```
--B.The recruiting department needs to know which courses are most popular
with
--the students. Please provide them with a query which lists the name of
each
--course and the number of students in that course.  The two columns
should have
--headers “Course Name” and “# Students”, and the output should be sorted
first
--by # Students descending and then by course name ascending.

SELECT COURSE.NAME AS "COURSE NAME",
       COUNT(STUDENTCOURSE.STUDENTID) AS "# STUDENTS"
FROM STUDENTCOURSE JOIN COURSE ON STUDENTCOURSE.COURSEID = COURSE.ID
GROUP BY COURSE.NAME
ORDER BY 2 DESC, 1 ASC;

--C.Quite a few students have been complaining that the professors are
absent
--from some of their courses.

--Write a query to list the names of all courses where the # of faculty
assigned
--to those courses is zero. The output should be in alphabetical order
--by course name.

SELECT COURSE.NAME
FROM COURSE LEFT JOIN FACULTYCOURSE ON COURSE.ID = FACULTYCOURSE.COURSEID
GROUP BY COURSE.NAME
HAVING COUNT(FACULTYCOURSE.FACULTYID)=0
ORDER BY COURSE.NAME;

--Using the above, write a query to list the course names and the # of
students
--in those courses for all courses where there is no assigned faculty.
--The output should be ordered first by # of students descending and then
--by course name ascending.

SELECT COUNT(STUDENTCOURSE.STUDENTID) AS "# STUDENTS",
       A.  NAME AS "COURSE NAME"
FROM
(SELECT COURSE.NAME,COURSE.ID
FROM COURSE LEFT JOIN FACULTYCOURSE ON COURSE.ID = FACULTYCOURSE.COURSEID
GROUP BY COURSE.NAME,COURSE.ID
HAVING COUNT(FACULTYCOURSE.FACULTYID)=0
ORDER BY COURSE.NAME) A,STUDENTCOURSE
WHERE STUDENTCOURSE.COURSEID = A.ID
GROUP BY A.NAME
ORDER BY 1 DESC,2 ASC;

--D. The enrollment team is gathering analytics about student enrollment
--throughout the years. Write a query that lists the total # of students
that
--were enrolled in classes during each school year. The first column
should have
```

```
--the header “Students”.  Provide a second “Year” column showing the
enrollment
--year. Display the records sorted in ascending order based on startDate.

SELECT COUNT(STUDENTID) AS "STUDENTS",
      TO_CHAR(STARTDATE,'YYYY') AS "YEAR"
FROM STUDENTCOURSE
GROUP BY TO_CHAR(STARTDATE,'YYYY')
ORDER BY 2 ASC;

--E. The enrollment team is gathering analytics about student enrollment
and
--they now want to know about August admissions specifically. Write a
query that
--lists the Start Date and # of Students who enrolled in classes in August
of
--each year. The Output should be ordered by start date ascending.

--CHECK AUGUST CLAUSE
SELECT STARTDATE AS "START DATE",
       COUNT(STUDENTID) AS "# STUDENTS"
FROM STUDENTCOURSE
WHERE TO_CHAR(STARTDATE,'MM')=08
GROUP BY STARTDATE
ORDER BY STARTDATE ASC;

--F. Students are required to take 4 courses, and at least two of these
courses
--must be from the department of their major. Write a query to list
students’
--First Name, Last Name, and the number of courses they are taking in
their
--major department. The output should be sorted first in increasing order
of
--the number of courses, then by student last name. Use aliases “First
Name”,
--“Last Name” and “Number of Courses”.

SELECT S.FIRSTNAME,S.LASTNAME,COUNT(SC.COURSEID)
FROM STUDENT S, DEPARTMENT D, COURSE C, STUDENTCOURSE SC
WHERE S.MAJORID = D.ID
  AND C.DEPTID = D.ID
  AND S.ID = SC.STUDENTID
  AND SC.COURSEID = C.ID
GROUP BY SC.COURSEID,S.FIRSTNAME,S.LASTNAME;

--G. Students making average progress in their courses of less than 50%
need to
--be offered tutoring assistance. Write a query to list First Name, Last
Name
--and Average Progress of all students achieving average progress of less
than
--50%.  The average progress as displayed should be rounded to one decimal
```

```
--place. Sort the output by average progress descending. Use aliases
“First
--Name”, “Last Name” and “Average Progress”.

SELECT STUDENT.FIRSTNAME AS "FIRST NAME",
       STUDENT.LASTNAME AS "LAST NAME",
       ROUND(A.AVERAGE,1) AS "AVERAGE PROGRESS"
FROM
(SELECT AVG(PROGRESS) AS "AVERAGE",STUDENTID
FROM STUDENTCOURSE
GROUP BY STUDENTID
HAVING AVG(PROGRESS)<50
ORDER BY 1 DESC) A, STUDENT
WHERE STUDENT.ID = A.STUDENTID;

--H. Faculty is awarded bonuses based on the progress made by students in
their
--courses.
--Write a query to list each course name and the average progress of
students in
--that course. The output should be sorted descending by average progress.
Use
--aliases “Course Name” and “Average Progress”.

SELECT COURSE.NAME AS "COURSE NAME",
       ROUND(AVG(PROGRESS),2) AS "AVERAGE PROGRESS"
FROM STUDENTCOURSE JOIN COURSE ON STUDENTCOURSE.COURSEID = COURSE.ID
GROUP BY COURSE.NAME
ORDER BY 2 DESC;

--Write a query that selects the maximum value of the average progress
reported
--by the previous query.

SELECT MAX(A.AVERAGEPROGRESS) FROM
(SELECT COURSE.NAME AS "COURSENAME",
        AVG(PROGRESS) AS "AVERAGEPROGRESS"
FROM STUDENTCOURSE JOIN COURSE ON STUDENTCOURSE.COURSEID = COURSE.ID
GROUP BY COURSE.NAME
ORDER BY 2 DESC) A;

--Write a query that outputs the faculty First Name, Last Name, and the
average
--of the progress made over all of their courses. Use aliases “First
Name”, “Last
--Name” and “Average Progress”.
--Sort the output by Average Progress ascending.


SELECT FACULTY.FIRSTNAME AS "FIRST NAME",
       FACULTY.LASTNAME AS "LAST NAME",
       ROUND(AVG(A."AVERAGE"),3) AS "AVERAGE PROGRESS"
FROM
  (SELECT COURSEID,AVG(STUDENTCOURSE.PROGRESS) AS "AVERAGE"
```

```
    FROM STUDENTCOURSE JOIN COURSE ON STUDENTCOURSE.COURSEID = COURSE.ID
    GROUP BY COURSE.ID) A,
    FACULTYCOURSE,
    FACULTY
WHERE FACULTYCOURSE.COURSEID = A.COURSEID
  AND FACULTY.ID = FACULTYCOURSE.FACULTYID
GROUP BY FACULTYCOURSE.FACULTYID,FACULTY.FIRSTNAME,FACULTY.LASTNAME
ORDER BY 3 ASC;

--Write a query just like #3, but where only those faculties where average
--progress in their courses is 90% or more of the maximum observed in #2.
--Display the records sorted in descending order based on “Average
Progress”.

SELECT FACULTY.FIRSTNAME AS "FIRST NAME",
       FACULTY.LASTNAME AS "LAST NAME",
       ROUND(AVG(A."AVERAGE"),3) AS "AVERAGE PROGRESS"
FROM
  (SELECT COURSEID,AVG(STUDENTCOURSE.PROGRESS) AS "AVERAGE"
    FROM STUDENTCOURSE JOIN COURSE ON STUDENTCOURSE.COURSEID = COURSE.ID
    GROUP BY COURSE.ID) A,
    FACULTYCOURSE,
    FACULTY
WHERE FACULTYCOURSE.COURSEID = A.COURSEID
  AND FACULTY.ID = FACULTYCOURSE.FACULTYID
GROUP BY FACULTYCOURSE.FACULTYID,FACULTY.FIRSTNAME,FACULTY.LASTNAME
HAVING "AVERAGE PROGRESS"> (SELECT MAX(A.AVERAGEPROGRESS) FROM
(SELECT COURSE.NAME AS "COURSENAME",
        AVG(PROGRESS) AS "AVERAGEPROGRESS"
FROM STUDENTCOURSE JOIN COURSE ON STUDENTCOURSE.COURSEID = COURSE.ID
GROUP BY COURSE.NAME
ORDER BY 2 DESC) A) OR "AVERAGE PROGRESS" > 90
ORDER BY 3 ASC;


--Students are awarded two grades based on the minimum and maximum
progress
--they are making in the courses. The grading scale is as follows:
--                        Progress < 40:            F
--                        Progress < 50:            D
--                        Progress < 60:            C
--                        Progress < 70:            B
--                        Progress >= 70:           A
--Write a query which displays each student’s “First Name”, “Last Name”,
--minimum progress with “Min Grade” as ALIAS, and maximum progress with
ALIAS
--“Max Grade”. Display the records sorted in ascending order based on
--“Average Progress”.

SELECT STUDENT.FIRSTNAME AS "FIRST NAME",
       STUDENT.LASTNAME AS "LAST NAME",
       CASE
       WHEN AVG(PROGRESS)<40 THEN 'F'
       WHEN AVG(PROGRESS)<50 THEN 'D'
```

```
        WHEN AVG(PROGRESS)<60 THEN 'C'
        WHEN AVG(PROGRESS)<70 THEN 'B'
        WHEN AVG(PROGRESS)>=70 THEN 'A'
        END
FROM STUDENT JOIN STUDENTCOURSE ON STUDENT.ID=STUDENTCOURSE.STUDENTID
GROUP BY STUDENTCOURSE.COURSEID;

SELECT STUDENTCOURSE.STUDENTID;


--J. Write a query that returns student's full name with "Student Name" as
alias
--whose progress is greater than the average progress for their course.

SELECT FIRSTNAME||' '||LASTNAME AS "STUDENT NAME"
FROM STUDENT JOIN STUDENTCOURSE SC ON STUDENT.ID = SC.STUDENTID
WHERE PROGRESS>;


SELECT FIRSTNAME||' '||LASTNAME AS "STUDENT NAME"
FROM
  (SELECT COURSEID, AVG(PROGRESS) AS "AVERAGE"
  FROM STUDENTCOURSE
  WHERE STUDENTCOURSE.STUDENTID = SC.STUDENTID
  GROUP BY COURSEID) A, STUDENT
WHERE STUDENT.ID = STUDENTCOURSE.STUDENTID
  AND STUDENTCOURSE.COURSEID = A.COURSEID
  AND STUDENTCOURSE.PROGRESS > A."AVERAGE";

--CALCULATING AVERAGE PROGRESS FOR EACH COURSE
SELECT COURSEID, AVG(PROGRESS)
FROM STUDENTCOURSE
GROUP BY COURSEID;
```