Mid

2.Find the total number of(distinct)students who have taken course sections taught by the instructor with ID 110011

```
SELECT COUNT(DISTINCT t.ID) AS total_students
FROM takes t
JOIN teaches te ON t.course_id = te.course_id AND t.sec_id = te.sec_id AND t.semester = te.semester AND t.year = te.year
WHERE te.ID = '110011';
```

3.find out the names of the instructors in the Computer Science department who have salary greater than $90,000.

```
SELECT name
FROM instructor
WHERE dept_name = 'Computer Science' AND salary > 90000;
```

4. find out all courses offered in Fall 2017 and in spring 2018.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2017) OR (semester = 'Spring' AND year = 2018);
```

5. increase salaries of instructors whose salary is over $100,000 by 3% and all others by 5%

```
UPDATE instructor
SET salary = CASE
   WHEN salary > 100000 THEN salary * 1.03
   ELSE salary * 1.05
END;
```

1. Find out the ID and salary of the instructors.

```
SELECT ID, salary
FROM instructor;
```

2. Find out the ID and salary of the instructor who gets more than $85,000.

```
SELECT ID, salary
FROM instructor
WHERE salary > 85000;
```

3. Find out the department names and their budget at the university.

```
SELECT dept_name, budget
FROM department;
```

4. find out the names of the instructors from Computer Science who have salary greater than $90,000.

```
SELECT name
FROM instructor
WHERE dept_name = 'Computer Science' AND salary > 90000;
```

5. For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.

```
SELECT DISTINCT instructor.name, teaches.course_id
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID;
```

6. Find the names of all instructors whose salary is greater than at least one instructor in the Biology department.

```
SELECT DISTINCT i1.name
FROM instructor i1, instructor i2
WHERE i1.salary > i2.salary AND i2.dept_name = 'Biology';
```

7. Find the advisor of the student with ID 12345

```
SELECT advisor.i_ID
FROM advisor
WHERE advisor.s_ID = '12345';
```

8. Find the average salary of all instructors.

```
SELECT AVG(salary) AS average_salary
FROM instructor;
```

9. Find the names of all departments whose building name includes the substring Watson.

```
SELECT dept_name
FROM department
WHERE building LIKE '%Watson%';
```

10. Find the names of instructors with salary amounts between $90,000 and $100,000.

```
SELECT name
FROM instructor
WHERE salary BETWEEN 90000 AND 100000;
```

11. Find the instructor names and the courses they taught for all instructors in the Biology department who have taught some course.

```
SELECT DISTINCT instructor.name, teaches.course_id
FROM instructor
JOIN teaches ON instructor.ID = teaches.ID
WHERE instructor.dept_name = 'Biology';
```

12. Find the courses taught in Fall-2009 semester.

```
SELECT course_id
FROM section
WHERE semester = 'Fall' AND year = 2009;
```

13. Find the set of all courses taught either in Fall-2009 or in Spring-2010.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2009) OR (semester = 'Spring' AND year = 2010);
```

14. Find the set of all courses taught in the Fall-2009 as well as in Spring-2010.

```
SELECT DISTINCT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2017) OR (semester = 'Spring' AND year = 2018);
```

15. Find all courses taught in the Fall-2009 semester but not in the Spring-2010 semester.

```
SELECT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2009)
INTERSECT
SELECT course_id
FROM section
WHERE (semester = 'Spring' AND year = 2010);
```

16. Find all instructors who appear in the instructor relation with null values for salary.

```
SELECT *
FROM instructor
WHERE salary IS NULL;
```

17. Find the average salary of instructors in the Finance department.

```
SELECT AVG(salary) AS average_salary
FROM instructor
WHERE dept_name = 'Finance';
```

18. Find the total number of instructors who teach a course in the Spring-2010 semester.

```
SELECT COUNT(DISTINCT ID) AS num_instructors
FROM teaches
WHERE (course_id, sec_id, semester, year) IN (
   SELECT course_id, sec_id, 'Spring', 2010
   FROM section
);
```

19. Find the average salary in each department.

```
SELECT dept_name, AVG(salary) AS avg_salary
FROM instructor
GROUP BY dept_name;
```

20. Find the number of instructors in each department who teach a course in the Spring-2010 semester.

```
SELECT dept_name, COUNT(DISTINCT ID) AS num_instructors
FROM teaches
WHERE (course_id, sec_id, semester, year) IN (
   SELECT course_id, sec_id, 'Spring', 2010
   FROM section
)
GROUP BY dept_name;
```

21. List out the departments where the average salary of the instructors is more than $42,000.

```
SELECT dept_name
FROM instructor
GROUP BY dept_name
HAVING AVG(salary) > 42000;
```

22. For each course section offered in 2009, find the average total credits (tot cred) of all students enrolled in the section, if the section had at least 2 students.

```
SELECT course_id, sec_id, AVG(tot_cred) AS avg_total_credits
FROM takes
JOIN student ON takes.ID = student.ID
WHERE (semester, year) IN ('Fall', 2009)
GROUP BY course_id, sec_id
HAVING COUNT(DISTINCT takes.ID) >= 2;
```

23. Find all the courses taught in both the Fall-2009 and Spring-2010 semesters.

```
SELECT course_id
FROM section
```

```
GROUP BY course_id
HAVING COUNT(DISTINCT CASE WHEN (semester, year) IN (('Fall', 2009), ('Spring',
2010)) THEN CONCAT(semester, year) END) = 2;
```

24. Find all the courses taught in the Fall-2009 semester but not in the Spring-2010 semester.

```
SELECT course_id
FROM section
WHERE (semester, year) = ('Fall', 2009)
EXCEPT
SELECT course_id
FROM section
WHERE (semester, year) = ('Spring', 2010);
```

25. Select the names of instructors whose names are neither <Mozart= nor <Einstein=.

```
SELECT name
FROM instructor
WHERE name NOT IN ('Mozart', 'Einstein');
```

26. Find the total number of (distinct) students who have taken course sections taught by the instructor with ID 110011.

```
SELECT COUNT(DISTINCT ID) AS num_students
FROM takes
WHERE (course_id, sec_id, semester, year) IN (
   SELECT course_id, sec_id, semester, year
   FROM teaches
   WHERE ID = '110011'
);
```

27. Find the ID and names of all instructors whose salary is greater than at least one instructor in the History department.

```
SELECT i1.ID, i1.name
FROM instructor i1, instructor i2
WHERE i1.salary > i2.salary AND i2.dept_name = 'History';
```

28. Find the names of all instructors that have a salary value greater than that of each instructor in the Biology department.

```
SELECT DISTINCT i1.name
FROM instructor i1
WHERE i1.salary > ALL (SELECT salary FROM instructor WHERE dept_name = 'Biology');
```

29. Find the departments that have the highest average salary.

```sql
SELECT dept_name
FROM (
   SELECT dept_name, AVG(salary) AS avg_salary
   FROM instructor
   GROUP BY dept_name
   ORDER BY avg_salary DESC
   LIMIT 1
);
```

30. Find all courses taught in both the Fall 2009 semester and in the Spring-2010 semester.

```sql
SELECT course_id
FROM section
WHERE (semester = 'Fall' AND year = 2009)
INTERSECT
SELECT course_id
FROM section
WHERE (semester = 'Spring' AND year = 2010);
```

31. Find all students who have taken all the courses offered in the Biology department.

```sql
SELECT s.ID, s.name
FROM student s
WHERE NOT EXISTS (
   SELECT c.course_id
   FROM course c
   WHERE c.dept_name = 'Biology'
   EXCEPT
   SELECT t.course_id
   FROM takes t
   WHERE t.ID = s.ID
);
```

32. Find all courses that were offered at most once in 2009.

```sql
SELECT course_id
FROM section
WHERE year = 2009
GROUP BY course_id
HAVING COUNT(*) <= 1;
```

33. Find all courses that were offered at least twice in 2009.

```sql
SELECT course_id
FROM section
WHERE year = 2009
GROUP BY course_id
HAVING COUNT(*) >= 2;
```

34. Find the average instructors salaries of those departments where the average salary is greater than $42,000.

```
SELECT dept_name, AVG(salary) AS avg_salary
FROM instructor
GROUP BY dept_name
HAVING AVG(salary) > 42000;
```

35. Find the maximum across all departments of the total salary at each department.

```
SELECT dept_name, MAX(total_salary) AS max_total_salary
FROM (
   SELECT dept_name, SUM(salary) AS total_salary
   FROM instructor
   GROUP BY dept_name
) AS department_salaries;
```

36. List all departments along with the number of instructors in each department.

```
SELECT dept_name, COUNT(*) AS num_instructors
FROM instructor
GROUP BY dept_name;
```

37. Find the titles of courses in the Comp. Sci. department that has 3 credits.

```
SELECT title
FROM course
WHERE dept_name = 'Comp. Sci.' AND credits = 3;
```

38. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

```
SELECT DISTINCT t.ID
FROM teaches t
JOIN instructor i ON t.ID = i.ID
WHERE i.name = 'Einstein';
```

39. Find the highest salary of any instructor.

```
SELECT MAX(salary) AS highest_salary
FROM instructor;
```

40. Find all instructors earning the highest salary (there may be more than one with the same salary).

```
SELECT ID, name, salary
FROM instructor
```

```
WHERE salary = (SELECT MAX(salary) FROM instructor);
```

41. Find the enrollment of each section that was offered in Autumn-2009.

```
SELECT course_id, sec_id, semester, year, COUNT(ID) AS enrollment
FROM takes
WHERE semester = 'Fall' AND year = 2009
GROUP BY course_id, sec_id, semester, year;
```

42. Find the maximum enrollment, across all sections, in Autumn-2009.

```
SELECT MAX(enrollment) AS max_enrollment
FROM (
    SELECT COUNT(ID) AS enrollment
    FROM takes
    WHERE semester = 'Fall' AND year = 2009
    GROUP BY course_id, sec_id, semester, year
) AS section_enrollments;
```

43. Find the salaries after the following operation: Increase the salary of each instructor in the Comp. Sci. department by 10%.

```
UPDATE instructor
SET salary = salary * 1.1
WHERE dept_name = 'Comp. Sci.';
```

44. Find all students who have not taken a course.

```
SELECT s.ID, s.name
FROM student s
WHERE NOT EXISTS (
    SELECT *
    FROM takes t
    WHERE t.ID = s.ID
);
```

45. List all course sections offered by the Physics department in the Fall-2009 semester, with the building and room number of each section.

```
SELECT s.course_id, s.sec_id, s.building, s.room_number
FROM section s
JOIN course c ON s.course_id = c.course_id
WHERE c.dept_name = 'Physics' AND s.semester = 'Fall' AND s.year = 2009;
```

46. Find the student names who take courses in Spring-2010 semester at Watson Building.

```
SELECT DISTINCT st.name
FROM student st
```

```
JOIN takes t ON st.ID = t.ID
JOIN section s ON t.course_id = s.course_id AND t.sec_id = s.sec_id
WHERE s.semester = 'Spring' AND s.year = 2010 AND s.building = 'Watson';
```

47. List the students who take courses teaches by Brandt.

```
SELECT DISTINCT st.name
FROM student st
JOIN takes t ON st.ID = t.ID
JOIN teaches te ON t.course_id = te.course_id AND t.sec_id = te.sec_id
JOIN instructor i ON te.ID = i.ID
WHERE i.name = 'Brandt';
```

48. Find out the average salary of the instructor in each department.

```
SELECT dept_name, AVG(salary) AS avg_salary
FROM instructor
GROUP BY dept_name;
```

49. Find the number of students who take the course titled " Intro. To Computer Science".

```
SELECT COUNT(DISTINCT ID) AS num_students
FROM takes
WHERE course_id = 'IntroCS';
```

50. Find out the total salary of the instructors of the Computer Science department who take
a
course(s) in Watson building.

```
SELECT SUM(salary) AS total_salary
FROM instructor
WHERE dept_name = 'Computer Science' AND ID IN (
    SELECT DISTINCT ID
    FROM teaches
    WHERE (building, room_number) IN (
        SELECT building, room_number
        FROM section
        WHERE building = 'Watson'
    )
);
```

51. Find out the course titles which starts between 10:00 to 12:00.

```
SELECT DISTINCT c.title
FROM course c
JOIN section s ON c.course_id = s.course_id
JOIN time_slot t ON s.time_slot_id = t.time_slot_id
WHERE t.start_hr >= 10 AND t.start_hr < 12;
```

52. List the course names where CS-1019 is the pre-requisite course.

```
SELECT c.title
FROM course c
JOIN prereq p ON c.course_id = p.course_id
WHERE p.prereq_id = 'CS-1019';
```

53. List the student names who get more than B+ grades in their respective courses.

```
SELECT s.name
FROM student s
JOIN takes t ON s.ID = t.ID
WHERE t.grade > 'B+';
```

54. Find the student who takes the maximum credit from each department.

```
SELECT s.ID, s.name, s.dept_name, MAX(tot_cred) AS max_credits
FROM student s
GROUP BY s.dept_name;
```

55. Find out the student ID and grades who take a course(s) in Spring-2009 semester.

```
SELECT t.ID, t.grade
FROM takes t
JOIN section s ON t.course_id = s.course_id AND t.sec_id = s.sec_id
WHERE s.semester = 'Spring' AND s.year = 2009;
```

56. Find the building(s) where the student takes the course titled Image Processing.

```
SELECT DISTINCT s.building
FROM student s
JOIN takes t ON s.ID = t.ID
JOIN section sec ON t.course_id = sec.course_id AND t.sec_id = sec.sec_id
JOIN course c ON sec.course_id = c.course_id
WHERE c.title = 'Image Processing';
```

57. Find the room no. and the building where the student from Fall-2009 semester can take a
course(s)

```
SELECT DISTINCT sec.room_number, sec.building
FROM section sec
JOIN takes t ON sec.course_id = t.course_id AND sec.sec_id = t.sec_id
WHERE t.semester = 'Fall' AND t.year = 2009;
```

58. Find the names of those departments whose budget is higher than that of Astronomy. List them in alphabetic order

```
SELECT dept_name
FROM department
WHERE budget > (SELECT budget FROM department WHERE dept_name = 'Astronomy')
ORDER BY dept_name;
```

59. Display a list of all instructors, showing each instructor's ID and the number of sections taught. Make sure to show the number of sections as 0 for instructors who have not taught any section

```
SELECT i.ID, i.name, COALESCE(COUNT(sec.course_id), 0) AS num_sections_taught
FROM instructor i
LEFT JOIN teaches t ON i.ID = t.ID
LEFT JOIN section sec ON t.course_id = sec.course_id AND t.sec_id = sec.sec_id
GROUP BY i.ID, i.name;
```

60. For each student who has retaken a course at least twice (i.e., the student has taken the course at least three times), show the course ID and the student's ID. Please display your results in order of course ID and do not display duplicate rows

```
SELECT t.course_id, t.ID
FROM takes t
WHERE t.grade IS NOT NULL
GROUP BY t.course_id, t.ID
HAVING COUNT(DISTINCT t.semester, t.year) >= 3
ORDER BY t.course_id, t.ID;
```

61. Find the names of Biology students who have taken at least 3 Accounting courses

```
SELECT s.name
FROM student s
JOIN takes t ON s.ID = t.ID
JOIN course c ON t.course_id = c.course_id
WHERE s.dept_name = 'Biology' AND c.dept_name = 'Accounting'
GROUP BY s.ID
HAVING COUNT(c.course_id) >= 3;
```

62. Find the sections that had maximum enrollment in Fall 2010

```
SELECT course_id, sec_id, MAX(enrollment) AS max_enrollment
FROM (
   SELECT course_id, sec_id, COUNT(ID) AS enrollment
   FROM takes
   WHERE semester = 'Fall' AND year = 2010
   GROUP BY course_id, sec_id
) AS section_enrollments
GROUP BY course_id, sec_id;
```

63. Find student names and the number of law courses taken for students who have taken at least half of the available law courses. (These courses are named things like 'Tort Law' or 'Environmental Law'

```
SELECT s.name, COUNT(DISTINCT c.course_id) AS num_law_courses
FROM student s
JOIN takes t ON s.ID = t.ID
JOIN course c ON t.course_id = c.course_id
WHERE c.title LIKE '%Law%'
GROUP BY s.name
HAVING COUNT(DISTINCT c.course_id) >= (SELECT COUNT(*) / 2 FROM course
WHERE title LIKE '%Law%');
```

64. Find the rank and name of the 10 students who earned the most A grades (A-, A, A+). Use alphabetical order by name to break ties.

```
SELECT RANK() OVER (ORDER BY COUNT(*) DESC) AS rank, ID, name, COUNT(*) AS
num_A_grades
FROM takes
WHERE grade IN ('A-', 'A', 'A+')
GROUP BY ID, name
ORDER BY num_A_grades DESC, name
LIMIT 10;
```

65. Find the titles of courses in the Comp. Sci. department that have 3 credits.

```
SELECT title
FROM course
WHERE dept_name = 'Comp. Sci.' AND credits = 3;
```

66. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.

```
SELECT DISTINCT t.ID
FROM teaches t
JOIN instructor i ON t.ID = i.ID
WHERE i.name = 'Einstein';
```

67. Find the ID and name of each student who has taken at least one Comp. Sci. course; make sure there are no duplicate names in the result.

```
SELECT DISTINCT s.ID, s.name
FROM student s
JOIN takes t ON s.ID = t.ID
JOIN course c ON t.course_id = c.course_id
WHERE c.dept_name = 'Comp. Sci.';
```

68. Find the course id, section id, and building for each section of a Biology course.

```
SELECT sec.course_id, sec.sec_id, sec.building
FROM section sec
JOIN course c ON sec.course_id = c.course_id
WHERE c.dept_name = 'Biology';
```

69. Output instructor names sorted by the ratio of their salary to their department's budget (in ascending order).

```
SELECT i.name, i.salary / d.budget AS salary_budget_ratio
FROM instructor i
JOIN department d ON i.dept_name = d.dept_name
ORDER BY salary_budget_ratio ASC;
```

70. Output instructor names and buildings for each building an instructor has taught in. Include instructor names who have not taught any classes (the building name should be NULL in this case).

```
SELECT i.name, sec.building
FROM instructor i
LEFT JOIN teaches t ON i.ID = t.ID
LEFT JOIN section sec ON t.course_id = sec.course_id AND t.sec_id = sec.sec_id;
```