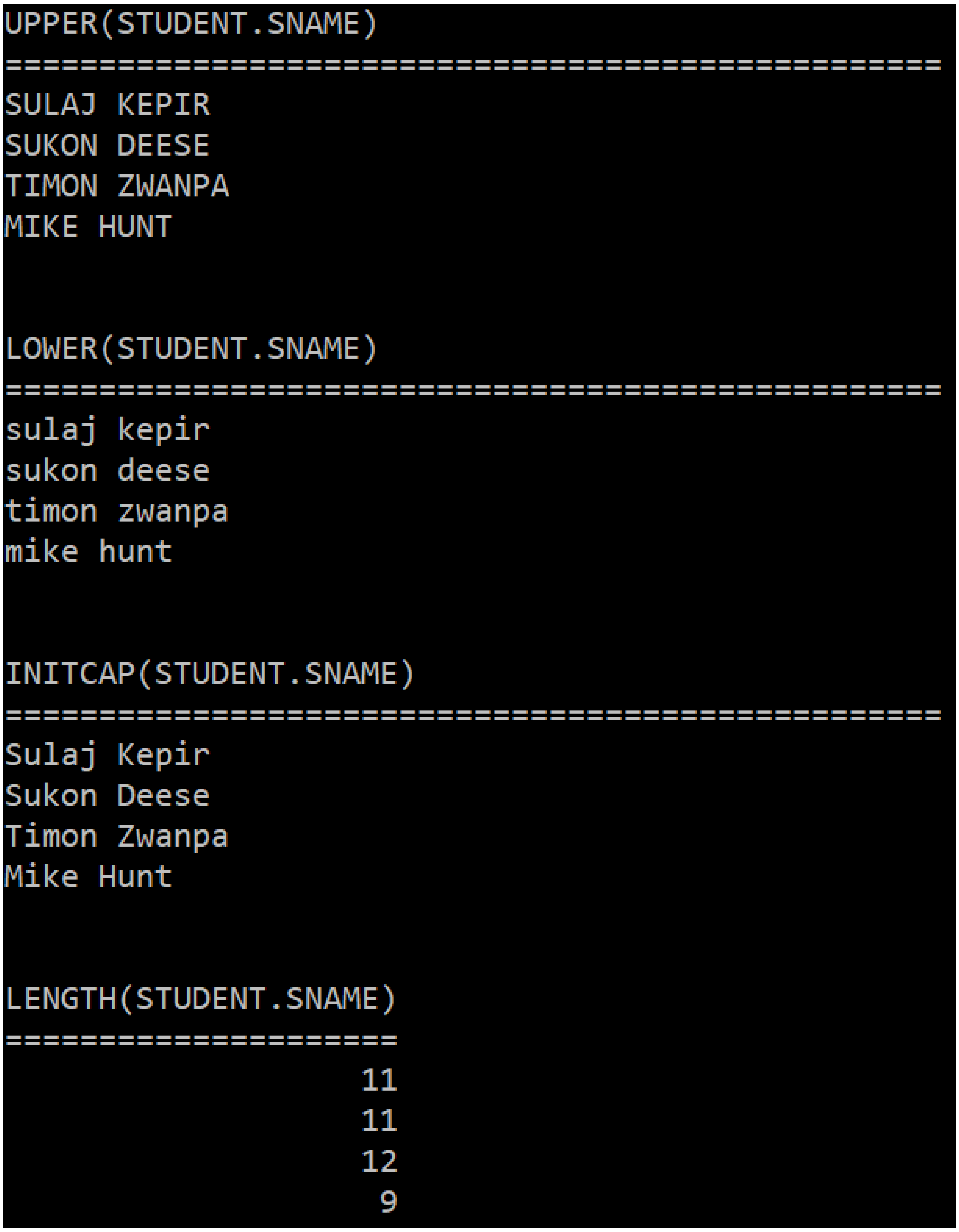
```sql
-- 3. (xxxv) Display the name of the departments and
-- the name of the program controlled by the department.
SELECT
    PROGRAMME.PROG_NAME,
    DEPARTMENT.DNAME
FROM
    PROGRAMME,
    DEPARTMENT
WHERE
    PROGRAMME.DEPT_ID = DEPARTMENT.DEPT_ID;
```



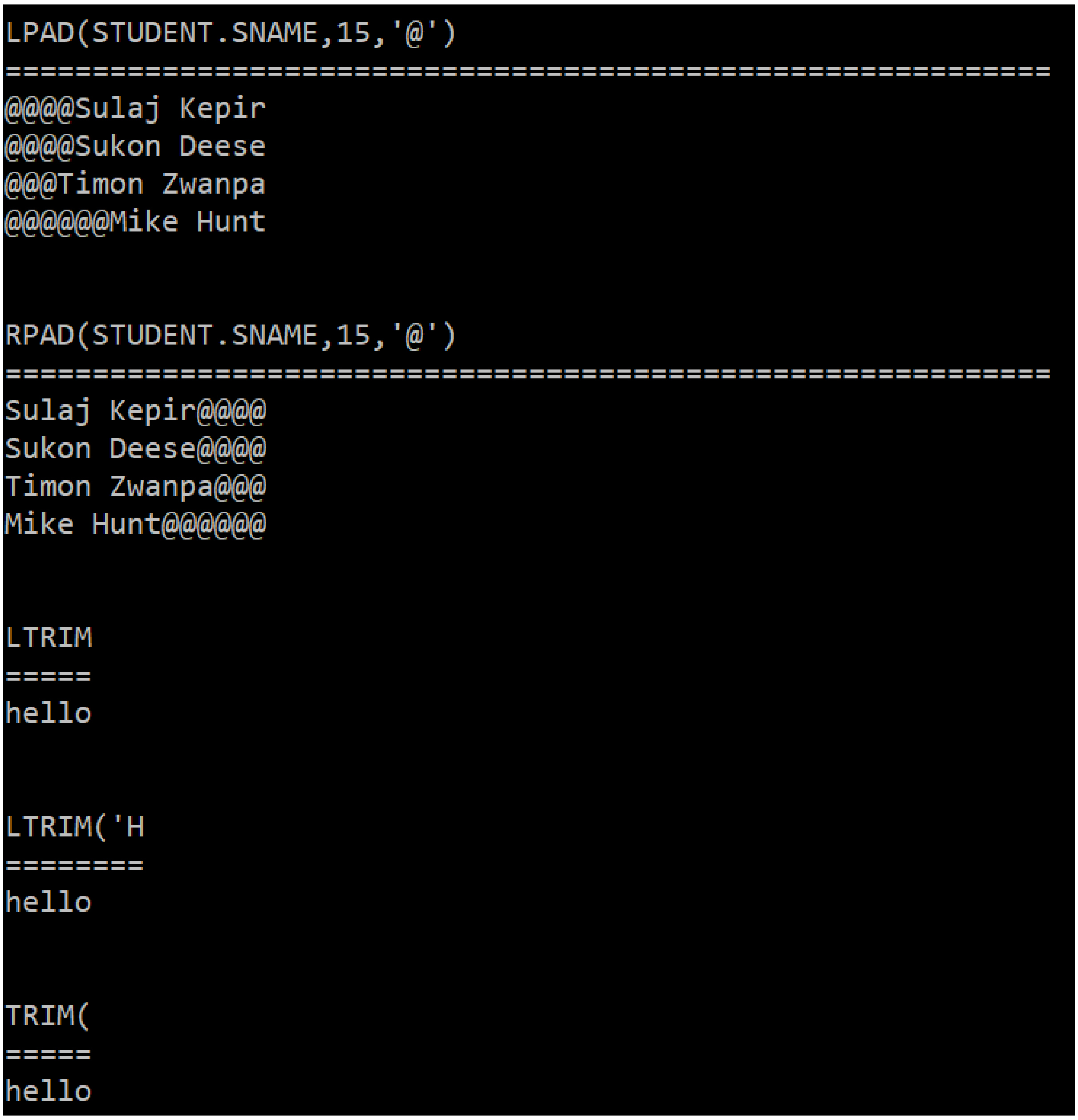
```
PROGRAMME             DEPARTMENT
NAME                  NAME
===================== ================================
Masters of Computer   NETWORKS
Application
```

```
-- 4. (i) Test the string manipulation functions –
-- UPPER, LOWER, INITCAP, LENGTH, LPAD, RPAD, LTRIM,
-- RTRIM and TRIM, using select queries on data
-- present in the tables. Use one query each for
-- demonstration of one function.
SELECT
    UPPER(STUDENT.SNAME)
FROM
    STUDENT;

SELECT
    LOWER(STUDENT.SNAME)
FROM
    STUDENT;

SELECT
    INITCAP(STUDENT.SNAME)
FROM
    STUDENT;

SELECT
    LENGTH(STUDENT.SNAME)
FROM
    STUDENT;

SELECT
    LPAD(STUDENT.SNAME,
    15,
    '@')
FROM
    STUDENT;

SELECT
    RPAD(STUDENT.SNAME,
    15,
```

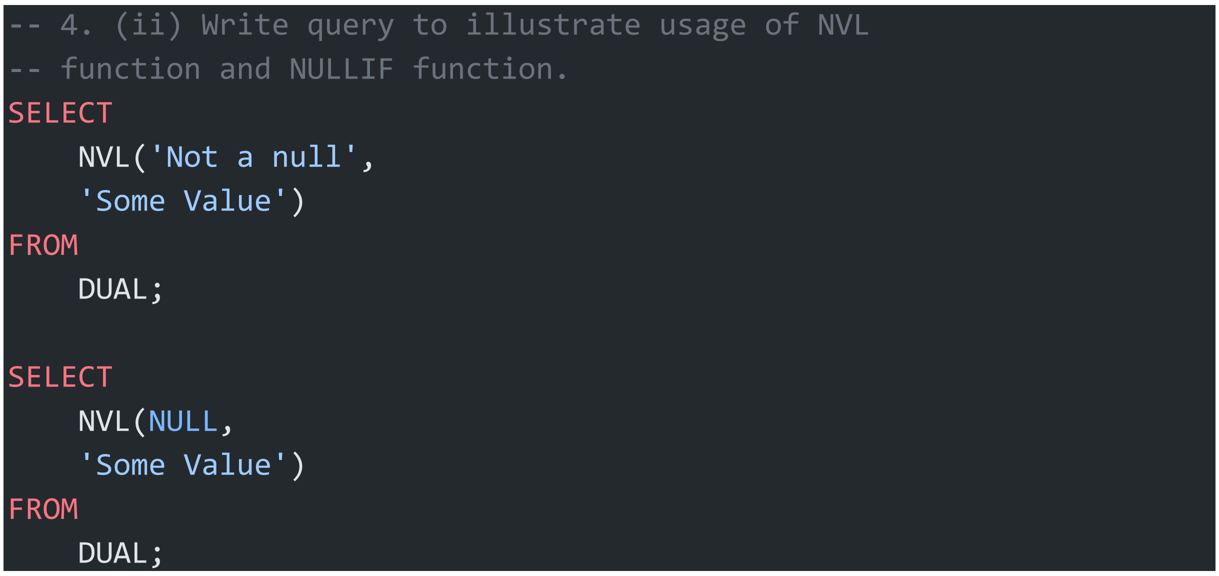
```sql
    '@')
FROM
    STUDENT;

SELECT
    LTRIM('   hello')
FROM
    DUAL;

SELECT
    LTRIM('hello   ')
FROM
    DUAL;

SELECT
    TRIM('   hello    ')
FROM
    DUAL;
```



```
UPPER(STUDENT.SNAME)
==================================================
SULAJ KEPIR
SUKON DEESE
TIMON ZWANPA
MIKE HUNT


LOWER(STUDENT.SNAME)
==================================================
sulaj kepir
sukon deese
timon zwanpa
mike hunt


INITCAP(STUDENT.SNAME)
==================================================
Sulaj Kepir
Sukon Deese
Timon Zwanpa
Mike Hunt


LENGTH(STUDENT.SNAME)
=====================
                   11
                   11
                   12
                    9
```

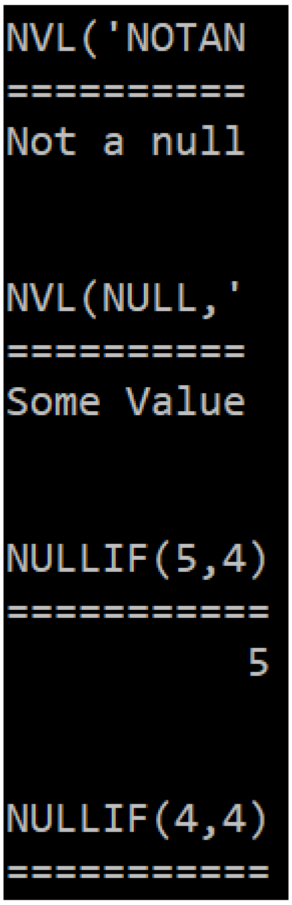


```
LPAD(STUDENT.SNAME,15,'@')
============================================================
@@@@Sulaj Kepir
@@@@Sukon Deese
@@@Timon Zwanpa
@@@@@@Mike Hunt


RPAD(STUDENT.SNAME,15,'@')
============================================================
Sulaj Kepir@@@@
Sukon Deese@@@@
Timon Zwanpa@@@
Mike Hunt@@@@@@


LTRIM
=====
hello


LTRIM('H
========
hello


TRIM(
=====
hello
```



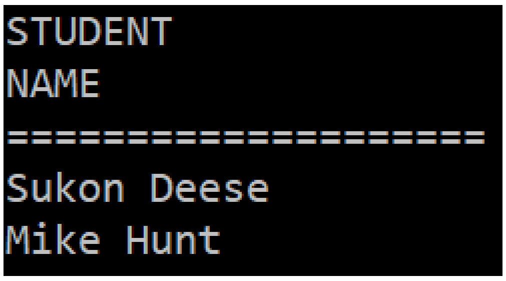
```
-- 4. (ii) Write query to illustrate usage of NVL
-- function and NULLIF function.
SELECT
    NVL('Not a null',
    'Some Value')
FROM
    DUAL;

SELECT
    NVL(NULL,
    'Some Value')
FROM
    DUAL;
```

```sql
SELECT
    NULLIF(5,
    4)
FROM
    DUAL;

SELECT
    NULLIF(4,
    4)
FROM
    DUAL;
```



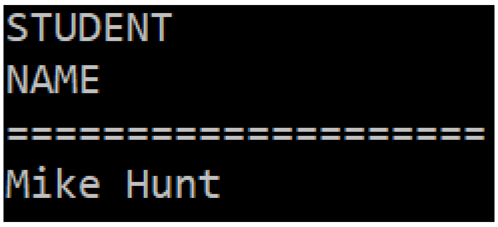
```
NVL('NOTAN
==========
Not a null


NVL(NULL,'
==========
Some Value


NULLIF(5,4)
===========
          5


NULLIF(4,4)
===========
```

```sql
-- 4. (iii) Display the name of the students who
-- were born on a specified month.
SELECT
    SNAME
FROM
    STUDENT
WHERE
    EXTRACT(MONTH FROM DOB) = 2;
```



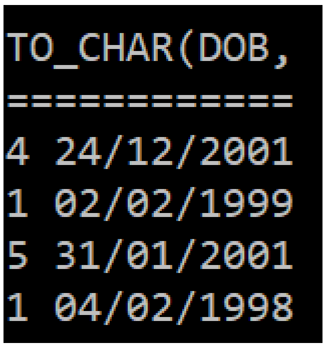
```
STUDENT
NAME
====================
Sukon Deese
Mike Hunt
```

```
-- 4. (iv) Display the name of the students with a
-- specified date of birth.
SELECT
    SNAME
FROM
    STUDENT
WHERE
    DOB = '04-FEB-1998';
```



```
STUDENT
NAME
====================
Mike Hunt
```

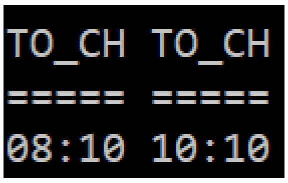
```
-- 4. (v) Display the date of birth of a specified
-- student in the format 'Day of week, Month dd, yyyy'
SELECT
    TO_CHAR(DOB,
    'W DD/MM/YYYY')
FROM
    STUDENT;
```



```
TO_CHAR(DOB,
============
4 24/12/2001
1 02/02/1999
5 31/01/2001
1 04/02/1998
```

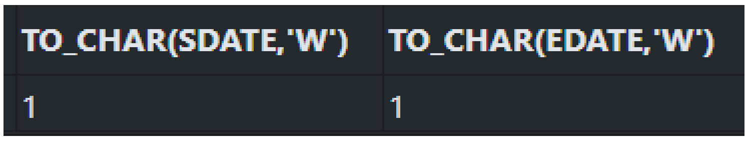
```
-- 4. (vi) Display the hour and minutes of the
-- start time and end time of a specified slot.

SELECT
```

```
    TO_CHAR(STIME,
    'HH:MI'),
    TO_CHAR(ETIME,
    'HH:MI')
FROM
    CLASS
WHERE
    SLOT = 'T1/A1';
```



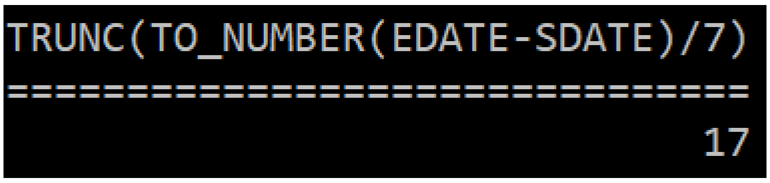
```
TO_CH TO_CH
===== =====
08:10 10:10
```

```
-- 4. (vii) Display the day of week of the start
-- date and end date of Winter semester 17-18
-- (Winter 2018).
SELECT
    TO_CHAR(SDATE,
    'W'),
    TO_CHAR(EDATE,
    'W')
FROM
    SEMESTER
WHERE
    SEM_CODE = 'WIN18';
```



TO_CHAR(SDATE,'W')	TO_CHAR(EDATE,'W')
1	1

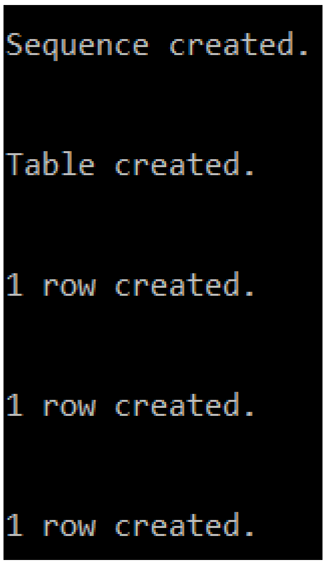
```
-- 4. (viii) Display the duration of Winter semester
-- 17-18 (Winter 2018) in terms of number of weeks
SELECT
    TRUNC(TO_NUMBER(EDATE - SDATE) / 7)
FROM
    SEMESTER
WHERE
    SEM_CODE = 'WIN18';
```



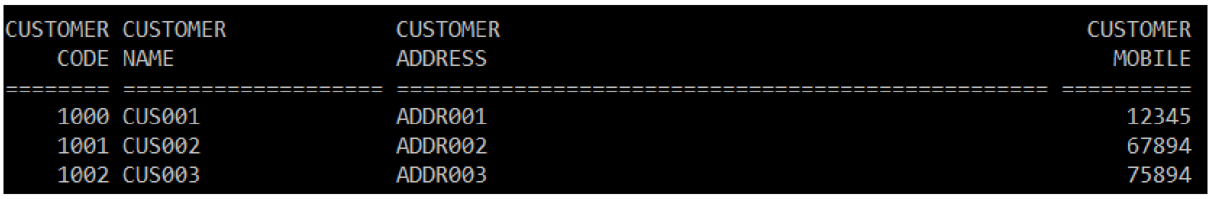
```
TRUNC(TO_NUMBER(EDATE-SDATE)/7)
==================================
                                17
```

```sql
-- 5. Create a sequence that starts with 1000 and
-- is incremented by 1. Use this sequence in the
-- following table for entering information about
-- at least three customers.
-- CUSTOMER(Cus_code, Cus_name, Cus_address, Cus_mobile)
CREATE SEQUENCE CODE_SEQUENCE START WITH 1000 INCREMENT BY 1;

CREATE TABLE CUSTOMER(
    CUS_CODE NUMBER,
    CUS_NAME VARCHAR(50),
    CUS_ADDRESS VARCHAR(100),
    CUS_MOBILE NUMBER,
    CONSTRAINT CUSTOMER_PK PRIMARY KEY(CUS_CODE)
);

INSERT INTO CUSTOMER VALUES(
    CODE_SEQUENCE.NEXTVAL,
    'CUS001',
    'ADDR001',
    12345
);

INSERT INTO CUSTOMER VALUES(
    CODE_SEQUENCE.NEXTVAL,
    'CUS002',
    'ADDR002',
    67894
);

INSERT INTO CUSTOMER VALUES(
    CODE_SEQUENCE.NEXTVAL,
    'CUS003',
    'ADDR003',
    75894
);

SELECT
```

```
    *
FROM
    CUSTOMER;
```



```
Sequence created.


Table created.


1 row created.


1 row created.


1 row created.
```



```
CUSTOMER CUSTOMER             CUSTOMER                                            CUSTOMER
    CODE NAME                 ADDRESS                                               MOBILE
======== ==================== ================================================== ==========
    1000 CUS001               ADDR001                                                 12345
    1001 CUS002               ADDR002                                                 67894
    1002 CUS003               ADDR003                                                 75894
```