The *Students*, *Enrollments*, *Courses, Departments* and *Faculty* table are defined as follows:

Column Name	Type
NetId	VARCHAR(10)
FirstName	VARCHAR(255)
LastName	VARCHAR(255)
Department	VARCHAR(100)

Column Name	Type
NetId	VARCHAR(10)
CRN	INT
Credits	INT
Score	REAL

Column Name	Type
CRN	INT
Title	VARCHAR(255)
Department	VARCHAR(100)
Instructor	VARCHAR(255)

Column Name	Type
Department	VARCHAR(100)
NumberOfStudents	INT
YearOfEstablishment	INT
DepartmentHead	VARCHAR(255)

Column Name	Type
FirstName	VARCHAR(255)
LastName	VARCHAR(255)
Department	VARCHAR(100)
ResearchArea	VARCHAR(255)

Q1. How big is your department?
For the table *Departments*, write a SQL query that returns the name (the column *Department*) of departments that are bigger (in terms of *NumberOfStudents*) compared to at least one other department.

Solution:

```
SELECT department
FROM   departments
WHERE  numberofstudents > ANY (SELECT numberofstudents
                               FROM   departments);
```

Alternative Solution:

```
SELECT department
FROM   departments
WHERE  numberofstudents > (SELECT Min(numberofstudents)
                           FROM   departments);
```

Q2. How balanced is your department?

For the table *Departments* and *Faculty*, write a SQL query that returns the name (the column *Department*) of Departments, in which the ratio of the number of faculty members and the number of students (*NumberOfStudents*) is greater than 0.25.

Solution:


```
SELECT D.department
FROM   departments D
WHERE  D.numberofstudents < 4 * (SELECT Count(*)
                                 FROM   faculty F
                                 WHERE  F.department = D.department);
```

Q3. How active is your department?
For the table *Students*, *Enrollments*, *Courses* and *Departments*, write a SQL query that returns the *DepartmentHead* for *Departments* where at least 50% of *NumberOfStudents* are enrolled in at least one course.

Solution:

```
SELECT departmenthead
FROM   departments D
WHERE  numberofstudents * 0.5 <= (SELECT Count(DISTINCT S.netid)
                                  FROM   students S,
                                         enrollments E
                                  WHERE  S.netid = E.netid
                                         AND S.department =
D.department)
```

Alternative Solution:

```
SELECT departmenthead
FROM   departments D,
       (SELECT R.dept,
               Count(*) AS ActiveStudentCount
        FROM   (SELECT S.department AS Dept,
                       Count(*)     AS coursecount
                FROM   students S,
                       enrollments E
                WHERE  S.netid = E.netid
                GROUP  BY S.department,
                          S.netid
                HAVING coursecount >= 1) AS R
        GROUP  BY R.dept) AS R2
WHERE  D.department = R2.dept
       AND R2.activestudentcount >= 0.5 * D.numberofstudents;
```

Q4. Who outperformed the non-departmental students?
For the table *Students*, *Enrollments*, *Courses* and *Departments*, write a SQL query that returns the *NetId* of *Students* from 'CS' *Department* whose *Score* in 'Database Systems' course is higher compared to all non-CS students taking the course. Sort the NetIds in ascending order.

Solution:

```
SELECT S.netid
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND S.department = 'CS'
       AND C.title = 'Database Systems'
       AND E.score > (SELECT Max(E.score)
                      FROM   students S,
                             enrollments E,
                             courses C
                      WHERE  S.netid = E.netid
                             AND E.crn = C.crn
                             AND S.department <> 'CS'
                             AND C.title = 'Database Systems')
ORDER  BY S.netid;
```


Alternative Solution:


```
SELECT S.netid
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND S.department = 'CS'
       AND C.title = 'Database Systems'
       AND E.score > ALL (SELECT E.score
                          FROM   students S,
                                 enrollments E,
                                 courses C
                          WHERE  S.netid = E.netid
                                 AND E.crn = C.crn
                                 AND C.title = 'Database Systems'
                                 AND S.department <> 'CS')
ORDER  BY S.netid
```

Q5. How Many CS Courses Are You Taking?
For the table *Students*, *Enrollments* and *Courses*, write a SQL query that returns the *FirstName* and number of CS courses (*Department* = *'CS'*) for all students who are enrolled in one or more CS courses. Sort your results first by number of CS courses (in descending order) and then by *FirstName* (in ascending order).

Solution:

```
SELECT S.firstname,
       Count(E.crn) AS numCSCrs
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND C.department = 'CS'
GROUP  BY S.netid,
          S.firstname
HAVING Count(E.crn) > 0
ORDER  BY numcscrs DESC,
          S.firstname ASC
```

Q6. What Course Are You Taking?
For the table *Students*, *Enrollments* and *Courses*, write a SQL query that returns the *NetId* and *Department* of all students who are taking courses offered by a different department than their own department.

Solution:

```
SELECT netid,
       department
FROM   students
WHERE  EXISTS (SELECT courses.department
               FROM   enrollments,
                      courses
               WHERE  enrollments.crn = courses.crn
                      AND enrollments.netid = students.netid
                      AND courses.department <> students.department);
```

Alternative Solution:

```
SELECT DISTINCT S.netid,
                S.department
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND C.crn IN (SELECT crn
                     FROM   courses
                     WHERE  crn = E.crn
                            AND department <> S.department)
```

Q7. New student on board

For the table *Students*, *Enrollments* and *Courses*:

1. Insert the following information into the appropriate table(s). A student 'Mickey Mouse' with *NetId* 'mm1' joins 'CS' department. He is taking 'Computer Graphics' course (*CRN* 195) offered by Prof. 'Donald Duck' from 'CS' department.
2. Write a SQL query that returns the *LastName* of student, *Title* of course he/she is taking, and the *Score*, for all students in 'CS' department.

Note that, you can write as many SQL statements as you need to insert data. Just maker sure, you write the SQL query after the insert statements.

Solution:

```
INSERT INTO students VALUE
            (
                        'mm1',
                        'Mickey',
                        'Mouse',
                        'CS'
            );INSERT INTO courses VALUE
            (
                        '195',
                        'Computer Graphics',
                        'CS',
                        'Donald Duck'
            );INSERT INTO enrollments
            (
                        netid,
                        crn
            )
            value
            (
                        'mm1',
                        '195'
            );
SELECT s.lastname,
       c.title,
       e.score
FROM   students s,
       courses c,
       enrollments e
WHERE  s.netid = e.netid
       AND c.crn = e.crn
       AND s.department = 'CS';
```

Q8. Who Scored the Highest?
For the table *Students*, *Enrollments* and *Courses*, write a SQL query that returns the *NetId* of students who received highest score for a non-departmental (a department other than his own) course among all students who attended that course.

Solution:

```
SELECT DISTINCT S.netid
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND S.department <> C.department
       AND E.score = (SELECT Max(score)
                      FROM   enrollments
                      WHERE  crn = E.crn)
```

Alternative Solution:

```
SELECT S.netid
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND S.department <> C.department
       AND E.score >= ALL (SELECT E2.score
                           FROM   enrollments E2
```

Q9. Do you know the department head?
For the table *Students*, *Enrollments*, *Courses* and *Departments*, write a SQL query that returns the *NetId* of all students who are taking courses offered by their *DepartmentHead*.

Solution:

```
SELECT S.netid
FROM   students S,
       enrollments E,
       courses C,
       departments D
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND S.department = C.department
       AND C.instructor = D.departmenthead;
```

Q10. Students taking same courses

For the table *Students*, *Enrollments* and *Courses*, write a SQL query that returns the pair of *FirstName* for *Students* who are enrolled in the exactly same courses. The resultant relation will list a pair once, having the alphabetically smaller *Firstname* in first column and the alphabetically bigger *Firstname* in second column, e.g., a tuple consisting of firstanmes 'Abcd' and 'Efg' will have 'Abcd' in first column and 'Efg' in second column. Also, the resultant relation will not have self pairs (a self pair involves two FirstNames of a single student).

Solution:

```
SELECT S1.firstname,
       S2.firstname
FROM   (SELECT E1.netid AS n1,
               E2.netid AS n2,
               Count(*) AS nosc
        FROM   enrollments E1,
               enrollments E2
        WHERE  E1.netid != E2.netid
               AND E1.crn = E2.crn
        GROUP  BY E1.netid,
                  E2.netid) AS Temp,
       students AS S1,
       students AS S2
WHERE  nosc = (SELECT Count(*)
               FROM   enrollments
               WHERE  netid = n1)
       AND nosc = (SELECT Count(*)
                   FROM   enrollments
                   WHERE  netid = n2)
       AND S1.netid = n1
       AND S2.netid = n2
       AND S1.firstname < S2.firstname
```

Q11. Inactive students and faculty members

For the table *Students*, *Enrollments*, *Courses, Departments* and *Faculty*, write a SQL query that returns the *LastName* of inactive students and faculty members. A student is inactive if he/she is not enrolled in any courses. A faculty member is inactive if he/she is not instructing any courses and is not serving as a department head.

Solution:

```
SELECT S.lastname
FROM   students S
WHERE  S.netid NOT IN (SELECT E2.netid
                       FROM   enrollments E2)
UNION
SELECT lastname
FROM   faculty
WHERE  Concat(firstname, ' ', lastname) NOT IN (SELECT departmenthead
                                                FROM   departments)
       AND Concat(firstname, ' ', lastname)NOT IN (SELECT instructor
                                                   FROM   courses);
```


Q12. Who outperformed the departmental students?

For the table *Students*, *Enrollments*, *Courses* and *Departments*, write a SQL query that returns the *NetId* of *Students* whose *Score* in a non-departmental course is higher compared to at least one departmental student taking the course. Sort the NetIds in ascending order. Also, the resultant relation should not list a *NetId* more than once.

Solution:


```
SELECT DISTINCT S.netid
FROM   students S,
       enrollments E,
       courses C
WHERE  S.netid = E.netid
       AND E.crn = C.crn
       AND S.department != C.department
       AND E.score > ANY (SELECT E1.score
                          FROM   students S1,
                                 enrollments E1,
                                 courses C1
                          WHERE  S1.netid = E1.netid
                                 AND E1.crn = E.crn
                                 AND S1.department = C.department)
ORDER  BY S.netid
```

Q13. Above average students I
For the table *Students*, *Enrollments*, *Courses* and *Departments*, write a SQL query that returns Title of each course and the number of students whose score in the course is higher than the average score for all students taking the course.

Solution:

```
SELECT DISTINCT C.title,
                Count(E.netid)
FROM   enrollments E,
       courses C
WHERE  E.crn = C.crn
       AND E.score > (SELECT Avg(score)
                      FROM   enrollments
                      WHERE  crn = E.crn)
GROUP  BY E.crn,
          C.title
```

Q14. Above average students II (is the same as Q9, with a different set of data)
For the table *Students*, *Enrollments*, *Courses* and *Departments*, write a SQL query that returns Title of each course and the number of students whose score in the course is higher than the average score for all students taking the course.

Solution:

```
SELECT R2.c1,
       Ifnull(R1.c2, 0)
FROM   (SELECT C.title AS C1,
               Count(*) AS C2
        FROM   students S,
               enrollments E,
               courses C
        WHERE  S.netid = E.netid
               AND E.crn = C.crn
               AND E.score > (SELECT Avg(E2.score)
                              FROM   enrollments E2
                              WHERE  E2.crn = E.crn)
        GROUP  BY C.title) AS R1
       RIGHT JOIN (SELECT title AS C1
                   FROM   courses) AS R2
               ON R1.c1 = R2.c1;
```

Q15. Deleting students

For the table *Students*, *Enrollments*, *Courses* and *Departments*:

1. Write a SQL statement that deletes the records of students from *Students* table whose average score for 'CS' courses is less than 80%.
2. Write a SQL query that returns the current *Students* table.

Note that, you need to write the statements in given order, first delete statement, then SQL query.

Solution:

```
DELETE FROM students
WHERE  netid IN (SELECT x.netid
                 FROM   (SELECT s1.netid
                         FROM   students s1,
                                enrollments e,
                                courses c,
                                departments d
                         WHERE  s1.netid = e.netid
                                AND e.crn = c.crn
                                AND c.department = 'CS'
                         GROUP  BY s1.netid
                         HAVING Avg(e.score) < 80) AS x);

SELECT *
FROM   students;
```