

NAME-Barsha Routh

Reg no-24MCA0164

### DATABASE SYSTEMS

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)

#### 1. Create the above tables.

### **SOLUTION**

create table PROFESSOR(Prof\_id varchar(5),Prof\_name varchar(20),Email varchar(20),Mobile number(10),Speciality varchar(10),Dept\_id varchar(5));

create table SCHOOL(SCode varchar(5),Scl\_name varchar(20),Prof\_id varchar(5),Location varchar(10));

create table DEPARTMENT(Dept\_id varchar(5),Dname varchar(20),Scode varchar(5),Prof\_id varchar(5));

create table COURSE (Crs\_code varchar(5),Crs\_name varchar(10),Description varchar(30),Credits number(2),Hours number(2));

CREATE TABLE CLASS (Cls\_code VARCHAR(5),Slot VARCHAR(5),Stime TIMESTAMP(0),Etime TIMESTAMP(0),Crs\_code VARCHAR(5), Prof\_id VARCHAR(5),Room\_no VARCHAR(5), Sem\_code VARCHAR(5), Day\_of\_week VARCHAR(10));

Create table SEMESTER(sem\_code varchar(5),term varchar(6),Year varchar(4),Sdate date,Edate date);

Create table STUDENT(Reg\_no varchar(5),Sname varchar(10),Address varchar(20),DoB date,Email varchar(20),Mobile varchar(10),Dept\_id varchar(5),Prof\_id varchar(5));

Create table ENROLL(Cls\_code varchar(5),Reg\_no varchar(5),Enroll\_time timestamp(0), Grade CHAR(1) CHECK (Grade IN ('S', 'A', 'B', 'C', 'D')));

Create table STUDENT\_VISA(Reg\_no varchar(5),Visa\_status Varchar(20));

Create table PROGRAMME(Prog\_code varchar(5), Prog\_name varchar(10), Prog\_preamble varchar(50), Scode varchar(5), Dept\_id varchar(5));

OUTPUT IN THE NEXT QUESTION--

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.

## SOLUTION

### INTERACTIVE ENTRY OF DATA

insert into PROFESSOR values(&Prof\_id,&Prof\_name,&Email,&Mobile,&Specialty,&Dept\_id);

insert into DEPARTMENT values(&Dept\_id, &Dname, &SCode, &Prof\_id);

insert into SCHOOL values(&SCode, &Scl\_name, &Prof\_id, &Location);

insert into COURSE values(&Crs\_code, &Crs\_name, &Description, &Credits, &Hours);

insert into CLASS values(&Cls\_code, &Slot, &Stime, &Etime, &Crs\_code, &Prof\_id, &Room\_no, &Sem\_code, &Day\_of\_week);

insert into SEMESTER values(&Sem\_code, &Term, &Year, &Sdate, &Edate);

insert into STUDENT values(&Reg\_no, &Sname, &Address, &DoB, &Email, &Mobile, &Dept\_id, &Prof\_id);

insert into ENROLL values(&Cls\_code, &Reg\_no, &Enroll\_time, &Grade);

insert into STUDENT\_VISA values(&Reg\_no, &Visa\_status);

insert into PROGRAMME values(&Prog\_code, &Prog\_name, &Prog\_preamble, &SCode, &Dept\_id);

## OUTPUT

### PROFESSOR

```
SQL> Select * from PROFESSOR;
```

PROFESSOR ID	PROFESSOR NAME	STUDENT EMAIL	MOBILE	PROFESSOR SPECIALITY	DEPARTMENT ID
P0001	Barsha	barsha@gmail.com	9647008681	Maths	D0001
P0002	Anil	anil@gmail.com	9876543210	Science	D0002
P0003	Sita	sita@gmail.com	8765432109	Arts	D0003
P0004	Ravi	ravi@gmail.com	7654321098	Commerce	D0004
P0005	Geeta	geeta@gmail.com	6543210987	Medical	D0005
P0006	Mohan	mohan@gmail.com	5432109876	Maths	D0001
P0007	Lata	lata@gmail.com	4321098765	Science	D0002
P0008	Raj	raj@gmail.com	3210987654	Arts	D0003
P0009	Meena	meena@gmail.com	2109876543	Commerce	D0004
P0010	Vijay	vijay@gmail.com	1098765432	Medical	D0005
P0011	O'Brien	brien@gmail.com	1098766432	Anatomy	D0005

### SCHOOL

```
SQL> Select * from SCHOOL;
```

SCHOOL CODE	SCL_NAME	PROFESSOR ID	LOCATION
S0001	Maths School	P0001	Building M
S0002	Science School	P0002	Building S
S0003	Arts School	P0003	Building A
S0004	Commerce School	P0004	Building C
S0005	Medical School	P0005	Building M

### DEPARTMENT

```
SQL> Select * from DEPARTMENT;
```

DEPARTMENT ID	DEPARTMENT NAME	SCHOOL CODE	PROFESSOR ID
D0001	Maths	S0001	P0006
D0002	Science Dept	S0002	P0007
D0003	Arts Dept	S0003	P0008
D0004	Commerce Dept	S0004	P0009
D0005	Medical Dept	S0005	P0010

### COURSE

```
SQL> Select * from COURSE;
```

COURSE CODE	COURSE NAME	COURSE DESCRIPTION	CREDITS	HOURS
C0001	MCA	Computer Applications Study	30	90
C0002	MBA	Business Administration	40	85
C0003	B.Tech	Engineering	50	80
C0004	B.Sc	Science	35	75
C0005	B.Com	Commerce	45	88
C0006	M.Sc	Advanced Science	25	92
C0007	Database	database here	11	12
C0008	Operating	OS HERE	20	82

### CLASS

CLASS CODE	SLOT	START	END_T	COURSE CODE	PROFESSOR ID	ROOM	SEM_C	DAY_OF_WEE
CL001	MOR	09:00	09:00	C0001	P0001	A114	Win01	Monday
CL002	AFT	13:00	13:00	C0002	P0002	B201	Win02	Tuesday
CL003	EVE	17:00	17:00	C0003	P0003	C301	Win01	Wednesday
CL004	MOR	09:00	09:00	C0004	P0004	D401	Fall1	Thursday
CL005	AFT	13:00	13:00	C0005	P0005	E501	Fall1	Friday
CL006	EVE	17:00	17:00	C0007	P0006	F609	Fall2	Tuesday
CL007	AFT	13:00	13:00	C0008	P0005	E502	Fall2	Saturday
CL008	EVE	17:00	17:00	C0007	P0006	F609	Win01	Tuesday

### SEMESTER

```
SQL> Select * from SEMESTER;
```

SEM_C	TERM	YEAR	START DATE	END DATE
Win01	Winter	2016	01-NOV-16	15-APR-17
Fall1	Fall	2016	01-MAR-16	15-NOV-16
Win02	Winter	2017	01-NOV-17	15-APR-18
Fall2	Fall	2017	01-MAR-17	15-NOV-17

### STUDENT

STUDENT REG NO	STUDENT NAME	STUDENT ADDRESS	DOB	STUDENT EMAIL	MOBILE	DEPARTMENT ID	PROFESSOR ID
2MCA1	Barshaa	Asansol	15-DEC-02	barsha7@gmail.com	9749806802	D0001	P0001
2MBA1	Rahul	Mumbai	20-JAN-01	rahul123@gmail.com	9876543210	D0003	P0003
2BTH1	Sneha	Delhi	05-MAR-00	sneha456@gmail.com	9123456780	D0002	P0002
2BSC1	Amit	Kolkata	10-JUL-02	amit789@gmail.com	9988776655	D0005	P0005
2BCM1	Priya	Chennai	25-SEP-01	priya101@gmail.com	9871234567	D0004	P0004
2MSC1	Vikram	Bangalore	15-NOV-00	vikram202@gmail.com	9765432109	D0001	P0007
2BBA1	Aadi	Katpadi	15-NOV-00	aadi202@gmail.com	9765432009	D0002	P0006
2BCA1	Ayul	Katpadi	15-NOV-19	ayu202@gmail.com	9765482009	D0003	P0006

### ENROLL

CLASS CODE	STUDENT REG NO	ENROLL TIME	GRADE
CL001	2MCA1	09:00	A
CL002	2MBA1	13:00	B
CL003	2BTH1	17:00	A
CL004	2BSC1	09:00	C
CL005	2BCM1	13:00	B
CL006	2MSC1	17:00	A
CL007	2MSC1	17:00	A
CL006	2BSC1	17:00	A
CL008	2MSC1	17:00	B

### STUDENT VISA

STUDENT REG NO	VISA_STATUS
2MCA1	APPLIED
2MBA1	APPROVED
2BTH1	PENDING
2BSC1	REJECTED
2BCM1	APPLIED
2MSC1	APPROVED

### PROGRAMME

SQL> Select \* from PROGRAMME;

PROGRAMME CODE	PROGRAMME NAME	PROGRAMME PREAMBLE	SCHOOL CODE	DEPARTMENT ID
PROG1	MCA	comprehensive understanding of computer science	S0001	D0001
PROG2	MBA	in-depth knowledge of business administration	S0003	D0003
PROG3	B.Tech	extensive training in engineering principles	S0002	D0002
PROG4	B.Sc	broad understanding of scientific concepts	S0002	D0002
PROG5	B.Com	comprehensive study of commerce and trade	S0004	D0004
PROG6	M.Sc	advanced exploration of scientific disciplines	S0002	D0002

**3. Alter or Recreate the above tables with primary key and foreign key and the following integrity constraints assigning name to integrity constrain**

**SOLUTION**

--PRIMARY KEYS

ALTER TABLE PROFESSOR ADD CONSTRAINT pk\_professor PRIMARY KEY (Prof\_id);

ALTER TABLE SCHOOL ADD CONSTRAINT pk\_school PRIMARY KEY (SCode);

ALTER TABLE DEPARTMENT ADD CONSTRAINT pk\_department PRIMARY KEY (Dept\_id);

ALTER TABLE COURSE ADD CONSTRAINT pk\_course PRIMARY KEY (Crs\_code);

ALTER TABLE CLASS ADD CONSTRAINT pk\_class PRIMARY KEY (Cls\_code);

ALTER TABLE SEMESTER ADD CONSTRAINT pk\_semester PRIMARY KEY (Sem\_code);

ALTER TABLE STUDENT ADD CONSTRAINT pk\_student PRIMARY KEY (Reg\_no);

ALTER TABLE ENROLL ADD CONSTRAINT pk\_enroll PRIMARY KEY (Cls\_code, Reg\_no);

ALTER TABLE STUDENT\_VISA ADD CONSTRAINT pk\_student\_visa PRIMARY KEY (Reg\_no);

ALTER TABLE PROGRAMME ADD CONSTRAINT pk\_programme PRIMARY KEY (Prog\_code);

--Add FK

alter table PROFESSOR add constraint fk\_Dept foreign key(Dept\_id) references DEPARTMENT(Dept\_id)  
deferrable initially deferred;

alter table SCHOOL add constraint fk\_Prof\_id foreign key(Prof\_id) references PROFESSOR(Prof\_id) deferrable  
initially deferred;

alter table DEPARTMENT add constraint fk\_Profid foreign key(Prof\_id) references PROFESSOR(Prof\_id)  
deferrable initially deferred;

alter table DEPARTMENT add constraint fk\_SCode foreign key(SCode) references SCHOOL(SCode) deferrable  
initially deferred;

alter table CLASS add constraint fk\_Crs\_code foreign key(Crs\_code) references COURSE(Crs\_code) deferrable  
initially deferred;

alter table CLASS add constraint fk\_Pr fid foreign key(Prof\_id) references PROFESSOR (Prof\_id) deferrable  
initially deferred;

alter table CLASS add constraint fk\_SemCode foreign key(Sem\_code) references Semester(Sem\_code)  
deferrable initially deferred;

alter table STUDENT add constraint fk\_Depid foreign key(Dept\_id) references DEPARTMENT (Dept\_id)  
deferrable initially deferred;

alter table STUDENT add constraint fk\_Proid foreign key(Prof\_id) references PROFESSOR (Prof\_id) deferrable  
initially deferred;

alter table Enroll add constraint fk\_ClsCode foreign key(Cls\_code) references CLASS (Cls\_code) deferrable  
initially deferred;

alter table Enroll add constraint fk\_RegNo foreign key(Reg\_no) references STUDENT (Reg\_no) deferrable initially deferred;

alter table PROGRAMME add constraint fk\_SC foreign key(Scode) references SCHOOL (SCode) deferrable initially deferred;

alter table PROGRAMME add constraint fk\_Dept foreign key(Dept\_id) references DEPARTMENT (Dept\_id) deferrable initially deferred;

4. 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.
- ii) Use timestamp data type without fractional parts of seconds for start time and end time column of class table
- iii) The Sem\_code should start with either 'Win' or 'Fall' and Term column can assume only one of two values {Winter, Fall}.
- iv) Email and mobile column in student table should have same characteristics as those in professor table.
- 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'}.
- vi) Use 'on delete cascade' or 'on delete set null' clause as requirements. Use deferrable constraint, if required.
- vii) Additional (innovative) integrity constraints, if any, may be specified by you.

--PROFESSOR

```
alter table PROFESSOR add constraint uk_email unique(Email);
alter table PROFESSOR add constraint uk_Mobile unique(Mobile);
alter table PROFESSOR add constraint chk_Len CHECK(LENGTH(Prof_id)=5);
alter table PROFESSOR add constraint chk_Len_Mob CHECK(LENGTH(Mobile)=10);
alter table PROFESSOR add constraint chk_Email_atTheRate CHECK(Email like '%@%');
```

--SEMESTER

```
alter table SEMESTER add constraint chk_SemesCod CHECK(sem_code Like 'Win%' or sem_code like 'Fall%');
alter table SEMESTER add constraint chk_Term CHECK(Term IN ('Winter', 'Fall'));
```

--STUDENT

```
alter table STUDENT add constraint uk_stu_email unique(Email);
alter table STUDENT add constraint uk_stu_Mobile unique(Mobile);
alter table STUDENT add constraint chk_Len_StuMob CHECK(LENGTH(Mobile)=10);
alter table STUDENT add constraint chk_stuEmail_atTheRate CHECK(Email like '%@%');
```

--ENROLL

```
alter table Enroll add constraint chk_value CHECK(Grade IN ('S', 'A', 'B', 'C', 'D'));
```

#### **4. . In built functions**

**(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(Sname) FROM STUDENT;

SELECT LOWER(Sname) FROM STUDENT;

SELECT INITCAP(Visa_status) FROM STUDENT_VISA;

SELECT LENGTH(Dname) FROM DEPARTMENT;

SELECT LPAD(Visa_status,12,'*') FROM STUDENT_VISA;

SELECT RPAD(Visa_status,12,'*') FROM STUDENT_VISA;

SELECT TRIM(emp_name) FROM Emp1;

SELECT RTRIM(emp_name,' ') FROM Emp1;

SELECT LTRIM(emp_name,' ') FROM Emp1;
```

### **SOLUTION**

```
SQL> SELECT UPPER(Sname) FROM STUDENT;

UPPER(SNAM
-----
BARSHAA
RAHUL
SNEHA
AMIT
PRIYA
VIKRAM
AADI
AYUL

8 rows selected.

SQL> SELECT LOWER(Sname) FROM STUDENT;

LOWER(SNAM
-----
barshaa
rahul
sneha
amit
priya
vikram
aadi
ayul
```

```

SQL> SELECT INITCAP(Visa_status) FROM STUDENT_VISA;

INITCAP(VISA_STATUS)
-----
Applied
Approved
Pending
Rejected
Applied
Approved

6 rows selected.

SQL> SELECT LENGTH(Dname) FROM DEPARTMENT;

LENGTH(DNAME)
-----
5
12
9
13
12

SQL> SELECT LPAD(Visa_status,12,'*') FROM STUDENT_VISA;

LPAD(VISA_STATUS,12,'*')
-----
*****APPLIED
*****APPROVED
*****PENDING
*****REJECTED
*****APPLIED
*****APPROVED

```

```

SQL> SELECT RPAD(Visa_status,12,'*') FROM STUDENT_VISA;

RPAD(VISA_STATUS,12,'*')
-----
APPLIED*****
APPROVED*****
PENDING*****
REJECTED*****
APPLIED*****
APPROVED*****

6 rows selected.

SQL> SELECT TRIM(emp_name) FROM Emp1;

TRIM(EMP_N
-----
Alice
Bob
Charlie
David
Eva

```



```
SQL> SELECT RTRIM(emp_name,' ') FROM Emp1;

RTRIM(EMP_
-----
  Alice
Bob
  Charlie
  David
Eva

SQL> SELECT LTRIM(emp_name, ' ') FROM Emp1;

LTRIM(EMP_
-----
Alice
Bob
Charlie
David
Eva
```

**(ii) Write query to illustrate usage of NVL function and NULLIF function.**

```
select Empid,emp_name,nvl(BonusAmt,0) as BonusAmt from emp1;
```

```
select Empid,emp_name,salary,bonusamt,nullif(salary,Bonusamt) as TotalAmt from emp1; --not returning null
when not equal
```

```
SQL> select Empid,emp_name,nvl(BonusAmt,0) as BonusAmt from emp1;

EMPID  EMP_NAME      BONUSAMT
-----
E0001  Alice          3000
E0002  Bob            60000
E0003  Charlie        6000
E0004  David          62000
E0005  Eva            2000

SQL> select Empid,emp_name,salary,bonusamt,nullif(salary,Bonusamt) as TotalAmt from emp1;

EMPID  EMP_NAME      SALARY  BONUSAMT  TOTALAMT
-----
E0001  Alice          50000    3000     50000
E0002  Bob            60000   60000
E0003  Charlie        55000    6000     55000
E0004  David          62000   62000
E0005  Eva            58000    2000     58000
```

**(iii) Display the name of the students who were born on a specified month.**

```
select sname from student where extract(month from DOB)=12;
```

```
STUDENT
NAME
-----
Barshaa
```

**(iv) Display the name of the students with a specified date of birth.**

```
select sname from student where DOB = TO_DATE('20-JAN-2001', 'DD-MON-YYYY');
```

```
SQL> select sname from student where DOB = TO_DATE('20-JAN-2001', 'DD-MON-YYYY');

STUDENT
NAME
-----
Rahul
```

**(v) Display the date of birth of a specified student in the format 'Day of week, Month dd, yyyy'.**

```
select TO_CHAR('Day , Month dd,yyyy') from student where sname='Barshaa';
```

```
TO_CHAR('DAY,MONTHD
-----
Day , Month dd,yyyy
```

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

```
SELECT TO_CHAR(Stime, 'HH24:MI') AS Start_Time, TO_CHAR(ETIME, 'HH24:MI') AS End_Time FROM CLASS
WHERE Slot = 'MOR';
```

```
START  END_T
-----
09:00  09:00
09:00  09:00
```

**(vii) Display the day of week of the start date and end date of Winter semester 17–18.**

```
SELECT
    TO_CHAR(Sdate, 'Day') AS Start_Day,
    TO_CHAR(EdatE, 'Day') AS End_Day
FROM SEMESTER
WHERE Term = 'Winter'
AND Year = '2017';
```

```
START_DAY          END_DAY
-----
Wednesday          Sunday
```

**(viii) Display the duration of Winter semester 17–18 in terms of number of weeks.**

```
SELECT ROUND((EdatE - Sdate) / 7) AS Duration_Weeks FROM SEMESTER WHERE Term = 'Winter' AND Year = '2017';
```

```
DURATION_WEEKS
-----
24
```

**(ix) Store date in the format dd/mm/yy for DOB of newly admitted student.**

```
SELECT sname, TO_CHAR(DOB, 'DD/MM/YY') AS NewDob FROM student;
```

STUDENT NAME	NEWDOB
Barshaa	15/12/02
Rahul	20/01/01
Sneha	05/03/00
Amit	10/07/02
Priya	25/09/01
Vikram	15/11/00
Aadi	15/11/00
Ayul	15/11/19
Bindu	15/08/04

(x) Test the numeric functions – CEIL, FLOOR, TRUNCATE, MIN, MAX, AVG, COUNT using select queries on data present in the tables. Use one query each for demonstration of one function.

```
SELECT CEIL(AVG(Credits)) FROM COURSE;
```

```
SELECT FLOOR(AVG(Credits)) FROM COURSE;
```

```
SELECT TRUNC(AVG(Credits), 1) FROM COURSE;
```

```
SELECT MIN(Credits) FROM COURSE;
```

```
SELECT MAX(Credits) FROM COURSE;
```

```
SELECT AVG(Credits) FROM COURSE;
```

```
SELECT COUNT(*) FROM ENROLL;
```

```
SQL> SELECT CEIL(AVG(Credits)) FROM COURSE;
CEIL(AVG(CREDITS))
-----
32

SQL> SELECT FLOOR(AVG(Credits)) FROM COURSE;
FLOOR(AVG(CREDITS))
-----
32

SQL> SELECT TRUNC(AVG(Credits), 1) FROM COURSE;
TRUNC(AVG(CREDITS),1)
-----
32

SQL> SELECT MIN(Credits) FROM COURSE;
MIN(CREDITS)
-----
11

SQL> SELECT MAX(Credits) FROM COURSE;
MAX(CREDITS)
-----
50

SQL> SELECT AVG(Credits) FROM COURSE;
AVG(CREDITS)
-----
32

SQL> SELECT COUNT(*) FROM ENROLL;
COUNT(*)
-----
9
```

5. Write Queries for

- 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 = 'Katpadi' AND sname LIKE '___l%';
```

STUDENT NAME	STUDENT EMAIL	STUDENT ADDRESS
Ayul	ayu202@gmail.com	Katpadi

- ii. Display name, email address and address for those students who are not from Tamil Nadu.

```
SELECT sname, Email, Address FROM Student WHERE Address != 'Tamil Nadu';
```

STUDENT NAME	STUDENT EMAIL	STUDENT ADDRESS
Barshaa	barsha7@gmail.com	Asansol
Rahul	rahul123@gmail.com	Mumbai
Sneha	sneha456@gmail.com	Delhi
Amit	amit789@gmail.com	Kolkata

- 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 NAME	STUDENT EMAIL	STUDENT ADDRESS
Barshaa	barsha7@gmail.com	Asansol
Rahul	rahul123@gmail.com	Mumbai
Sneha	sneha456@gmail.com	Delhi
Amit	amit789@gmail.com	Kolkata

- (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 = 'Medical School' AND SCHOOL.PROF_ID = PROFESSOR.PROF_ID;
```

PROFESSOR NAME	PROFESSOR SPECIALITY
Geeta	Medical

- 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;
```

SCL_NAME	PROFESSOR NAME
Maths School	Barsha
Science School	Anil
Arts School	Sita
Commerce School	Ravi
Medical School	Geeta

- vi. List course code, course name and course description in alphabetic order of course code.

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

COURSE CODE	COURSE NAME	COURSE DESCRIPTION
C0001	MCA	Computer Applications Study
C0002	MBA	Business Administration
C0003	B.Tech	Engineering
C0004	B.Sc	Science
C0005	B.Com	Commerce
C0006	M.Sc	Advanced Science
C0007	Database	database here
C0008	Operating	OS HERE

- vii. Change the mobile number of a student interactively.

```
UPDATE STUDENT SET MOBILE='&MOBILE' WHERE REG_NO='&REG_NO';
```

```
SQL> UPDATE STUDENT SET MOBILE='&MOBILE' WHERE REG_NO='&REG_NO';
Enter value for mobile: 9749800899
Enter value for reg_no: 3BCA1
old 1: UPDATE STUDENT SET MOBILE='&MOBILE' WHERE REG_NO='&REG_NO'
new 1: UPDATE STUDENT SET MOBILE='9749800899' WHERE REG_NO='3BCA1'
```

(viii) Remove enrollment information of a student from a particular course interactively. How would you recover the data?

```
SAVEPOINT BEFORE_del;
```

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

--we can recover data with the help of ROLLBACK

```
ROLLBACK TO BEFORE_del;
```

```
SQL> SAVEPOINT BEFORE_del;
Savepoint created.

SQL> DELETE FROM ENROLL WHERE REG_NO='&REG_NO';
Enter value for reg_no: 2MSC1
old 1: DELETE FROM ENROLL WHERE REG_NO='&REG_NO'
new 1: DELETE FROM ENROLL WHERE REG_NO='2MSC1'

3 rows deleted.

SQL> ROLLBACK TO BEFORE_del;
Rollback complete.
```

**(ix) Create a duplicate of course table.**

```
CREATE TABLE COURSE_DUPLICATED AS SELECT * FROM COURSE;
```

```
SELECT * FROM COURSE_DUPLICATED;
```

```
SQL> CREATE TABLE COURSE_DUPLICATED AS SELECT * FROM COURSE;
Table created.
SQL> SELECT * FROM COURSE_DUPLICATED;
```

COURSE CODE	COURSE NAME	COURSE DESCRIPTION	CREDITS	HOURS
C0001	MCA	Computer Applications Study	30	90
C0002	MBA	Business Administration	40	85
C0003	B.Tech	Engineering	50	80
C0004	B.Sc	Science	35	75
C0005	B.Com	Commerce	45	88
C0006	M.Sc	Advanced Science	25	92
C0007	Database	database here	11	12
C0008	Operating	OS HERE	20	82

```
8 rows selected.
```

**(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 STUDENTS_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 STUDENTS_COURSE_VIEW;
```

```
SQL> SELECT * FROM STUDENTS_COURSE_VIEW;
```

STUDENT REG NO	STUDENT NAME	COURSE NAME	PROFESSOR NAME
2MCA1	Barshaa	MCA	Barsha
2MBA1	Rahul	MBA	Anil
2BTH1	Sneha	B.Tech	Sita
2BSC1	Amit	Database	Mohan
2BSC1	Amit	B.Sc	Ravi
2BCM1	Priya	B.Com	Geeta
2MSC1	Vikram	Database	Mohan
2MSC1	Vikram	Operating	Geeta
2MSC1	Vikram	Database	Mohan

**(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, TO_CHAR(Stime, 'HH24:MI') AS STIME, TO_CHAR(etime, 'HH24:MI') AS ETIME,
EXTRACT (HOUR FROM ETIME - STIME) AS "DURATION" FROM CLASS WHERE DAY_OF_WEEK = 'Wednesday'
ORDER BY ROOM_NO DESC;
```

ROOM	SLOT	STARTING TIME	END TIME	DURATION
C301	EVE	17:00	17:00	0

**(xii) Display the name and grade of a student in different courses underwent in fall semester 2017 – 18.**

```
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 NAME	COURSE NAME	GRADE
Amit	Database	A
Vikram	Database	A
Vikram	Operating	A

**(xiii) Find out name of students who have taken Database Systems course as well as Operating Systems course in fall semester 2016 – 17.**

```
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 ('C0007',
```

```
'C0008')
```

```
INNER JOIN SEMESTER
```

```
ON SEMESTER.SEM_CODE = CLASS.SEM_CODE
```

```
AND SEMESTER.TERM = 'Fall'
```

```
AND SEMESTER.YEAR = 2017
```

```
GROUP BY
```

```
STUDENT.SNAME
```

```
HAVING
```

```
COUNT(DISTINCT COURSE.CRS_CODE) = 2;
```

STUDENT NAME
Vikram

**(xiv) Find out name of students who have taken Database Systems course but have not taken Operating Systems course in winter semester 2017 – 18.**

```

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 ('C0007','C0008')
INNER JOIN SEMESTER ON SEMESTER.SEM_CODE = CLASS.SEM_CODE AND SEMESTER.YEAR = 2017
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 = 'C0008' INNER JOIN SEMESTER ON SEMESTER.SEM_CODE = CLASS.SEM_CODE AND
SEMESTER.YEAR = 2018

AND SEMESTER.TERM = 'Win01';

```

no rows selected

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

```

SELECT S.Reg_no, S.Sname
FROM STUDENT S
JOIN ENROLL E ON S.Reg_no = E.Reg_no
JOIN CLASS C ON E.Cls_code = C.Cls_code
JOIN COURSE CR ON C.Crs_code = CR.Crs_code
JOIN SEMESTER SEM ON C.Sem_code = SEM.Sem_code
WHERE SEM.Term = 'Win01'
AND SEM.Year = '2016'
GROUP BY S.Reg_no, S.Sname
HAVING SUM(CR.Credits) = (
    SELECT MAX(Total_Credits)
    FROM (
        SELECT SUM(CR.Credits) AS Total_Credits
        FROM STUDENT S

```



```

JOIN ENROLL E ON S.Reg_no = E.Reg_no

JOIN CLASS C ON E.Cls_code = C.Cls_code

JOIN COURSE CR ON C.Crs_code = CR.Crs_code

JOIN SEMESTER SEM ON C.Sem_code = SEM.Sem_code

WHERE SEM.Term = 'Win01'

AND SEM.Year = '2016'

GROUP BY S.Reg_no

)

);

```

```
no rows selected
```

**(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
Operating	1	AFT
B.Tech	1	EVE
Database	1	EVE
Database	2	EVE
MBA	1	AFT
MCA	1	MOR
B.Sc	1	MOR
B.Com	1	AFT

**(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.**

```

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
-----
Ayusla

```

**(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, COURSE

WHERE ENROLL.REG_NO = STUDENT.REG_NO

```

```

AND ENROLL.CLS_CODE = CLASS.CLS_CODE

AND CLASS.CRS_CODE = COURSE.CRS_CODE

AND TO_CHAR(ENROLL.ENROLL_TIME, 'DD-MM-YYYY') = '17-11-2017'

AND COURSE.CRS_NAME = 'Database';

```

```

STUDENT
REG NO
-----
2BSC1
2MSC1

```

**(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.**

```

SELECT ENROLL.REG_NO

FROM ENROLL, CLASS, SEMESTER

WHERE ENROLL.CLS_CODE = CLASS.CLS_CODE

AND CLASS.SEM_CODE = SEMESTER.SEM_CODE

AND SEMESTER.SEM_CODE LIKE 'Fall%'

AND SEMESTER.YEAR = '2017'

AND ENROLL.REG_NO LIKE '&REG_NO';

```

```

SQL> SELECT ENROLL.REG_NO
  2     FROM ENROLL, CLASS, SEMESTER
  3     WHERE ENROLL.CLS_CODE = CLASS.CLS_CODE
  4     AND CLASS.SEM_CODE = SEMESTER.SEM_CODE
  5     AND SEMESTER.SEM_CODE LIKE 'Fall%'
  6     AND SEMESTER.YEAR = '2017'
  7     AND ENROLL.REG_NO LIKE '&REG_NO';
Enter value for reg_no: 2MSC1
old 7:      AND ENROLL.REG_NO LIKE '&REG_NO'
new 7:      AND ENROLL.REG_NO LIKE '2MSC1'

STUDENT
REG NO
-----
2MSC1
2MSC1

```

**(xx) List the name of departments and the name professors who is in charge of the department.**

```

SELECT Dept.Dname AS Department_Name, Prof.Prof_name AS Professor_Name

FROM DEPARTMENT Dept

JOIN PROFESSOR Prof ON Dept.Prof_id = Prof.Prof_id;

```

DEPARTMENT_NAME	PROFESSOR_NAME
Maths	Mohan
Science Dept	Lata
Arts Dept	Raj
Commerce Dept	Meena
Medical Dept	Vijay
English Dept	O'Brien

6 rows selected.

**(xxi) List the name of schools with students' strength higher than 7000.**

```
SELECT Sch.Scl_name AS School_Name, COUNT(Stu.Reg_no) AS Student_Strength FROM STUDENT Stu JOIN
DEPARTMENT Dept ON Stu.Dept_id = Dept.Dept_id JOIN SCHOOL Sch ON Dept.SCode = Sch.SCode GROUP BY
Sch.Scl_name HAVING COUNT(Stu.Reg_no) > 7000;
```

```
SQL> SELECT Sch.Scl_name AS School_Name, COUNT(Stu.Reg_no) AS Student_Strength
2 FROM STUDENT Stu
3 JOIN DEPARTMENT Dept ON Stu.Dept_id = Dept.Dept_id
4 JOIN SCHOOL Sch ON Dept.SCode = Sch.SCode
5 GROUP BY Sch.Scl_name
6 HAVING COUNT(Stu.Reg_no) > 7000;

no rows selected
```

**(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.**

**(xxiii) Given the registration number of a student, display the total credits registered by him/her in Winter 17-18**

```
SELECT S.Reg_no, SUM(CR.Credits) AS Total_Credits FROM ENROLL E
JOIN CLASS C ON E.Cls_code = C.Cls_code
JOIN COURSE CR ON C.Crs_code = CR.Crs_code
JOIN SEMESTER SEM ON C.Sem_code = SEM.Sem_code
JOIN STUDENT S ON E.Reg_no = S.Reg_no
WHERE S.Reg_no = '2MSC1' AND SEM.Term = 'Win01' AND SEM.Year = '2018' GROUP BY S.Reg_no;
```

```
SQL> SELECT S.Reg_no, SUM(CR.Credits) AS Total_Credits FROM ENROLL E
2 JOIN CLASS C ON E.Cls_code = C.Cls_code
3 JOIN COURSE CR ON C.Crs_code = CR.Crs_code
4 JOIN SEMESTER SEM ON C.Sem_code = SEM.Sem_code
5 JOIN STUDENT S ON E.Reg_no = S.Reg_no
6 WHERE S.Reg_no = '2MSC1' AND SEM.Term = 'Win01' AND SEM.Year = '2018' GROUP BY S.Reg_no;

no rows selected
```

**(xxiv) Given the registration number of a student, display her/his grade in the course she/he registered in Fall 17-18.**

```
SELECT E.Cls_code, C.Crs_code, C.Crs_name, E.Grade
FROM ENROLL E
```

```

JOIN CLASS CL ON E.Cls_code = CL.Cls_code

JOIN COURSE C ON CL.Crs_code = C.Crs_code

JOIN SEMESTER SEM ON CL.Sem_code = SEM.Sem_code

WHERE SEM.Term = 'Fall1'

AND SEM.Year = '2017'

AND E.Reg_no = : &reg_no;

```

```

SQL> SELECT E.Cls_code, C.Crs_code, C.Crs_name, E.Grade
2  FROM ENROLL E
3  JOIN CLASS CL ON E.Cls_code = CL.Cls_code
4  JOIN COURSE C ON CL.Crs_code = C.Crs_code
5  JOIN SEMESTER SEM ON CL.Sem_code = SEM.Sem_code
6  WHERE SEM.Term = 'Fall1'
7    AND SEM.Year = '2017'
8    AND E.Reg_no = : &reg_no;
Enter value for reg_no: 2MSC1
old 8:  AND E.Reg_no = : &reg_no
new 8:  AND E.Reg_no = : 2MSC1
SP2-0552: Bind variable "2" not declared.
SQL> |

```

**(xxv) Display the name of the courses that are not being offered in Winter 17–18.**

```

SELECT C.Crs_name FROM COURSE C WHERE C.Crs_code NOT IN ( SELECT CL.Crs_code FROM CLASS CL JOIN
SEMESTER SEM ON CL.Sem_code = SEM.Sem_code WHERE SEM.Term = 'Win01' AND SEM.Year = '2017' );

```

```

COURSE
NAME
-----
MCA
MBA
B.Tech
B.Sc
B.Com
M.Sc
Database
Operating

8 rows selected.

```

**(xxvi) Write necessary SQL statement to advance the start time and end time of every class by ten minutes in Fall 18–19.**

```

UPDATE CLASS SET Stime = Stime + INTERVAL '10' MINUTE, Etime = Etime + INTERVAL '10' MINUTE WHERE
Sem_code = ( SELECT Sem_code FROM SEMESTER WHERE Term = 'Fall' AND Year = '2017' );

```

```

SQL> UPDATE CLASS SET Stime = Stime + INTERVAL '10' MINUTE, Etime = Etime + INTERVAL '10' MINUTE WHERE
Sem_code = ( SELECT Sem_code FROM SEMESTER WHERE Term = 'Fall' AND Year = '2017' );

2 rows updated.

```

**(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.**

**BEFORE**

```
SQL> SELECT Sdate, Edate
2 FROM SEMESTER
3 WHERE Term = 'Fall'
4 AND Year = '2017';
```

START DATE	END DATE
-----	-----
01-MAR-17	15-NOV-17

UPDATE SEMESTER

SET Sdate = Sdate + INTERVAL '7' DAY,

Edate = Edate + INTERVAL '7' DAY

WHERE Term = 'Fall'

AND Year = '2017';

```
SQL> UPDATE SEMESTER
2 SET Sdate = Sdate + INTERVAL '7' DAY,
3 Edate = Edate + INTERVAL '7' DAY
4 WHERE Term = 'Fall'
5 AND Year = '2017';

1 row updated.
```

AFTER

```
SQL> SELECT Sdate, Edate
2 FROM SEMESTER
3 WHERE Term = 'Fall'
4 AND Year = '2017';
```

START DATE	END DATE
-----	-----
08-MAR-17	22-NOV-17

**(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 S.Sname

FROM STUDENT S

WHERE

(SELECT COUNT(\*)

FROM ENROLL E

WHERE E.Reg\_no = S.Reg\_no AND E.Grade = 'S') >=

(SELECT COUNT(\*) / 2

FROM ENROLL E

WHERE E.Reg\_no = S.Reg\_no);

```
STUDENT
NAME
-----
Aadi
Ayuł
```

**(xxix) Given the registration number of a student, find out his/her free slots.**

```
SELECT DISTINCT Cl.Slot
FROM CLASS Cl
WHERE Cl.Slot NOT IN (
    SELECT C.Slot
    FROM ENROLL E
    JOIN CLASS C ON E.Cls_code = C.Cls_code
    WHERE E.Reg_no = '2MSC1'
);
```

```
SQL> SELECT DISTINCT Cl.Slot
2 FROM CLASS Cl
3 WHERE Cl.Slot NOT IN (
4     SELECT C.Slot
5     FROM ENROLL E
6     JOIN CLASS C ON E.Cls_code = C.Cls_code
7     WHERE E.Reg_no = '2MSC1'
8 );

SLOT
-----
MOR
```

**(xxx) Find out the name list of students who have classes in the afternoon session only a specific day of the week.**

```
SELECT DISTINCT S.Sname FROM STUDENT S JOIN ENROLL E ON S.Reg_no = E.Reg_no JOIN CLASS C ON
E.Cls_code = C.Cls_code WHERE C.Day_of_week = 'Monday' -- Specify the day of the week AND Slot='Aft';
```

```
STUDENT
NAME
-----
Barshaa
```

**(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 INTERVAL DAY TO SECOND;
```

```
UPDATE CLASS SET Duration = Etime - Stime;
```

```
ERROR at line 2:
ORA-01873: the leading precision of the interval is too small
```

**(xxxiii) Find out the list of students who are undergoing MCA program.**

```
SELECT S.Reg_no, S.Sname FROM STUDENT S JOIN PROGRAMME P ON S.Dept_id = P.Dept_id WHERE  
P.Prog_name = 'MCA';
```

STUDENT REG NO	STUDENT NAME
-----	-----
2MCA1	Barshaa
2MSC1	Vikram

**(xxxiv) Display the name of programs and the name of school offering the program.**

```
SELECT Prog.Prog_name AS Program_Name, Sch.Scl_name AS School_Name FROM PROGRAMME Prog JOIN  
SCHOOL Sch ON Prog.SCode = Sch.SCode;
```

PROGRAM_NA	SCHOOL_NAME
-----	-----
MCA	Maths School
MBA	Arts School
B.Tech	Science School
B.Sc	Science School
B.Com	Commerce School
M.Sc	Science School

**(xxxv) Display the name of the departments and the name of the program controlled by the department.**

```
SELECT Dept.Dname AS Department_Name, Prog.Prog_name AS Program_Name FROM DEPARTMENT Dept  
JOIN PROGRAMME Prog ON Dept.Dept_id = Prog.Dept_id;
```

DEPARTMENT_NAME	PROGRAM_NA
-----	-----
Maths	MCA
Arts Dept	MBA
Science Dept	B.Tech
Science Dept	B.Sc
Commerce Dept	B.Com
Science Dept	M.Sc

**(xxxvi) Find the school which has highest school strength (i.e number of students)**

```
SELECT Scl_name, Student_Count FROM ( SELECT Sch.Scl_name, COUNT(Stu.Reg_no) AS Student_Count FROM  
STUDENT Stu JOIN DEPARTMENT Dept ON Stu.Dept_id = Dept.Dept_id JOIN SCHOOL Sch ON Dept.SCode =  
Sch.SCode GROUP BY Sch.Scl_name ORDER BY Student_Count DESC ) WHERE ROWNUM = 1;
```

SCL_NAME	STUDENT_COUNT
-----	-----
Arts School	3

---