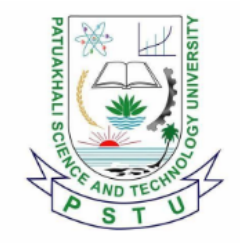

Patuakhali Science and Technology University
Faculty of Computer Science and Engineering

CCE 224 :: Database System Sessional

Sessional Project Report

Title : Full lab report
Submission Date : Sat 15, June 2025

Submitted to,

Prof. Dr. Md Samsuzzaman
Professor,
Department of Computer and Communication Engineering,
Patuakhali Science and Technology University.

Submitted by,

Md. Sharafat Karim
ID : 2102024,
Reg: 10151

Contents

Lab Problems

1. Problem set 1

1. 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');
```

```
# dept_name
'Athletics'
'Biology'
'Cybernetics'
'Finance'
'History'
'Math'
'Physics'
'Psychology'
```

2. 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, COUNT(T.ID) as number_of_sections
FROM
    instructor AS I
        NATURAL LEFT JOIN
    teaches AS T
GROUP BY I.ID
ORDER BY number_of_sections;
```

```
# ID, number_of_sections
'35579', '0'
'52647', '0'
'50885', '0'
'57180', '0'
'58558', '0'
'59795', '0'
'63395', '0'
'64871', '0'
'72553', '0'
'4034', '0'
'37687', '0'
'74426', '0'
'78699', '0'
'79653', '0'
'31955', '0'
```

```
'95030', '0'
'96895', '0'
'16807', '0'
'97302', '0'
'15347', '1'
'73623', '1'
'65931', '1'
'80759', '1'
'90376', '1'
'90643', '1'
'48570', '1'
'25946', '1'
'50330', '1'
'4233', '1'
'42782', '1'
'48507', '1'
'14365', '2'
'63287', '2'
'3335', '2'
'28400', '2'
'81991', '2'
'28097', '2'
'41930', '3'
'19368', '3'
'34175', '3'
'43779', '4'
'95709', '4'
'3199', '4'
'36897', '5'
'77346', '6'
'79081', '6'
'74420', '6'
'99052', '9'
'6569', '10'
'22591', '13'
```

3. 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 distinct course_id, ID
  from takes
    group by ID, course_id having count(*) > 2
    order by course_id;
```

```
# course_id, ID
'362', '16480'
'362', '16969'
'362', '27236'
'362', '39925'
'362', '39978'
'362', '44881'
'362', '49611'
'362', '5414'
'362', '69581'
'362', '9993'
```

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

```
select name
from student natural join (
  select ID from takes
  where course_id in ( select course_id from course
    where dept_name = "Accounting")
    group by ID having count(*) > 2
    ) as T where dept_name = "Biology";
```

```
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(*) > 2;
```

```
# name
'Michael'
'Dalton'
'Shoji'
'Wehen'
'Uchiyama'
'Schill'
'Kaminsky'
'Giannoulis'
```

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

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

```
select course_id, sec_id
from takes
WHERE semester = 'Fall' AND year = 2010
group by course_id, sec_id
order by count(*) desc
limit 1;
```

```
# course_id, sec_id
'867', '2'
```

6. 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').

```sql
select name, count(*)
from student as st
natural join takes as tt
where tt.course_id in (
  select course_id
    from course
    where title like "%Law%"
)
group by st.ID
having count(*) > (
  select count(*)/2
    from course
    where title like "%Law%"
);
```

```
# name, count(*)
'Nakajima', '4'
'Nikut', '4'
'Hahn-', '4'
'Nanda', '4'
'Schinag', '4'
```

7. 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. Note: the browser SQLite does not support window functions.

```sql
select row_number() over () as rnk, P.name from (
  select name, count(*) as cnt
  from student S
  join takes T on S.ID = T.ID
  where T.grade in ("A+", "A", "A-")
  group by T.ID
    ) as P
order by P.cnt desc, P.name
limit 10;

select row_number() over(order by count(*) desc, name) as rnk, name
from student st
natural join takes tt
where tt.grade in ("A+", "A", "A-")
group by ID
order by count(*) desc, name
limit 10;
```

```
# rnk, name
'1', 'Lepp'
'2', 'Eller'
'3', 'Masri'
'4', 'Vries'
'5', 'Åström'
'6', 'Gandhi'
'7', 'Greene'
'8', 'Haigh'
'9', 'McCarter'
'10', 'Sanchez'
```

2. Problem set 2

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

```
select ID, salary from instructor;
```

```
-- small database
# ID, salary
'10101', '65000.00'
'12121', '90000.00'
'15151', '40000.00'
'22222', '95000.00'
'32343', '60000.00'
'33456', '87000.00'
'45565', '75000.00'
'58583', '62000.00'
'76543', '80000.00'
'76766', '72000.00'
'83821', '92000.00'
'98345', '80000.00'

-- big database
# ID, salary
'14365', '32241.56'
'15347', '72140.88'
'16807', '98333.65'
'19368', '124651.41'
'22591', '59706.49'
'25946', '90891.69'
'28097', '35023.18'
'28400', '84982.92'
'31955', '71351.42'
'3199', '82534.37'
'3335', '80797.83'
'34175', '115469.11'
'35579', '62579.61'
'36897', '43770.36'
'37687', '104563.38'
'4034', '61387.56'
'41930', '50482.03'
'4233', '88791.45'
'42782', '34272.67'
'43779', '79070.08'
'48507', '107978.47'
'48570', '87549.80'
'50330', '108011.81'
'50885', '32570.50'
'52647', '87958.01'
'57180', '43966.29'
'58558', '66143.25'
'59795', '48803.38'
'63287', '103146.87'
'63395', '94333.99'
'64871', '45310.53'
'6569', '105311.38'
'65931', '79866.95'
'72553', '46397.59'
```

```
'73623', '90038.09'
'74420', '121141.99'
'74426', '106554.73'
'77346', '99382.59'
'78699', '59303.62'
'79081', '47307.10'
'79653', '89805.83'
'80759', '45538.32'
'81991', '77036.18'
'90376', '117836.50'
'90643', '57807.09'
'95030', '54805.11'
'95709', '118143.98'
'96895', '119921.41'
'97302', '51647.57'
'99052', '93348.83'
```

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

```
select ID, salary from instructor
where salary > 85000;

-- small db
# ID, salary
'12121', '90000.00'
'22222', '95000.00'
'33456', '87000.00'
'83821', '92000.00'

-- big db
# ID, salary
'16807', '98333.65'
'19368', '124651.41'
'25946', '90891.69'
'34175', '115469.11'
'37687', '104563.38'
'4233', '88791.45'
'48507', '107978.47'
'48570', '87549.80'
'50330', '108011.81'
'52647', '87958.01'
'63287', '103146.87'
'63395', '94333.99'
'6569', '105311.38'
'73623', '90038.09'
'74420', '121141.99'
'74426', '106554.73'
'77346', '99382.59'
'79653', '89805.83'
'90376', '117836.50'
'95709', '118143.98'
'96895', '119921.41'
'99052', '93348.83'
```

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

```
select dept_name, budget from department;
```

```
-- small db
# dept_name, budget
'Biology', '90000.00'
'Comp. Sci.', '100000.00'
'Elec. Eng.', '85000.00'
'Finance', '120000.00'
'History', '50000.00'
'Music', '80000.00'
'Physics', '70000.00'

-- big db
# dept_name, budget
'Accounting', '441840.92'
'Astronomy', '617253.94'
'Athletics', '734550.70'
'Biology', '647610.55'
'Civil Eng.', '255041.46'
'Comp. Sci.', '106378.69'
'Cybernetics', '794541.46'
'Elec. Eng.', '276527.61'
'English', '611042.66'
'Finance', '866831.75'
'Geology', '406557.93'
'History', '699140.86'
'Languages', '601283.60'
'Marketing', '210627.58'
'Math', '777605.11'
'Mech. Eng.', '520350.65'
'Physics', '942162.76'
'Pol. Sci.', '573745.09'
'Psychology', '848175.04'
'Statistics', '395051.74'
```

4. List out the names of the instructors from Computer Science who have more than $70,000.

```
desc instructor;
select name from instructor
where salary > 70000;

-- small db
# name
'Wu'
'Einstein'
'Gold'
'Katz'
'Singh'
'Crick'
'Brandt'
'Kim'

-- big db
# name
'Bawa'
'Yazdi'
'Wieland'
'Liley'
'Atanassov'
```

```
'Moreira'
'Gustafsson'
'Bourrier'
'Bondi'
'Arias'
'Luo'
'Romero'
'Lent'
'Sarkar'
'Shuming'
'Bancilhon'
'Jaekel'
'McKinnon'
'Mingoz'
'Pimenta'
'Sullivan'
'Voronina'
'Kenje'
'Mahmoud'
'Levine'
'Valtchev'
'Bietzk'
'Sakurai'
'Mird'
'Dale'
```

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 I.name, T.course_id
from instructor I
natural join teaches T
order by I.name;

-- small db
# name, course_id
'Brandt', 'CS-190'
'Brandt', 'CS-190'
'Brandt', 'CS-319'
'Crick', 'BIO-101'
'Crick', 'BIO-301'
'Einstein', 'PHY-101'
'El Said', 'HIS-351'
'Katz', 'CS-101'
'Katz', 'CS-319'
'Kim', 'EE-181'
'Mozart', 'MU-199'
'Srinivasan', 'CS-101'
'Srinivasan', 'CS-315'
'Srinivasan', 'CS-347'
'Wu', 'FIN-201'

-- large db
# name, course_id
'Atanassov', '603'
```

```
'Atanassov', '604'
'Bawa', '457'
'Bietzk', '158'
'Bondi', '571'
'Bondi', '274'
'Bondi', '747'
'Bourrier', '960'
'Bourrier', '949'
'Choll', '461'
'DAgostino', '663'
'DAgostino', '338'
'DAgostino', '338'
'DAgostino', '352'
'DAgostino', '400'
'DAgostino', '400'
'DAgostino', '991'
'DAgostino', '642'
'DAgostino', '599'
'DAgostino', '482'
'DAgostino', '962'
'DAgostino', '972'
'DAgostino', '867'
'Dale', '893'
'Dale', '237'
'Dale', '629'
'Dale', '237'
'Dale', '927'
'Dale', '748'
'Dale', '802'
'Dale', '158'
'Dale', '496'
'Gustafsson', '169'
'Gustafsson', '169'
'Gustafsson', '631'
'Gustafsson', '561'
'Jaekel', '852'
'Jaekel', '334'
'Kean', '366'
'Kean', '808'
'Lembr', '200'
'Lembr', '843'
'Lent', '626'
'Liley', '192'
'Luo', '679'
'Mahmoud', '704'
'Mahmoud', '735'
'Mahmoud', '735'
'Mahmoud', '493'
'Mahmoud', '864'
'Mahmoud', '486'
'Mingoz', '362'
'Mingoz', '527'
'Mingoz', '137'
'Mingoz', '362'
'Mingoz', '426'
'Mingoz', '304'
```

```
'Mingoz', '319'
'Mingoz', '445'
'Mingoz', '349'
'Mingoz', '362'
'Morris', '696'
'Morris', '791'
'Morris', '795'
'Morris', '313'
'Morris', '242'
'Pimenta', '875'
'Queiroz', '559'
'Romero', '105'
'Romero', '489'
'Romero', '105'
'Romero', '476'
'Sakurai', '468'
'Sakurai', '960'
'Sakurai', '258'
'Sakurai', '270'
'Sarkar', '867'
'Shuming', '468'
'Sullivan', '694'
'Tung', '692'
'Tung', '401'
'Tung', '421'
'Ullman ', '408'
'Ullman ', '974'
'Ullman ', '345'
'Ullman ', '200'
'Ullman ', '760'
'Ullman ', '408'
'Valtchev', '702'
'Valtchev', '415'
'Vicentino', '793'
'Voronina', '376'
'Voronina', '239'
'Voronina', '959'
'Voronina', '443'
'Voronina', '612'
'Voronina', '443'
'Wieland', '581'
'Wieland', '591'
'Wieland', '545'
```

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

```
select name
from instructor
where dept_name = "Biology"
and salary > ( select min(S.salary) from (
  select salary
    from instructor
    where dept_name = "Biology"
) as S );

-- big db
```

```
# name
'Valtchev'
```

7. Find the advisor of the student with ID 12345

```
select * from advisor
where s_ID = 12345;

-- small db
# s_ID, i_ID
'12345', '10101'
```

8. Find the average salary of all instructors.

```
select avg(I.salary) average_salary from
(select salary from instructor) as I;

-- small db
# average_salary
'74833.333333'

-- big db
# average_salary
'77600.188200'
```

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

```
select dept_name from department
where building like "%Watson%";

-- small db
# dept_name
'Biology'
'Physics'
```

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;

-- small db
# name
'Wu'
'Einstein'
'Brandt'

-- large db
# name
'Yazdi'
'Liley'
'McKinnon'
'Sullivan'
'Mahmoud'
'Dale'
```

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

have taught some course.

```
select name, course_id
from instructor
natural left join teaches
where dept_name = "Biology";

-- small db
# name, course_id
'Crick', 'BIO-101'
'Crick', 'BIO-301'

-- large db
# name, course_id
'Queiroz', '559'
'Valtchev', '415'
'Valtchev', '702'
```

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

```
select course_id from teaches
where semester = "Fall" and year = 2009;

-- big db
# course_id
'105'
'237'
'242'
'304'
'334'
'486'
'960'
```

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

```
select course_id from teaches
where (semester = "Fall" and year = 2009) or (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
union ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- big db
# course_id
'105'
'237'
'242'
'270'
'304'
'334'
'443'
'486'
'493'
'679'
'692'
'735'
'960'
```

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

```
select course_id from teaches
where (semester = "Fall" and year = 2009) and (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
intersect ( select course_id from teaches
where semester = "Spring" and year = 2010 );
```

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

```
select course_id from teaches
where (semester = "Fall" and year = 2009) and not (semester = "Spring" and year =
2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
except ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- big db
# course_id
'105'
'237'
'242'
'304'
'334'
'486'
'960'
```

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

```
select * from instructor
where salary = NULL;
```

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

```
select avg(T.salary) from
( select salary from instructor
where dept_name = "Finance" ) as T;

-- small db
# avg(T.salary)
'85000.000000'

-- big db
# avg(T.salary)
'105311.380000'
```

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

```
select  count(T.id) from
( select id from instructor
  natural join teaches
    where semester = "Spring"
    and year = 2010 ) as T ;

-- big db
# count(T.id)
'6'
```

19. Find the average salary in each department.

```
select dept_name, avg(salary)
from department
natural join instructor
group by dept_name;

-- small db
# dept_name, avg(salary)
'Biology', '72000.000000'
'Comp. Sci.', '77333.333333'
'Elec. Eng.', '80000.000000'
'Finance', '85000.000000'
'History', '61000.000000'
'Music', '40000.000000'
'Physics', '91000.000000'

-- big db
# dept_name, avg(salary)
'Accounting', '48716.592500'
'Athletics', '77098.198000'
'Pol. Sci.', '100053.073333'
'Psychology', '61143.050000'
'Languages', '57421.856667'
'English', '72089.050000'
'Statistics', '67795.441667'
'Elec. Eng.', '74162.740000'
'Comp. Sci.', '98133.470000'
'Marketing', '84097.437500'
'Astronomy', '79070.080000'
'Mech. Eng.', '79813.020000'
'Physics', '114576.900000'
'Cybernetics', '96346.567500'
'Finance', '105311.380000'
'Geology', '99382.590000'
'Biology', '61287.250000'
```

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

```
select dept_name, count(ID)
from department
natural join instructor
where ID in (
  select ID from teaches
    where semester = "Spring"
    and year = 2010
)
group by dept_name;

--big db
# dept_name, count(ID)
'Athletics', '1'
'English', '2'
'Physics', '1'
'Geology', '1'
```

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

```sql
select dept_name
from department D
where ( select avg(salary) from (
  select salary
    from instructor I
    where D.dept_name = I.dept_name
) as T ) > 42000;

select distinct dept_name
from instructor
group by dept_name
having avg(salary) > 42000;

-- small db
# dept_name
'Biology'
'Comp. Sci.'
'Elec. Eng.'
'Finance'
'History'
'Physics'

-- big db
# dept_name
'Accounting'
'Astronomy'
'Athletics'
'Biology'
'Comp. Sci.'
'Cybernetics'
'Elec. Eng.'
'English'
'Finance'
'Geology'
'Languages'
'Marketing'
'Mech. Eng.'
'Physics'
'Pol. Sci.'
'Psychology'
'Statistics'
```

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.

```sql
select course_id, sec_id, avg(tot_cred)
from takes
natural join student
where year = 2009
group by sec_id, course_id
having count(*) > 1;

-- big db
# course_id, sec_id, avg(tot_cred)
'105', '1', '68.3578'
```

```
'237', '2', '65.6656'
'242', '1', '64.4576'
'304', '1', '64.9023'
'334', '1', '62.8806'
'486', '1', '64.8980'
'604', '1', '65.7233'
'960', '1', '66.0847'
'972', '1', '65.2607'
```

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

```
select course_id from teaches
where (semester = "Fall" and year = 2009) and (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
intersect ( select course_id from teaches
where semester = "Spring" and year = 2010 );
```

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

```
select course_id from teaches
where (semester = "Fall" and year = 2009) and not (semester = "Spring" and year =
2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
except ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- big db
# course_id
'105'
'237'
'242'
'304'
'334'
'486'
'960'
```

25. Select the names of instructors whose names are neither ‘Mozart’ nor ‘Einstein’.

```
select name from instructor
where name not in ("Mozart", "Einstein");

-- small db
# name
'Srinivasan'
'Wu'
'El Said'
'Gold'
'Katz'
'Califieri'
'Singh'
'Crick'
'Brandt'
'Kim'
```

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

with ID 110011.

```
select count(T.ID)
from takes T
natural join section S
where "110011" in (
  select ID
    from teaches ST
    where T.course_id = ST.course_id
    and T.sec_id = ST.sec_id
    and T.semester = ST.semester
    and T.year = ST.year
);

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';
```

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

department.

```
select ID, name
from instructor
where salary > ( select min(T.salary) from (
  select salary
    from instructor
    where dept_name = "History"
) as T );

-- small db
# ID, name
'10101', 'Srinivasan'
'12121', 'Wu'
'22222', 'Einstein'
'33456', 'Gold'
'45565', 'Katz'
'58583', 'Califieri'
'76543', 'Singh'
'76766', 'Crick'
'83821', 'Brandt'
'98345', 'Kim'
```

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

Biology department.

```
select name
from instructor
where salary > ( select max(T.salary) from (
  select salary
    from instructor
    where dept_name = "Biology"
) as T );
```

```
-- small db
# name
'Wu'
'Einstein'
'Gold'
'Katz'
'Singh'
'Brandt'
'Kim'

-- big db
# name
'Yazdi'
'Wieland'
'Liley'
'Atanassov'
'Gustafsson'
'Bourrier'
'Bondi'
'Arias'
'Luo'
'Romero'
'Lent'
'Sarkar'
'Shuming'
'Bancilhon'
'Jaekel'
'McKinnon'
'Mingoz'
'Pimenta'
'Sullivan'
'Voronina'
'Kenje'
'Mahmoud'
'Levine'
'Bietzk'
'Sakurai'
'Mird'
'Dale'
```

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

```
select dept_name
from instructor
group by dept_name
order by avg(salary) desc
limit 1;

-- small db
# dept_name
'Physics'

-- small db
# dept_name
'Physics'
```

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

```
select course_id from teaches
where (semester = "Fall" and year = 2009) and (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
intersect ( select course_id from teaches
where semester = "Spring" and year = 2010 );
```

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

```
with cnt as (
  select count(course_id) as ct
    from course
    where dept_name = "Biology"
)
select ID
from takes
where course_id in (
  select course_id
    from course
    where dept_name = "Biology"
)
group by ID
having count(*) = (
  select ct from cnt
);
```

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

```
select course_id
from takes
where year = 2009
group by course_id
having count(*) = 1;
```

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

```
select course_id
from takes
where year = 2009
group by course_id
having count(*) > 1;
```

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

```
select dept_name
from department D
where ( select avg(salary) from (
  select salary
    from instructor I
    where D.dept_name = I.dept_name
) as T ) > 42000;

select distinct dept_name
from instructor
group by dept_name
having avg(salary) > 42000;
```

```
-- small db
# dept_name
'Biology'
'Comp. Sci.'
'Elec. Eng.'
'Finance'
'History'
'Physics'

-- big db
# dept_name
'Accounting'
'Astronomy'
'Athletics'
'Biology'
'Comp. Sci.'
'Cybernetics'
'Elec. Eng.'
'English'
'Finance'
'Geology'
'Languages'
'Marketing'
'Mech. Eng.'
'Physics'
'Pol. Sci.'
'Psychology'
'Statistics'
```

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

```
select sum(salary)
from department
natural join instructor
group by dept_name
order by sum(salary) desc
limit 1;

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

-- small db
# max_total_salary
'232000.00'

-- big db
# max_total_salary
'406772.65'
```

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

```
select dept_name, count(ID)
from department
natural left join instructor
```

```
group by dept_name;

-- small data
# dept_name, count(ID)
'Biology', '1'
'Comp. Sci.', '3'
'Elec. Eng.', '1'
'Finance', '2'
'History', '2'
'Music', '1'
'Physics', '2'

-- big data
# dept_name, count(ID)
'Accounting', '4'
'Astronomy', '1'
'Athletics', '5'
'Biology', '2'
'Civil Eng.', '0'
'Comp. Sci.', '2'
'Cybernetics', '4'
'Elec. Eng.', '4'
'English', '4'
'Finance', '1'
'Geology', '1'
'History', '0'
'Languages', '3'
'Marketing', '4'
'Math', '0'
'Mech. Eng.', '2'
'Physics', '2'
'Pol. Sci.', '3'
'Psychology', '2'
'Statistics', '6'
```

3. Problem set 3

1. 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;

-- small db
# title
'Robotics'
'Image Processing'
'Database System Concepts'

-- large db
# title
'International Finance'
'Computability Theory'
'Japanese'
```

2. Find the IDs of all students who were taught by an instructor named Einstein; make

sure there are no duplicates in the result.

```
select takes_table.ID
from takes takes_table
join teaches teaches_table
  on takes_table.course_id = teaches_table.course_id
    and takes_table.sec_id = teaches_table.sec_id
    and takes_table.semester = teaches_table.semester
    and takes_table.year = teaches_table.year
where teaches_table.ID = ( select ID from instructor
where name = "Einstein" ) ;

-- small db
# ID
'44553'
```

3. 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 ID, name
from student
natural join takes
where takes.course_id in (
  select course_id
    from course
    where dept_name = "Comp. Sci."
);

-- small db
# ID, name
'00128', 'Zhang'
'12345', 'Shankar'
'45678', 'Levy'
'54321', 'Williams'
'76543', 'Brown'
'98765', 'Bourikas'

-- big db
-- big output
```

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

```
select course_id, sec_id, building
from section
natural join course
where dept_name = "Biology";

-- small db
# course_id, sec_id, building
'BIO-101', '1', 'Painter'
'BIO-301', '1', 'Painter'

-- big db
# course_id, sec_id, building
'415', '1', 'Lamberton'
'559', '1', 'Lamberton'
'702', '1', 'Saucon'
```

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

```
select name
from instructor
natural left join department
order by (salary/ budget) asc;
```

```
-- small db
# name
'Mozart'
'Srinivasan'
'Singh'
'Wu'
'Katz'
'Crick'
'Brandt'
'Kim'
'El Said'
'Califieri'
'Gold'
'Einstein'
```

```
-- big data
# name
'Konstantinides'
'Kean'
'Tung'
'Queiroz'
'DAgostino'
'Lembr'
'Soisalon-Soininen'
'Yin'
'Desyl'
'Murata'
'Bawa'
'Bertolino'
'Hau'
'Pimenta'
'Ullman '
'Shuming'
'Gutierrez'
'Dale'
'McKinnon'
'Valtchev'
'Mingoz'
'Vicentino'
'Romero'
'Voronina'
'Yazdi'
'Arinb'
'Jaekel'
'Luo'
'Choll'
'Bietzk'
'Pingr'
```

```
'Liley'
'Sarkar'
'Bancilhon'
'Moreira'
'Sakurai'
'Lent'
'Morris'
'Atanassov'
'Wieland'
'Mahmoud'
'Arias'
'Gustafsson'
'Dusserre'
'Levine'
'Sullivan'
'Kenje'
'Mird'
'Bourrier'
'Bondi'
```

6. 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 name, building
from instructor
natural left join teaches
natural left join section;

-- small db
# name, building
'Srinivasan', 'Packard'
'Srinivasan', 'Watson'
'Srinivasan', 'Taylor'
'Wu', 'Packard'
'Mozart', 'Packard'
'Einstein', 'Watson'
'El Said', 'Painter'
'Gold', NULL
'Katz', 'Packard'
'Katz', 'Watson'
'Califieri', NULL
'Singh', NULL
'Crick', 'Painter'
'Crick', 'Painter'
'Brandt', 'Taylor'
'Brandt', 'Taylor'
'Brandt', 'Taylor'
'Kim', 'Taylor'

-- big db
# name, building
'Lembr', 'Saucon'
'Lembr', 'Fairchild'
'Bawa', 'Saucon'
'Yazdi', NULL
```

```
'Wieland', 'Saucon'
'Wieland', 'Alumni'
'Wieland', 'Saucon'
'DAgostino', 'Fairchild'
'DAgostino', 'Stabler'
'DAgostino', 'Lambeau'
'DAgostino', 'Lambeau'
'DAgostino', 'Main'
'DAgostino', 'Whitman'
'DAgostino', 'Chandler'
'DAgostino', 'Saucon'
'DAgostino', 'Fairchild'
'DAgostino', 'Taylor'
'DAgostino', 'Nassau'
'DAgostino', 'Taylor'
'DAgostino', 'Lamberton'
'Liley', 'Polya'
'Kean', 'Saucon'
'Kean', 'Polya'
'Atanassov', 'Taylor'
'Atanassov', 'Bronfman'
'Moreira', NULL
'Gustafsson', 'Gates'
'Gustafsson', 'Drown'
'Gustafsson', 'Main'
'Gustafsson', 'Taylor'
'Bourrier', 'Saucon'
'Bourrier', 'Lamberton'
'Bondi', 'Main'
'Bondi', 'Power'
'Bondi', 'Gates'
'Soisalon-Soininen', NULL
'Morris', 'Fairchild'
'Morris', 'Chandler'
'Morris', 'Saucon'
'Morris', 'Polya'
'Morris', 'Lamberton'
'Arias', NULL
'Murata', NULL
'Tung', 'Saucon'
'Tung', 'Gates'
'Tung', 'Taylor'
'Luo', 'Saucon'
'Vicentino', 'Nassau'
'Romero', 'Chandler'
'Romero', 'Taylor'
'Romero', 'Drown'
'Romero', 'Lamberton'
'Lent', 'Lamberton'
'Sarkar', 'Lamberton'
'Shuming', 'Power'
'Konstantinides', NULL
'Bancilhon', NULL
'Hau', NULL
'Dusserre', NULL
'Desyl', NULL
```

```
'Jaekel', 'Taylor'
'Jaekel', 'Gates'
'McKinnon', NULL
'Gutierrez', NULL
'Mingoz', 'Fairchild'
'Mingoz', 'Lamberton'
'Mingoz', 'Rathbone'
'Mingoz', 'Saucon'
'Mingoz', 'Lamberton'
'Mingoz', 'Alumni'
'Mingoz', 'Bronfman'
'Mingoz', 'Lamberton'
'Mingoz', 'Alumni'
'Mingoz', 'Saucon'
'Pimenta', 'Power'
'Yin', NULL
'Sullivan', 'Alumni'
'Voronina', 'Taylor'
'Voronina', 'Power'
'Voronina', 'Whitman'
'Voronina', 'Gates'
'Voronina', 'Lamberton'
'Voronina', 'Saucon'
'Kenje', NULL
'Mahmoud', 'Whitman'
'Mahmoud', 'Lamberton'
'Mahmoud', 'Taylor'
'Mahmoud', 'Drown'
'Mahmoud', 'Taylor'
'Mahmoud', 'Power'
'Pingr', NULL
'Ullman ', 'Chandler'
'Ullman ', 'Taylor'
'Ullman ', 'Taylor'
'Ullman ', 'Taylor'
'Ullman ', 'Garfield'
'Ullman ', 'Polya'
'Levine', NULL
'Queiroz', 'Lamberton'
'Valtchev', 'Lamberton'
'Valtchev', 'Saucon'
'Bietzk', 'Whitman'
'Cholll', 'Main'
'Arinb', NULL
'Sakurai', 'Main'
'Sakurai', 'Power'
'Sakurai', 'Lambeau'
'Sakurai', 'Power'
'Mird', NULL
'Bertolino', NULL
'Dale', 'Taylor'
'Dale', 'Power'
'Dale', 'Fairchild'
'Dale', 'Taylor'
'Dale', 'Stabler'
'Dale', 'Saucon'
```

```
'Dale', 'Saucon'
'Dale', 'Fairchild'
'Dale', 'Saucon'
```

4. 70 Queries (SQL Server)

```
--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. List out the names of the instructors from Computer Science who have more than
$70,000.
select name from instructor where dept_name='Comp. Sci.' and salary>70000;

--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 name, course_id
from instructor
left 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 name from instructor
where salary > (select min(salary)
  from instructor where dept_name='Biology');

--7. Find the advisor of the student with ID 12345
select i_id from advisor where s_ID=12345;

--8. Find the average salary of all instructors.
select avg(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 name, course.title from instructor
join teaches on instructor.ID=teaches.ID
join course on teaches.course_id=course.course_id
where instructor.dept_name='Biology';
```

```sql
--12. Find the courses taught in Fall-2009 semester.
select course_id from teaches
where semester='Fall' and year=2009;

--13. Find the set of all courses taught either in Fall-2009 or in Spring-2010.
select course_id from teaches
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 course_id from teaches
where (semester='Fall' and year=2009)
 and (semester='Spring' and year=2010);

--15. Find all courses taught in the Fall-2009 semester but not in the Spring-2010
semester.
select course_id from teaches
where (semester='Fall' and year=2009)
 and not (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) 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) from teaches
where (semester='Spring' and year=2010);

--19. Find the average salary in each department.
select dept_name, 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 count(distinct ID) from teaches
where (semester='Spring' and year=2010);

--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 average_tot
from takes join student
on takes.ID=student.ID
```

```
group by course_id, sec_id
having count(student.ID)>1;

--23. Find all the courses taught in both the Fall-2009 and Spring-2010 semesters.
select course_id from teaches
where (semester='Fall' and year=2009)
 and (semester='Spring' and year=2010);

--24. Find all the courses taught in the Fall-2009 semester but not in the
Spring-2010 semester.
select course_id from teaches
where (semester='Fall' and year=2009)
 and not (semester='Spring' and year=2010);

--25. Select the names of instructors whose names are neither &lt;Mozart= nor
&lt;Einstein=.
select name from instructor
where name != 'Mozart' and name !='Einstein';

--26. Find the total number of (distinct) students who have taken course sections
taught by the
--instructor with ID 110011.
select count(distinct takes.ID)
from instructor
join teaches on teaches.ID=instructor.ID
join takes on
 takes.course_id=teaches.course_id and
 takes.sec_id=teaches.sec_id and
 takes.semester=teaches.semester and
 takes.year=teaches.year
where instructor.ID=10101
group by instructor.ID;

--27. Find the ID and names of all instructors whose salary is greater than at least
one instructor in the
--History department.
select ID, name from instructor
where salary > (select min(salary)
 from instructor where 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 name
from instructor
where salary > (
 select min(salary)
 from instructor
 where dept_name='Biology'
);

--29. Find the departments that have the highest average salary.
select top 1
dept_name
from instructor
group by dept_name
```

```sql
order by avg(salary) desc;

--30. Find all courses taught in both the Fall 2009 semester and in the Spring-2010
semester.
select course_id
from teaches
where semester='Fall' and year=2009
 and semester='Spring' and year=2010;

--31. Find all students who have taken all the courses offered in the Biology
department.
select student.ID, name, count(student.ID)
from student
join takes
 on takes.ID=student.ID
join course
 on takes.course_id=course.course_id
where course.dept_name='Biology'
group by student.ID, name
having count(student.ID) = (
 select count(*)
 from course
 where dept_name='Biology'
);

--32. Find all courses that were offered at most once in 2009.
select teaches.course_id, title
from teaches
join course on course.course_id=teaches.course_id
where year=2009
group by teaches.course_id, title
having count(*)=1;

--33. Find all courses that were offered at least twice in 2009.
select teaches.course_id, title
from teaches
join course on course.course_id=teaches.course_id
where year=2009
group by teaches.course_id, title
having count(*)>=2;

--34. Find the average instructors9 salaries of those departments where the average
salary is greater
--than $42,000.
select dept_name
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 top 1
dept_name, sum(salary)
from instructor
group by dept_name
order by sum(salary) desc;
```

```sql
--36. List all departments along with the number of instructors in each department.
select dept_name, count(*)
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 takes.ID
from instructor
join teaches
 on teaches.ID=instructor.ID
join takes
 on takes.course_id=teaches.course_id
 and takes.sec_id=teaches.sec_id
 and takes.semester=teaches.semester
 and takes.year=teaches.year
where name='Einstein';

--39. Find the highest salary of any instructor.
select max(salary)
from instructor;

--40. Find all instructors earning the highest salary (there may be more than one
with the same
--salary).
select ID, name
from instructor
group by ID, name
order by sum(salary) desc;

--41. Find the enrollment of each section that was offered in Autumn-2009.
select sec_id, course_id
from takes
where semester='Autumn' and year=2009;

--42. Find the maximum enrollment, across all sections, in Autumn-2009.
select top 1
sec_id, course_id
from takes
where semester='Autumn' and year=2009
group by sec_id, course_id
order by count(*) desc;

--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 * 1.10;
where dept_name = 'Comp. Sci.';
```

```sql
--44. Find all students who have not taken a course.
select student.ID, name
from student
left join takes
 on student.ID=takes.ID
where course_id is NULL;

--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  teaches.sec_id, teaches.course_id, section.building, room_number
from teaches
join course
 on teaches.course_id=course.course_id
join section
 on teaches.course_id=section.course_id
 and teaches.sec_id=section.sec_id
 and teaches.semester=section.semester
 and teaches.year=section.year
join department
 on course.dept_name=department.dept_name
where course.dept_name='Physics'
 and teaches.semester='Fall'
 and teaches.year=2009
group by teaches.sec_id, teaches.course_id, section.building, room_number;

--46. Find the student names who take courses in Spring-2010 semester at Watson
Building.
select name
from student
join takes
 on student.ID=takes.ID
join section
 on takes.course_id=section.course_id
 and takes.sec_id=section.sec_id
 and takes.semester=section.semester
 and takes.year=section.year
where building='Watson'
 and takes.semester='Spring'
 and takes.year=2010;

--47. List the students who take courses teaches by Brandt.
select distinct takes.ID
from instructor
join teaches
 on teaches.ID=instructor.ID
join takes
 on takes.course_id=teaches.course_id
 and takes.sec_id=teaches.sec_id
 and takes.semester=teaches.semester
 and takes.year=teaches.year
where name='Brandt';

--48. Find out the average salary of the instructor in each department.
select dept_name, avg(salary) from instructor group by dept_name;
```

```sql
--49. Find the number of students who take the course titled 8Intro. To Computer
Science.
select count(*)
from student
join takes
 on takes.ID=student.ID
join course
 on takes.course_id=course.course_id
where title='Intro. To Computer Science';

--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)
from instructor
join teaches
 on instructor.ID=teaches.ID
join section
 on teaches.course_id=section.course_id
 and teaches.sec_id=section.sec_id
 and teaches.semester=section.semester
 and teaches.year=section.year
where dept_name='Comp. Sci.'
 and building = 'Watson';

--51. Find out the course titles which starts between 10:00 to 12:00.
select title
from course
join section
 on course.course_id=section.course_id
join time_slot
 on time_slot.time_slot_id=section.time_slot_id
where start_hr = 10
 or (start_hr > 10 and start_hr < 12)
 or (start_hr = 12 and start_min = 0);

--52. List the course names where CS-1019 is the pre-requisite course.
select title
from course
join prereq
 on course.course_id=prereq.course_id
where prereq_id = 'CS-1019';

--53. List the student names who get more than B+ grades in their respective courses.
select name
from student
join takes
 on student.ID=takes.ID
where grade like '%A%';

--54. Find the student who takes the maximum credit from each department.
select top 1
student.name, student.dept_name, sum(credits)
from student
join takes
```

```
 on student.ID = takes.ID
join course
 on course.course_id = takes.course_id
group by student.dept_name, student.name, student.ID
order by sum(credits) desc;

--55. Find out the student ID and grades who take a course(s) in Spring-2009
semester.
select student.ID, grade
from student
join takes
 on student.ID=takes.ID
where semester='Spring'
 and year=2009;

--56. Find the building(s) where the student takes the course titled Image
Processing.
select building
from section
join course
 on section.course_id = course.course_id
where title = 'Image Processing';

--57. Find the room no. and the building where the student from Fall-2009 semester
can take a
--course(s)
select building, room_number
from section
where semester = 'Fall'
 and 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'
);

--59. Display a list of all instructors, showing each instructor&amp;#39;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 instructor.ID, count(distinct sec_id)
from instructor
left join teaches
 on instructor.ID = teaches.ID
group by instructor.ID;

--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&amp;#39;s ID. Please
```

```
display your
--results in order of course ID and do not display duplicate rows
select ID, course_id
from takes
group by ID, course_id
having count(*) > 2;

--61. Find the names of Biology students who have taken at least 3 Accounting courses
select name
from student
join takes
 on student.ID = takes.ID
where dept_name = 'Biology'
 and course_id in (
  select course_id
  from course
  where dept_name = 'Accounting'
 )
group by student.ID, name
having count(*) > 2;

--62. Find the sections that had maximum enrollment in Fall 2010
select top 1
sec_id, course_id
from takes
where semester='Fall' and year=2010
group by sec_id, course_id
order by count(ID) desc;

SELECT course_id, sec_id
FROM takes
WHERE semester = 'Fall' AND year = 2010
GROUP BY course_id, sec_id
HAVING COUNT(ID) = (
    SELECT MAX(enrollment_count)
    FROM (
        SELECT COUNT(ID) AS enrollment_count
        FROM takes
        WHERE semester = 'Fall' AND year = 2010
        GROUP BY course_id, sec_id
    ) AS subquery
);

--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
&#39;Tort Law&#39; or
--&#39;Environmental Law&#39;
select name, count(*)
from student
join takes on student.ID=takes.ID
join course on takes.course_id=course.course_id
where title like '%Law%'
group by student.ID, name
having count(*) > (
 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 top 10
row_number() over(order by count(*) desc, name) as rank, name
from student
join takes
 on student.id = takes.id
where grade in ('A-', 'A', 'A+')
group by name, student.ID
order by count(*) desc, name;

--65. Find the titles of courses in the Comp. Sci. department that have 3 credits.
select title
from course
where credits = 3 and dept_name = 'Comp. Sci.';

--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 takes.ID
from instructor
join teaches
 on teaches.ID=instructor.ID
join takes
 on takes.course_id=teaches.course_id
 and takes.sec_id=teaches.sec_id
 and takes.semester=teaches.semester
 and takes.year=teaches.year
where 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 student.ID, name
from student
join takes
 on student.id = takes.id
where course_id in (
 select course_id
 from  course
 where dept_name = 'Comp. Sci.'
);


--68. Find the course id, section id, and building for each section of a Biology
course.
select section.course_id, sec_id, building
from section
join course
 on section.course_id = course.course_id
where dept_name = 'Biology';
```

```
--69. Output instructor names sorted by the ratio of their salary to their
department&amp;#39;s budget (in
--ascending order).
select name
from instructor
join department
 on instructor.dept_name = department.dept_name
order by (salary / budget);

--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 distinct name, building
from instructor
left join teaches
 on instructor.ID = teaches.ID
left join section
 on teaches.course_id = section.course_id
 and teaches.sec_id = section.sec_id
 and teaches.year = section.year
 and teaches.semester = section.semester;
```

5. W3 Resources

MySQL Basic Select Statement:

1.

```
-- Selecting the first_name column and aliasing it as "First Name"
SELECT first_name "First Name",
-- Selecting the last_name column and aliasing it as "Last Name"
last_name "Last Name"
-- Selecting data from the employees table
FROM employees;
```

2.

```
-- Selecting distinct values from the department_id column
SELECT DISTINCT department_id
-- Selecting data from the employees table
FROM employees;
```

3.

```
-- Selecting all columns from the employees table
SELECT *
-- Selecting data from the employees table
FROM employees
-- Ordering the result set by the first_name column in descending order
ORDER BY first_name DESC;
```

4.

```
-- Selecting the first_name, last_name, and salary columns
SELECT first_name, last_name, salary,
-- Calculating 15% of the salary and aliasing it as "PF" (Provident Fund)
salary * 0.15 AS PF
-- Selecting data from the employees table
FROM employees;
```

5.
```sql
-- Selecting specific columns: employee_id, first_name, last_name, and salary
SELECT employee_id, first_name, last_name, salary
-- Selecting data from the employees table
FROM employees
-- Ordering the result set by the salary column in ascending order
ORDER BY salary;
```

6.
```sql
-- Calculating the sum of salaries for all employees
SELECT SUM(salary)
-- Selecting data from the employees table
FROM employees;
```

7.
```sql
-- Selecting the maximum and minimum salary values from the employees table
SELECT MAX(salary), MIN(salary)
-- Selecting data from the employees table
FROM employees;
```

8.
```sql
-- Calculating the average salary and counting the total number of employees
SELECT AVG(salary), COUNT(*)
-- Selecting data from the employees table
FROM employees;
```

9.
```sql
-- Counting the total number of records (rows) in the employees table
SELECT COUNT(*)
-- Selecting data from the employees table
FROM employees;
```

10.
```sql
-- Counting the total number of distinct job IDs in the employees table
SELECT COUNT(DISTINCT job_id)
-- Selecting data from the employees table
FROM employees;
```

11.
```sql
-- Converting the first_name column values to uppercase
SELECT UPPER(first_name)
-- Selecting data from the employees table
FROM employees;
```

12.
```sql
-- Extracting the substring of the first three characters from the first_name column
SELECT SUBSTRING(first_name, 1, 3)
-- Selecting data from the employees table
FROM employees;
```

13.
```sql
-- Performing arithmetic operations: multiplication and addition
SELECT 171 * 214 + 625 Result;
```

14.

```sql
-- Concatenating the first_name and last_name columns with a space in between
SELECT CONCAT(first_name, ' ', last_name) 'Employee Name'
-- Selecting data from the employees table
FROM employees;
```

15.

```sql
-- Removing leading and trailing whitespace characters from the first_name column
SELECT TRIM(first_name)
-- Selecting data from the employees table
FROM employees;
```

16.

```sql
-- Selecting the first_name and last_name columns
SELECT first_name, last_name,
-- Calculating the sum of the lengths of first_name and last_name columns and
aliasing it as 'Length of Names'
LENGTH(first_name) + LENGTH(last_name) 'Length of Names'
-- Selecting data from the employees table
FROM employees;
```

17.

```sql
-- Selecting all columns from the employees table
SELECT *
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the first_name column contains
a digit
WHERE first_name REGEXP '[0-9]';
```

18.

```sql
-- Selecting the employee_id and first_name columns
SELECT employee_id, first_name
-- Selecting data from the employees table
FROM employees
-- Limiting the result set to only include the first 10 rows
LIMIT 10;
```

19.

```sql
-- Selecting the first_name, last_name, and calculating the monthly salary
SELECT first_name, last_name,
-- Dividing the salary by 12 to calculate the monthly salary and rounding it to 2
decimal places
ROUND(salary / 12, 2) AS 'Monthly Salary'
-- Selecting data from the employees table
FROM employees;
MySQL Restricting and Sorting data:
```

1.

```sql
-- Selecting the first_name, last_name, and salary columns
SELECT first_name, last_name, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the salary is not between
10000 and 15000
WHERE salary NOT BETWEEN 10000 AND 15000;
```

```
2.
-- Selecting the first_name, last_name, and department_id columns
SELECT first_name, last_name, department_id
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the department_id is either 30
or 100
WHERE department_id IN (30, 100)
-- Ordering the result set by the department_id column in ascending order
ORDER BY department_id ASC;

3.
-- Selecting the first_name, last_name, salary, and department_id columns
SELECT first_name, last_name, salary, department_id
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the salary is not between
10000 and 15000
-- and the department_id is either 30 or 100
WHERE salary NOT BETWEEN 10000 AND 15000
AND department_id IN (30, 100);

4.
-- Selecting the first_name, last_name, and hire_date columns
SELECT first_name, last_name, hire_date
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the year part of the hire_date
is like '1987%'
WHERE YEAR(hire_date) LIKE '1987%';

5.
-- Selecting the first_name column
SELECT first_name
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the first_name column contains
both 'b' and 'c'
WHERE first_name LIKE '%b%'
AND first_name LIKE '%c%';

6.
-- Selecting the last_name, job_id, and salary columns
SELECT last_name, job_id, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the job_id is either 'IT_PROG'
or 'SH_CLERK'
-- and the salary is not 4500, 10000, or 15000
WHERE job_id IN ('IT_PROG', 'SH_CLERK')
AND salary NOT IN (4500, 10000, 15000);

7.
-- Selecting the last_name column
SELECT last_name
-- Selecting data from the employees table
```

```sql
FROM employees
-- Filtering the result set to include only rows where the last_name column consists
of exactly six characters
WHERE last_name LIKE '______';

8.
-- Selecting the last_name column
SELECT last_name
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the last_name column starts
with two characters followed by 'e' and any other characters
WHERE last_name LIKE '__e%';

9.
-- Selecting distinct values from the job_id column
SELECT DISTINCT job_id
-- Selecting data from the employees table
FROM employees;

10.
-- Selecting the first_name, last_name, salary columns, and calculating 15% of the
salary as PF (Provident Fund)
SELECT first_name, last_name, salary, salary * 0.15 AS PF
-- Selecting data from the employees table
FROM employees;

11.
-- Selecting all columns from the employees table
SELECT *
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the last_name column matches
one of the specified values
WHERE last_name IN ('JONES', 'BLAKE', 'SCOTT', 'KING', 'FORD');

Aggregate Functions and Group by :

1.
-- Counting the number of distinct job IDs in the employees table
SELECT COUNT(DISTINCT job_id)
-- Selecting data from the employees table
FROM employees;

2.
-- Calculating the total sum of salaries for all employees
SELECT SUM(salary)
-- Selecting data from the employees table
FROM employees;

3.
-- Retrieving the minimum salary from the employees table
SELECT MIN(salary)
-- Selecting data from the employees table
FROM employees;
```

```
4.
-- Retrieving the maximum salary among employees with the job_id 'IT_PROG'
SELECT MAX(salary)
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees with the job_id 'IT_PROG'
WHERE job_id = 'IT_PROG';

5.
-- Calculating the average salary and counting the number of employees in department
90
SELECT AVG(salary), COUNT(*)
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees in department 90
WHERE department_id = 90;

6.
-- Calculating various statistics (maximum, minimum, sum, average) for the salary
column
SELECT
    -- Rounding the maximum salary to 0 decimal places and aliasing it as 'Maximum'
    ROUND(MAX(salary), 0) 'Maximum',
    -- Rounding the minimum salary to 0 decimal places and aliasing it as 'Minimum'
    ROUND(MIN(salary), 0) 'Minimum',
    -- Rounding the sum of salaries to 0 decimal places and aliasing it as 'Sum'
    ROUND(SUM(salary), 0) 'Sum',
    -- Rounding the average salary to 0 decimal places and aliasing it as 'Average'
    ROUND(AVG(salary), 0) 'Average'
-- Selecting data from the employees table
FROM employees;

7.
-- Counting the number of employees for each job ID
SELECT job_id, COUNT(*)
-- Selecting data from the employees table
FROM employees
-- Grouping the result set by the job_id column
GROUP BY job_id;

8.
-- Calculating the difference between the maximum and minimum salaries
SELECT MAX(salary) - MIN(salary) DIFFERENCE
-- Selecting data from the employees table
FROM employees;

9.
-- Selecting the manager_id and the minimum salary for each manager
SELECT manager_id, MIN(salary)
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where manager_id is not NULL
WHERE manager_id IS NOT NULL
-- Grouping the result set by manager_id
GROUP BY manager_id
-- Sorting the result set by the minimum salary in descending order
```

```
ORDER BY MIN(salary) DESC;
```

```
10.
-- Calculating the total salary for each department
SELECT department_id, SUM(salary)
-- Selecting data from the employees table
FROM employees
-- Grouping the result set by department_id
GROUP BY department_id;
```

```
11.
-- Calculating the average salary for each job, excluding 'IT_PROG'
SELECT job_id, AVG(salary)
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the job_id is not 'IT_PROG'
WHERE job_id <> 'IT_PROG'
-- Grouping the result set by job_id
GROUP BY job_id;
```

```
12.
-- Calculating various statistics for salaries of employees in department 90, grouped
by job_id
SELECT job_id, SUM(salary), AVG(salary), MAX(salary), MIN(salary)
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only rows where the department_id is '90'
WHERE department_id = '90'
-- Grouping the result set by job_id
GROUP BY job_id;
```

```
13.
-- Retrieving the maximum salary for each job title, filtering out job titles where
the maximum salary is less than 4000
SELECT job_id, MAX(salary)
-- Selecting data from the employees table
FROM employees
-- Grouping the result set by job_id
GROUP BY job_id
-- Filtering the result set to include only groups where the maximum salary is
greater than or equal to 4000
HAVING MAX(salary) >= 4000;
```

```
14.
-- Calculating the average salary and counting the number of employees for each
department, filtering out departments with fewer than 10 employees
SELECT department_id, AVG(salary), COUNT(*)
-- Selecting data from the employees table
FROM employees
-- Grouping the result set by department_id
GROUP BY department_id
-- Filtering the result set to include only groups where the count of employees is
greater than 10
HAVING COUNT(*) > 10;
```

MySQL Subquery :

```
1.
-- Selecting the first name, last name, and salary of employees whose salary is
higher than that of the employee with the last name 'Bull'
SELECT FIRST_NAME, LAST_NAME, SALARY
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is higher than the
salary of the employee with the last name 'Bull'
WHERE SALARY >
    -- Subquery to fetch the salary of the employee with the last name 'Bull'
    (SELECT salary FROM employees WHERE last_name = 'Bull');

2.
-- Selecting the first name and last name of employees
SELECT first_name, last_name
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose department_id is in the
set of department_ids where the department_name is 'IT'
WHERE department_id
IN (SELECT department_id FROM departments WHERE department_name='IT');

3.
-- Selecting the first name and last name of employees
SELECT first_name, last_name
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose manager_id is in the set
of employee_ids
-- where the department_id is in the set of department_ids associated with locations
in the US
WHERE manager_id in
    (SELECT employee_id
    -- Subquery to select employee_ids from the employees table
    FROM employees
    -- Filtering the employee_ids to include only those associated with departments
    -- where the location_id is in the set of location_ids associated with countries
having country_id 'US'
    WHERE department_id
    IN
        (SELECT department_id
        -- Subquery to select department_ids from the departments table
        FROM departments
        -- Filtering the department_ids to include only those associated with
locations
        -- where the country_id is 'US'
        WHERE location_id
        IN
            (SELECT location_id
            -- Subquery to select location_ids from the locations table
            FROM locations
            -- Filtering the location_ids to include only those associated with
countries
            -- having country_id 'US'
            WHERE country_id='US')
```

```
        )
    );
```

4.

```
-- Selecting the first name and last name of employees
SELECT first_name, last_name
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose employee_id is in the set
of manager_ids
WHERE (employee_id IN
    -- Subquery to select manager_ids from the employees table
    (SELECT manager_id FROM employees)
);
```

5.

```
-- Selecting the first name, last name, and salary of employees
SELECT first_name, last_name, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is greater than
the average salary of all employees
WHERE salary >
    -- Subquery to calculate the average salary from the employees table
    (SELECT AVG(salary) FROM employees);
```

6.

```
-- Selecting the first name, last name, and salary of employees
SELECT first_name, last_name, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary matches the
minimum salary defined for their job position
WHERE employees.salary =
    -- Subquery to select the minimum salary for each job position from the jobs
table
    (SELECT min_salary
    FROM jobs
    -- Matching the job_id of each employee with the job_id in the jobs table
    WHERE employees.job_id = jobs.job_id);
```

7.

```
Selecting the first name, last name, and salary of employees
SELECT first_name, last_name, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees who belong to departments with
names starting with 'IT'
-- and have a salary higher than the average salary of all employees
WHERE department_id IN
    -- Subquery to select department_ids from the departments table where the
department_name starts with 'IT'
    (SELECT department_id FROM departments WHERE department_name LIKE 'IT%')
-- Additional condition: Salary of employees should be higher than the average salary
of all employees
AND salary >
```

```
    -- Subquery to calculate the average salary from the employees table
    (SELECT avg(salary) FROM employees);

8.
S-- Selecting the first name, last name, and salary of employees
SELECT first_name, last_name, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is higher than
that of the employee with the last name 'Bell'
WHERE salary >
    -- Subquery to fetch the salary of the employee with the last name 'Bell'
    (SELECT salary FROM employees WHERE last_name = 'Bell')
-- Sorting the result set in ascending order based on the first name of employees
ORDER BY first_name;

9.
-- Selecting all columns from the employees table
SELECT *
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is equal to the
minimum salary among all employees
WHERE salary =
    -- Subquery to find the minimum salary from the employees table
    (SELECT MIN(salary) FROM employees);

10.
-- Selecting all columns from the employees table
SELECT *
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is greater than
all average salaries within their respective departments
WHERE salary >-- Selecting all columns from the employees table
SELECT *
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is greater than
all average salaries within their respective departments
WHERE salary >
    -- Subquery to calculate the average salary for each department and compare with
each employee's salary
    ALL(SELECT avg(salary) FROM employees GROUP BY department_id);

11.
-- Selecting the first name, last name, job ID, and salary of employees
SELECT first_name, last_name, job_id, salary
-- Selecting data from the employees table
FROM employees
-- Filtering the result set to include only employees whose salary is greater than
all salaries of employees with job_id 'SH_CLERK'
WHERE salary >
    -- Subquery to select all salaries of employees with job_id 'SH_CLERK' and
compare with each employee's salary
    ALL (SELECT salary FROM employees WHERE job_id = 'SH_CLERK')
```

```sql
-- Sorting the result set in ascending order based on salary
ORDER BY salary;
```

12.

```sql
-- Selecting the first name and last name of employees who are not managers
SELECT b.first_name, b.last_name
-- Selecting data from the employees table, aliasing it as 'b'
FROM employees b
-- Filtering the result set to include only employees who are not managers
WHERE NOT EXISTS
    -- Subquery to check if there is no employee with manager_id equal to the
employee_id of each employee in the outer query
    (SELECT 'X' FROM employees a WHERE a.manager_id = b.employee_id);
```

13.

```sql
-- Selecting the employee_id, first name, last name, and department name of
employees
SELECT employee_id, first_name, last_name,
-- Subquery to fetch the department name for each employee's department_id
(SELECT department_name FROM departments d
 -- Joining the employees table with the departments table based on the department_id
 WHERE e.department_id = d.department_id) department
-- Selecting data from the employees table, aliasing it as 'e'
 FROM employees e
-- Sorting the result set based on the department name
 ORDER BY department;
```

14.

```sql
-- Selecting the employee_id and first name of employees
SELECT employee_id, first_name
-- Selecting data from the employees table, aliasing it as 'A'
FROM employees AS A
-- Filtering the result set to include only employees whose salary is greater than
the average salary of their department
WHERE salary >
    -- Subquery to calculate the average salary for each department and compare with
each employee's salary
    (SELECT AVG(salary) FROM employees WHERE department_id = A.department_id);
```

15.

```sql
-- Setting user-defined variable @i to 0
SET @i = 0;
-- Selecting the sequential number and employee_id of every second row in the
employees table
SELECT i, employee_id
-- Subquery to generate a sequential number for each row in the employees table,
starting from 1
FROM (SELECT @i := @i + 1 AS i, employee_id FROM employees) a
-- Filtering the result set to include only rows with even sequential numbers
WHERE MOD(a.i, 2) = 0;
```

16.

```sql
-- Selecting distinct salaries from the employees table for which there are exactly 5
distinct salaries greater than or equal to it
SELECT DISTINCT salary
-- Selecting data from the employees table, aliasing it as 'e1'
```

```sql
FROM employees e1
-- Filtering the result set to include only those salaries for which there are
exactly 5 distinct salaries greater than or equal to it
WHERE 5 =
    -- Subquery to count the number of distinct salaries greater than or equal to
each salary
    (SELECT COUNT(DISTINCT salary)
    -- Selecting data from the employees table, aliasing it as 'e2'
    FROM employees e2
    -- Filtering the result set to include only distinct salaries greater than or
equal to the salary in the outer query (e1.salary)
    WHERE e2.salary >= e1.salary);

17.
-- Selecting distinct salaries from the employees table for which there are exactly 4
distinct salaries less than or equal to it
SELECT DISTINCT salary
-- Selecting data from the employees table, aliasing it as 'e1'
FROM employees e1
-- Filtering the result set to include only those salaries for which there are
exactly 4 distinct salaries less than or equal to it
WHERE 4 =
    -- Subquery to count the number of distinct salaries less than or equal to each
salary
    (SELECT COUNT(DISTINCT salary)
    -- Selecting data from the employees table, aliasing it as 'e2'
    FROM employees e2
    -- Filtering the result set to include only distinct salaries less than or equal
to the salary in the outer query (e1.salary)
    WHERE e2.salary <= e1.salary);

18.
-- Selecting all columns from a subset of the employees table, ordered by employee_id
in descending order, and limiting the result to the first 10 rows
SELECT * FROM (
    SELECT * FROM employees ORDER BY employee_id DESC LIMIT 10
) sub
-- Sorting the result set from the subset in ascending order based on employee_id
ORDER BY employee_id ASC;

19.
-- Selecting all columns from the departments table
SELECT *
-- Selecting data from the departments table
FROM departments
-- Filtering the result set to include only departments whose department_id is not
present in the set of department_ids associated with employees
WHERE department_id
NOT IN
    -- Subquery to select department_ids from the employees table
    (select department_id FROM employees);

20.
-- Selecting distinct salary values from the employees table
SELECT DISTINCT salary
-- Selecting data from the employees table, aliasing it as 'a'
```

```
FROM employees a
-- Filtering the result set to include only salary values where there are at most 3
distinct salary values greater than or equal to it
WHERE 3 >=
    -- Subquery to count the number of distinct salaries greater than or equal to
each salary
    (SELECT COUNT(DISTINCT salary)
    -- Selecting data from the employees table, aliasing it as 'b'
    FROM employees b
    -- Filtering the result set to include only distinct salaries greater than or
equal to the salary in the outer query (a.salary)
    WHERE b.salary >= a.salary)
-- Sorting the result set in descending order based on salary
ORDER BY a.salary DESC;

21.
-- Selecting distinct salary values from the employees table
SELECT DISTINCT salary
-- Selecting data from the employees table, aliasing it as 'a'
FROM employees a
-- Filtering the result set to include only salary values where there are at most 3
distinct salary values less than or equal to it
WHERE 3 >=
    -- Subquery to count the number of distinct salaries less than or equal to each
salary
    (SELECT COUNT(DISTINCT salary)
    -- Selecting data from the employees table, aliasing it as 'b'
    FROM employees b
    -- Filtering the result set to include only distinct salaries less than or equal
to the salary in the outer query (a.salary)
    WHERE b.salary <= a.salary)
-- Sorting the result set in descending order based on salary
ORDER BY a.salary DESC;


22.
-- Selecting all columns from the employees table, aliasing it as 'emp1'
SELECT *
-- Selecting data from the employees table, aliasing it as 'emp1'
FROM employees emp1
-- Filtering the result set to include only employees where the count of distinct
salaries greater than the salary of the current employee is 1
WHERE (1) = (
    -- Subquery to count the number of distinct salaries greater than the salary of
each employee
    SELECT COUNT(DISTINCT(emp2.salary))
    -- Selecting data from the employees table, aliasing it as 'emp2'
    FROM employees emp2
    -- Filtering the result set to include only distinct salaries greater than the
salary of the employee in the outer query (emp1.salary)
    WHERE emp2.salary > emp1.salary
);

MySQL Joins :

1.
```

```sql
-- This SQL query selects specific columns from the 'locations' table after
performing a natural join with the 'countries' table.

SELECT
    location_id, -- Selecting the 'location_id' column from the result set.
    street_address, -- Selecting the 'street_address' column from the result set.
    city, -- Selecting the 'city' column from the result set.
    state_province, -- Selecting the 'state_province' column from the result set.
    country_name -- Selecting the 'country_name' column from the result set.
FROM
    locations -- Specifying the 'locations' table.
NATURAL JOIN
    countries; -- Performing a natural join with the 'countries' table based on any
common columns.

2.
-- This SQL query selects specific columns from the 'employees' table after
performing an inner join with the 'departments' table using the 'department_id'
column.

SELECT
    first_name, -- Selecting the 'first_name' column from the result set.
    last_name, -- Selecting the 'last_name' column from the result set.
    department_id, -- Selecting the 'department_id' column from the result set.
    department_name -- Selecting the 'department_name' column from the result set.
FROM
    employees -- Specifying the 'employees' table.
JOIN
    departments -- Specifying the 'departments' table.
USING
    (department_id); -- Performing an inner join using the 'department_id' column,
which is common in both tables.



3.
-- This SQL query selects specific columns from the 'employees' and 'departments'
tables, as well as the 'locations' table, to retrieve information about employees in
the London city.

SELECT
    e.first_name, -- Selecting the 'first_name' column from the 'employees' table and
aliasing it as 'e'.
    e.last_name, -- Selecting the 'last_name' column from the 'employees' table and
aliasing it as 'e'.
    e.job_id, -- Selecting the 'job_id' column from the 'employees' table and
aliasing it as 'e'.
    e.department_id, -- Selecting the 'department_id' column from the 'employees'
table and aliasing it as 'e'.
    d.department_name -- Selecting the 'department_name' column from the
'departments' table and aliasing it as 'd'.
FROM
    employees e -- Specifying the 'employees' table and aliasing it as 'e'.
JOIN
    departments d -- Specifying the 'departments' table and aliasing it as 'd'.
ON
```

```
    (e.department_id = d.department_id) -- Performing a join between 'employees' and
'departments' based on the 'department_id' column.
JOIN
    locations l ON -- Joining the 'locations' table and aliasing it as 'l'.
    (d.location_id = l.location_id) -- Performing a join between 'departments' and
'locations' based on the 'location_id' column.
WHERE
    LOWER(l.city) = 'London'; -- Filtering the result to only include rows where the
city in lowercase is 'London'.
4.
-- This SQL query selects specific columns from the 'employees' table, twice aliased
as 'e' and 'm', to retrieve information about employees and their managers.

SELECT
    e.employee_id 'Emp_Id', -- Selecting the 'employee_id' column from the
'employees' table, aliased as 'e', and renaming it as 'Emp_Id'.
    e.last_name 'Employee', -- Selecting the 'last_name' column from the 'employees'
table, aliased as 'e', and renaming it as 'Employee'.
    m.employee_id 'Mgr_Id', -- Selecting the 'employee_id' column from the
'employees' table, aliased as 'm', and renaming it as 'Mgr_Id'.
    m.last_name 'Manager' -- Selecting the 'last_name' column from the 'employees'
table, aliased as 'm', and renaming it as 'Manager'.
FROM
    employees e -- Specifying the 'employees' table and aliasing it as 'e'.
JOIN
    employees m -- Joining the 'employees' table again and aliasing it as 'm'.
ON
    (e.manager_id = m.employee_id); -- Performing a join between 'employees' and
itself based on the 'manager_id' column to associate employees with their managers.

5.
-- This SQL query selects specific columns from the 'employees' table to retrieve
information about employees hired after an employee with the last name 'Jones'.

SELECT
    e.first_name, -- Selecting the 'first_name' column from the 'employees' table.
    e.last_name, -- Selecting the 'last_name' column from the 'employees' table.
    e.hire_date -- Selecting the 'hire_date' column from the 'employees' table.
FROM
    employees e -- Specifying the 'employees' table and aliasing it as 'e'.
JOIN
    employees davies -- Joining the 'employees' table again and aliasing it as
'davies'.
ON
    (davies.last_name = 'Jones') -- Performing a join based on the condition where
the last name in 'davies' is 'Jones'.
WHERE
    davies.hire_date < e.hire_date; -- Filtering the result to include only employees
hired after the employee with the last name 'Jones'.

6.
-- This SQL query retrieves the count of employees in each department, along with the
department names, from the 'departments' and 'employees' tables.

SELECT
    department_name AS 'Department Name', -- Selecting the 'department_name' column
```

```
from the 'departments' table and aliasing it as 'Department Name'.
    COUNT(*) AS 'No of Employees' -- Counting the number of records (employees) in
each department and aliasing it as 'No of Employees'.
FROM
    departments -- Specifying the 'departments' table.
INNER JOIN
    employees -- Performing an inner join with the 'employees' table.
ON
    employees.department_id = departments.department_id -- Joining the 'employees'
and 'departments' tables based on the 'department_id' column.
GROUP BY
    departments.department_id, department_name -- Grouping the result set by
department ID and department name.
ORDER BY
    department_name; -- Ordering the result set by department name in ascending
order.

7.
-- This SQL query retrieves specific columns from the 'job_history' table and
calculates the duration of each job in days for employees in the specified
department.

SELECT
    employee_id, -- Selecting the 'employee_id' column from the result set.
    job_title, -- Selecting the 'job_title' column from the result set.
    end_date - start_date AS Days -- Calculating the difference between 'end_date'
and 'start_date' columns and aliasing it as 'Days'.
FROM
    job_history -- Specifying the 'job_history' table.
NATURAL JOIN
    jobs -- Performing a natural join with the 'jobs' table.
WHERE
    department_id = 90; -- Filtering the result to include only records where the
department ID is 90.

8.
-- This SQL query retrieves specific columns from the 'departments' and 'employees'
tables to get information about department managers.

SELECT
    d.department_id, -- Selecting the 'department_id' column from the 'departments'
table.
    d.department_name, -- Selecting the 'department_name' column from the
'departments' table.
    d.manager_id, -- Selecting the 'manager_id' column from the 'departments' table.
    e.first_name -- Selecting the 'first_name' column from the 'employees' table and
aliasing it as 'e'.
FROM
    departments d -- Specifying the 'departments' table and aliasing it as 'd'.
INNER JOIN
    employees e -- Performing an inner join with the 'employees' table and aliasing
it as 'e'.
ON
    (d.manager_id = e.employee_id); -- Joining the 'departments' and 'employees'
tables based on the 'manager_id' column to associate departments with their managers.
```

```
9.
-- This SQL query retrieves specific columns from the 'departments', 'employees', and
'locations' tables to get information about department managers and their
corresponding locations.

SELECT
    d.department_name, -- Selecting the 'department_name' column from the
'departments' table.
    e.first_name, -- Selecting the 'first_name' column from the 'employees' table and
aliasing it as 'e'.
    l.city -- Selecting the 'city' column from the 'locations' table and aliasing it
as 'l'.
FROM
    departments d -- Specifying the 'departments' table and aliasing it as 'd'.
JOIN
    employees e -- Performing a join with the 'employees' table and aliasing it as
'e'.
ON
    (d.manager_id = e.employee_id) -- Joining the 'departments' and 'employees'
tables based on the 'manager_id' column to associate departments with their managers.
JOIN
    locations l USING (location_id); -- Performing a join with the 'locations' table
based on the 'location_id' column.

10.
-- This SQL query calculates the average salary for each job title by joining the
'employees' and 'jobs' tables.

SELECT
    job_title, -- Selecting the 'job_title' column from the result set.
    AVG(salary) -- Calculating the average salary and selecting it from the result
set.
FROM
    employees -- Specifying the 'employees' table.
NATURAL JOIN
    jobs -- Performing a natural join with the 'jobs' table.
GROUP BY
    job_title; -- Grouping the result set by job title to calculate the average
salary for each job.

11.
-- This SQL query selects specific columns from the 'employees' and 'jobs' tables and
calculates the difference between each employee's salary and the minimum salary for
their job title.

SELECT
    job_title, -- Selecting the 'job_title' column from the result set.
    first_name, -- Selecting the 'first_name' column from the result set.
    salary - min_salary AS 'Salary - Min_Salary' -- Calculating the difference
between the salary and the minimum salary for each job title and aliasing it as
'Salary - Min_Salary'.
FROM
    employees -- Specifying the 'employees' table.
NATURAL JOIN
    jobs; -- Performing a natural join with the 'jobs' table.
```

12.

```
-- This SQL query retrieves all columns from the 'job_history' table for employees
whose salary is greater than 10000.

SELECT
    jh.* -- Selecting all columns from the 'job_history' table.
FROM
    job_history jh -- Specifying the 'job_history' table and aliasing it as 'jh'.
JOIN
    employees e -- Performing a join with the 'employees' table and aliasing it as
'e'.
ON
    (jh.employee_id = e.employee_id) -- Joining the 'job_history' and 'employees'
tables based on the 'employee_id' column to associate job history with employees.
WHERE
    salary > 10000; -- Filtering the result to include only records where the salary
is greater than 10000.
```

13.

```
-- This SQL query selects specific columns from the 'departments' and 'employees'
tables to retrieve information about department managers with more than 15 years of
experience.

SELECT
    first_name, -- Selecting the 'first_name' column from the result set.
    last_name, -- Selecting the 'last_name' column from the result set.
    hire_date, -- Selecting the 'hire_date' column from the result set.
    salary, -- Selecting the 'salary' column from the result set.
    (DATEDIFF(now(), hire_date))/365 Experience -- Calculating the experience in
years and aliasing it as 'Experience'.
FROM
    departments d -- Specifying the 'departments' table and aliasing it as 'd'.
JOIN
    employees e -- Joining the 'employees' table and aliasing it as 'e'.
ON
    (d.manager_id = e.employee_id) -- Joining the 'departments' and 'employees'
tables based on the 'manager_id' column to associate department managers with their
departments.
WHERE
    (DATEDIFF(now(), hire_date))/365 > 15; -- Filtering the result to include only
records where the experience in years is greater than 15.
```

Date and Time functions :

1.

```
-- This SQL query calculates a date that is three months prior to the current date.

SELECT
date(((PERIOD_ADD -- Calculates a period by adding a specified number of months to a
given period.
```

```
    (EXTRACT(YEAR_MONTH -- Extracts the year and month from the current date
(CURDATE()).
    FROM CURDATE()) -- Specifies the current date.
    ,-3)*100)+1)); -- Subtracts three months from the current date, then multiplies
by 100 to convert it to a period, and finally adds 1 to convert it back to a date.

2.
-- This SQL query calculates the last day of the current month.

SELECT
    (SUBDATE(ADDDATE -- Calculates the date obtained by adding a specified interval
(1 month) to the current date.
    (CURDATE(),INTERVAL 1 MONTH), -- Adds 1 month to the current date (CURDATE()).
    INTERVAL DAYOFMONTH(CURDATE()) DAY)) -- Subtracts the number of days from the
current date to get the last day of the current month.
    AS LastDayOfTheMonth; -- Alias for the calculated last day of the month.

3.
-- This SQL query retrieves distinct dates representing the first day of the week for
each hire date in the 'employees' table.

SELECT DISTINCT( -- Selects distinct values of the result set.
    STR_TO_DATE( -- Converts a string into a date value.
CONCAT( -- Concatenates multiple strings into one string.
YEARWEEK(hire_date), -- Gets the year and week number for each hire date.
            '1' -- Appends '1' to the end of the concatenated string.
        ),
        '%x%v%w' -- Specifies the format of the input string.
    )
)
FROM
employees; -- Specifies the 'employees' table.

4.
-- This SQL query creates a date using the year extracted from the current date and
the day '1'.

SELECT
MAKEDATE( -- Creates a date using the specified year and day.
EXTRACT(YEAR FROM CURDATE()), -- Extracts the year from the current date (CURDATE()).
        1 -- Specifies the day as '1'.
    );

5.
-- This SQL query creates a date using the year extracted from the current date, the
month '12', and the day '31'.

SELECT
    STR_TO_DATE( -- Converts a string into a date value.
CONCAT( -- Concatenates multiple strings into one string.
            12, -- Specifies the month as '12'.
            31, -- Specifies the day as '31'.
EXTRACT(YEAR FROM CURDATE()) -- Extracts the year from the current date (CURDATE()).
        ),
        '%m%d%Y' -- Specifies the format of the input string.
    );
```

```
6.
-- This SQL query calculates the age based on the difference between the current year
and the year of birth.

SELECT
YEAR(CURRENT_TIMESTAMP) - -- Calculates the current year.
YEAR("1967-06-08") - -- Calculates the year from the given birthdate.
    (RIGHT(CURRENT_TIMESTAMP, 5) < -- Checks if the month and day of the current date
are before the month and day of the birthdate.
RIGHT("1967-06-08", 5)) -- Extracts the month and day from the birthdate.
as age; -- Alias for the calculated age.

7.
-- This SQL query formats the current date in a specified format and aliases it as
'Current_date'.

SELECT
    DATE_FORMAT( -- Formats a date value according to the specified format.
CURDATE(), -- Retrieves the current date using the CURDATE() function.
        '%M %e, %Y' -- Specifies the desired format for the date: '%M' for full month
name, '%e' for day of the month without leading zeros, and '%Y' for four-digit year.
    )
    AS 'Current_date'; -- Alias for the formatted current date.

8.
-- This SQL query formats the current date and time in a specified format.

SELECT
    DATE_FORMAT( -- Formats a date and time value according to the specified format.
NOW(), -- Retrieves the current date and time using the NOW() function.
        ''%W %M %Y' -- Specifies the desired format for the date and time: '%W' for
the full weekday name, '%M' for the full month name, and '%Y' for the four-digit
year.
    );

9.
-- This SQL query extracts the year component from the current date and time.

SELECT
EXTRACT(YEAR FROM NOW());

10.
-- This SQL query converts a number representing the number of days since year 0 to a
date value.

SELECT
    FROM_DAYS(730677);

11.
-- This SQL query retrieves the first name and hire date of employees hired within a
specific date range.

SELECT
    FIRST_NAME, -- Selecting the 'FIRST_NAME' column from the 'employees' table.
    HIRE_DATE -- Selecting the 'HIRE_DATE' column from the 'employees' table.
```

```
FROM
employees -- Specifying the 'employees' table.
WHERE
    HIRE_DATE  -- Filtering the rows based on the hire date being within a specific
date range.
    BETWEEN '1987-06-01 00:00:00' -- Specifies the start of the date range.
    AND '1987-07-30  23:59:59'; -- Specifies the end of the date range.

12.
-- This SQL query formats the current date and time in a specific format.

SELECT
date_format( -- Formats a date and time value according to the specified format.
CURDATE(), -- Retrieves the current date using the CURDATE() function.
        '%W %D %M %Y %T' -- Specifies the desired format for the date and time.
    );


13.
-- This SQL query formats the current date in a specific date format.

SELECT
date_format( -- Formats a date value according to the specified format.
CURDATE(), -- Retrieves the current date using the CURDATE() function.
        '%d/%m/%Y' -- Specifies the desired format for the date.
    );

14.
-- This SQL query formats the current date and time in a specific format.

SELECT
date_format( -- Formats a date and time value according to the specified format.
CURDATE(), -- Retrieves the current date using the CURDATE() function.
        '%l:%i %p %b %e, %Y' -- Specifies the desired format for the date and time.
    );

15.
-- This SQL query retrieves the first name and last name of employees hired in June.

SELECT
first_name, -- Selecting the 'first_name' column from the 'employees' table.
last_name -- Selecting the 'last_name' column from the 'employees' table.
FROM
employees -- Specifying the 'employees' table.
WHERE
MONTH(HIRE_DATE) =  6; -- Filtering the rows to include only those where the hire
date month is June.

16.
-- This SQL query retrieves the year part of the hire date for employees and groups
them by year, filtering only those years where the count of employees hired is
greater than 10.

SELECT
    DATE_FORMAT(HIRE_DATE,'%Y') -- Formats the hire date to extract the year part and
returns it as 'YYYY'.
```

```
FROM
employees -- Specifies the 'employees' table.
GROUP BY
    DATE_FORMAT(HIRE_DATE,'%Y') -- Groups the result set by the year part of the hire
date.
HAVING
COUNT(EMPLOYEE_ID) > 10; -- Filters the grouped results to include only those years
where the count of employees is greater than 10.

17.
-- This SQL query retrieves the first name and hire date of employees hired in the
year 1987.

SELECT
    FIRST_NAME, -- Selecting the 'FIRST_NAME' column from the 'employees' table.
    HIRE_DATE -- Selecting the 'HIRE_DATE' column from the 'employees' table.
FROM
employees -- Specifying the 'employees' table.
WHERE
YEAR(HIRE_DATE)=1987; -- Filtering the rows to include only those where the hire date
year is 1987.

18.
-- This SQL query retrieves the department name, first name, and salary of employees
who have been working for more than 5 years and are also managers of their respective
departments.

SELECT
    DEPARTMENT_NAME, -- Selecting the 'DEPARTMENT_NAME' column from the 'departments'
table.
    FIRST_NAME, -- Selecting the 'FIRST_NAME' column from the 'employees' table.
    SALARY -- Selecting the 'SALARY' column from the 'employees' table.
FROM
departments D -- Specifying the 'departments' table and aliasing it as 'D'.
JOIN
employees E -- Joining the 'employees' table and aliasing it as 'E'.
ON
    (D.MANAGER_ID=E.MANAGER_ID) -- Joining the 'departments' and 'employees' tables
based on the manager ID to associate managers with their departments.
WHERE
    (SYSDATE()-HIRE_DATE) / 365 > 5; -- Filtering the result to include only records
where the duration of employment is more than 5 years.

19.
-- This SQL query retrieves the employee ID, last name, hire date, and the last day
of the month for each hire date.

SELECT
employee_id, -- Selecting the 'employee_id' column from the 'employees' table.
last_name, -- Selecting the 'last_name' column from the 'employees' table.
hire_date, -- Selecting the 'hire_date' column from the 'employees' table.
    LAST_DAY(hire_date) -- Using the LAST_DAY() function to calculate the last day of
the month for each hire date.
FROM
employees; -- Specifying the 'employees' table.
```

```
20.
-- This SQL query retrieves the first name of employees, the current date, their hire
date, and calculates their years of employment.

SELECT
    FIRST_NAME, -- Selecting the 'FIRST_NAME' column from the 'employees' table.
SYSDATE(), -- Retrieving the current date using the SYSDATE() function.
    HIRE_DATE, -- Selecting the 'HIRE_DATE' column from the 'employees' table.
DATEDIFF(SYSDATE(), hire_date)/365 -- Calculating the difference in days between the
current date and the hire date, then dividing by 365 to get years of employment.
FROM
employees; -- Specifying the 'employees' table.

21.
-- This SQL query calculates the count of employees hired in each department for each
year, ordered by department ID.

SELECT
    DEPARTMENT_ID, -- Selecting the 'DEPARTMENT_ID' column from the 'employees'
table.
    DATE_FORMAT(HIRE_DATE,'%Y'), -- Formatting the hire date to extract the year and
selecting it as 'YEAR'.
COUNT(EMPLOYEE_ID) -- Counting the number of employees in each department for each
year.
FROM
employees -- Specifying the 'employees' table.
GROUP BY
    DEPARTMENT_ID, DATE_FORMAT(HIRE_DATE, '%Y') -- Grouping the result set by
department ID and year of hire date.
ORDER BY
    DEPARTMENT_ID; -- Ordering the result set by department ID.

MySQL String :

1.
-- This SQL query selects the job_id and concatenates the employee_id values
separated by a space for each group of job_id.

SELECT
job_id,
    GROUP_CONCAT(employee_id, ' ') AS 'Employees ID' -- Concatenates employee_ids
with a space separator and renames the resulting column as 'Employees ID'
FROM
employees
GROUP BY
job_id; -- Groups the result by job_id, so each row represents a unique job_id with
concatenated employee_ids.
2.
-- This SQL statement updates the 'phone_number' column in the 'employees' table.
-- It replaces any occurrence of the substring '124' in the 'phone_number' column
with '999'.
-- The WHERE clause ensures that only rows with phone numbers containing '124' are
updated.

UPDATE employees
SET phone_number = REPLACE(phone_number, '124', '999')
```

```
WHERE phone_number LIKE '%124%';

3.
-- This SQL query selects all columns from the 'employees' table where the length of
the first name is greater than or equal to 8.

SELECT
    * -- Selecting all columns from the 'employees' table.
FROM
employees -- Specifying the 'employees' table.
WHERE
LENGTH(first_name) >= 8; -- Filtering the rows to include only those where the length
of the first name is greater than or equal to 8.

4.
-- This SQL query selects the job_id, maximum salary, and minimum salary from the
jobs table.
-- The LPAD function is used to left-pad the salary values with zeros to ensure they
have a total width of 7 characters.

SELECT
job_id,
LPAD(max_salary, 7, '0') AS ' Max Salary', -- Left-pads the max_salary column with
zeros to ensure a width of 7 characters, and renames the resulting column as 'Max
Salary'.
LPAD(min_salary, 7, '0') AS ' Min Salary'  -- Left-pads the min_salary column with
zeros to ensure a width of 7 characters, and renames the resulting column as 'Min
Salary'.
FROM
jobs;

5.
-- Updating the email addresses of employees
UPDATE employees
-- Setting the email column to a concatenation of the existing email value and
'@example.com'
SET email = CONCAT(email, '@example.com');

6.
-- Selecting the employee_id, first_name, and hire month from the employees table
SELECT employee_id, first_name,
-- Extracting the month portion of the hire_date using the MID function, starting
from the 6th character and taking 2 characters
MID(hire_date, 6, 2) as hire_month
-- Selecting data from the employees table
FROM employees;

7.
-- This SQL query selects the employee ID and extracts a portion of the reversed
email address to create a new column called Email_ID.

SELECT
employee_id, -- Selecting the employee ID from the employees table.

    -- Reverses the email address, extracts a substring starting from the fourth
character, and then reverses it back.
```

```
REVERSE(SUBSTR(REVERSE(email), 4)) as Email_ID

FROM
employees; -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

8.
-- This SQL query selects all columns from the employees table where the first name
is in uppercase.

SELECT
    * -- Selecting all columns from the employees table.

FROM
employees -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

WHERE
first_name = BINARY UPPER(first_name);

9.
-- This SQL query selects the last four digits of the phone numbers stored in the
'phone_number' column and aliases the result as 'Ph.No.'.

SELECT
RIGHT(phone_number, 4) as 'Ph.No.' -- Selecting the last four digits of the phone
numbers and aliasing the result as 'Ph.No.'.

FROM
employees; -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

10.
-- This SQL query selects the location ID, street address, and extracts the last word
from the street address.

SELECT
location_id, -- Selecting the location ID from the locations table.
street_address, -- Selecting the street address from the locations table.

    -- Extracting the last word from the street address by replacing punctuation
marks with spaces,
    -- splitting the address into words, and then selecting the last word.
    SUBSTRING_INDEX(
REPLACE(
REPLACE(
REPLACE(street_address',',' '),
            ')',' '),
        '(',' '),
' ',-1) AS 'Last--word-of-street_address'

FROM
locations; -- Specifies the table from which data is being retrieved, in this case,
it's the 'locations' table.

11.
```

```sql
-- This SQL query selects all columns from the locations table where the length of
the street address is less than or equal to the minimum length of all street
addresses in the locations table.

SELECT
    * -- Selecting all columns from the locations table.

FROM
locations -- Specifies the table from which data is being retrieved, in this case,
it's the 'locations' table.

WHERE
LENGTH(street_address) <= ( -- Filters the rows where the length of the street
address is less than or equal to...

        SELECT
MIN(LENGTH(street_address)) -- ... the minimum length of all street addresses in the
locations table.

        FROM
locations
    );

12.
-- This SQL query selects a portion of the job title from the jobs table.

SELECT
job_title, -- Selecting the job title from the jobs table.

    -- Extracting the substring from the job title starting from the first character
up to the position of the first space.
SUBSTR(job_title, 1, INSTR(job_title, ' ') - 1)

FROM
jobs; -- Specifies the table from which data is being retrieved, in this case, it's
the 'jobs' table.

13.
-- This SQL query selects the first name and last name of employees whose last name
contains the letter 'C' after the second position.

SELECT
first_name, last_name -- Selecting the first name and last name from the employees
table.

FROM
employees -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

WHERE
INSTR(last_name,'C') > 2; -- Filters the rows where the letter 'C' is found in the
last name after the second position.

14.
-- This SQL query selects the first name and its length from the employees table for
names starting with 'J', 'M', or 'A', and sorts the result by first name.
```

```sql
SELECT
first_name "Name", -- Selecting the first name and aliasing it as "Name".
LENGTH(first_name) "Length" -- Calculating the length of the first name and aliasing
it as "Length".

FROM
employees -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

WHERE
first_name LIKE 'J%' -- Filters the rows where the first name starts with 'J'.
    OR first_name LIKE 'M%' -- Filters the rows where the first name starts with 'M'.
    OR first_name LIKE 'A%' -- Filters the rows where the first name starts with 'A'.

ORDER BY
first_name; -- Orders the result set by the first name in ascending order.
```

15.

```sql
-- This SQL query selects the first name of employees and left-pads their salary
values with '$' characters up to a total width of 10 characters.

SELECT
first_name, -- Selecting the first name from the employees table.

    -- Left-pads the salary values with '$' characters up to a total width of 10
characters.
LPAD(salary, 10, '$') SALARY

FROM
employees; -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.
```

16.

```sql
-- This SQL query selects the first 8 characters of the first name, repeats '$'
characters based on the salary divided by 1000, and retrieves the original salary
from the employees table, ordering the result by salary in descending order.

SELECT
LEFT(first_name, 8), -- Selecting the first 8 characters of the first name.

    -- Repeats '$' characters a number of times based on the salary divided by 1000.
REPEAT('$', FLOOR(salary/1000)) 'SALARY($)',

salary -- Selecting the original salary from the employees table.

FROM
employees -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

ORDER BY
salary DESC; -- Orders the result set by salary in descending order.
```

17.

```sql
-- This SQL query selects the employee ID, first name, last name, and hire date from
the employees table
-- where the hire date contains the month and day "07" in the specified format, '%d
%m %Y'.

SELECT
employee_id, -- Selecting the employee ID from the employees table.
first_name, -- Selecting the first name from the employees table.
last_name, -- Selecting the last name from the employees table.
hire_date -- Selecting the hire date from the employees table.

FROM
employees -- Specifies the table from which data is being retrieved, in this case,
it's the 'employees' table.

WHERE
POSITION("07" IN DATE_FORMAT(hire_date, '%d %m %Y')) > 0;
    -- Checks if the position of "07" exists in the formatted hire date string,
    -- indicating that the month and day of hiring are "07".
```

6. Lab Day 3

```sql
-- lab 03

create database universipedia;
use universipedia;


-- classroom test
select * from classroom;

-- # building, room_number, capacity
-- 'Alumni', '143', '47'
-- 'Alumni', '547', '26'
-- 'black', '431', '1000'
-- 'Bronfman', '700', '12'
-- 'Chandler', '375', '10'
-- 'Chandler', '804', '11'
-- 'dark', '431', '9999'
-- 'Drown', '757', '18'
-- 'Fairchild', '145', '27'
-- 'Garfield', '119', '59'
-- 'Gates', '314', '10'
-- 'Gates', '707', '65'
-- 'Grace', '40', '34'
-- 'Lambeau', '348', '51'
-- 'Lamberton', '134', '10'
-- 'Lamberton', '143', '10'
-- 'Main', '425', '22'
-- 'Main', '45', '30'
-- 'Nassau', '45', '92'
-- 'Painter', '86', '97'
-- 'pink', '431', '9999'
-- 'Polya', '808', '28'
-- 'Power', '717', '12'
-- 'Power', '972', '10'
-- 'Rathbone', '261', '60'
```

```
  -- 'Saucon', '113', '109'
  -- 'Saucon', '180', '15'
  -- 'Saucon', '844', '24'
  -- 'Stabler', '105', '113'
  -- 'Taylor', '183', '71'
  -- 'Taylor', '812', '115'
  -- 'White', '432', '100'
  -- 'Whitman', '134', '120'
  -- 'Whitman', '434', '32'

  describe classroom;

  insert into classroom ( building, room_number, capacity )
  value ("dark", 431, 9999);

  -- 10:29:15 insert into classroom ( building, room_number, capacity ) value
("dark", 431, 9999) 1 row(s) affected 0.0036 sec

  CREATE TABLE department (
    dept_name VARCHAR(20),
    building VARCHAR(15),
    budget NUMERIC(12 , 2 ),
    PRIMARY KEY (dept_name)
  );

  CREATE TABLE department (
    dept_name VARCHAR(20),
    building VARCHAR(15),
    budget NUMERIC(12 , 2 ),
    PRIMARY KEY (dept_name)
  );

  create table course (
    course_id varchar(7),
    title varchar(50),
    dept_name varchar (20),
    credits numeric (2, 0),
    primary key (course_id),
    foreign key (dept_name) references department(dept_name)
  );

  create table instructor (
    ID varchar(5),
    number varchar (20) not null,
    dept_name varchar (20),
    salary numeric(8,2),
    primary key (ID),
    foreign key (dept_name) references department
  );

  CREATE TABLE section (
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6),
    year NUMERIC(4 , 0 ),
    building VARCHAR(15),
```

```
    room_number VARCHAR(7),
    time_slot_id VARCHAR(4),
    PRIMARY KEY (course_id , sec_id , semester , year),
    FOREIGN KEY (course_id)
      REFERENCES course
  );

  create table teaches (
    ID varchar (5),
    course_id varchar(8),
    sec_id varchar(8),
    semester varchar(6),
    year numeric (4, 0),
    primary key (ID, course_id, sec_id, semester, year),
    foreign key (course_id, sec_id, semester, year) references section,
    foreign key (ID) references instructor
  );

  DROP TABLE IF EXISTS parent;

  CREATE TABLE parent (
    Name VARCHAR(1),
    Age NUMERIC(2, 0),
    PRIMARY KEY (Name, Age)
  );

  DESCRIBE parent;

  CREATE TABLE child (
    Name VARCHAR(1),
    Age NUMERIC(2, 0),
    PRIMARY KEY (Name, Age),
    FOREIGN KEY (Name, Age)
      REFERENCES parent (Name, Age)
  );

  drop table child;
  desc child;

  -- 11:01:04 create table course (  course_id varchar(7),     title varchar(50),
dept_name varchar (20),     credits numeric (2, 0),     primary key (course_id),
foreign key (dept_name) references department(dept_name) ) 0 row(s) affected 0.020
sec
  -- 11:03:46 create table instructor (  ID varchar(5),     number varchar (20) not
null,     dept_name varchar (20),     salary numeric(8,2),     primary key (ID),
foreign key (dept_name) references department )  0 row(s) affected 0.022 sec
  -- 11:06:53 create table section (  course_id varchar(8),     sec_id varchar(8),
semester varchar (6),     year numeric(4, 0),     building varchar (15),
room_number varchar(7),     time_slot_id varchar(4),     primary key (course_id,
sec_id, semester, year),     foreign key (course_id) references course ) 0 row(s)
affected 0.025 sec
  -- 11:14:51 create table teaches (  ID varchar (5),     course_id varchar(8),
sec_id varchar(8),     semester varchar(6),     year numeric (4, 0),     primary key
(ID, course_id, sec_id, semester, year),     foreign key (course_id, sec_id,
semester, year) references section,     foreign key (ID) references instructor )  0
row(s) affected 0.028 sec
```

```
-- use university
use university;

-- select name from instructor
select name from instructor;

-- 'Lembr'
-- 'Bawa'
-- 'Yazdi'
-- 'Wieland'
-- 'DAgostino'
-- 'Liley'
-- 'Kean'
-- 'Atanassov'
-- 'Moreira'
-- 'Gustafsson'
-- 'Bourrier'
-- 'Bondi'
-- 'Soisalon-Soininen'
-- 'Morris'
-- 'Arias'
-- 'Murata'
-- 'Tung'
-- 'Luo'
-- 'Vicentino'
-- 'Romero'
-- 'Lent'
-- 'Sarkar'
-- 'Shuming'
-- 'Konstantinides'
-- 'Bancilhon'
-- 'Hau'
-- 'Dusserre'
-- 'Desyl'
-- 'Jaekel'
-- 'McKinnon'
-- 'Gutierrez'
-- 'Mingoz'
-- 'Pimenta'
-- 'Yin'
-- 'Sullivan'
-- 'Voronina'
-- 'Kenje'
-- 'Mahmoud'
-- 'Pingr'
-- 'Ullman '
-- 'Levine'
-- 'Queiroz'
-- 'Valtchev'
-- 'Bietzk'
-- 'Choll'
-- 'Arinb'
-- 'Sakurai'
-- 'Mird'
-- 'Bertolino'
```

```
-- 'Dale'

-- select departments from instructor
select dept_name from instructor;

-- 'Accounting'
-- 'Accounting'
-- 'Accounting'
-- 'Accounting'
-- 'Astronomy'
-- 'Athletics'
-- 'Athletics'
-- 'Athletics'
-- 'Athletics'
-- 'Athletics'
-- 'Biology'
-- 'Biology'
-- 'Comp. Sci.'
-- 'Comp. Sci.'
-- 'Cybernetics'
-- 'Cybernetics'
-- 'Cybernetics'
-- 'Cybernetics'
-- 'Elec. Eng.'
-- 'Elec. Eng.'
-- 'Elec. Eng.'
-- 'Elec. Eng.'
-- 'English'
-- 'English'
-- 'English'
-- 'English'
-- 'Finance'
-- 'Geology'
-- 'Languages'
-- 'Languages'
-- 'Languages'
-- 'Marketing'
-- 'Marketing'
-- 'Marketing'
-- 'Marketing'
-- 'Mech. Eng.'
-- 'Mech. Eng.'
-- 'Physics'
-- 'Physics'
-- 'Pol. Sci.'
-- 'Pol. Sci.'
-- 'Pol. Sci.'
-- 'Psychology'
-- 'Psychology'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
```

```sql
-- select department names but with all
select all dept_name from instructor;

-- 'Accounting'
-- 'Accounting'
-- 'Accounting'
-- 'Accounting'
-- 'Astronomy'
-- 'Athletics'
-- 'Athletics'
-- 'Athletics'
-- 'Athletics'
-- 'Athletics'
-- 'Biology'
-- 'Biology'
-- 'Comp. Sci.'
-- 'Comp. Sci.'
-- 'Cybernetics'
-- 'Cybernetics'
-- 'Cybernetics'
-- 'Cybernetics'
-- 'Elec. Eng.'
-- 'Elec. Eng.'
-- 'Elec. Eng.'
-- 'Elec. Eng.'
-- 'English'
-- 'English'
-- 'English'
-- 'English'
-- 'Finance'
-- 'Geology'
-- 'Languages'
-- 'Languages'
-- 'Languages'
-- 'Marketing'
-- 'Marketing'
-- 'Marketing'
-- 'Marketing'
-- 'Mech. Eng.'
-- 'Mech. Eng.'
-- 'Physics'
-- 'Physics'
-- 'Pol. Sci.'
-- 'Pol. Sci.'
-- 'Pol. Sci.'
-- 'Psychology'
-- 'Psychology'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'
-- 'Statistics'

-- select only distinct departments from instructor
select distinct dept_name from instructor;
```

```
-- 'Accounting'
-- 'Astronomy'
-- 'Athletics'
-- 'Biology'
-- 'Comp. Sci.'
-- 'Cybernetics'
-- 'Elec. Eng.'
-- 'English'
-- 'Finance'
-- 'Geology'
-- 'Languages'
-- 'Marketing'
-- 'Mech. Eng.'
-- 'Physics'
-- 'Pol. Sci.'
-- 'Psychology'
-- 'Statistics'

-- select all districts from instructor
select distinct * from instructor;

-- '14365', 'Lembr', 'Accounting', '32241.56'
-- '15347', 'Bawa', 'Athletics', '72140.88'
-- '16807', 'Yazdi', 'Athletics', '98333.65'
-- '19368', 'Wieland', 'Pol. Sci.', '124651.41'
-- '22591', 'DAgostino', 'Psychology', '59706.49'
-- '25946', 'Liley', 'Languages', '90891.69'
-- '28097', 'Kean', 'English', '35023.18'
-- '28400', 'Atanassov', 'Statistics', '84982.92'
-- '31955', 'Moreira', 'Accounting', '71351.42'
-- '3199', 'Gustafsson', 'Elec. Eng.', '82534.37'
-- '3335', 'Bourrier', 'Comp. Sci.', '80797.83'
-- '34175', 'Bondi', 'Comp. Sci.', '115469.11'
-- '35579', 'Soisalon-Soininen', 'Psychology', '62579.61'
-- '36897', 'Morris', 'Marketing', '43770.36'
-- '37687', 'Arias', 'Statistics', '104563.38'
-- '4034', 'Murata', 'Athletics', '61387.56'
-- '41930', 'Tung', 'Athletics', '50482.03'
-- '4233', 'Luo', 'English', '88791.45'
-- '42782', 'Vicentino', 'Elec. Eng.', '34272.67'
-- '43779', 'Romero', 'Astronomy', '79070.08'
-- '48507', 'Lent', 'Mech. Eng.', '107978.47'
-- '48570', 'Sarkar', 'Pol. Sci.', '87549.80'
-- '50330', 'Shuming', 'Physics', '108011.81'
-- '50885', 'Konstantinides', 'Languages', '32570.50'
-- '52647', 'Bancilhon', 'Pol. Sci.', '87958.01'
-- '57180', 'Hau', 'Accounting', '43966.29'
-- '58558', 'Dusserre', 'Marketing', '66143.25'
-- '59795', 'Desyl', 'Languages', '48803.38'
-- '63287', 'Jaekel', 'Athletics', '103146.87'
-- '63395', 'McKinnon', 'Cybernetics', '94333.99'
-- '64871', 'Gutierrez', 'Statistics', '45310.53'
-- '6569', 'Mingoz', 'Finance', '105311.38'
-- '65931', 'Pimenta', 'Cybernetics', '79866.95'
-- '72553', 'Yin', 'English', '46397.59'
```

```
-- '73623', 'Sullivan', 'Elec. Eng.', '90038.09'
-- '74420', 'Voronina', 'Physics', '121141.99'
-- '74426', 'Kenje', 'Marketing', '106554.73'
-- '77346', 'Mahmoud', 'Geology', '99382.59'
-- '78699', 'Pingr', 'Statistics', '59303.62'
-- '79081', 'Ullman ', 'Accounting', '47307.10'
-- '79653', 'Levine', 'Elec. Eng.', '89805.83'
-- '80759', 'Queiroz', 'Biology', '45538.32'
-- '81991', 'Valtchev', 'Biology', '77036.18'
-- '90376', 'Bietzk', 'Cybernetics', '117836.50'
-- '90643', 'Choll', 'Statistics', '57807.09'
-- '95030', 'Arinb', 'Statistics', '54805.11'
-- '95709', 'Sakurai', 'English', '118143.98'
-- '96895', 'Mird', 'Marketing', '119921.41'
-- '97302', 'Bertolino', 'Mech. Eng.', '51647.57'
-- '99052', 'Dale', 'Cybernetics', '93348.83'

-- list all properties of instructor
describe instructor;

-- 'ID', 'varchar(5)', 'NO', 'PRI', NULL, ''
-- 'name', 'varchar(20)', 'NO', '', NULL, ''
-- 'dept_name', 'varchar(20)', 'YES', 'MUL', NULL, ''
-- 'salary', 'decimal(8,2)', 'YES', '', NULL, ''

-- select all from instructor but with their salaries multiplied by 0
select name, dept_name, salary * 0 from instructor;

-- 'Lembr', 'Accounting', '0.00'
-- 'Bawa', 'Athletics', '0.00'
-- 'Yazdi', 'Athletics', '0.00'
-- 'Wieland', 'Pol. Sci.', '0.00'
-- 'DAgostino', 'Psychology', '0.00'
-- 'Liley', 'Languages', '0.00'
-- 'Kean', 'English', '0.00'
-- 'Atanassov', 'Statistics', '0.00'
-- 'Moreira', 'Accounting', '0.00'
-- 'Gustafsson', 'Elec. Eng.', '0.00'
-- 'Bourrier', 'Comp. Sci.', '0.00'
-- 'Bondi', 'Comp. Sci.', '0.00'
-- 'Soisalon-Soininen', 'Psychology', '0.00'
-- 'Morris', 'Marketing', '0.00'
-- 'Arias', 'Statistics', '0.00'
-- 'Murata', 'Athletics', '0.00'
-- 'Tung', 'Athletics', '0.00'
-- 'Luo', 'English', '0.00'
-- 'Vicentino', 'Elec. Eng.', '0.00'
-- 'Romero', 'Astronomy', '0.00'
-- 'Lent', 'Mech. Eng.', '0.00'
-- 'Sarkar', 'Pol. Sci.', '0.00'
-- 'Shuming', 'Physics', '0.00'
-- 'Konstantinides', 'Languages', '0.00'
-- 'Bancilhon', 'Pol. Sci.', '0.00'
-- 'Hau', 'Accounting', '0.00'
-- 'Dusserre', 'Marketing', '0.00'
-- 'Desyl', 'Languages', '0.00'
```

```
-- 'Jaekel', 'Athletics', '0.00'
-- 'McKinnon', 'Cybernetics', '0.00'
-- 'Gutierrez', 'Statistics', '0.00'
-- 'Mingoz', 'Finance', '0.00'
-- 'Pimenta', 'Cybernetics', '0.00'
-- 'Yin', 'English', '0.00'
-- 'Sullivan', 'Elec. Eng.', '0.00'
-- 'Voronina', 'Physics', '0.00'
-- 'Kenje', 'Marketing', '0.00'
-- 'Mahmoud', 'Geology', '0.00'
-- 'Pingr', 'Statistics', '0.00'
-- 'Ullman ', 'Accounting', '0.00'
-- 'Levine', 'Elec. Eng.', '0.00'
-- 'Queiroz', 'Biology', '0.00'
-- 'Valtchev', 'Biology', '0.00'
-- 'Bietzk', 'Cybernetics', '0.00'
-- 'Choll', 'Statistics', '0.00'
-- 'Arinb', 'Statistics', '0.00'
-- 'Sakurai', 'English', '0.00'
-- 'Mird', 'Marketing', '0.00'
-- 'Bertolino', 'Mech. Eng.', '0.00'
-- 'Dale', 'Cybernetics', '0.00'


-- selecting instructor whose dept name is comp sci and salary is 70k
select name from instructor where dept_name = 'Comp. Sci.' and salary > 70000;

-- 'Bourrier'
-- 'Bondi'

-- custom names support
select T.name, S.course_id
from instructor as T, teaches as S
where T .ID= S.ID;

-- # name, course_id
-- 'Romero', '105'
-- 'Romero', '105'
-- 'Mingoz', '137'
-- 'Bietzk', '158'
-- 'Dale', '158'
-- 'Gustafsson', '169'
-- 'Gustafsson', '169'
-- 'Liley', '192'
-- 'Lembr', '200'
-- 'Ullman ', '200'
-- 'Dale', '237'
-- 'Dale', '237'
-- 'Voronina', '239'
-- 'Morris', '242'
-- 'Sakurai', '258'
-- 'Sakurai', '270'
-- 'Bondi', '274'
-- 'Mingoz', '304'
-- 'Morris', '313'
-- 'Mingoz', '319'
```

```
-- 'Jaekel', '334'
-- 'DAgostino', '338'
-- 'DAgostino', '338'
-- 'Ullman ', '345'
-- 'Mingoz', '349'
-- 'DAgostino', '352'
-- 'Mingoz', '362'
-- 'Mingoz', '362'
-- 'Mingoz', '362'
-- 'Kean', '366'
-- 'Voronina', '376'
-- 'DAgostino', '400'
-- 'DAgostino', '400'
-- 'Tung', '401'
-- 'Ullman ', '408'
-- 'Ullman ', '408'
-- 'Valtchev', '415'
-- 'Tung', '421'
-- 'Mingoz', '426'
-- 'Voronina', '443'
-- 'Voronina', '443'
-- 'Mingoz', '445'
-- 'Bawa', '457'
-- 'Choll', '461'
-- 'Sakurai', '468'
-- 'Shuming', '468'
-- 'Romero', '476'
-- 'DAgostino', '482'
-- 'Mahmoud', '486'
-- 'Romero', '489'
-- 'Mahmoud', '493'
-- 'Dale', '496'
-- 'Mingoz', '527'
-- 'Wieland', '545'
-- 'Queiroz', '559'
-- 'Gustafsson', '561'
-- 'Bondi', '571'
-- 'Wieland', '581'
-- 'Wieland', '591'
-- 'DAgostino', '599'
-- 'Atanassov', '603'
-- 'Atanassov', '604'
-- 'Voronina', '612'
-- 'Lent', '626'
-- 'Dale', '629'
-- 'Gustafsson', '631'
-- 'DAgostino', '642'
-- 'DAgostino', '663'
-- 'Luo', '679'
-- 'Tung', '692'
-- 'Sullivan', '694'
-- 'Morris', '696'
-- 'Valtchev', '702'
-- 'Mahmoud', '704'
-- 'Mahmoud', '735'
-- 'Mahmoud', '735'
```

```
-- 'Bondi', '747'
-- 'Dale', '748'
-- 'Ullman ', '760'
-- 'Morris', '791'
-- 'Vicentino', '793'
-- 'Morris', '795'
-- 'Dale', '802'
-- 'Kean', '808'
-- 'Lembr', '843'
-- 'Jaekel', '852'
-- 'Mahmoud', '864'
-- 'DAgostino', '867'
-- 'Sarkar', '867'
-- 'Pimenta', '875'
-- 'Dale', '893'
-- 'Dale', '927'
-- 'Bourrier', '949'
-- 'Voronina', '959'
-- 'Bourrier', '960'
-- 'Sakurai', '960'
-- 'DAgostino', '962'
-- 'DAgostino', '972'
-- 'Ullman ', '974'
-- 'DAgostino', '991'

select distinct T.name
from instructor as T , instructor as S
where T.salary > S.salary and S.dept_name = 'Biology';

-- # name
-- 'Bawa'
-- 'Yazdi'
-- 'Wieland'
-- 'DAgostino'
-- 'Liley'
-- 'Atanassov'
-- 'Moreira'
-- 'Gustafsson'
-- 'Bourrier'
-- 'Bondi'
-- 'Soisalon-Soininen'
-- 'Arias'
-- 'Murata'
-- 'Tung'
-- 'Luo'
-- 'Romero'
-- 'Lent'
-- 'Sarkar'
-- 'Shuming'
-- 'Bancilhon'
-- 'Dusserre'
-- 'Desyl'
-- 'Jaekel'
-- 'McKinnon'
-- 'Mingoz'
-- 'Pimenta'
```

```
-- 'Yin'
-- 'Sullivan'
-- 'Voronina'
-- 'Kenje'
-- 'Mahmoud'
-- 'Pingr'
-- 'Ullman '
-- 'Levine'
-- 'Valtchev'
-- 'Bietzk'
-- 'Choll'
-- 'Arinb'
-- 'Sakurai'
-- 'Mird'
-- 'Bertolino'
-- 'Dale'

-- selecting building with custom like
select distinct dept_name, building
from department
where building like '%auc_n%';

-- # dept_name, building
-- 'Accounting', 'Saucon'

-- select all order by salary descending and name ascending
select *
from instructor
order by salary desc, name asc;

-- # ID, name, dept_name, salary
-- '19368', 'Wieland', 'Pol. Sci.', '124651.41'
-- '74420', 'Voronina', 'Physics', '121141.99'
-- '96895', 'Mird', 'Marketing', '119921.41'
-- '95709', 'Sakurai', 'English', '118143.98'
-- '90376', 'Bietzk', 'Cybernetics', '117836.50'
-- '34175', 'Bondi', 'Comp. Sci.', '115469.11'
-- '50330', 'Shuming', 'Physics', '108011.81'
-- '48507', 'Lent', 'Mech. Eng.', '107978.47'
-- '74426', 'Kenje', 'Marketing', '106554.73'
-- '6569', 'Mingoz', 'Finance', '105311.38'
-- '37687', 'Arias', 'Statistics', '104563.38'
-- '63287', 'Jaekel', 'Athletics', '103146.87'
-- '77346', 'Mahmoud', 'Geology', '99382.59'
-- '16807', 'Yazdi', 'Athletics', '98333.65'
-- '63395', 'McKinnon', 'Cybernetics', '94333.99'
-- '99052', 'Dale', 'Cybernetics', '93348.83'
-- '25946', 'Liley', 'Languages', '90891.69'
-- '73623', 'Sullivan', 'Elec. Eng.', '90038.09'
-- '79653', 'Levine', 'Elec. Eng.', '89805.83'
-- '4233', 'Luo', 'English', '88791.45'
-- '52647', 'Bancilhon', 'Pol. Sci.', '87958.01'
-- '48570', 'Sarkar', 'Pol. Sci.', '87549.80'
-- '28400', 'Atanassov', 'Statistics', '84982.92'
-- '3199', 'Gustafsson', 'Elec. Eng.', '82534.37'
-- '3335', 'Bourrier', 'Comp. Sci.', '80797.83'
```

```
-- '65931', 'Pimenta', 'Cybernetics', '79866.95'
-- '43779', 'Romero', 'Astronomy', '79070.08'
-- '81991', 'Valtchev', 'Biology', '77036.18'
-- '15347', 'Bawa', 'Athletics', '72140.88'
-- '31955', 'Moreira', 'Accounting', '71351.42'
-- '58558', 'Dusserre', 'Marketing', '66143.25'
-- '35579', 'Soisalon-Soininen', 'Psychology', '62579.61'
-- '4034', 'Murata', 'Athletics', '61387.56'
-- '22591', 'DAgostino', 'Psychology', '59706.49'
-- '78699', 'Pingr', 'Statistics', '59303.62'
-- '90643', 'Choll', 'Statistics', '57807.09'
-- '95030', 'Arinb', 'Statistics', '54805.11'
-- '97302', 'Bertolino', 'Mech. Eng.', '51647.57'
-- '41930', 'Tung', 'Athletics', '50482.03'
-- '59795', 'Desyl', 'Languages', '48803.38'
-- '79081', 'Ullman ', 'Accounting', '47307.10'
-- '72553', 'Yin', 'English', '46397.59'
-- '80759', 'Queiroz', 'Biology', '45538.32'
-- '64871', 'Gutierrez', 'Statistics', '45310.53'
-- '57180', 'Hau', 'Accounting', '43966.29'
-- '36897', 'Morris', 'Marketing', '43770.36'
-- '28097', 'Kean', 'English', '35023.18'
-- '42782', 'Vicentino', 'Elec. Eng.', '34272.67'
-- '50885', 'Konstantinides', 'Languages', '32570.50'
-- '14365', 'Lembr', 'Accounting', '32241.56'

-- select instructor name with defined salary
select name
from instructor
where salary between 90000 and 100000;

-- # name
-- 'Yazdi'
-- 'Liley'
-- 'McKinnon'
-- 'Sullivan'
-- 'Mahmoud'
-- 'Dale'

-- union to combine 2 diff tables
(select course_id, semester
from section
where semester = 'Fall' and year= 2007)
union
(select course_id, semester
from section
where semester = 'Spring' and year= 2008);

-- # course_id, semester
-- '893', 'Fall'
-- '489', 'Fall'
-- '612', 'Fall'
-- '258', 'Fall'
-- '468', 'Fall'
-- '949', 'Fall'
-- '362', 'Spring'
```

```
-- '852', 'Spring'
-- '991', 'Spring'
-- '962', 'Spring'
-- '237', 'Spring'
-- '349', 'Spring'
-- '345', 'Spring'
-- '158', 'Spring'
-- '704', 'Spring'

-- union to combine 2 diff tables but with duplicates
(select course_id, semester
from section
where semester = 'Fall' and year= 2007)
union all
(select course_id, semester
from section
where semester = 'Spring' and year= 2008);

-- # course_id, semester
-- '893', 'Fall'
-- '489', 'Fall'
-- '612', 'Fall'
-- '258', 'Fall'
-- '468', 'Fall'
-- '949', 'Fall'
-- '362', 'Spring'
-- '852', 'Spring'
-- '991', 'Spring'
-- '962', 'Spring'
-- '237', 'Spring'
-- '349', 'Spring'
-- '345', 'Spring'
-- '158', 'Spring'
-- '704', 'Spring'

-- same thing just course_id
(select course_id
from section
where semester = 'Fall')
union all
(select course_id
from section
where semester = 'Spring');

-- # course_id
-- '893'
-- '489'
-- '612'
-- '258'
-- '468'
-- '949'
-- '362'
-- '852'
-- '991'
-- '962'
-- '237'
```

```
  -- '349'
  -- '345'
  -- '158'
  -- '704'

  -- intersecting two tables
  (select course_id
  from section
  where semester = 'Fall' order by course_id)
  intersect all
  (select course_id
  from section
  where semester = 'Spring' order by course_id);

  -- # course_id
  -- '362'
  -- '200'
  -- '169'
  -- '237'
  -- '400'
  -- '158'

  describe instructor;

  -- finds the average salary
  select avg (salary)
  from instructor
  where dept_name = 'Comp. Sci.';

  -- # avg (salary)
  -- '98133.470000'

  SELECT sum(value)
  FROM (
    SELECT 2 AS value
    UNION ALL
    SELECT 1
    UNION ALL
    SELECT 3
  ) AS temp_table;
```

7. Lab day 4

```
use university;

-- select instructor's dept name and building from 2 tables,
-- where these 2 are same
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name= department.dept_name;

-- 'Lembr', 'Accounting', 'Saucon'
-- 'Bawa', 'Athletics', 'Bronfman'
-- 'Yazdi', 'Athletics', 'Bronfman'
-- 'Wieland', 'Pol. Sci.', 'Whitman'
-- 'DAgostino', 'Psychology', 'Thompson'
-- 'Liley', 'Languages', 'Linderman'
```

```
-- 'Kean', 'English', 'Palmer'
-- 'Atanassov', 'Statistics', 'Taylor'
-- 'Moreira', 'Accounting', 'Saucon'
-- 'Gustafsson', 'Elec. Eng.', 'Main'
-- 'Bourrier', 'Comp. Sci.', 'Lamberton'
-- 'Bondi', 'Comp. Sci.', 'Lamberton'
-- 'Soisalon-Soininen', 'Psychology', 'Thompson'
-- 'Morris', 'Marketing', 'Lambeau'
-- 'Arias', 'Statistics', 'Taylor'
-- 'Murata', 'Athletics', 'Bronfman'
-- 'Tung', 'Athletics', 'Bronfman'
-- 'Luo', 'English', 'Palmer'
-- 'Vicentino', 'Elec. Eng.', 'Main'
-- 'Romero', 'Astronomy', 'Taylor'
-- 'Lent', 'Mech. Eng.', 'Rauch'
-- 'Sarkar', 'Pol. Sci.', 'Whitman'
-- 'Shuming', 'Physics', 'Wrigley'
-- 'Konstantinides', 'Languages', 'Linderman'
-- 'Bancilhon', 'Pol. Sci.', 'Whitman'
-- 'Hau', 'Accounting', 'Saucon'
-- 'Dusserre', 'Marketing', 'Lambeau'
-- 'Desyl', 'Languages', 'Linderman'
-- 'Jaekel', 'Athletics', 'Bronfman'
-- 'McKinnon', 'Cybernetics', 'Mercer'
-- 'Gutierrez', 'Statistics', 'Taylor'
-- 'Mingoz', 'Finance', 'Candlestick'
-- 'Pimenta', 'Cybernetics', 'Mercer'
-- 'Yin', 'English', 'Palmer'
-- 'Sullivan', 'Elec. Eng.', 'Main'
-- 'Voronina', 'Physics', 'Wrigley'
-- 'Kenje', 'Marketing', 'Lambeau'
-- 'Mahmoud', 'Geology', 'Palmer'
-- 'Pingr', 'Statistics', 'Taylor'
-- 'Ullman ', 'Accounting', 'Saucon'
-- 'Levine', 'Elec. Eng.', 'Main'
-- 'Queiroz', 'Biology', 'Candlestick'
-- 'Valtchev', 'Biology', 'Candlestick'
-- 'Bietzk', 'Cybernetics', 'Mercer'
-- 'Choll', 'Statistics', 'Taylor'
-- 'Arinb', 'Statistics', 'Taylor'
-- 'Sakurai', 'English', 'Palmer'
-- 'Mird', 'Marketing', 'Lambeau'
-- 'Bertolino', 'Mech. Eng.', 'Rauch'
-- 'Dale', 'Cybernetics', 'Mercer'

select T.name, S.course_id
from instructor as T , teaches as S
where T.ID= S.ID;

-- # name, course_id
-- 'Lembr', '200'
-- 'Lembr', '843'
-- 'Bawa', '457'
-- 'Wieland', '545'
-- 'Wieland', '581'
-- 'Wieland', '591'
```

```
-- 'DAgostino', '338'
-- 'DAgostino', '338'
-- 'DAgostino', '352'
-- 'DAgostino', '400'
-- 'DAgostino', '400'
-- 'DAgostino', '482'
-- 'DAgostino', '599'
-- 'DAgostino', '642'
-- 'DAgostino', '663'
-- 'DAgostino', '867'
-- 'DAgostino', '962'
-- 'DAgostino', '972'
-- 'DAgostino', '991'
-- 'Liley', '192'
-- 'Kean', '366'
-- 'Kean', '808'
-- 'Atanassov', '603'
-- 'Atanassov', '604'
-- 'Gustafsson', '169'
-- 'Gustafsson', '169'
-- 'Gustafsson', '561'
-- 'Gustafsson', '631'
-- 'Bourrier', '949'
-- 'Bourrier', '960'
-- 'Bondi', '274'
-- 'Bondi', '571'
-- 'Bondi', '747'
-- 'Morris', '242'
-- 'Morris', '313'
-- 'Morris', '696'
-- 'Morris', '791'
-- 'Morris', '795'
-- 'Tung', '401'
-- 'Tung', '421'
-- 'Tung', '692'
-- 'Luo', '679'
-- 'Vicentino', '793'
-- 'Romero', '105'
-- 'Romero', '105'
-- 'Romero', '476'
-- 'Romero', '489'
-- 'Lent', '626'
-- 'Sarkar', '867'
-- 'Shuming', '468'
-- 'Jaekel', '334'
-- 'Jaekel', '852'
-- 'Mingoz', '137'
-- 'Mingoz', '304'
-- 'Mingoz', '319'
-- 'Mingoz', '349'
-- 'Mingoz', '362'
-- 'Mingoz', '362'
-- 'Mingoz', '362'
-- 'Mingoz', '426'
-- 'Mingoz', '445'
-- 'Mingoz', '527'
```

```
-- 'Pimenta', '875'
-- 'Sullivan', '694'
-- 'Voronina', '239'
-- 'Voronina', '376'
-- 'Voronina', '443'
-- 'Voronina', '443'
-- 'Voronina', '612'
-- 'Voronina', '959'
-- 'Mahmoud', '486'
-- 'Mahmoud', '493'
-- 'Mahmoud', '704'
-- 'Mahmoud', '735'
-- 'Mahmoud', '735'
-- 'Mahmoud', '864'
-- 'Ullman ', '200'
-- 'Ullman ', '345'
-- 'Ullman ', '408'
-- 'Ullman ', '408'
-- 'Ullman ', '760'
-- 'Ullman ', '974'
-- 'Queiroz', '559'
-- 'Valtchev', '415'
-- 'Valtchev', '702'
-- 'Bietzk', '158'
-- 'Choll', '461'
-- 'Sakurai', '258'
-- 'Sakurai', '270'
-- 'Sakurai', '468'
-- 'Sakurai', '960'
-- 'Dale', '158'
-- 'Dale', '237'
-- 'Dale', '237'
-- 'Dale', '496'
-- 'Dale', '629'
-- 'Dale', '748'
-- 'Dale', '802'
-- 'Dale', '893'
-- 'Dale', '927'

select distinct T .name
from instructor as T , instructor as S
where T.salary > S.salary and S.dept_name = 'Biology';

-- # name
-- 'Bawa'
-- 'Yazdi'
-- 'Wieland'
-- 'DAgostino'
-- 'Liley'
-- 'Atanassov'
-- 'Moreira'
-- 'Gustafsson'
-- 'Bourrier'
-- 'Bondi'
-- 'Soisalon-Soininen'
-- 'Arias'
```

```
-- 'Murata'
-- 'Tung'
-- 'Luo'
-- 'Romero'
-- 'Lent'
-- 'Sarkar'
-- 'Shuming'
-- 'Bancilhon'
-- 'Dusserre'
-- 'Desyl'
-- 'Jaekel'
-- 'McKinnon'
-- 'Mingoz'
-- 'Pimenta'
-- 'Yin'
-- 'Sullivan'
-- 'Voronina'
-- 'Kenje'
-- 'Mahmoud'
-- 'Pingr'
-- 'Ullman '
-- 'Levine'
-- 'Valtchev'
-- 'Bietzk'
-- 'Choll'
-- 'Arinb'
-- 'Sakurai'
-- 'Mird'
-- 'Bertolino'
-- 'Dale'

select dept_name
from department
where building like '%a%';

-- # dept_name
-- 'Accounting'
-- 'Astronomy'
-- 'Athletics'
-- 'Biology'
-- 'Civil Eng.'
-- 'Comp. Sci.'
-- 'Elec. Eng.'
-- 'English'
-- 'Finance'
-- 'Geology'
-- 'History'
-- 'Languages'
-- 'Marketing'
-- 'Math'
-- 'Mech. Eng.'
-- 'Pol. Sci.'
-- 'Statistics'

select count (distinct_ID)
from teaches
```

```sql
where semester = 'Spring';

describe teaches;


select course_ID
from teaches
where semester = 'Spring';

select count(*)
from course;

select dept_name, AVG(salary) as avg_salary
  from instructor
    group by dept_name;

select dept_name, avg_salary
from (select dept_name, AVG(salary) as avg_salary
      from instructor
      group by dept_name) as T(dept_name, avg_salary)
      where avg_salary>42000;

-- select all distinct course_id
-- where semester is Fall
select distinct course_id
from section
where semester = 'Fall' and
course_id not in (select course_id
from section
where semester = 'Spring');

-- # course_id
-- '694'
-- '105'
-- '313'
-- '476'
-- '242'
-- '843'
-- '893'
-- '421'
-- '468'
-- '415'
-- '559'
-- '867'
-- '960'
-- '304'
-- '489'
-- '612'
-- '626'
-- '274'
-- '461'
-- '258'
-- '561'
-- '192'
-- '808'
-- '974'
```

```
-- '376'
-- '527'
-- '642'
-- '401'
-- '545'
-- '748'
-- '927'
-- '949'
-- '959'
-- '366'
-- '239'
-- '334'
-- '496'
-- '603'
-- '486'
-- '482'

--
select distinct course_id
from section
where semester = 'Fall' and
course_id not in (select course_id
from section
where semester = 'Spring');

-- # course_id
-- '694'
-- '105'
-- '313'
-- '476'
-- '242'
-- '843'
-- '893'
-- '421'
-- '468'
-- '415'
-- '559'
-- '867'
-- '960'
-- '304'
-- '489'
-- '612'
-- '626'
-- '274'
-- '461'
-- '258'
-- '561'
-- '192'
-- '808'
-- '974'
-- '376'
-- '527'
-- '642'
-- '401'
-- '545'
-- '748'
```

```
-- '927'
-- '949'
-- '959'
-- '366'
-- '239'
-- '334'
-- '496'
-- '603'
-- '486'
-- '482'

select * from section;

select distinct course_id
from section
where semester = 'Fall' and
course_id in (select course_id from section where semester = 'Spring');

select S.ID, S.name
from student as S
where exists ((select course_id
from course
where dept_name = 'Biology')
except
(select T.course_id
from takes as T
where S.ID = T.ID));
```

8. Lab day 5

```
use universipedia;
use university;

-- simple instruction to list all instructors
select name
from instructor;

SELECT
    name, course_id
FROM
    student,
    takes
WHERE
    student.ID = takes.ID;

SELECT
    name, course_id
FROM
    student
        NATURAL JOIN
    takes;

-- 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');

-- # dept_name
-- 'Athletics'
-- 'Biology'
-- 'Cybernetics'
-- 'Finance'
-- 'History'
-- 'Math'
-- 'Physics'
-- 'Psychology'

SELECT
    *
FROM
    instructor;

SELECT
    I.ID, COUNT(*)
FROM
    instructor AS I,
    teaches AS T
WHERE
    I.ID = T.ID
GROUP BY I.ID
ORDER BY I.ID;

SELECT
    I.ID, COUNT(T.ID) as number_of_sections
FROM
    instructor AS I
        NATURAL LEFT JOIN
    teaches AS T
GROUP BY I.ID
ORDER BY number_of_sections;

-- # ID, number_of_sections
-- '35579', '0'
-- '52647', '0'
-- '50885', '0'
-- '57180', '0'
-- '58558', '0'
-- '59795', '0'
-- '63395', '0'
-- '64871', '0'
-- '72553', '0'
-- '4034', '0'
```

```
-- '37687', '0'
-- '74426', '0'
-- '78699', '0'
-- '79653', '0'
-- '31955', '0'
-- '95030', '0'
-- '96895', '0'
-- '16807', '0'
-- '97302', '0'
-- '15347', '1'
-- '73623', '1'
-- '65931', '1'
-- '80759', '1'
-- '90376', '1'
-- '90643', '1'
-- '48570', '1'
-- '25946', '1'
-- '50330', '1'
-- '4233', '1'
-- '42782', '1'
-- '48507', '1'
-- '14365', '2'
-- '63287', '2'
-- '3335', '2'
-- '28400', '2'
-- '81991', '2'
-- '28097', '2'
-- '41930', '3'
-- '19368', '3'
-- '34175', '3'
-- '43779', '4'
-- '95709', '4'
-- '3199', '4'
-- '36897', '5'
-- '77346', '6'
-- '79081', '6'
-- '74420', '6'
-- '99052', '9'
-- '6569', '10'
-- '22591', '13'

-- 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&#39;s ID. Please display
your results in order
-- of course ID and do not display duplicate rows.

describe takes;

select distinct takes.course_id, D.ID
from
  ( select distinct S.ID, count(T.ID) as cnt
    from student as S natural left join takes as T
    group by S.ID ) as D natural inner join takes
where D.cnt>1 ;
```

```sql
select distinct course_id, ID
  from takes
    group by ID, course_id having count(*) > 2
    order by course_id;

-- # course_id, ID
-- '362', '16480'
-- '362', '16969'
-- '362', '27236'
-- '362', '39925'
-- '362', '39978'
-- '362', '44881'
-- '362', '49611'
-- '362', '5414'
-- '362', '69581'
-- '362', '9993'



-- select distinct S.ID, count(T.ID) as cnt
-- from student as S natural left join takes as T
-- group by S.ID;

show tables;
describe course;
select * from course;
describe student;

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

select name
from student natural inner join (
select ID, course_id
from takes
group by id, course_id having count(*) > 2 ) as T
where T.course_id in ( select course_id from course
  where dept_name = "Accounting" );

select ID from takes
where course_id in ( select course_id from course
  where dept_name = "Accounting")
    group by ID having count(*) > 2
    ;

    describe student;
select ID, course_id
from takes
group by id, course_id having count(*) > 2;

SELECT
    name
FROM
    student
        NATURAL JOIN
    (SELECT
        ID
```

```
    FROM
        takes
    WHERE
        course_id IN (SELECT
                course_id
            FROM
                course
            WHERE
                dept_name = 'Accounting')
    GROUP BY ID
    HAVING COUNT(*) > 2) AS T
WHERE
    dept_name = 'Biology';

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(*) > 2;

-- # name
-- 'Michael'
-- 'Dalton'
-- 'Shoji'
-- 'Wehen'
-- 'Uchiyama'
-- 'Schill'
-- 'Kaminsky'
-- 'Giannoulis'

-- Find the sections that had maximum enrollment in Fall 2010.
show tables;
desc takes;
select * from section;


select max(cnt) from (
  select count(*) as cnt from takes as T2
  group by course_id, sec_id
) as D;

select count(*) as cnt from takes as T2
  group by course_id, sec_id
    order by cnt;

select course_id, sec_id from takes as T1
where semester = "Fall" and year = 2010
group by course_id, sec_id
having count(*) = ( select max(cnt) from (
  select count(*) as cnt from takes as T2
    where semester = "Fall" and year = 2010
  group by course_id, sec_id
) as D );
```

```sql
SELECT course_id, sec_id
FROM takes
WHERE semester = 'Fall' AND year = 2010
GROUP BY course_id, sec_id
HAVING COUNT(ID) = (
    SELECT MAX(enrollment_count)
    FROM (
        SELECT COUNT(ID) AS enrollment_count
        FROM takes
        WHERE semester = 'Fall' AND year = 2010
        GROUP BY course_id, sec_id
    ) AS subquery
);

-- # course_id, sec_id
-- '867', '2'

--  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').

describe student;
describe takes;
describe course;

select name, count(*)
from student as S
join takes T on S.ID = T.ID
where T.course_id in (
  select course_id
    from course
    where title like "%Law%" )
group by S.ID
having count(*) >= (
  select count(course_id) / 2
  from (
    select course_id
    from course
    where title like "%Law%" )
        as D
  );

-- # name, count(*)
-- 'Nakajima', '4'
-- 'Nikut', '4'
-- 'Hahn-', '4'
-- 'Nanda', '4'
-- 'Schinag', '4'

-- 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. Note: the browser SQLite does not
support window functions.
show tables;
describe student;
```

```sql
describe takes;

select * from student;
select * from takes;

select name, grade
from student S
join takes T on S.ID = T.ID
where grade in ("A+", "A", "A-")
;

select row_number() over () as rnk, P.name from (
  select name, count(*) as cnt
  from student S
  join takes T on S.ID = T.ID
  where T.grade in ("A+", "A", "A-")
  group by T.ID
    ) as P
order by P.cnt desc, P.name
limit 10;

WITH StudentAGrades AS (
    SELECT s.ID, s.name, COUNT(*) AS a_count
    FROM student s
    JOIN takes t ON s.ID = t.ID
    WHERE t.grade IN ('A', 'A-', 'A+')
    GROUP BY s.ID, s.name
)
-- SELECT RANK() OVER (ORDER BY a_count DESC, name ASC) AS rnk, name
-- FROM StudentAGrades
-- ORDER BY rank
-- LIMIT 10
select ID, name, a_count
from StudentAGrades
order by a_count desc;

-- weirdness overloaded
SELECT s.ID, s.name, COUNT(*) AS a_count
    FROM student s
    JOIN takes t ON s.ID = t.ID
    WHERE t.grade IN ('A', 'A-', 'A+')
    GROUP BY s.ID, s.name
    order by a_count desc;

FROM student s
    JOIN takes t ON s.ID = t.ID
    WHERE t.grade IN ('A', 'A-', 'A+')
    GROUP BY s.ID, s.name
)
SELECT name, a_count
FROM (
    SELECT name, a_count, @curRank := @curRank + 1 AS rnk
    FROM StudentAGrades, (SELECT @curRank := 0) r
    ORDER BY a_count DESC, name ASC
) ranked
WHERE rnk <= 10
```

```
ORDER BY rnk;

-- # rnk, name
-- '1', 'Lepp'
-- '2', 'Eller'
-- '3', 'Masri'
-- '4', 'Vries'
-- '5', 'Åström'
-- '6', 'Gandhi'
-- '7', 'Greene'
-- '8', 'Haigh'
-- '9', 'McCarter'
-- '10', 'Sanchez'
```

9. Lab day 6

```
create database small_uni;
use small_uni;
use university;

show tables;

desc instructor;
select ID, salary from instructor;

-- small database
-- # ID, salary
-- '10101', '65000.00'
-- '12121', '90000.00'
-- '15151', '40000.00'
-- '22222', '95000.00'
-- '32343', '60000.00'
-- '33456', '87000.00'
-- '45565', '75000.00'
-- '58583', '62000.00'
-- '76543', '80000.00'
-- '76766', '72000.00'
-- '83821', '92000.00'
-- '98345', '80000.00'

-- use big database
-- # ID, salary
-- '14365', '32241.56'
-- '15347', '72140.88'
-- '16807', '98333.65'
-- '19368', '124651.41'
-- '22591', '59706.49'
-- '25946', '90891.69'
-- '28097', '35023.18'
-- '28400', '84982.92'
-- '31955', '71351.42'
-- '3199', '82534.37'
-- '3335', '80797.83'
-- '34175', '115469.11'
-- '35579', '62579.61'
-- '36897', '43770.36'
-- '37687', '104563.38'
```

```
-- '4034', '61387.56'
-- '41930', '50482.03'
-- '4233', '88791.45'
-- '42782', '34272.67'
-- '43779', '79070.08'
-- '48507', '107978.47'
-- '48570', '87549.80'
-- '50330', '108011.81'
-- '50885', '32570.50'
-- '52647', '87958.01'
-- '57180', '43966.29'
-- '58558', '66143.25'
-- '59795', '48803.38'
-- '63287', '103146.87'
-- '63395', '94333.99'
-- '64871', '45310.53'
-- '6569', '105311.38'
-- '65931', '79866.95'
-- '72553', '46397.59'
-- '73623', '90038.09'
-- '74420', '121141.99'
-- '74426', '106554.73'
-- '77346', '99382.59'
-- '78699', '59303.62'
-- '79081', '47307.10'
-- '79653', '89805.83'
-- '80759', '45538.32'
-- '81991', '77036.18'
-- '90376', '117836.50'
-- '90643', '57807.09'
-- '95030', '54805.11'
-- '95709', '118143.98'
-- '96895', '119921.41'
-- '97302', '51647.57'
-- '99052', '93348.83'

-- big database
-- # ID, salary
-- '14365', '32241.56'
-- '15347', '72140.88'
-- '16807', '98333.65'
-- '19368', '124651.41'
-- '22591', '59706.49'
-- '25946', '90891.69'
-- '28097', '35023.18'
-- '28400', '84982.92'
-- '31955', '71351.42'
-- '3199', '82534.37'
-- '3335', '80797.83'
-- '34175', '115469.11'
-- '35579', '62579.61'
-- '36897', '43770.36'
-- '37687', '104563.38'
-- '4034', '61387.56'
-- '41930', '50482.03'
-- '4233', '88791.45'
```

```
-- '42782', '34272.67'
-- '43779', '79070.08'
-- '48507', '107978.47'
-- '48570', '87549.80'
-- '50330', '108011.81'
-- '50885', '32570.50'
-- '52647', '87958.01'
-- '57180', '43966.29'
-- '58558', '66143.25'
-- '59795', '48803.38'
-- '63287', '103146.87'
-- '63395', '94333.99'
-- '64871', '45310.53'
-- '6569', '105311.38'
-- '65931', '79866.95'
-- '72553', '46397.59'
-- '73623', '90038.09'
-- '74420', '121141.99'
-- '74426', '106554.73'
-- '77346', '99382.59'
-- '78699', '59303.62'
-- '79081', '47307.10'
-- '79653', '89805.83'
-- '80759', '45538.32'
-- '81991', '77036.18'
-- '90376', '117836.50'
-- '90643', '57807.09'
-- '95030', '54805.11'
-- '95709', '118143.98'
-- '96895', '119921.41'
-- '97302', '51647.57'
-- '99052', '93348.83'

-- Find out the ID and salary of the instructor who gets more than $85,000.
select ID, salary from instructor
where salary > 85000;

-- small db
-- # ID, salary
-- '12121', '90000.00'
-- '22222', '95000.00'
-- '33456', '87000.00'
-- '83821', '92000.00'

-- big db
-- # ID, salary
-- '16807', '98333.65'
-- '19368', '124651.41'
-- '25946', '90891.69'
-- '34175', '115469.11'
-- '37687', '104563.38'
-- '4233', '88791.45'
-- '48507', '107978.47'
-- '48570', '87549.80'
-- '50330', '108011.81'
-- '52647', '87958.01'
```

```
-- '63287', '103146.87'
-- '63395', '94333.99'
-- '6569', '105311.38'
-- '73623', '90038.09'
-- '74420', '121141.99'
-- '74426', '106554.73'
-- '77346', '99382.59'
-- '79653', '89805.83'
-- '90376', '117836.50'
-- '95709', '118143.98'
-- '96895', '119921.41'
-- '99052', '93348.83'

-- Find out the department names and their budget at the university.
show tables;
desc department;

select dept_name, budget from department;

-- List out the names of the instructors from Computer Science who have more than
$70,000.
desc instructor;
select name from instructor
where salary > 70000;

-- For all instructors in the university who have taught some course, find their
names and the course ID of
-- all courses they taught.
desc instructor;
desc teaches;

select I.name, T.course_id
from instructor I
natural join teaches T
order by I.name;

-- Find the names of all instructors whose salary is greater than at least one
instructor in the Biology
-- department.
select name
from instructor
where dept_name = "Biology"
and salary > ( select min(S.salary) from (
  select salary
    from instructor
    where dept_name = "Biology"
) as S );

select min(S.salary) from ( select salary
    from instructor
    where dept_name = "Biology") as S;

select * from instructor;

-- Find the advisor of the student with ID 12345
select * from advisor
```

```sql
where s_ID = 12345;

-- Find the average salary of all instructors.
select avg(I.salary) average_salary from
(select salary from instructor) as I;

-- Find the names of all departments whose building name includes the substring
'Watson'.
select dept_name from department
where building like "%Watson%";

-- Find the names of instructors with salary amounts between $90,000 and $100,000.
select name from instructor
where salary between 90000 and 100000;

-- Find the instructor names and the courses they taught for all instructors in the
Biology department who
-- have taught some course.
select name, course_id
from instructor
natural left join teaches
where dept_name = "Biology";

-- Find the courses taught in Fall-2009 semester.
select course_id from teaches
where semester = "Fall" and year = 2009;

-- Find the set of all courses taught either in Fall-2009 or in Spring-2010.
select course_id from teaches
where (semester = "Fall" and year = 2009) or (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
union ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- Find the set of all courses taught in the Fall-2009 as well as in Spring-2010.
select course_id from teaches
where (semester = "Fall" and year = 2009) and (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
intersect ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- Find all courses taught in the Fall-2009 semester but not in the Spring-2010
semester.
select course_id from teaches
where (semester = "Fall" and year = 2009) and not (semester = "Spring" and year =
2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
except ( select course_id from teaches
where semester = "Spring" and year = 2010 );
```

```
-- Find all instructors who appear in the instructor relation with null values for
salary.
select * from instructor
where salary = NULL;

select * from instructor;

-- Find the average salary of instructors in the Finance department.
select avg(T.salary) from
( select salary from instructor
where dept_name = "Finance" ) as T;

-- Find the total number of instructors who teach a course in the Spring-2010
semester.
describe teaches;
select  count(T.id) from
( select id from instructor
  natural join teaches
    where semester = "Spring"
    and year = 2010 ) as T ;

-- Find the average salary in each department.
select dept_name, avg(salary)
from department
natural join instructor
group by dept_name;

-- Find the number of instructors in each department who teach a course in the
Spring-2010 semester.
select dept_name, count(ID)
from department
natural join instructor
where ID in (
  select ID from teaches
    where semester = "Spring"
    and year = 2010
)
group by dept_name;

-- List out the departments where the average salary of the instructors is more than
$42,000.
select dept_name
from department D
where ( select avg(salary) from (
  select salary
    from instructor I
    where D.dept_name = I.dept_name
) as T ) > 42000;

select distinct dept_name
from instructor
group by dept_name
having avg(salary) > 42000;

-- For each course section offered in 2009, find the average total credits (tot cred)
of all students enrolled
```

```sql
-- in the section, if the section had at least 2 students.
select course_id, sec_id, avg(tot_cred)
from takes
natural join student
where year = 2009
group by sec_id, course_id
having count(*) > 1;

-- Find all the courses taught in both the Fall-2009 and Spring-2010 semesters.
select course_id from teaches
where (semester = "Fall" and year = 2009) and (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
intersect ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- Select the names of instructors whose names are neither 'Mozart' nor 'Einstein'.
select name from instructor
where name not in ("Mozart", "Einstein");

select name from instructor;

-- Find the total number of (distinct) students who have taken course sections taught
by the instructor
-- with ID 110011.
select count(T.ID)
from takes T
natural join section S
where "110011" in (
  select ID
    from teaches ST
    where T.course_id = ST.course_id
    and T.sec_id = ST.sec_id
    and T.semester = ST.semester
    and T.year = ST.year
);

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';


select * from teaches;

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

select ID, name
from instructor
```

```sql
where salary > ( select min(T.salary) from (
  select salary
    from instructor
    where dept_name = "History"
) as T );

-- Find the names of all instructors that have a salary value greater than that of
each instructor in the
-- Biology department.
select name
from instructor
where salary > ( select max(T.salary) from (
  select salary
    from instructor
    where dept_name = "Biology"
) as T );

-- Find the departments that have the highest average salary.
select dept_name
from department
natural join instructor
group by dept_name
order by avg(salary) desc
limit 1;

select dept_name
from instructor
group by dept_name
order by avg(salary) desc
limit 1;

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

select course_id from teaches
where (semester = "Fall" and year = 2009) and (semester = "Spring" and year = 2010);

( select course_id from teaches
where semester = "Fall" and year = 2009 )
intersect ( select course_id from teaches
where semester = "Spring" and year = 2010 );

-- Find all students who have taken all the courses offered in the Biology
department.
with cnt as (
  select count(course_id) as ct
    from course
    where dept_name = "Biology"
)
select ID
from takes
where course_id in (
  select course_id
    from course
    where dept_name = "Biology"
)
```

```sql
group by ID
having count(*) = (
  select ct from cnt
);

-- Find all courses that were offered at most once in 2009.
select course_id
from takes
where year = 2009
group by course_id
having count(*) = 1;

-- Find all courses that were offered at least twice in 2009.
select course_id
from takes
where year = 2009
group by course_id
having count(*) > 1;

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

-- Find the maximum across all departments of the total salary at each department.
select sum(salary)
from department
natural join instructor
group by dept_name
order by sum(salary) desc
limit 1;

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

-- List all departments along with the number of instructors in each department.
select dept_name, count(ID)
from department
natural left join instructor
group by dept_name;

SELECT d.dept_name, COUNT(i.ID) AS num_instructors
FROM department d
LEFT JOIN instructor i ON d.dept_name = i.dept_name
GROUP BY d.dept_name;

-- 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;

-- Find the IDs of all students who were taught by an instructor named Einstein; make
```

```sql
-- sure there are no duplicates in the result.
select ID from instructor
where name = "Einstein";

select takes_table.ID
from takes takes_table
join teaches teaches_table
  on takes_table.course_id = teaches_table.course_id
    and takes_table.sec_id = teaches_table.sec_id
    and takes_table.semester = teaches_table.semester
    and takes_table.year = teaches_table.year
where teaches_table.ID = ( select ID from instructor
where name = "Einstein" ) ;

select * from teaches;

-- 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 ID, name
from student
natural join takes
where takes.course_id in (
  select course_id
    from course
    where dept_name = "Comp. Sci."
);

-- Find the course id, section id, and building for each section of a Biology course.
select course_id, sec_id, building
from section
natural join course
where dept_name = "Biology";

-- Output instructor names sorted by the ratio of their salary to their department's
budget
-- (in ascending order).
select name
from instructor
natural left join department
order by (salary/ budget) asc;

-- 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 name, building
from instructor
natural left join teaches
natural left join section;
```

10. Lab day 7

```sql
use university;

SELECT
    title
```

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

select distinct takes.ID
from takes, instructor, teaches
where takes.course_id = teaches.course_id and
takes.sec_id = teaches.sec_id and
takes.semester = teaches.semester and
takes.year = teaches.year and
teaches.id = instructor.id and
instructor.name = 'Einstein';

desc instructor;
select salary
from instructor
order by salary desc
limit 1;

select max(salary)
from
 instructor;

-- a. 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;
-- b. Find the IDs of all students who were taught by an instructor named Einstein;
make sure there are no duplicates in the result.
select takes_table.ID
from takes takes_table
join teaches teaches_table
  on takes_table.course_id = teaches_table.course_id
    and takes_table.sec_id = teaches_table.sec_id
    and takes_table.semester = teaches_table.semester
    and takes_table.year = teaches_table.year
where teaches_table.ID = ( select ID from instructor
where name = "Einstein" ) ;
-- c. Find the highest salary of any instructor.
desc instructor;
select salary
from instructor
order by salary desc
limit 1;

-- Find all instru tors earning the highest salary (there may be more than
-- one with the same salary).

select ID, name
from instructor
where salary = (
  select salary
  from instructor
  order by salary desc
  limit 1
```

```
);

select ID, name
from instructor
where salary = (select max(salary) from instructor);

-- Find the enrollment of each section that was offered in Fall 2017.
desc takes;

select course_id, sec_id, count(*)
from takes
where year = 2017 and
semester = "Fall"
group by course_id, sec_id;

select takes.course_id, takes.sec_id, count(ID)
from section, takes
where takes.course_id= section.course_id
  and takes.sec_id = section.sec_id
  and takes.semester = section.semester
  and takes.year = section.year
    and takes.semester = 'Fall'
    and takes.year = 2017
group by takes.course_id, takes.sec_id ;

-- Find the maximum enrollment, a ross all se tions, in Fall 2017.
SELECT course_id, sec_id
FROM takes
WHERE semester = 'Fall' AND year = 2017
GROUP BY course_id, sec_id
HAVING COUNT(ID) = (
    SELECT MAX(enrollment_count)
    FROM (
        SELECT COUNT(ID) AS enrollment_count
        FROM takes
        WHERE semester = 'Fall' AND year = 2017
        GROUP BY course_id, sec_id
    ) AS subquery
);

with sec_enrollment as (
  select takes.course_id, takes.sec_id, count(ID) as enrollment
    from section, takes
    where takes.year = section.year
    and takes.semester = section.semester
    and takes.course_id = section.course_id
        and takes.sec_id = section.sec_id
        and takes.semester = 'Fall' and takes.year = 2017
  group by takes.course_id, takes.sec_id)
select course_id, sec_id
from sec_enrollment
where enrollment = (select max(enrollment) from sec_enrollment) ;

-- Find the names of Biology students who have taken at least 3 Accounting courses.
select name
from student natural join (
```

```
  select ID from takes
  where course_id in ( select course_id from course
    where dept_name = "Accounting")
    group by ID having count(*) > 2
    ) as T where dept_name = "Biology";

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(*) > 2;

-- Find the sections that had the maximum enrollment in Fall 2017.
desc section;
desc takes;

select sec_id, course_id
from takes
where year = 2017
and semester = "Fall"
group by sec_id, course_id
having count(*)
order by count(*) desc
limit 1;

select * from section;

WITH sec_enrollment AS (
    SELECT
        t.course_id, t.sec_id, COUNT(t.ID) AS enrollment
    FROM section s
    JOIN takes t
        ON t.year = s.year
        AND t.semester = s.semester
        AND t.course_id = s.course_id
        AND t.sec_id = s.sec_id
    WHERE t.semester = 'Fall'
      AND t.year = 2017
    GROUP BY t.course_id, t.sec_id
)
SELECT course_id, sec_id
FROM sec_enrollment
WHERE enrollment = (SELECT MAX(enrollment) FROM sec_enrollment);

-- Find the total grade points earned by the student with ID '12345', across all
courses taken by the student.

desc course;

select sum(credits * points)
from takes, course, grade_points
where takes.grade = grade_points.grade
and takes.course_id = course.course_id
```

```sql
and ID = '12345';

select
 sum(credits * points)/sum(credits) as GPA
from
 takes, course, grade_points
where
 takes.grade = grade_points.grade
and takes.course_id = course.course_id
and ID= '12345';

select
 ID, sum(credits * points)/sum(credits) as GPA
from
 takes, ourse, grade_points
where
 takes.grade = grade_points.grade
and takes.course_id = course.course_id
group by ID;

--  In rease the salary of ea h instru tor in the Comp. S i. department by
-- 10%.

update instructor
set salary = salary * 1
where dept_name = "Comp. Sci.";

-- b. Delete all courses that have never been offered (i.e., do not occur in the
section relation).
delete from course
where course_id not in (
  select course_id from section
);

-- c. Insert every student whose tot_cred attribute is greater than 100 as an
instructor in the same department, with a salary of $10,000.
desc instructor;
desc student;

insert into instructor
select ID, name, dept_name, 100000
from student
where tot_cred > 100;

select
 count(distinct person.driver_id)
from
 accident, participated, person, owns
where
 accident.report_number = participated.report_number
and owns.driver_id = person.driver_id
and owns.license_plate = participated.license_plate
and year = 2017;

select * from student where ID = "12345";
create index studentID_index on student(ID);
```

```
desc student;

  select ID,
    case
      when name=score < 40 then "F"
      when name=score < 60 then "C"
      when name=score < 80 then "B"
      else "A"
    end
  from marks;

with grades as (
select ID,
  case
    when name=score < 40 then "F"
        when name=score < 60 then "C"
        when name=score < 80 then "B"
        else "A"
    end
from marks )
select grade, count(ID)
from grades
group by grade;

select dept_name
from department
where lower(dept_name) like "%sci%";

-- a. Find the ID, name, and city of residence of each employee who works for
-- “First Bank Corporation”.
select e.ID, e.person name, city
from employee as e, works as w
where w.company_name = "First Bank Corporation" and
  w.ID = e.ID;

-- b. Find the ID, name, and city of residence of each employee who works for “First
Bank Corporation” and earns more than $10000.
select e.ID, e.person name, city
from employee as e, works as w
where w.company_name = "First Bank Corporation" and
  w.salary > 10000 and
  w.ID = e.ID;

describe instructor;

with tb as (
  select salary
    from instructor
)
select salary from tb
;

describe student;
SELECT DISTINCT student.ID, student.name
FROM student
```

```sql
JOIN takes ON student.ID = takes.ID
JOIN course ON takes.course_id = course.course_id
WHERE course.dept_name = 'Comp. Sci.';

    -- 11. Write the following queries in SQL, using the university schema:
-- a. 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 student.ID, student.name
FROM student
JOIN takes ON student.ID = takes.ID
JOIN course ON takes.course_id = course.course_id
WHERE course.dept_name = 'Comp. Sci.';

-- b. Find the ID and name of each student who has not taken any course offered
before 2017.
SELECT student.ID, student.name
FROM student
WHERE student.ID NOT IN (
    SELECT takes.ID
    FROM takes
    WHERE takes.year < 2017
);

-- c. For each department, find the maximum salary of instructors in that department.
You may assume that every department has at least one instructor.
SELECT dept_name, MAX(salary) AS max_salary
FROM instructor
GROUP BY dept_name;
-- d. Find the lowest, across all departments, of the per-department maximum salary
computed by the preceding query.
SELECT MIN(max_salary) AS lowest_max_salary
FROM (
    SELECT dept_name, MAX(salary) AS max_salary
    FROM instructor
    GROUP BY dept_name
) AS dept_max_salaries;

-- Write the SQL statements using the university schema to perform the following
operations:
-- a. Create a new course “CS-001”, titled “Weekly Seminar”, with 0 credits.
desc course;
show create table course;
SHOW COLUMNS FROM course;
INSERT INTO course (course_id, title, dept_name, credits)
VALUES ('CS-001', 'Weekly Seminar', 'Comp. Sci.', 0);
-- b. Create a section of this course in Fall 2017, with sec_id of 1, and with the
location of this section not yet specified.
desc section;
INSERT INTO section (course_id, sec_id, semester, year)
VALUES ('CS-101', 1, 'Fall', 2017);
select * from course;
-- c. Enroll every student in the Comp. Sci. department in the above section.
INSERT INTO takes (ID, course_id, sec_id, semester, year)
SELECT student.ID, 'CS-001', 1, 'Fall', 2017
FROM student
WHERE student.dept_name = 'Comp. Sci.';
```

```sql
-- d. Delete enrollments in the above section where the student’s ID is 12345.
DELETE FROM takes
WHERE course_id = 'CS-001'
  AND sec_id = 1
  AND semester = 'Fall'
  AND year = 2017
  AND ID = '12345';
-- e. Delete the course CS-001. What will happen if you run this delete statement
without first deleting offerings (sections) of this course?
DELETE FROM course
WHERE course_id = 'CS-001';
-- f. Delete all takes tuples corresponding to any section of any course with the
word “advanced” as a part of the title; ignore case when matching the word with the
title.
DELETE FROM takes
WHERE course_id IN (
    SELECT course_id
    FROM course
    WHERE LOWER(title) LIKE '%advanced%'
);

-- Write SQL DDL corresponding to the schema in Figure 3.17. Make any reasonable
assumptions about data types, and be sure to declare primary and foreign keys.
CREATE DATABASE InsuranceDB;
USE InsuranceDB;

-- Drop things a bit
DROP TABLE person;
DROP TABLE car;
DROP TABLE accident;
DROP TABLE owns;
DROP TABLE participated;

-- Person Table
CREATE TABLE person (
    driver_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    address VARCHAR(100)
);

-- Car Table
CREATE TABLE car (
    license_plate VARCHAR(20) PRIMARY KEY,
    model VARCHAR(50),
    year YEAR
);

-- Accident Table
CREATE TABLE accident (
    report_number INT PRIMARY KEY,
    year YEAR,  -- Changed from VARCHAR(10) to YEAR
    location VARCHAR(100)
);

-- Owns Table
-- (Relationship between person and car)
```

```sql
CREATE TABLE owns (
    driver_id INT,
    license_plate VARCHAR(20),
    PRIMARY KEY (driver_id, license_plate),
    FOREIGN KEY (driver_id) REFERENCES person(driver_id) ON DELETE CASCADE,
    FOREIGN KEY (license_plate) REFERENCES car(license_plate) ON DELETE CASCADE
);

-- Participated Table
-- (Records cars and drivers involved in accidents)
CREATE TABLE participated (
    report_number INT,
    license_plate VARCHAR(20),
    driver_id INT,
    damage_amount DECIMAL(10,2) DEFAULT 0.00,
    PRIMARY KEY (report_number, license_plate, driver_id),
    FOREIGN KEY (report_number) REFERENCES accident(report_number) ON DELETE
CASCADE,
    FOREIGN KEY (license_plate) REFERENCES car(license_plate) ON DELETE SET NULL,
    FOREIGN KEY (driver_id) REFERENCES person(driver_id) ON DELETE SET NULL
);

SELECT employee.ID, employee.name
FROM employee
JOIN works ON employee.ID = works.ID
WHERE works.salary > (
    SELECT AVG(salary)
    FROM works w2
    WHERE w2.company_name = works.company_name
);

SELECT AVG(borrowed_count) AS avg_books_borrowed
FROM (
    SELECT m.memb_no, COUNT(b.isbn) AS borrowed_count
    FROM member m
    LEFT JOIN borrowed b ON m.memb_no = b.memb_no
    GROUP BY m.memb_no
) AS member_borrow_counts;

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

SELECT t.ID
FROM takes t
WHERE t.grade IS NOT NULL
AND 2 <= (
  select count(course_id)
    from takes t2
    where t2.ID = t.ID
    and t2.course_id = t.course_id
)
GROUP BY t.ID
```

```sql
HAVING COUNT(DISTINCT t.course_id) >= 3;

SELECT i.name, i.ID
FROM instructor i
WHERE NOT EXISTS (
    SELECT c.course_id
    FROM course c
    WHERE c.dept_name = i.dept_name
    EXCEPT
    SELECT t.course_id
    FROM teaches t
    WHERE t.ID = i.ID
)
ORDER BY i.name;

SELECT i.name, i.ID
FROM instructor i
WHERE NOT EXISTS (
    SELECT c.course_id
    FROM course c
    WHERE c.dept_name = i.dept_name
    AND NOT EXISTS (
        SELECT t.course_id
        FROM teaches t
        WHERE t.ID = i.ID AND t.course_id = c.course_id
    )
)
ORDER BY i.name;

SELECT s.ID, s.name
FROM student s
WHERE s.dept_name = 'History'
AND s.name LIKE 'D%'
AND (
    SELECT COUNT(*)
    FROM takes t
    WHERE t.ID = s.ID
    AND t.course_id IN (
        SELECT c.course_id
        FROM course c
        WHERE c.dept_name = 'Music'
    )
) < 5;

  SELECT AVG(salary) - (SUM(salary) / COUNT(*))
  FROM instructor;

SELECT distinct i.ID, i.name
FROM instructor i
LEFT JOIN teaches t ON i.ID = t.ID
LEFT JOIN takes tk ON t.course_id = tk.course_id
WHERE tk.grade != 'A' AND EXISTS (
  select tk2.grade
    from takes tk2
    where t.course_id = tk2.course_id
    and tk2.grade is not null
```

```
);

SELECT i.ID, i.name
FROM instructor i
LEFT JOIN teaches t ON i.ID = t.ID
LEFT JOIN takes tk ON t.course_id = tk.course_id
WHERE tk.grade != 'A' OR tk.grade IS NULL
GROUP BY i.ID;

SELECT DISTINCT c.course_id, 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.end_hr >= 12 AND c.dept_name = 'Comp. Sci.';

SELECT s.course_id, s.sec_id, s.year, s.semester, COUNT(t.ID) AS num
FROM section s
LEFT JOIN takes t ON s.course_id = t.course_id
AND s.sec_id = t.sec_id
GROUP BY course_id, sec_id, year, semester
HAVING num > 0;

WITH section_enrollment AS (
    SELECT s.course_id, s.sec_id, s.year, s.semester, COUNT(t.ID) AS num
    FROM section s
    LEFT JOIN takes t ON s.course_id = t.course_id AND s.sec_id = t.sec_id
    GROUP BY course_id, sec_id, year, semester
)
SELECT course_id, sec_id, year, semester, num
FROM section_enrollment
WHERE num = (SELECT MAX(num) FROM section_enrollment);

SELECT instructor.name
FROM instructor
WHERE instructor.dept_name = 'History';

SELECT instructor.ID, department.dept_name
FROM instructor,department
WHERE instructor.dept_name = department.dept_name
AND department.budget > 95000;
```

11. Lab day 8

```
use small_uni;

-- Create a relational database for employee salary maintenance with attributes
EmployeeID (PK), EmployeeName, Department, Salary, Month.
-- From Employee Table, transform rows into columns in MySQL


CREATE TABLE employee (
    employee_id INT,
    employee_name VARCHAR(50),
    department VARCHAR(50),
    salary INT,
    month VARCHAR(50),
    PRIMARY KEY (employee_id, month)
```

```
);

truncate employee;

INSERT INTO employee (employee_id, employee_name, department, salary, month)
VALUES
(1, 'Alice', 'HR', 1, 'January'),
(1, 'Alice', 'HR', 2, 'February'),
(1, 'Alice', 'HR', 3, 'March'),
(2, 'BOB', 'IT', 4, 'January'),
(2, 'BOB', 'IT', 5, 'February'),
(2, 'BOB', 'IT', 6, 'March');

INSERT INTO employee (employee_id, employee_name, department, salary, month)
VALUES
(3, 'BOB', 'IT', 5555, 'March');

select * from employee;

SELECT
    employee_id,
    employee_name,
    department,
    MAX(CASE WHEN Month = 'January' THEN Salary END) AS January,
    MAX(CASE WHEN Month = 'February' THEN Salary END) AS February,
    MAX(CASE WHEN Month = 'March' THEN Salary END) AS March
FROM employee
GROUP BY employee_id, employee_name, department;

SELECT
    employee_id,
    employee_name,
    department
FROM employee
GROUP BY employee_id, employee_name, department;

SELECT
    Employee_ID,
    Employee_Name,
    Department,
    MAX(CASE WHEN Month = 'January' THEN Salary END) AS January,
    MAX(CASE WHEN Month = 'February' THEN Salary END) AS February,
    MAX(CASE WHEN Month = 'March' THEN Salary END) AS March
FROM employee
GROUP BY Employee_ID, Employee_Name, Department;

SELECT e.Employee_ID, e.Employee_Name, e.Department, s.Salary, s.Month
FROM employee e
INNER JOIN employee s ON e.Employee_ID = s.Employee_ID;

DELETE FROM employee
WHERE Employee_ID NOT IN (
    SELECT MIN(Employee_ID) FROM employee
    GROUP BY Employee_ID, Month
);
```

```sql
SELECT MIN(Employee_ID) FROM employee
    GROUP BY Employee_ID, Month;

SELECT course.DEPT_NAME ,instructor.SALARY
FROM course ,instructor
WHERE SALARY > all (SELECT SALARY FROM instructor WHERE DEPT_NAME = 'Astronomy')
ORDER BY DEPT_NAME;

select * from instructor;
```