

<https://nptel.ac.in/courses/106/105/106105175/>

<https://tutorial.techaltum.com/sql-queries-for-practice.html>

<https://www.slideshare.net/AzizulMamun1/dbmsintermediate-sql>

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>tot_cred</i>
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Figure 4.1 The *student* relation.

<i>ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>	<i>grade</i>
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	null

Figure 4.2 The *takes* relation.

**The Natural Join: query outputs only students who have taken some course**

**select name, course id**

**from student, takes**

**where student. ID = takes. ID ;**

**Both are same**

**select name, course id  
from student natural join takes;**

**select  $A_1, A_2, \dots, A_n$   
from  $r_1$  natural join  $r_2$  natural join  $\dots$  natural join  $r_m$   
where  $P$ ;**

**List the names of students along with the titles of  
courses that they have taken**

**select name, title  
from student natural join takes, course  
where takes.course id = course.course id;**

**Join Conditions:join expression containing the on con-  
dition**

**select \*  
from student join takes on student. ID = takes. ID ;**

**Both are same**

**select \*  
from student, takes  
where student. ID = takes. ID ;**

**Outer Joins**

a list of all students, displaying their ID , and name, dept name, and tot cred, along with the courses that they have taken.

There are **three forms of outer join**:

The **left outer join** preserves tuples only in the relation named before (to the left of) the left outer join operation.

The **right outer join** preserves tuples only in the relation named after (to the right of) the right outer join operation.

- The **full outer join** preserves tuples in both relations.

**select \***

**from student natural left outer join takes;**

2/11/2020

### 1)Rename

```
select name as instructor_name, course_id as  
id_of_the_course  
from instructor, teaches  
where instructor. ID = teaches. ID ;
```

instructor_name	id_of_the_course
....	.....

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

```
select T .name, S.course id  
from instructor as T , teaches as S  
where T . ID = S. ID ;
```

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

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

4)String Operations:::Find the names of all departments whose building name includes 'Wat-son'

```
select dept name  
from department  
where building like '%Watson%';    IT adWatsonsc
```

CS Watson ,EC adcWatson,,ME Watsonme

5)“ \* ”-->“all attributes.”

All attributes of instructor are to be selected

**select instructor.\***

**from instructor, teaches**

**where instructor. ID = teaches. ID ;**

**Select \***

**from instructor;**

11	bbb
10	aaa

6)Ordering the Display of Tuples by order by

*classroom(building, room\_number, capacity)*  
*department(dept\_name, building, budget)*  
*course(course\_id, title, dept\_name, credits)*  
*instructor(ID, name, dept\_name, salary)*  
*section(course\_id, sec\_id, semester, year, building, room\_number, time\_slot\_id)*  
*teaches(ID, course\_id, sec\_id, semester, year)*  
*student(ID, name, dept\_name, tot\_cred)*  
*takes(ID, course\_id, sec\_id, semester, year, grade)*  
*advisor(s\_ID, i\_ID)*  
*time\_slot(time\_slot\_id, day, start\_time, end\_time)*  
*prereq(course\_id, prereq\_id)*

To list in alphabetic order all instructors in the Physics department

**select name**

**from instructor**

**where dept name = 'Physics'**

**order by name desc;**

7)to list the entire instructor relation in descending order of salary. If several instructors have the same salary, we order them in ascending order by name.

**select \***

**from instructor**

**order by salary desc, name asc;**

**dfg 7500**

**fty 7250**

**aaa—7000**

**bbb-7000**

**asd 5000**

8)find the names of instructors with salary amounts between \$90,000 and \$100,000

**select name**

**from instructor**

**where salary between 90000 and 100000;**

**OR**

**select name**

**from instructor**

**where salary <= 100000 and salary >= 90000;**

9)To find the set of all courses taught either in Fall 2017 or in Spring 2018, or both

(select course id

from section

where semester = 'Fall' and year= 2017)

union

(select course id

from section

where semester = 'Spring' and year= 2018);

10)To find the set of all courses taught in both the Fall 2017 and Spring 2018

(select course id

from section

where semester = 'Fall' and year= 2017)

intersect

(select course id

from section

where semester = 'Spring' and year= 2018);

11)To find all courses taught in the Fall 2017 semester but not in the Spring 2018 semester,

(select course id

from section


where semester = 'Fall' and year= 2017)

except

(select course id

from section

where semester = 'Spring' and year= 2018);



## Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- The predicate **is null** can be used to check for null values.
  - 🔑 E.g. Find all loan number which appear in the *loan* relation with null values for *amount*.

```
select loan-number
from loan
where amount is null
```
- The result of any arithmetic expression involving *null* is *null*
  - 🔑 E.g. 5 + null returns null
- However, aggregate functions simply ignore nulls

5-null-->null, 5-3=2, 5>null-->null

No two null values are equal. Comparisons between two null values, or between a null value and any other value, return unknown because the value of each NULL is unknown.

SQL treats as **unknown** the result of any comparison involving a null value.

$(a+b) == (c*d)$



• **and**: The result of *true and unknown* is *unknown*, *false and unknown* is *false* while  
 The following table shows the results of applying an AND operator to two Boolean expressions where one expression returns UNKNOWN.

Expression 1	Expression 2	Result
TRUE	UNKNOWN	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN
FALSE	UNKNOWN	FALSE

The following table shows the results of applying an OR operator to two Boolean expressions where one expression returns UNKNOWN.

Expression 1	Expression 2	Result
TRUE	UNKNOWN	TRUE
UNKNOWN	UNKNOWN	UNKNOWN
FALSE	UNKNOWN	UNKNOWN

## where color is not null

100	Chilly powder	red
<div> <div> SELECT ProductID,  Name,  Color  FROM Production.Product  WHERE Color IS NULL </div> </div>		
101	Wheat	yellow
102	Rice	NULL
103	Turm	Yell

```
SELECT ProductID,  
       Name,  
       Color  
FROM Production.Product  
WHERE Color IS NOT NULL
```

**12)To find all instructors who appear in the instructor**

**relation with null values for salary**

**select name**

**from instructor**

**where salary is null;**

**13)Find the average salary of instructors in the Computer Science department.**

**select avg (salary)**

**from instructor**

**where dept name = 'Comp. Sci.';**

**14)Aggregation with Grouping::“Find the average salary in each department.”**

**Select dept name, avg (salary) as avg\_salary,MAX( salary) AS  
MAXIMUM\_SALARY**

**from instructor**

**group by dept name;**

dept_name	avg_salary	<b>MAXIMUM_SALARY</b>
Biology	72000	72000
Comp.sci	X	92000
Ele.eng	80000	80000
finance	Y	90000
history	Z	62000
music	40000	40000
phy	P	95000

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

Tuples of the *instructor* relation, grouped by the *dept\_name* attribute.

**15)Find the average salary of all instructors**

**select avg (salary)**

**from instructor;**

**16)Find the number of instructors in each department who teach a course in the Spring 2018 semester.”**

select dept name, count (distinct ID ) as instr count  
 from instructor, teaches  
 where instructor. ID = teaches. ID and  
 semester = 'Spring' and year = 2018  
 group by dept name;

<i>dept_name</i>	<i>instr_count</i>
Comp. Sci.	3
Finance	1
History	1
Music	1

in  
 where the average salary is more than \$42,000.”

17)Find the average salary of instructors those departments

select dept name, avg (salary) as avg salary  
 from instructor  
 group by dept name  
 having avg (salary) > 42000;

<i>dept_name</i>	<i>avg_salary</i>
Physics	91000
Elec. Eng.	80000
Finance	85000
Comp. Sci.	77333
Biology	72000
History	61000

18)For each course section offered in 2017, find the average total

credits (tot cred) of all students enrolled in the section, if the section has at least 2 students.”

```
select course_id, semester, year, sec_id, avg (tot_cred)  
from student, takes  
where student.ID= takes.ID and year = 2017  
group by course_id, semester, year, sec_id  
having count (ID) >= 2;
```

30/10/2020

Friday

SQL

I)Set Operations

The Union Operation

The Intersect Operation

## The Except Operation

### II)Null Values

### III)Aggregate Functions

Find the average salary and max salary,minimum of instructors in the Computer Science department.

Find out the total salary of all teachers who is working in physics department and getting more than 50,000 salary.

```
select sum(salary) as total_salary  
from instructor  
where dept name = 'physics' and  
salary>50,000;
```

total_salary
200000

```
classroom(building, room_number, capacity)  
department(dept_name, building, budget)  
course(course_id, title, dept_name, credits)  
instructor(ID, name, dept_name, salary)  
section(course_id, sec_id, semester, year, building, room_number, time_slot_id)  
teaches(ID, course_id, sec_id, semester, year)  
student(ID, name, dept_name, tot_cred)  
takes(ID, course_id, sec_id, semester, year, grade)  
advisor(s_ID, i_ID)  
time_slot(time_slot_id, day, start_time, end_time)  
prereq(course_id, prereq_id)
```

For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught and the faculty should belongs to computer science.

```
select name, course id
from instructor, teaches
where instructor.ID = teaches.ID and
dept_name='com_sci';
```

```
select name, instructor.dept name, building  
from instructor, department  
where instructor.dept name= department.dept  
name;  
join(algebra)
```

```
select <<columns>>  
from <<tables>>  
where <<conditions>>
```

```
select distinct name  
from instructor  
where dept name = 'Comp. Sci.' or salary > 70000;
```



<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

**Figure 2.4** Unsorted display of the *instructor* relation.

**select**  
**ID , name, dept name, salary \* 1.1**  
**from instructor;**

*classroom*(building, room\_number, capacity)  
*department*(dept\_name, building, budget)  
*course*(course\_id, title, dept\_name, credits)  
*instructor*(ID, name, dept\_name, salary)  
*section*(course\_id, sec\_id, semester, year, building, room\_number, time\_slot\_id)  
*teaches*(ID, course\_id, sec\_id, semester, year)  
*student*(ID, name, dept\_name, tot\_cred)  
*takes*(ID, course\_id, sec\_id, semester, year, grade)  
*advisor*(s\_ID, i\_ID)  
*time\_slot*(time\_slot\_id, day, start\_time, end\_time)  
*prereq*(course\_id, prereq\_id)

**The result of “Retrieve the names of all instructors, along with their department names and department building name.”**

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

**If we wished to find only instructor names and course identifiers for instructors in the Computer Science department,**

**First,**  
**two relations in the from clause may have attributes with the same name, in which case an attribute name is duplicated in the result.**

**Second,**  
**if we use an arithmetic expression in the select clause, the resultant attribute does not have a name.**

**Third,**  
even if an attribute name can be derived from the base relations as in the preceding example, we may want to change the attribute name in the result.

Hence, SQL provides a way of renaming the attributes of a result relation. It uses the as clause

## **Data def commands in sql**

- 1)create**
- 2)drop**
- 3)alter**

## **Data Man**

algebra	sql
---------	-----

project(id,name)instructor

<i>ID</i>	<i>name</i>
22222	Einstein
12121	Wu
32343	El Said
45565	Katz
98345	Kim
76766	Crick
10101	Srinivasan

**select id,name  
from instructor;**

*classroom*(*building*, *room\_number*, *capacity*)

*department*(*dept\_name*, *building*, *budget*)

*course*(*course\_id*, *title*, *dept\_name*, *credits*)

*instructor*(*ID*, *name*, *dept\_name*, *salary*)

*section*(*course\_id*, *sec\_id*, *semester*, *year*, *building*, *room\_number*, *time\_slot\_id*)

*teaches*(*ID*, *course\_id*, *sec\_id*, *semester*, *year*)

*student*(*ID*, *name*, *dept\_name*, *tot\_cred*)

*takes*(*ID*, *course\_id*, *sec\_id*, *semester*, *year*, *grade*)

*advisor*(*s\_ID*, *i\_ID*)

*time\_slot*(*time\_slot\_id*, *day*, *start\_time*, *end\_time*)

*prereq*(*course\_id*, *prereq\_id*)